SCRATCH: Shot Consultation and Refinement Applied Through Computer Hardware

Group 17 Authors:

Luke Ambray	Mark Nelson	Goran Lalich	Mena Mishriky
Computer Engineering	*Electrical Engineering*	*Electrical Engineering*	*Electrical Engineering*

Mentor:

Dr. Chung Yong Chan

TABLE OF CONTENTS

List of Figures

List of Tables

1. EXECUTIVE SUMMARY

The game of billiards is commonly known as pool and has been played for many years and has players from all different skill levels, backgrounds, and cultures. Unfortunately, this variety of cultures and backgrounds does not include a significant portion of our modern day society: visually-impaired personnel. These can lead to a feeling of exclusion and lack of appreciation amongst the visually impaired community. Additionally, because of the same reasons that constrain billiards from the visually impaired, many other activities are not visually impaired friendly. As a result of this clear lack of empathy, visually impaired people have a very limited pool (pun not intended) of activities to participate in. This project is aimed at addressing this issue. Our team created a system by which visually impaired people are able to both enjoy and even improve in the game of billiards. Hopefully, other entities and entrepreneurs can follow suit and devise such entertainment that is inclusive to not just the visually impaired community, but rather the entirety of the special needs community.

SCRATCH is a billiards training tool which can improve the performance of both visually impaired users as well as non-visually impaired users. This project contains four main subsystems that will communicate data that can be used to provide shot feedback as well as being able to help the visually impaired user with navigation and shot taking, including live aim assistance. Each of these subsystems performs specific tasks that help the user while at the same time not negatively impact performance. The subsystems are designed in a way that is most convenient to the user, with no protrusions, heavy components, or components suffering from uncontrollable heat generation and power dissipation.

This paper documents SCRATCH's design process, this includes discussing key technologies, communications, theory, implementation, testing procedures, as well as fabrication. We first supply the context and justification for the project and then provide a detailed explanation of the engineering requirements for SCRATCH. Important technologies that relate to the project are briefly discussed and then the important components used for each of the subsystems are compared and then selected. These are analyzed by performance and functionality as well as cost to determine the best fit for the system. The design is then discussed in more detail this includes a fabrication plan as well as PCB schematics and real world best practices for PCB design. Additionally, the software flow for each subsystem, as well as for the overall system, is discussed in detail via flowcharts and supplementary materials. Administrative content is then mentioned at the end of the document, this includes project milestones, project budget, and an index of sources used to assist with the development of this project.

2. PROJECT DESCRIPTION

The game of billiards, commonly known as pool, is enjoyed by millions of players daily. In many situations, the game of billiards can serve as an icebreaker and a facilitator of social interactions between different people. In other situations, the game is played competitively by professional athletes. Nevertheless, there are many people who cannot reap the social benefits of the game due to a lack of proficiency. Additionally, there has not been any significant effort to include visually impaired individuals in the game by making it accessible to them. While there have been a couple of papers written, information is scarce and the projects were never commercialized. This creates two problems: (1) there exists a proficiency wall that beginners face and find difficult to overcome when trying to play the game of billiards and (2) visually impaired personnel are excluded from the game, regardless of talent or ability. As engineers, it is our job to solve such problems through technology.

2.1 PROJECT FUNCTION

Overall Functionality: The overall functionality of the project is to solve the two problems mentioned above. In short, it is to provide a lightweight, low-power, and convenient training system that aids users in improving their proficiency in the game of billiards by providing pre-shot guidance and post-shot feedback to the user in a simple-to-understand manner. The other goal is to make the game of billiards accessible to the visually impaired user. SCRATCH focuses on data collection, processing, and presentation via wearable technologies and modified equipment that is custom-made for this project. Most of the data that scratch focuses on comes from interaction with the user. SCRATCH consists of four main subsystems; their functionalities are highlighted below.

HUD Functionality: The HUD consists of three different subsystems, these are an audio system, a camera system, and a display system. The audio and display serve to provide impaired and unimpaired users the pre and post-shot information. This information will aid in improving user performance. The camera is on the HUD to place it on the user's head and ensure it has a clear view of the cue stick's impact point on the ball for post-shot feedback. The camera will aid the system in determining the point of impact between the cue stick and the cue ball. Overall, the HUD is to provide a low power, low heat, and lightweight solution which will aid in making both visually impaired and non-visually impaired users better at the game of billiards. This is through providing pre (including aim assistance for the visually impaired user) and post-shot information.

The display and related peripherals will be used to clearly project the following required information directly in front of the user's point of view. Before the shot is taken, cue impact point, shot strength, and the target ball color will be communicated to the user. After the shot is taken, there will be two feedback screens. The first will show the target cue impact placement VS. the actual cue impact point, and the second will show the target shot strength VS. the actual shot strength.

For impaired users, the display will be turned off to conserve power and reduce heat output, and the audio system will be enabled. A speaker will be used to convey all instructions to the impaired individual. This includes the information received via the VISION team for localizing and directing the user to the table in the correct orientation. Then, the hand placement and cue aim will be fine-tuned via the other SCRATCH subsystems.

The camera system is being used to help gather the unimpaired user's post-shot feedback. The camera will know to take the picture based on IMU data received from the cue stick and then, working in conjunction with a laser on the cue stick, the point of impact between the cue stick and cue ball will be computed.

Cue Stick Functionality: The primary function of the cue stick subsystem is to accurately determine the orientation and speed of the user's cue stick. This gives the rest of the system the data necessary to provide meaningful and constructive feedback to both impaired and non-impaired users. The secondary function of the cue stick is to give the user control of the GUI displayed on the HUD.

As this system is mounted to a cue stick, it is critical that this system be as lightweight as possible to minimize changes to the user's shot. As this system does not have a physical connection to any other subsystem, it must be able to wirelessly communicate with the other subsystems. For that same reason, the stick's system must be a low-power solution as it will be battery-powered.

Glove Functionality: The primary function of the glove is to assist the vision-impaired user with pre-shot feedback in order to be facing the cue ball at the correct angle as well as the at the correct distance from the ball, this angle will be provided by the VISION team. This system will have to be very compact in order to not impact the user's shot.

Central Control Unit (CCU) Functionality: As the name implies, the central control unit is the subsystem responsible for sending and receiving data to and from the other subsystems. This is also the portion of our project that would communicate with the VISION team's system.

When communicating with the HUD, the CCU has two major responsibilities. The first is rendering and sending the images that will be displayed on the HUD's display when in normal use. When the system is being used by the visually impaired, the CCU is responsible for telling the HUD's audio system which audio clip to play. The second responsibility is to interface with the HUD's camera system. The CCU is responsible for controlling when the camera begins recording. The CCU then receives the image from the HUD; with this data, the CCU renders a low-resolution image of the location of cue impact on the ball that is sent back to the HUD to provide feedback to the user.

When communicating with the cue stick, the CCU's primary responsibility is to receive the motion from the cue stick and use it to provide meaningful feedback to the user. The secondary responsibility is to receive the button input data from the stick and change the

state of the CCU's program accordingly. Related to both of these responsibilities is the fact that the CCU occasionally has to send switching information to the cue stick depending on the button inputs.

When communicating with the glove, the CCU receives sensor data from the glove. With this data, the CCU determines whether the user should position their hand closer or farther from the cue ball.

When communicating with the VISION team, the CCU is responsible for receiving the angle and speed data for the desired shot. The CCU is also responsible for notifying the other system when SCRATCH is ready for the next shot.

An additional planned feature for the CCU was to make it responsible for keeping track of user data. This includes the user's mapped cue speed, the number of games played, as well as other useful metrics for gauging user performance. The CCU would have saved each user's data in separate files and store them on the CCU's SD card. However, due to time constraints, our team was unable to implement this feature

2.2 PROJECT GOALS AND OBJECTIVES

The objective of SCRATCH is to provide a training system that can improve the performance of visually-impaired and non-visually impaired users in the game of billiards. This project occurs in conjunction with VISION, which has the same final objective. While VISION focuses on table analysis and getting the visually-impaired user to the table (among other things), SCRATCH specifically focuses on interaction with the user. This is done in the form of pre- and post-shot guidance and feedback suitable to the visually-impaired and the unimpaired user. The data is provided in a simple-to-understand manner in both display and audio formats in order to assist different users. While a small portion of this data will be procured from the VISION team (the data can be easily simulated in case of VISION failure), most of the data procurement is done via a variety of sensors placed on different parts of the user in the form of low-power, lightweight, and convenient wearable technologies that are custom made for the project.

In order to have the ability to complete the project without relying on the success of the VISION team, we have the option to simulate the data that they would be sending. This includes setting up basic shots and simulating the angle of the user in order to take a shot and the force of the shot.

- HUD Goals:
    - Basic
        - HUD can provide basic audio commands to guide the visually impaired user (commands include turn right/left, move hand forward/backward)
        - HUD display system provides clear visual information to guide the unimpaired user on pre-shot information such as cue ball impact point and shot strength
    - Advanced

        - HUD can use camera to show point of impact between cue stick and ball
        - HUD Display system will provide post-shot information to give user feedback on actual impact point and strength compared to desired impact point and strength
    - Stretch
        - Use an FPGA to do image processing instead of the Pi
- Stick Goals:
    - Basic
        - The system can collect and transmit data regarding the orientation and force applied to the cue stick
        - The system can collect and transmit button input data
- Glove Goals:
    - Basic
        - Read angle of users body to align shot
        - Locate ball position along angle
- Central Control Unit Goals:
    - Basic
        - The system can interpret cue stick motion data and send visual and auditory feedback to the HUD
        - The system can respond to button inputs and change display output accordingly
    - Stretch
        - The system can maintain user profiles that contain shot speed data and play history
        - The system can communicate with and use data from the VISION team

2.3 REQUIREMENT SPECIFICATIONS

Specifications for each individual subsystem are listed below:

HUD Requirement Specifications:

- Should weigh less than 1500 grams
- Does not protrude more than 2’’ out from the head of the user
- Temperature of all the skin touching portions of HUD must not exceed 120 degrees Fahrenheit
- Should be able to provide post-shot feedback within 8 seconds of impact with the ball
    - This includes data roundtrip and Bluetooth latency
- Display should be crisp and easy to view in standard, well-lit room
- Sampling user aim and providing audio feedback pre-shot should be done on 2-second intervals. This is the time between the end of the first audio feedback and the start of the second

Cue Stick Requirement Specifications:

- Determine the orientation of the cue stick within 5 degrees
- Determine the speed of the cue stick within 1 meter per second
- The latency between data being collected and sent to other subsystems must be less than 1 second
- The latency between a button being pressed and the rest of the system recognizing it must be less than 500 milliseconds
- System must be able to be used by both impaired and non-impaired players
- System must be able to sustain operation for 30 minutes
- Entire subsystem must weigh less than 1 kilograms

Glove Requirement Specifications:

- Glove weight less than 500 grams
- Glove design should not obstruct players shot
- Electronics should not exceed 100 degrees Fahrenheit
- For detection of user direction latency should be under 1 second
- Total circuitry area should not exceed size of back of palm less than 80cm^2

Central Control Unit Specifications:

- Be able to render and send images to HUD within 4 seconds of receiving images taken by camera
- Be able to respond to button inputs from the cue stick within 500 milliseconds
- Be able to trigger audio feedback within 1 second of correction being recognized

3D Printed Housings:

- Housings will entirely enclose sensitive components to prevent damage
- Balance and ergonomics will be a top priority as they will all be held or worn

The specifications that were our team's primary focus are summarized in Table 2.1 below.

Table 2.1 Prioritized Specifications

Specification	Value
HUD should be lightweight	< 1500 grams
HUD should be able to provide post-shot feedback after impact with the ball	< 8 seconds
Sampling user aim and providing audio feedback pre-shot should be done on predetermined intervals.	< 2 seconds
Cue stick should be able to determine the orientation of the cue stick within an acceptable margin of error	< 3 degrees
Cue stick should be able to determine the speed of the cue stick within an acceptable margin of error	< 1 meter per second
Electronics should not reach uncomfortable temperatures	< 100 degrees Fahrenheit
Central control unit should be able to respond to button inputs from the cue stick with minimal latency	< 500 milliseconds

2.4 PROJECT BLOCK DIAGRAM

For implementing our project we followed the block diagram shown above in Figure 2.1, this includes four total subsystems that will be in charge of performing a task and then communicating this information back to the Central Control Unit (CCU). Since this design is used to be worn by the user each subsystem is using wireless communication to send this information back to the CCU, but on the inside of each system we used wired connections. One of the features also associated with this project is the ability to communicate with the VISION team, this team will send us some shot selection data that we can then use to help guide the user, this information will be sent to the CCU to be later transmitted to the individual subsystems.

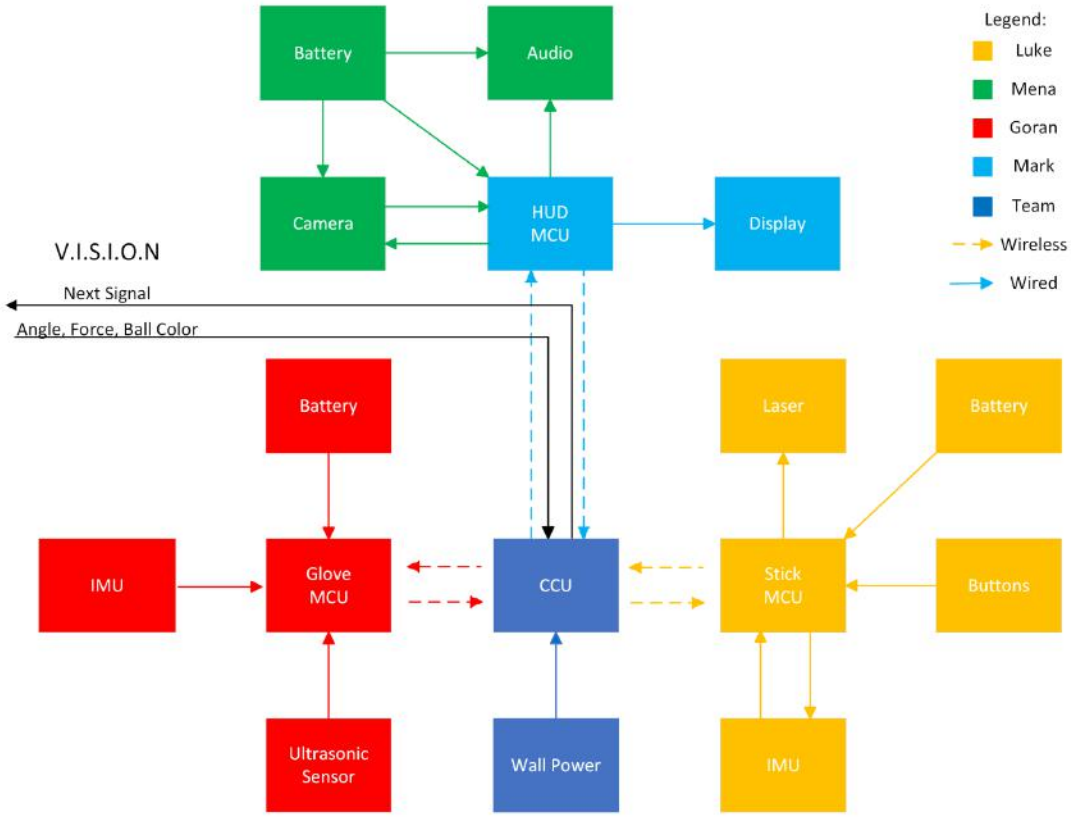


Figure 2.1: Project Hardware Diagram

2.5 HOUSE OF QUALITY

The House of Quality shown below in Figure 2.2 was extremely helpful when establishing and relating the requirements set by both the marketing and engineering teams. This table allowed the two teams to work together and better understand the relationships and consequences of each of their desired elements. The direction of each arrow clearly shows the direction of improvement relating to each requirement.

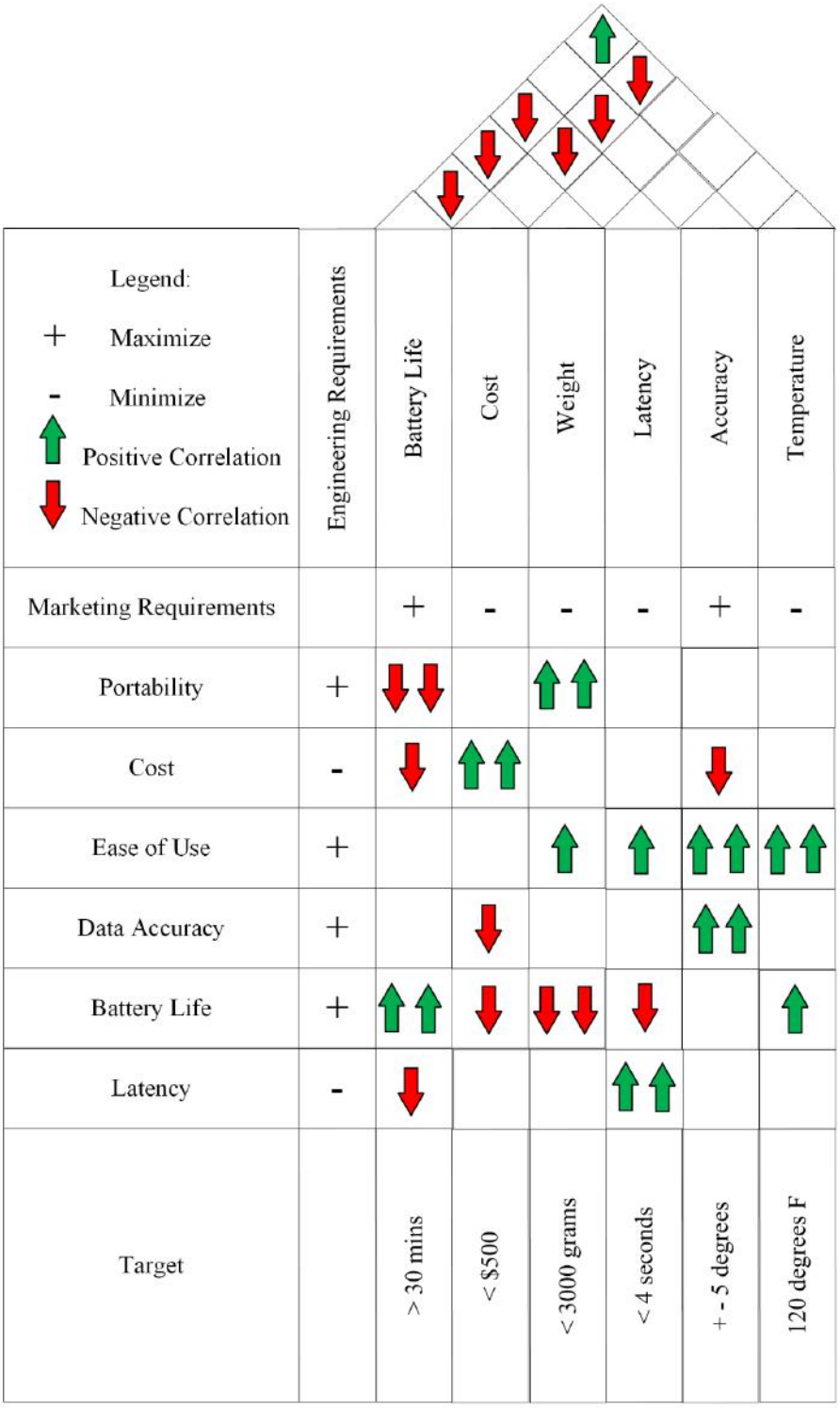


Figure 2.2: House of Quality

3. RESEARCH

In this section, our team will discuss the research conducted prior to the design of our project. This includes relevant technologies, similar projects, and components that may be used in the development of SCRATCH. When discussing possible components, each section will have a table comparing the various options and the option we chose will be highlighted.

3.1 RELEVANT TECHNOLOGIES AND SIMILAR PROJECTS

In the section directly below, research related directly to relevant technologies and similar projects will be explored. These projects can provide great insight into learning what does and doesn't work, especially as fully developed products could potentially have years of R&D at their backs.

3.1.1 SMART GLASSES

Projector Based Smart Glasses: Smart glasses are currently one of the many next big things in the tech world with the idea of enhancing human capability and productivity as much as possible. Research and improvements are constantly being made and perhaps the most desirable feature of these glasses is the prospect of Augmented Reality (AR) becoming, well, a reality. Augments Reality creates a technological link between the user and the surrounding world. Unlike Virtual Reality, which inserts the user into a completely new and computer-generated world, AR overlays information that is directly reality to what is happening in front of the user as they go about their lives. One simple example of this would be walking directions that update in real time based on where the user is and project arrows onto the glasses which appear as though they are on and around the sidewalks. For this purpose, however, the AR portion of these glasses is not what is most interesting, but rather what is interesting is simply the act of displaying an image in front of a user in real time. There are a huge variety of products available in the Smart Glasses market that project the image directly onto the lens in front of the user's eye. They accomplish this in many different ways, but one way is by using a curved mirror and reflecting a display's light, similar to the technique of the automotive Heads Up Displays that will be discussed in further detail below. Another way is by projecting the image through a waveguide. This is a new technology that often utilizes specifically a diffractive waveguide which bends the light waves into the correct orientation and projects it onto the lens.

Display-Based Smart Glasses: The second and much less common technique is to hover the display in front of the user's eye. The purpose of this style of smart glasses is much more in line with this project's intentions of developing smart glasses. Rather than implementing an AR system, the goal is simple to display information in a more accessible way. There are varying results associated with hovering the display in front of the user's eye rather than using a miniaturized projector of sorts. While the largest disadvantages are that it becomes much more cumbersome and clunky, there are of course some major advantages, these being the cost and simplicity. This is the somewhat

obvious result of avoiding the difficult design and inevitable manufacturing task of manipulating light waves. Another additional advantage of using a display rather than a miniaturized projector is the additional reduction in design difficulty associated with using the heavily established display technology. These can be in the form of both wireless, or to even further lessen costs, wired.

One specific example of this is the Vufine VUF-110. [3.1] This option is wired and utilizes a unique two-part mounting system that makes it a universal attachment to any user's glasses they already own which is an extremely intriguing advantage that could be employed by this project's system as well.

Figure 3.1: Example of Display Based Smart Glasses. Reproduction permission obtained from Vufine.

3.1.2 AUTOMOTIVE HEADS UP DISPLAYS

Integrated HUDs: Many automotive manufacturers have made it a standard option to integrate a Heads Up Display into the dashboard of the vehicle. [3.2] By using a system of mirrors and lenses, important vehicle telemetry that is normally placed on the speedometer, tachometer, or navigation system can be placed directly in the user's sight line, allowing them to view the information without looking away from the road. The system can be summarized as a larger, and as a result of the size, a somewhat simpler version of what is used in Smart Glasses. Understanding of how these HUDs are engineered can allow for better understanding of how Smart Glasses are designed, and perhaps a combination of techniques can be used to engineer the final product.

Aftermarket HUDs: As some cars do not come standard with the HUD, there are aftermarket options available as well. These work in a much simpler manner, as rather than utilizing the windshield, they often come with their own, flat, clear surface that, like the display based smart glasses discussed above, skips past much of the complex optics and photonics challenges associated with the more complex, integrated HUD. A valuable piece of information observed by these, is that they require the glass to be coated to provide the clearest reflection possible.

3.1.3 MOTION CONTROLS IN VIDEO GAMES

As one of the primary goals of our project is to accurately record the motion of the player's cue stick, it is important that we look at some real-world examples of motion tracking. One industry where this has become widely popular is video games. In the following sections, we will look at how various systems implemented motion controls and see which implementations would be most suitable for applications.

Nintendo Wii: The Nintendo Wii launched in North America on November 19, 2006. The primary gimmick that allowed the Wii to become the best-selling console of its generation was the introduction of the Wii Remote, commonly referred to as the "Wiimote", a wireless controller that, besides traditional video game controls, could be used as a pointing device or for gesture recognition.

The main way that the Wii Remote tracks motion is through its integrated three-axis MEMS accelerometers. These accelerometers record the direction and strength of the force being applied to the Wii Remote. In addition to measuring applied on the X-, Y-, and Z-axis, the accelerometers also allow the remote to sense its bank and pitch. It does so by measuring the angle that gravity makes with the remote.

However, while the accelerometers can measure movement, they are not able to measure relative position. This is where the Wii Sensor Bar comes into play. Contrary to popular belief, the sensor bar does not send any data; it just contains two sets of five infrared lights. The Wii Remote has infrared sensors on its front in order to triangulate its position relative to the bar. If the lights are at the bottom of the remote's view, the pointer should be at the top of the screen. The same goes for all the other directions.

In order to send all of its position, acceleration, and button data to the console, the Wiimote uses a Broadcom BCM2042 to send data. It uses the standard Bluetooth HID protocol, which is based on the USB HID, to communicate with the console. In addition to the Bluetooth chip, the Wiimote also has a 16KB EEPROM chip and a microcontroller that is used to process the sensor and button data.

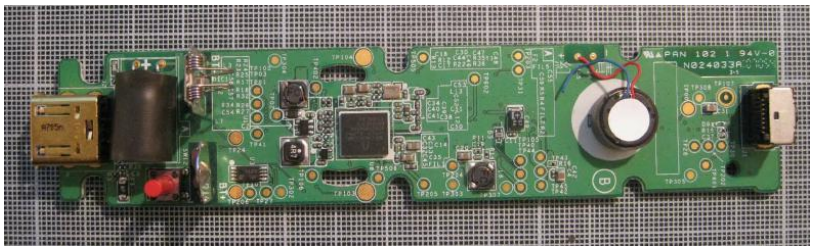


Figure 3.2: Nintendo Wii Remote internals. Reproduction permission requested from Stuart Smith.

Xbox Kinect: The Xbox Kinect was released for the Xbox 360 on November 4, 2010. Unlike the Wii which used a specialized controller, the Kinect does its motion tracking through the use of a camera, depth sensor, and specialized software. The camera is RGB, has a pixel resolution of 640x480, and has a frame rate of thirty frames per second. The

depth sensor has a CMOS sensor and an infrared projector. This helps it create 3D imagery as it measures the time of flight of the infrared light after reflecting off of the object being tracked. The data from both of these sensors are then interpreted by Kinect's software in order to create a 3D avatar.

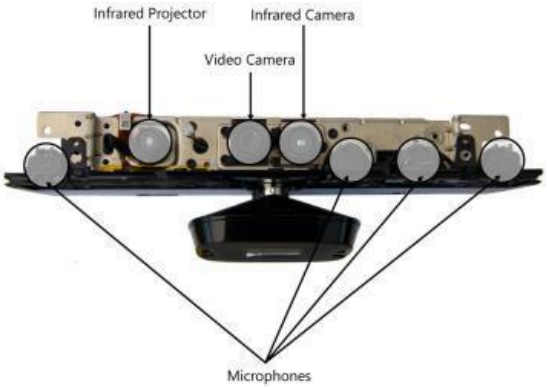


Figure 3.3: Internals of an Xbox 360 Kinect. Reproduction permission requested from Microsoft Press.

Meta Quest 2: The Meta Quest 2, formerly the Oculus Quest 2, is the newest platform with motion tracking that will be discussed, and released on October 13, 2020. Unlike other virtual reality headsets, the Quest 2 does not use external cameras to track where the player is in 3D space, but rather it uses a system called "inside-out tracking". With inside-out tracking, the cameras that track where the player and the controllers are on the headset itself. On each controller, there is a set of infrared LED rings that the cameras continuously take pictures of. Using these pictures, as well as the accelerometer and gyroscope data that is sent via Bluetooth Low Energy (BLE), the headset is able to triangulate the position of the controllers.

Figure 3.4: Meta Quest 2 infrared tracking system. Reproduction permission requested from Mechatech.

Key Takeaways: After reviewing these three different methods of implementing motion controls in video games, the approach that we found most relevant to our project was the Nintendo Wii's Wiimote. This is due to the system's relative simplicity as it primarily relies on accelerometer data as opposed to the other two's highly complex tracking software.

3.1.4 UNIVERSITY OF COLORADO PROPOSAL

In the process of doing background research, our team came across a capstone project proposal from a group of students at the University of Colorado at Boulder in 2009. Their proposal states that they would construct a pool cue attachment that would measure the movements of the cue. They primarily planned to do this using accelerometers and gyroscopes and would then send that data to a computer using Bluetooth. Like our project, they would provide the user with feedback to improve their shots.

Their proposal also stated other goals that we did not include in our portion of this project. The first extra feature that they proposed was the ability to distinguish between a break shot and a regular shot as they involve different motions. The second feature was the ability to use multiple cues simultaneously. The last feature that they mention is a tracking system for the balls; this is a feature that is being implemented by the table-focused team of our project.

Their bill of materials includes, but is not limited to:

- Various accelerometers
- Various gyroscopes
- Multiple ATmega microcontrollers
- Bluetooth modules
- Various PCBs
- Webcam
- Lithium-ion batteries
- Electric toothbrush

The estimated cost of their project was over $1000.

3.1.5 COIN OPERATED POOL TABLES

We can look at automatic pool tables that are commonly used in areas where you have to pay to play a round of pool. These billiards tables are able to differentiate between the cue ball and the other balls. One of the possible ways that these automated billiards tables can differentiate is by using a cue ball that is larger in size than the other balls so the mechanical system underneath the table is unable to let the cue ball through so that it will come out to continue play instead of being locked until the next game like the other balls. This system would not really affect our target audience which is people who are trying to get better at billiards as well as assisting visually impaired users. This system does affect better players who would not be adapted to the larger ball. This system would make it difficult to get an accurate result regarding hand position for our visually impaired users. Another way that we can differentiate the balls is using a cue ball with a magnetic core. In an automatic pool table they use a magnetic cue ball that would trigger a magnetic sensor most likely a hall sensor and would then trigger a separating system. A similar way that the billiards tables use is having a metal core that can be attracted to magnets that would separate the cue ball from other balls allowing it to go down a different track to be recovered.

3.2 CORE COMPONENTS AND PARTS SELECTION

3.2.1 DISPLAY

As discussed above, there are many different ideas towards providing this easy access to information. The best solution likely lies in creating a combination of the above ideas. For example, when looking towards buying any pair of the smart glasses, they, while being very useful and efficient, are often extremely expensive and reliant on proprietary software that can add severe complexity in configuring the glasses to specific use cases and integrating them into an already complex system. Therefore, the best option lies in creating a unique set of glasses.

Smart glasses typically use what is effectively a miniaturized projector to place the display with the information on the lens directly in front of the user's eye. This allows the user to receive the information at any time without moving their body or losing focus on the world ahead of them. This is exactly the inspiration of the automotive Heads-Up Display. Using these both as inspiration, a "simple" HUD integrated into a set of glasses will be used in this project to allow one to easily access the valuable information provided by the other components of the training system, while still in the position to make the shot.

The task of designing a Heads-Up Display from scratch to display the information poses many challenges, the largest of which relate to the ergonomics of the device, with parameters such as size, weight, and their resulting layout restrictions

The first and most important thing to consider is how the image will be displayed in front of the user. The two primary options of displaying the image are either an LED display or a projector, such as those used in smart glasses. The format chosen for this project, is rather than using a miniature projector, an LED display will be used in conjunction with mirrors and lenses to project the image onto a surface just past the lens of the actual glasses surface in front of the user's eye. While the projector has the advantage in ensuring small weight and size, it, like the smart glasses, has disadvantages which lie in its lack of simplicity and high cost.

This technique using mirrors and lenses is inspired by car manufacturers' Heads-Up Displays as shown in the figure below. [3.2] Therefore, the first consideration in designing the entire Heads-Up display assembly is tied to choosing an LED display. The below sections will explore the considerations made in choosing the correct LED display for this application. Design and configuration using the concepts displayed in the figures will be further discussed in the HUD design section.

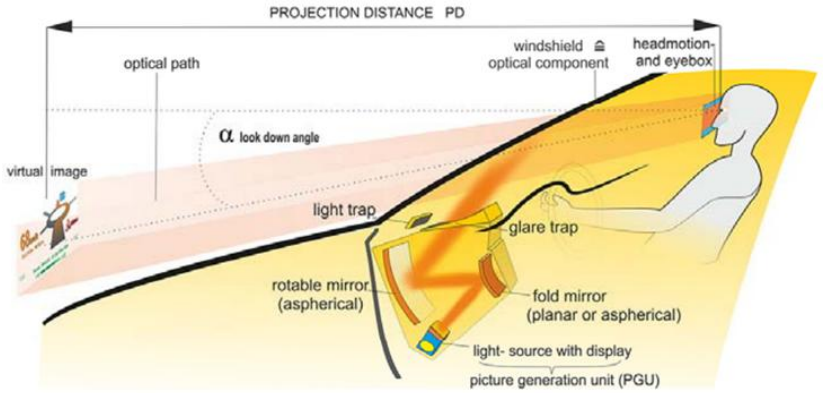


Figure 3.5: Demonstration of car industry HUD technique that will be used as inspiration for this HUD. Reproduction permission requested from Continental Automotive.

Display Types

LED: LED displays are flat panels that use a large array of Light-Emitting Diodes to project an image.[3.3] In most cases, every LED represents one pixel of the overall image. These can range from extremely small to extremely large, and in this case, will be approximately 100 pixels wide by 100 pixels tall, creating a complete image by coloring each of the 1000 points. Modern displays are also often referred to as OLED or Organic Light-Emitting Diode displays. Traditional LED displays are based on a p-n diode structure and use a backlight. Then, after attaching a forward biased voltage to the diode, light is emitted every time an electron crosses over the junction due to the releasing of surplus energy. OLEDs, on the other hand, while working in a similar fashion, use organic molecules to produce the electrons and holes rather than p-type and n-type metals. These layers are shown in Figure 3.6 and when a forward biased voltage is applied between the cathode and anode, the electrons and holes move together. Like the traditional LED display, when the two combine in the emissive layer, energy is released in the form of a photon.

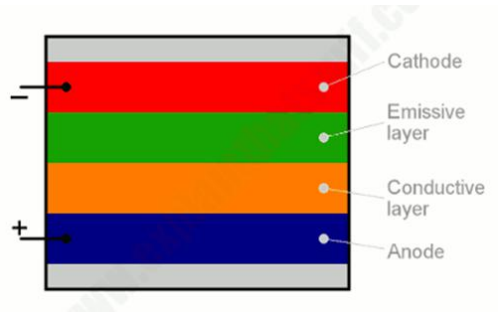


Figure 3.6: Diagram showing the components of OLED displays. Reproduction permission obtained from Explain That Stuff.

LCD: LCD, or Liquid Crystal Display, is another extremely common way manufacturers choose to produce high quality displays. [3.4] Unlike LED displays, which use an array of diodes by which each individual one emits light, LCD displays require one large backlight to produce the image which initially comes as one solid color, also known as monochrome. Then, the overall system uses many layers to alter the light into the desired image. The LCD panel works in conjunction with filters and films that produce the final desired image. These filters and films give the image the correct color and clarity, while the LCD panel's crystals can be electrically manipulated to bend the backlight's light rays into the correct shape. These displays used to be used for phones, televisions, and computer monitors, however nowadays they've been replaced by the onset of LED technology. Still, however, these are a very popular option for segmented displays and other applications where LCD's high quality contrast, bright, and clear monochrome images can best be applied.

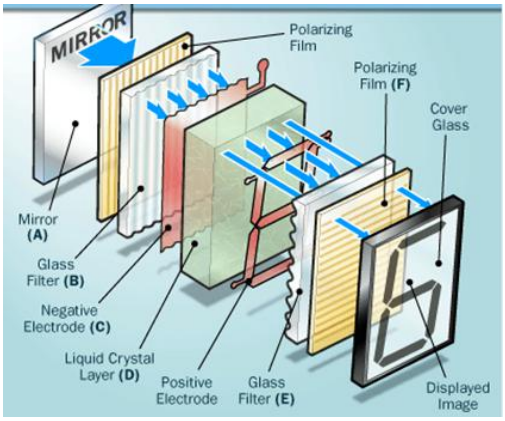


Figure 3.7: Layout of entire LCD display. Reproduction permission requested from Xenarc.

LCOS: One popular display type specifically used in Augment/Virtual reality as well as automotive Heads Up Display applications is the Liquid Crystal on Silicon Display. [3.5] The easiest way to understand how a LCOS display is to understand the one it is based on, which is Digital Light Processing (DLP). DLP works by controlling very tiny mirrors and flipping them between on and off rapidly. When they are on, they are projecting the light towards the lens and the brightness of their resulting pixel is determined by the length that they are on. This idea is used in the Liquid Crystal on Silicon display by using the layers of crystals to reflect the light rather than mirrors. In this way, it is similar in concept to an LCD display as well. Typically, these displays are very small by comparison to typical LCD or LED displays as a result of their architecture. In color displays, there is a red, green, and blue LCOS structure which all combine to produce the final image. This is shown in Figure 3.7. The white light is split into each layer and then recombined through the projection lens to produce the image seen on the display. As a result of the size restriction, this technique is also commonly used in TV/Movie projectors.

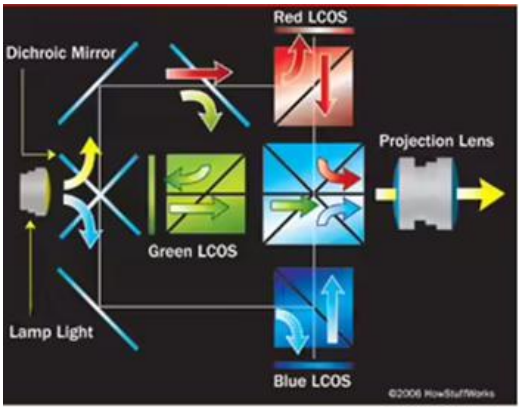


Figure 3.8: Layout of multiple colored LCOS systems that combine to create a color image. Reproduction permission requested from How Stuff Works**.**

Information Required on Display: The first consideration in choosing the display is determining how large the size needs to be. This is mostly determined by what information is required to be displayed on the HUD. The feedback that the user will receive in total is:

1) Shot selection
2) Cue Ball Aiming point
3) Shot strength/power
4) Post Shot Analysis Feedback

Of these, 1, 3, 4, 5, and 6 will need to be displayed legibly on the HUD, while the shot path is left on the monitor fed directly from the solution algorithm. According to professional pool players, the goal is to almost entirely visualize the shot before the player gets into position. This is because how one approaches the table plays a large role in aim comfort and as a result of that, precision. What this means, is the player will already have the shot selection and path determined before they get in position to aim. Leaving only aim (Both slight path adjustments and cue ball striking point) and power as the last considerations.

To break apart the information slightly, and reduce the size of the display wherever possible, a pre- and post-shot screen may have to be used.

Display Options: The largest contributor to the layout isn't the information, but rather the size. To ensure that the HUD isn't bulky and uncomfortable to wear, the display must be as small as possible while maintaining a clear image after being both reflected and refracted in the process of projecting the image in front of the user's eye.

As a size constraint, only displays with dimensions of approximately 1"x1" or smaller will be considered at first. If necessary, a larger size may be explored in the future. First,

potential displays will be explored to consider what options are available with the given size restraint. Then possible layouts given those display sizes will be configured.

The most important parameters that need to be considered when choosing a display are the outer dimensions, viewing area dimensions, viewing area resolution, voltage requirements, data transfer form factor, whether the display is color, brightness and finally, because the team consists of college students, the cost.

The different data transfer form factors that are found below are SPI and I2C. The advantages and disadvantages of each will be discussed in further detail before the final decision is made.

The first option is the 0.95" ARDUINO OLED COLOR 96X64. The overall outer dimensions of the module is 1.07” W x 1.21” H (27.4 mm W x 30.7 mm H) with a viewing area of 0.87” W x 0.61” H (22.1 mm W x 15.40 mm H). This OLED display has a resolution of 96x64, and the communication interface is 4-wire SPI. The voltage requirement is 3.33V and the brightness level is 100 cd/m^2. The final notable feature of this display is that it has Red, Green, and Blue (RGB) graphics coloring. These displays can be found between $5 - $12. [1]

Figure 3.9: Display Option 1. Reproduction permission requested from ARDUINO.

The next option is much skinnier than the others, but also much longer. The overall dimensions of the device are 1.5” x .47” (38 mm x 12 mm) while the active area is 0.9” diagonally, or with a resolution of 128x32. The data transfer form factor is I2C, the OLED display is mono-color with a blue backlight, and the voltage input requirement is 3.3-5 V. This device is $18 for 5 units on Amazon, or $3.60 each. [3]

Figure 3.10: Display Option 2. Reproduction permission requested from HiLetgo.

The next two options are the HiLetgo 0.96” SSD1306 I2C IIC SPI Serial 128x64 OLED LCD display and the very similar UCRTONICS 0.96” Module 12864 128x64 Yellow Blue Display Board. The HiLetgo modules have many options that give it a high potential advantage in the final decision. Both displays have outer dimensions of 1.06” x

1.06” x .16” (27mm x 27mm x 4.1mm) and viewing areas of 0.85” x 0.44” (21.74mm x 11.2mm). The HiLetgo OLED display comes in either I2C or SPI with mono-color options of white and blue, and a dual color option of blue/yellow. This module costs $7.29 on Amazon. [2] The UCTRONICS display uses I2C and has a blue/yellow display. One major flaw in the blue/yellow display is that not the entire display has the option available, ¼ of the screen is yellow while ¾ of the screen is blue. It costs $6.99 on Amazon. [4]

Figure 3.11: Display Option 3 (left) and Display Option 4 (right). Reproduction permission requested from HiLetgo.

The largest display considered has a diagonal screen width of 1.5” (38.1 mm) which yields an active viewing area of 1.06”x1.06” (26.86 mm x 26.86 mm) and a resolution of 128x128. This option will only be considered if the size constraints become too large to display the required information. The overall dimensions of the module are 1.75” x 1.45” (44.5 mm x 37.0 mm). The device interfaces with either I2C or 4-wire SPI and costs $18.99 on Amazon.

The five above options have varying sizes and resolutions and allow for the best decision to be made with regard to display choice with the given restraints regarding size and complexity.

Figure 3.12: Display Option 5. Reproduction permission requested from HiLetgo

OmniVision is a leading developer of LCOS display panels with integrated drivers. They have multiple options with resolutions ranging from 1280x720 to 1920x1080. Since intense clarity is not necessarily required for this application, the focus will be applied to the 1280x720 models. With a pixel size of only 6.4 μm, and an overall size of .92"x.35" (23.4 mm x 9.0 mm) and a diagonal screen size of .37" (9.4 mm), this is by far the smallest screen size considered. However, as a result of its use of the LCOS projection system it achieves outstanding clarity at this size and with the use of multiple screens, one will still be able to achieve displaying the required information. The power requirements when fully active is 200 mW and only needs a 1.5V input to operate.

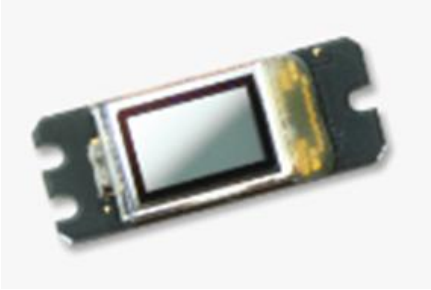


Figure 3.13: Display Option 6. Reproduction permission requested from OmniVision Technologies**.**

Communication Comparison for Display Purposes: When comparing the two communication options given by the displays of I2C and SPI, there are of course advantages and disadvantages to both. Additionally, given that the communication will be strictly within its own subsystem that is only responsible for displaying and projecting the image, integration with the rest of the main system is a requirement only of the wireless communication module within the HUD and doesn't have to be considered here. This wireless communication module will also dictate the maximum data transfer rate to the display itself.

When comparing the two directly, I2C is slower and synchronous, meaning communication can only happen in one direction at a time as a result of being a half-duplex configuration. SPI on the other hand has a full-duplex and can communicate faster as a result of both the full-duplex and the lack of data framing requirements. While I2C is slower, its integration is comparatively simpler, having only two wires when compared to SPI's four. This weight and clutter reduction gives I2C a major advantage in this application. The additional wires allow for SPI to communicate in both directions at the same time, however, in this application of a display, most if not all the data transfer is only going to go in one direction anyway, being into the display's driver module.

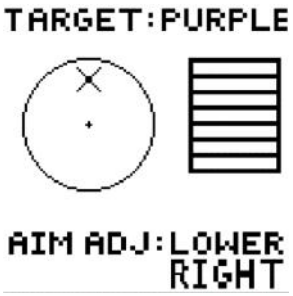
All in all, unless the speed is a high priority, SPI's additional complexity does not add utility in this use case, which leaves I2C as the preferred communication protocol for the display.

Display Layout Options: Given the options that were listed above, the following resolutions are available: 96X64 (in B&W or RGB), 128x32 (in B&W), and 128x64 (in B&W or Blue/Yellow).

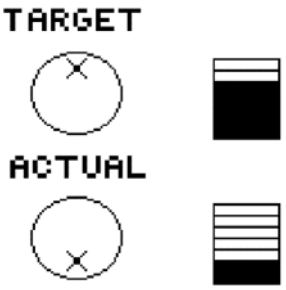
The following shows potential layouts that could be used across the varying display sizes. These were created on a pixel art creating software, showing exactly what they could look like at the varying screen resolutions, allowing for much better visualization and further assistance in the decision-making process. The first display shown will start with the largest available option, and work down from there to view the limitations.

128x128 (1:1) Display, Mono-color

PRE SHOT SCREEN



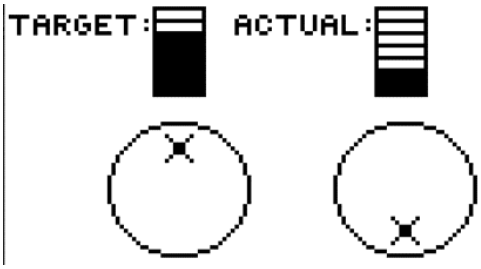
POST SHOT SCREEN



128x64 (2:1) OLED Display, Mono-color

PRE SHOT SCREEN

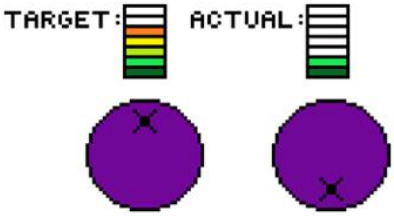
POST SHOT SCREEN



128x64 (2:1) Display, Color

PRE SHOT SCREEN

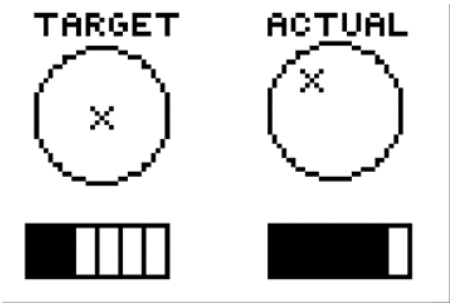
POST SHOT SCREEN



96X64 (3:2) Display, Mono-color

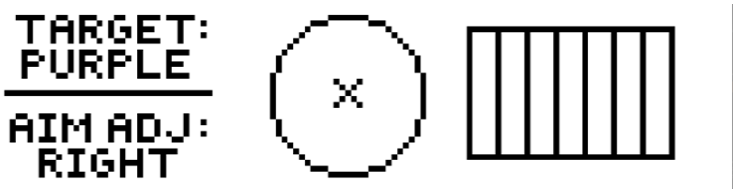
PRE SHOT SCREEN

POST SHOT SCREEN

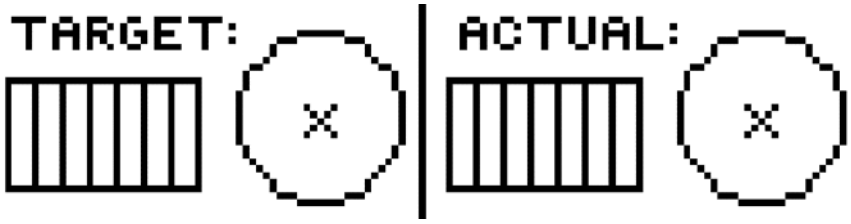


128x32 (4:1) Display, Mono-color

PRE SHOT SCREEN



POST SHOT SCREEN



Chosen Display: Based on the above drawings, it would appear as though every display can reasonably fit the required information without too much trouble. Of course, the more space there is the clearer the message will be, but in this case, the best option is likely to pick one that is roughly in the middle when choosing between the smallest size possible and the most light projected. Due to all of the mono-color options fitting the information perfectly fine, and in the interest of maintaining the clearest image possible, the multi-color displays will not be considered a priority, however they of course can still be used as just a mono-color display. The varying wavelengths of light and multiple colors may distort the image, and it is not worth the overall risk. The other narrowing down of options is only considering displays that use I2C, as the reduction of wires will be helpful in reducing clutter when compared with 4-wire SPI. Table 3.1 shows the remaining options.

Table 3.1: Summary of Remaining Display Options

Display Name /MFR	Resolution	Overall Size (in)	Weight (oz)	Voltage	Price
Maker Focus	128x32	1.5 x .47 x .36	0.845	3.3 - 5	$4.99
Wave Share	128x128	1.75 x 1.46	0.67	5	$18.99
HiLetgo	128x64	1.06 x 1.06 x .16	NOT LISTED	3.0 - 5	$7.29

Of these options, the best all around choice was the HiLetgo, 128x64 display. The size is the perfect balance between the three options. It allows an in-between option in terms of resolution, while having the best design of the three which minimizes wasted space due to display module peripherals. Finally, the price point was very competitive and this specific display and identical reproductions of it were widely available online from retailers such as Amazon or DigiKey.

3.2.2 MICROCONTROLLERS AND MICROPROCESSORS

Due to the various subsystems that our project consists of, several microcontrollers needed to be implemented. For each component of the system, every microcontroller must fulfill two main responsibilities. First, the microcontroller needed to be able to read from a variety of sensors. Second, the microcontroller needed to be able to wirelessly communicate to both the main processing unit, as well as other microcontrollers in other subsystems. The first responsibility must be natively supported by the controller; the second responsibility could either be natively supported or can be enabled through the use of a separate wireless adapter.

Four main factors must be taken into account when selecting a microcontroller board. The first factor is processing power. The processing power of the microcontroller is important as it determines how fast data can be read, processed, sent, and received. This can be quantified by the number of cores and the clock speed of those cores. The second factor is memory. Each board has three pools of memory, however, only two of them are important: flash and SRAM. Flash memory is used to store the program that will be run, therefore it is important to select a board that has enough flash. SRAM is used to store and manipulate variables. This is arguably the more important of the two pools as SRAM is where all of the sensor data will be stored and manipulated. Not having enough SRAM to hold all of the data would result in the system crashing. The third factor is power consumption. As all of the peripherals in our project will be battery-powered, all components must draw as minimal power as possible. The last factor to be taken into account is cost. It was important to minimize the ratio of cost to performance.

In the following sections, we will discuss the specific requirements each subsystem has when selecting the ideal microcontroller/microprocessor.

HUD Requirements and Responsibilities: One of the many microprocessors included in this project was the one located on the heads-up display that drives all of the associated components. The components that interact with this microprocessor include the display,

the audio system, the camera, and finally for all data communication back to the main system, a wireless communication module. Each of these components has unique requirements both in terms of communication interface as well as processing needs. As such, the first step was to determine whether the best approach is to have one localized processor, or multiple that will receive and transmit data via the Bluetooth module. An important note is that none of the actual processing occurs on the HUD to maintain low power and size requirements that keep the device as lightweight and safe as possible. In general, and like all processors chosen across the project, the main goals were as follows:

- Have the ability to interface with and supply power to two or more peripheral devices
- Has wireless communication capabilities
- Can be battery-powered
- Small and lightweight

Display Processor Requirements: The display requirements were perhaps the simplest of the three peripherals located on the HUD. Most of the processing was done by the display's internal drivers and multiplexing, meaning that just about any processor will provide enough computational power to drive the display. The display chosen was 128x64 pixels, which means requireed approximately 1 KB of RAM to buffer the display. Finally, the chosen communication protocol to drive the display was I2C and the required voltage is a generous 3-5VDC, meaning that the processor must have been compatible with those two specifications.

Audio Processor Requirements: The audio system required a microprocessor capable of supplying I2S audio information. At the time, we were not such if the microcontroller would or would not have an onboard DAC. If an onboard DAC was present, no further action was to be required; however, if a DAC was not present, an external DAC/amplifier board would have needed to been used. The microcontroller did not need to supply power for the DAC board; external power sources were used.

Camera Processor Requirements: The requirements of the camera were not as large as one might expect. The module chosen for this project had a 8 MByte frame buffer, which alleviates much of the burden it might have had on the processor. The communication requirements were one I2C interface for the sensor configuration and one SPI interface for the camera and data stream.

Cue Stick Requirements and Responsibilities: For the microcontroller in charge of the cue stick, not much computing power was required. Its primary responsibility was to read and interpret the data from the accelerometer and gyroscope and wirelessly communicate with the central control unit and the HUD. Its secondary responsibility was to provide and regulate the power to any peripheral attached to the cue stick. The microcontroller also needed to be able to be powered via batteries as a wired connection could affect the player's movement. Similarly, the microcontroller needed to also be light enough so as to not substantially affect the weight of the cue stick.

Summary of Cue Stick Microcontroller Requirements:

- Has the ability to interface with and supply power to two or more peripheral devices
- Has wireless communication capabilities
- Can be battery-powered
- Small and lightweight

Glove Requirements and Responsibilities: For the microcontroller in charge of the glove, not much computing power was required. Its primary responsibility was to read and interpret the data from the sensors such as the ultrasonic sensor and the inertial measurement unit, and wirelessly communicate this data to the central control unit and the HUD. Its secondary responsibility was to provide and regulate the power to any peripheral attached to the glove, such as the circuitry around the ultrasonic sensor and IMU. The microcontroller also needed to be able to be powered via batteries as a wired connection could affect the player's movement. Similarly, the microcontroller needed to be light enough so as to not substantially affect the weight and movement of the player's hand.

Central Control Unit Requirements and Responsibilities: The last system that needed computing power is the central control unit (CCU). As this system performs the majority of the processing in our project and will run our main program, it needed to be much more powerful and required a microprocessor instead of a microcontroller.

The first responsibility of the CCU is to communicate with the table team's system. This is important as we must take their ideal shot data and interpret it in a way that is meaningful to the user. In order to allow for this communication, the microprocessor needed wireless communication capabilities, either natively supported or implemented through an external adapter. Speaking of wireless communications, the second responsibility of the CCU is to receive data from the other subsystems and use it to determine how accurate the player's shot was compared to the ideal shot. After taking the user's shot tendencies into account, the CCU makes recommendations to the user to improve their shot. In order to allow the HUD to be as small as possible, the CCU needed to be in charge of rendering the image viewed by the player and sends it the HUD for it to display. As the CCU will be stationary, it was not necessary that it be powered by batteries and therefore could be plugged into AC power.

Summary of CCU Microprocessor Requirements and Responsibilities:

- Ability to communicate with the other team's system
- Wirelessly communicate with the cue stick, glove, and HUD
- Manage multiple user profiles
- Process shot data and recommend changes to the user
- Render and transmit image displayed on HUD
- Can be plugged into AC power

We will now discuss the various microcontrollers and microprocessors that were available to us.

Arduino: Arduino is an open-source platform that consists of a microcontroller and an IDE. The Arduino is popular with beginners due to its ease of use. The Arduino IDE uses a simplified version of C++ and a standard API known as the 'Arduino language'. Due to the platform's popularity, it has been well documented and code for a majority of use cases has already been created. Also, since the platform is open source, there are several third-party boards available; however, to discuss components for selection, only first-party options will be discussed.

Nano Family: The Nano Family is made up of Arduino's smallest offerings. However, despite their small size, even the smallest of boards can keep up with, and in some cases surpass, the capabilities of their older, larger brothers. The majority of the family has integrated Bluetooth and WiFi radio modules, with the exceptions being the original Arduino Nano and its replacement, the Arduino Nano Every.

1) *Nano Every*

   The Nano Every is Arduino's smallest offering, measuring 45 mm x 18 mm. It is based on the ATMega4809 microcontroller. It has a clock speed of 20 MHz, 48 KB of flash memory, and 6 KB of SRAM. The Nano Every has a total of fourteen digital I/O pins, eight of which can be used as analog inputs. Among those pins, there are also sets of pins to support UART, SPI, and I2C separately. It also has six PWM pins. As mentioned previously, it lacks any native wireless capabilities, therefore, WiFi and Bluetooth would need to be handled by an attached adapter. The Nano Every has an operating voltage of 5 V. Arduino does not state the operating current of the Nano Every, but it can be assumed it has a similar current draw as its predecessor, the Nano, at 19 mA. Therefore, the estimated power consumption is 95 mW. A single Nano Every can be purchased for $10.20, however, they can be purchased in a pack of three for $29.20.

2) *Nano 33 BLE*

   The Nano 33 BLE, like the Nano Every, measures 45 mm x 18 mm. It is based on the nRF52840 microcontroller. It has a clock speed of 64 MHz, 1 MB of flash memory, and 256 KB of SRAM. It has the same pinout as the Every with fourteen digital pins, eight of which can be analog, and six PWM pins. The 33 BLE also features USB, SPI, I2C, I2S, and UART interfaces. As the name suggests, the Nano 33 BLE has a Bluetooth 5 multiprotocol radio that supports Bluetooth, Bluetooth Low Energy, and Zigbee. As a bonus, the board also contains a built-in 9-axis IMU. The operating voltage of the board is 3.3 V. Like the Every, it can be assumed that the current should be the same. This results in an estimated power consumption of 62.7 mW. The Arduino Nano 33 BLE retails for $22.30.

MKR Family: The MKR series of boards are optimized for experimenting with IoT devices. Most of the family come equipped with a radio module that supports WiFi,

Bluetooth, LoRa, Sigfox, and NB-IoT communication. All of the boards are based on the Cortex-M0 32-bit SAMD21 low-power processor. Therefore, each board has a clock speed of 48MHz, 256KB of internal flash memory, and 32 KB of SRAM.

1) *MKR WiFi 1010*

The MKR WiFi 1010 measures 61.5 mm x 25 mm. It features USB, SPI, I2C, I2S, and UART interfaces. It has twenty-two pins that can be used for digital I/O, twelve pins that can be used for PWM, seven pins that can be used as analog inputs, and one pin that can be used as an analog output. The WiFi 1010 has an input voltage of 5 V but operates at 3.3 V. The board can be configured to only draw 30 mA, resulting in an overall power consumption of 99 mW. In addition to USB power, the board has a Li-Po charging circuit that allows it to run on battery power or an external 5 V source. The Arduino MKR WiFi 1010 can normally be obtained for $35.40.

2) *MKR WAN 1310*

The MKR WAN 1310 measures 67.64 mm x 25 mm. In addition to the 256 KB of internal flash, the MKR WiFi also has 2 MB of external flash memory. It features the same interfaces and pinout as the MKR WiFi 1010. It also has the same input and operating voltages. The WAN 1310 can be configured to draw as little as 104 uA, however, during normal use, the current is usually 30mA. This results in an average power consumption of 99 mW. Like the WiFi 1010, it features a Li-Po battery charger. The Arduino MKR WAN 1310 sells for $42.20.

Summary of Arduino Development Boards: The four Arduino boards that have been considered, along with their technical specifications, are compared in Table 3.2.

Table 3.2: Comparison of Arduino Development Boards

Board	**Nano Every**	**Nano 33 BLE**	**MKR WiFi 1010**	**MKR WAN 1310**
Processor	ATMega4809 (20 MHz)	nRF52840 (64 MHz)	SAMD21 Cortex®-M0+ (48 MHz)	SAMD21 Cortex®-M0+ (48 MHz)
SRAM	6 KB	256 KB	32 KB	32 KB
Flash	48 KB	1 MB	256 KB	256 KB
Wired Connectivity	UART, SPI, IPC	USB, SPI, I2C, I2S, UART	USB, SPI, I2C, I2S, UART	USB, SPI, I2C, I2S, UART
Wireless Connectivity	None	Bluetooth 5.0, BLE, Zigbee	802.11 b/g/n, Bluetooth 5.0, BLE	802.11 b/g/n, Bluetooth 5.0, BLE
Size	45 mm x 18 mm	45 mm x 18 mm	61.5 mm x 25 mm	67.64 mm x 25 mm
Power Consumption	95 mW	62.7 mW	99 mW	99 mW
Price	$10.20	$22.30	$35.40	$42.20

ESP32 Family: The ESP32 family consists of MCUs with integrated WiFi and Bluetooth connectivity. It is developed by Espressif Systems and is manufactured by TSMC, however, the platform is open source. The ESP32 family can be programmed using a variety of platforms, the three main ones being VSC with the ESP-IDF extension, the Arduino IDE with the ESP32 Arduino Core, and MicroPython.

ESP32 Series: The original ESP32 series of MCUs feature either one or two cores with an adjustable clock frequency, ranging from 80 MHz to 240 MHz. Each MCU has 520 KB of SRAM and 448 KB of ROM. In terms of wireless connectivity, the ESP32 series supports 802.11 b/g/n, Bluetooth version 4.2, and BLE. They also have thirty-four GPIO pins and interfaces for SPI, I2S, I2C, and UART.

1) *Adafruit ESP32 Feather V2*

   The Adafruit ESP32 Feather V2 is one of Adafruit's most popular ESP32 boards. It features the ESP-PICO-MINI-02 module which contains the dual-core 240 MHz Xtensa processor. It has 520 KB of SRAM, 4 MB flash, and 2 MB PSRAM. The board measures 52.3 mm x 22.8 mm x 7.2 mm. The recommended operating conditions from Espressif state a supply voltage of 3.3 V and a supply current of 0.5 A, resulting in the board consuming 1.65 W. However, current consumption can be lowered by changing between active and sleep modes. The board can be purchased for $19.95.

2) *NodeMCU ESP-32S*

   The NodeMCU ESP-32S uses Ai-Thinker's ESP32-S, which is equivalent to Espressif's ESP-WROOM-32 module. It features two Tensilica LX6 cores and 512KB of SRAM. Like the Adafruit ESP32 Feather V2, it typically consumes 1.65 W but this can be lowered through configurations. The board measures 48.26 mm x 25.4 mm. The boards can be purchased in a pack of three, resulting in each board costing $6.63.

ESP32-S2 Series: The ESP32-S2 series is a variation of the ESP32 optimized for low-power performance. This series comes with a single-core 240 MHz Xtensa LX7 CPU, 128 KB of ROM, 320 KB of SRAM, and 16 KB of RTC SRAM. The S2 series has support for 2.4 GHz WiFi but does not have Bluetooth capabilities; therefore, that functionality would need to be added through the use of an external adapter.

1) *ESP32-S2-Saola-1*

   The ESP32-S2-Saola is an ESP32-S2-based development board from Espressif. This board comes with an ESP32-S2-WROVER module containing 4 MB of flash and 2 MB of PSRAM. It has forty I/O headers and supports SPI, I2S, UART, and I2C. The board measures 53.9 mm x 29.8 mm. Like the other ESP32 modules, the ESP-32-WROVER consumes 1.65W and can be configured to consume less. There are three ways to power: the Micro-USB port (recommended), the 5V header, and the 3V3 header. This development kit can be purchased for $14.50.

2) *Adafruit ESP32-S2 Feather*

Like the ESP32-S2-Saola, the Adafruit ESP32-S2 Feather comes with a 240 MHz Tensilica processor, 4 MB of flash, and 2 MB PSRAM. The ESP32-S2 Feather measures 52.4 mm x 22.8 mm x 7.2 mm. Once again, the board can consume 1.65 W, less if configured properly. The Feather can be powered using either the USB type C port or using a Lipoly battery. This board retails for $17.50.

ESP32-S3 Series: The ESP32-S3 series is the newest version of the ESP32. All modules come with an Xtensa 32-bit LX7 dual-core processor that runs at 240 MHz. For memory, all S3 modules come with 512 KB of SRAM, and 384 KB of ROM. This series also has additional support for vector instructions, accelerating neural network and signal processing workloads. In terms of connectivity, there are forty-five GPIOs, SPI, I2S, I2C, PWM, ADC, and UART. The S3 series has support for 2.4 GHz WiFi and Bluetooth 5 (BLE) but does not support Bluetooth Classic.

1) *ESP32-S3-DevKitM-1-N8*

The ESP32-S3-DevKitM-1 is equipped with the ESP32-S3-MINI-1. It comes with 8 MB of flash memory. Currently, there is limited support for this development kit as there is only support for ESP IDF. Like the ESP32-S2-Saola-1, this board can be powered via USB, the 5V and GND pins, or the 3v3 and GND pins. Like other ESP32s, it consumes 1.65 W and can be configured to consume less. This board measures 62.74 mm x 25.40 mm. This development kit can be purchased for $15.95.

2) *ESP32-S3-DevKitC-1*

The ESP32-S3-DevKitC-1 comes with the ESP32-S3-WROOM-2. It has 32 MB of flash memory and 8 MB of PSRAM. Like the other S3 development boards, there is limited support and only ESP-IDF is supported. It consumes 1.65 W, however, the CPU and radio can be configured to reduce power consumption. Like the ESP32-S3-DevKitM-1-N8, this board can be powered via USB, the 5V and GND pins, or the 3v3 and GND pins. This board measures 62.74 mm x 25.40 mm and is sold for $19.95.

Summary of ESP32 Development Boards: The three ESP32 series that have been considered, along with their technical specifications, are compared in Table 3.3.

Table 3.3: Comparison of ESP32 Families

Board	ESP32	ESP32-S2	ESP32-S3
Processor	Dual-core Xtensa/Tensilica (240 MHz)	Single-core Xtensa/Tensilica (240 MHz)	Dual-core Xtensa/Tensilica (240 MHz)
SRAM	520 KB	320 KB	512 KB
Flash	448 KB - 4 MB	4 MB	8 - 32 MB
Wired Connectivity	SPI, I2C, I2S, UART	USB, SPI, I2C, I2S, UART	USB, SPI, I2C, I2S, UART
Wireless Connectivity	802.11 b/g/n, Bluetooth 4.2, BLE	802.11 b/g/n	802.11 b/g/n, BLE
Size	52.3 mm x 22.8 mm, 48.26 mm x 25.4 mm	53.9 mm x 29.8 mm, 52.4 mm x 22.8 mm	62.74 mm x 25.40 mm
Maximum Power Consumption	1.65 W	1.65 W	1.65 W
Price	$6.63 - $19.95	$14.50 - $17.50	$15.95 - $19.95

Raspberry Pi Family: Raspberry Pi is a series of microcontroller boards developed by the Raspberry Pi Foundation. Raspberry Pis were designed to be small, low-cost computers, making them ideal for both small projects as well as being an everyday computer. All Raspberry Pis have a microSD card slot that allows the user to run one of many OSs available. The official supported OS is Raspberry Pi OS, formerly known as Raspbian. Raspberry Pi OS is based on Debian, one of the more popular Linux distributions

1) *Raspberry Pi Zero 2 W:*

The Raspberry Pi Zero 2 W is Raspberry Pi's budget computing option with an MSRP of just $15. At the heart of the Zero 2 W is the quad-core 64-bit Arm Cortex-A53 CPU which runs at 1 GHz. The Zero 2 W comes with 512 MB of SDRAM. In terms of wireless connectivity, the Zero 2 W has onboard antennas and supports 2.4 GHz 802.11 b/g/n WLAN, Bluetooth 4.2, and BLE. As for wired connectivity, the board has a mini HDMI port, a Micro-USB OTG port, a CSI-2 camera connector, and an unpopulated HAT-compatible 40-pin header footprint. The board runs off of Micro-USB power and is 65 mm x 30 mm.

2) *Raspberry Pi 4 Model B:*

The Raspberry Pi 4 Model B is Raspberry Pi's latest mainline release. The board's processor is a Broadcom BCM2711, a quad-core Cortex-A72 64-bit SoC running at 1.5 GHz. The Pi 4 Model B comes in four different SDRAM configurations: 1 GB, 2 GB, 4 GB, and 8 GB; the SDRAM is LPDDR4-3200. For network connectivity, the board supports 2.4/5 GHz 802.11ac. Bluetooth 5.0, BLE, and gigabit Ethernet. In terms of I/O, there are 2 USB 3.0 ports; 2 USB 2.0 ports, the Raspberry Pi standard 40 pin GPIO header, 2 micro-HDMI ports, a 2-lane MIPI DSI display port, a 2-lane MIPI CSI camera port, 4-pole stereo audio, and a

composite video port. The Raspberry Pi 4 Model B can be powered via USB-C, the GPIO header, or through Power over Ethernet. The board measures 56 mm x 85 mm. The MSRP for this board ranges from $35 for the 1 GB version to $75 for the 8 GB version.

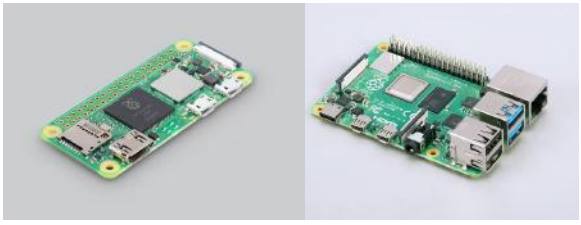


Figure 3.14: Raspberry Pi Zero 2 W (left) and Raspberry Pi 4 Model B (right). Reproduction permission requested from Raspberry Pi.

Summary of Raspberry Pis: The Raspberry Pis that have been considered, along with their technical specifications, are compared below in Table 3.4.

Table 3.4: Comparison of Raspberry Pis

Board	Raspberry Pi Zero 2 W	Raspberry Pi 4 Model B
Processor	Quad-core Arm Cortex-A53 (1 GHz)	Quad-core Broadcom BCM2711 (1.5 GHz)
RAM	512 MB SDRAM	1 - 8 GB LPDDR4
Wired Connectivity	mini-HDMI, Micro-USB OTG, CSI-2, 40-pin header	USB, SPI, I2C, I2S, UART, Ethernet
Wireless Connectivity	802.11 b/g/n, Bluetooth 4.2, BLE	802.11ac. Bluetooth 5.0, BLE
Dimensions	65 mm x 30 mm	56 mm x 85 mm
Price	MSRP: $15 Market: $90	MSRP: $35 - $75 Market: $114.99 - $156.99

Raspberry Pi Alternatives: In the case that we were not able to procure a Raspberry Pi for a reasonable price, it was important that we had some alternative options at our disposal. The Raspberry Pi alternatives that we considered, along with their technical specifications, are compared in Table 3.5.

Table 3.5: Comparison of Raspberry Pi Alternatives

Board	Rock Pi 4 Model C	Pine RockPro64	Odroid-C4
Processor	Hexa-core Rockchip RK3399 (1.8 GHz)	Quad-core Corex-A53 (1.4 GHz)	Quad-core Amlogic S905X3 (1.91 GHz)
RAM	4 GB LPDDR4-3200	4 GB LPDDR4	4 GB LPDDR4
Wired Connectivity	USB, UART, SPI, I2C, I2S, SPDIF, Ethernet	USB, PCIe 4x	USB, UART, SPI, I2C, Ethernet
Wireless Connectivity	802.11ac, Bluetooth 5.0	802.11ac, Bluetooth 5.0	N/A
Dimensions	85 mm x 54 mm	133 mm x 80 mm	85 mm x 56 mm
Price	MSRP: $59 Market: $107.11-$135	MSRP: $79.99	MSRP: $54.00

HUD Controller Selection: The HUD required a microcontroller that is versatile and lightweight. The microprocessor needs to have Bluetooth and WiFi capabilities to transmit and receive data to and from the Central Control Unit (CCU). Versatility was needed as the microcontrollers drive both visual and auditory systems. Lastly, the lightweight requirement was there since the microcontroller will be positioned on the HUD, a wearable. Due to all these requirements, the ESP32 was selected. This also maintained uniformity with other system components.

Cue Stick Microcontroller Selection: While the majority of the boards that have been discussed met our requirements for the cue stick, we decided that we would use a microcontroller board from the ESP32 family. While the Arduino family does have a significant amount of support, the majority of the ESP32 family could be programmed using the Arduino IDE, giving the ESP32 family the same functionality as an Arduino. Also, most ESP32 boards are significantly cheaper than Arduinos with the same functionality.

As for which series of the ESP32, the original ESP32 series, despite being the oldest, was our best option. As was mentioned in the opening paragraph on the ESP32 family, this series could be programmed using a variety of platforms. While the ESP32-S3 series is the newest iteration of the ESP32 and would probably be slightly more power-efficient, its “newest” ended up being its downfall as it had the least support and could only be programmed using the ESP-IDF extension in VSC. As for the ESP32-S2 series, the need to add Bluetooth functionality through the use of an external adapter immediately eliminated it from being a viable option as it would add an additional point of failure in the system.

Moving on to which particular board we are using for the cue stick, the NodeMCU ESP-32S was our best option due to it having the same functionality as the Adafruit ESP32 Feather V2 while costing significantly less. As a result, the final prototype of our systems will use ESP-WROOM-32 modules or their third-party equivalents.

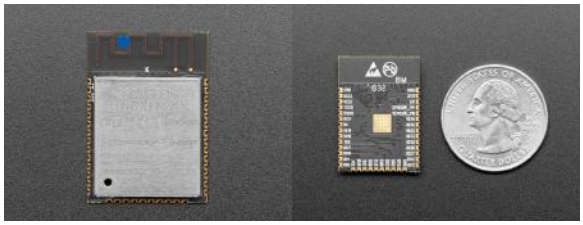


Figure 3.15: ESP32 WiFi-BT-BLE MCU Module / ESP-WROOM-32. Reproduction permission requested from Adafruit.

As was mentioned in the section discussing the ESP32, we had multiple methods for programming the boards: ESP-IDF, ESP32 Arduino Core, and MicroPython. In a later section, we will look at the pros and cons of each language and decide which to use moving forward.

Glove Controller Selection: The Glove required a microcontroller that is versatile and lightweight. The microprocessor needed to have Bluetooth and WiFi capabilities to transmit and receive data to and from the Central Control Unit (CCU). The lightweight requirement was there since the microcontroller will be positioned inside the glove. Due to all these requirements, the ESP32 was selected. This also maintained uniformity with other system components.

CCU Microprocessor Selection: Selecting from a microprocessor for the central processing unit may have been the simplest component to choose. We concluded that the Raspberry Pi Zero 2 W, while small and power-efficient, did not have enough computing power to suit our needs. The main disadvantage to using the Zero 2 W is its small amount of SDRAM. Currently, we are not completely sure how much RAM our application would need but ideally we would have as much available to us as possible.

For this reason, we have decided to select the Raspberry Pi 4 Model B as the microprocessor board for the CCU. At the most expensive configuration, we would have 8 GB of LPDDR4-3200; this should be more than enough to run our application. The large amount of RAM paired with the quad-core processor should allow for the CCU to seamlessly communicate with both our own subsystems, as well as the table team's.

Due to the current chip shortage, the prices for the Raspberry Pi 4 Model B have risen dramatically. The MSRP for the 8 GB variant is $75, however, on the majority of marketplaces such as Amazon, the board is either out of stock or being sold for more than $150. Therefore it is important for us to have a backup option in the event of our team not being able to procure a board. Among the alternatives that were compared, our next best option would be the Pine RockPro64. While it has less RAM, 4 GB instead of 8 GB, this should be enough to run our central application.

Analog-to-Digital Converters: When using the ESP32, we had an analog input from 0 to 3.3 V where a set voltage is equal to 1-bit all the way up to 4095-bits but unfortunately for our case the ADC in the ESP32 was not entirely linear so we would lose some

accuracy if we were to use the full ADC. But due to this input range we needed to use amplifiers to have a better range for the values in order to maximize our resolution.

Input Methods: There are three types of common input methods for sending analog signals to ADCs, single ended, fully differential, and pseudo-differential. A single-ended input compares all input signals to a common node, usually the analog ground. This is sufficient for most applications. A fully differential input takes both a positive and negative terminal for each input. This provides good 'common mode' noise rejection—any noise signal present on both the positive and negative inputs is canceled out when the difference between inputs is taken. A pseudo-differential input samples from the noninverting input with respect to its common ground but switches the input connection to the inverting input during the hold stage, effectively using it as a reference.

3.2.3 SENSORS

Ultrasonic Sensor: One way to locate the ball would be to use UltraSonic Sensors [3.6]. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. For this case we could use the ultrasonic sensor because the ball has a hard surface it would be easily detectable by the ultrasonic sensor giving us a reading of distance. Some issues that this may pose is the size of the ultrasonic sensor, the majority of these sensors are multiple centimeters in length, width and height. Another issue posed is that the user would already have to be aligned with the ball hand-wise in order to have this work since this would not be able to differentiate between balls.

1) *HC-SR04*

This is the most common ultrasonic sensor on the market. It will offer some decent specs. This can provide us with accuracy from 2-400 cm from the ultrasonic transmitter, receiver and control circuit. This would need to be powered by a 5 V input with 0 V GND; it has a working current of 15 mA and a working frequency of 40 Hz. One issue posed is these tend to work best and most accurately on flat surfaces which we would not be using it on.

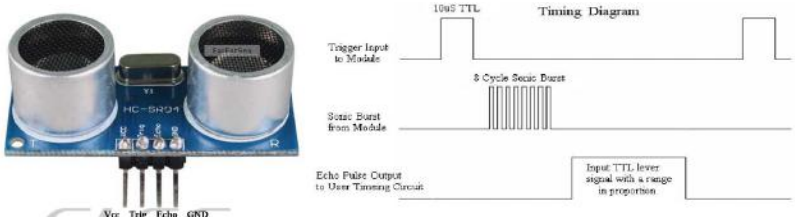


Figure 3.16: HC-SR04. Reproduction permission requested from Elec Freaks.

2) *RCWL-1601*

The RCWL ultrasonic sensor is very similar to the HC-SR04. The only main difference is that it has a 3.3V input which would be ideal since this is regulated output of the ESP32. This would simplify our system, and make it so we do not require external circuitry.

3) *Ping Ultrasonic Sensor*

The Ping ultrasonic sensor is very similar to the HC-SR04. The only main difference is that it has a better range and it is only a 3-pin configuration which is Vdd,Vss, and I/O Signal. This would simplify our system, but due to cost we believe the better choice would be to go with the HC-SR04. Figure 3.17: Ping Ultrasonic Sensor. Reproduction permission obtained from Parallax

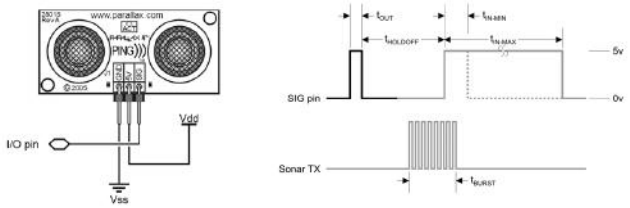


Figure 3.17: Ping Ultrasonic Sensor. Reproduction permission obtained from Parallax.

Table 3.6: Ultrasonic Sensor Selection

Chip	RCWL-1601	HC-SR04	Ping Ultrasonic
Size	40x18mm	45x20 cm	46x22cm
Range	2cm-450cm	2cm-400cm	3cm-300cm
Working Voltage	3V-5.5V	5V	5V
Price	$4.00	$3.50	$35.00

We selected the RCWL-1601 due to its input voltage level being 3V-5.5V which will allow us to use the power off of the ESP32, this also has a superior range compared to the other two ultrasonic sensors. The price is also significantly better than the ping ultrasonic sensor and comparable to the HC-SR04 sensor.

Hall Sensor: For our glove we are going to need to be able to detect where the cue ball is on the billiards table for this to be compatible with assisting vision impaired individuals[3.7]. For this we are going to use hall sensors to be able to locate the ball. For an automatic billiards table they use a ball return system so instead of having to get the balls out of the pocket each time, this is mainly used in places where you would pay to play a game due to the ability to hold the balls underneath the table and make the customer pay before playing a game, but in this design they make the cue ball have a

magnet inside it so that it can differentiate from the numbered balls. This is much more efficient than using light sensors to detect the reflected light from the ball.

Hall sensors are a really good way to determine a magnetic field by using this magnetic field to cause a voltage difference that can be read. Hall sensors are used in many applications in all fields of the industry such as automotive where they are used to detect whether a seatbelt is fastened or in manufacturing for detecting position of objects. In addition to these, automatic billiard tables use hall sensors to detect the magnetic field from the cue ball so that it can differentiate them, in our case we just need to identify when the hand of the user is directly in front of the cue ball once they are placed in the correct place. We are able to use the hall sensor to measure the strength of the magnetic field and using that we can determine where the cue ball is located in relation to the user.

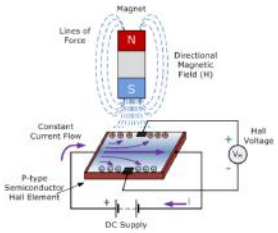


Figure 3.18: Hall Sensor Physics. Reproduction permission obtained from Electronics Tutorials.

The basic principle behind using a hall sensor is that it is a thin p-type semiconductor such as gallium arsenide (GaAs) and when the device is placed within a magnetic field, the magnetic flux lines exert a force on the semiconductor material which deflects the charge carriers, electrons and holes, to either side of the semiconductor slab. As these electrons and holes move sidewards a potential difference is produced between the two sides of the semiconductor material by the build-up of these charge carriers. This principle is called the hall effect.

$$V = Re \times (\frac{I}{t} \times B)$$

That is used to determine the magnetic flux which is the magnetic field through a specified area. The larger the magnetic flux going through the sensor the larger the magnetic field, this will be put to good use in our case for having the user locate the ball. The output of our sensor will just be a voltage which we can send to our bluetooth module and eventually send to our microcontroller to determine whether the user needs to adjust their hand.

Possible Choices:

1) *KY-003*

   We can simply integrate this sensor into our glove by using the KY-003, it has an operating voltage of 4.5V-24V, operating temperature of -40 C - 85 C, as well as it has a footprint of 18.5mm x 15mm which would definitely fit on the glove. But

this only has a pin for the digital output which wouldn't be able to get the specification that we would want to determine the strength of the magnetic field so that we can determine proximity to the ball. With only the digital output we could only tell them whether there is a magnetic field that is strong enough to detect.

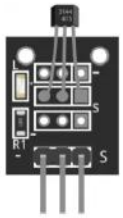


Figure 3.19: KY-003. Reproduction permission requested from Arduino Modules.

2) *KY-026*

The KY-026 will also perform the same functions but this adds in the analog pin which will allow us to read the value and then send this via bluetooth module to the microcontroller so that we can tell the user how far away they should be from the ball. This will allow for much more precision when placing the hand for the shot.

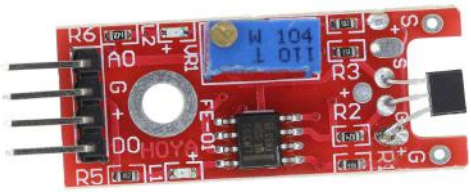


Figure 3.20: KY-026. Reproduction permission requested from Arduino Modules.

Even though both these hall sensors offer great options, our project used the RCWL-1601 ultrasonic sensor in order to detect the ball due to its great range and precision as well as its simple integration into the glove subsystem compared to having to integrate a magnet into the cue ball to achieve the functionality of the hall sensor.

Flex Sensor: Even when we get the vision impaired person's hand in front of the ball we see the issue that they might not even be at the correct height to hit the ball. When analyzing where an ideal shot would be we get many different options depending on what the circumstances are. When a non-vision impaired person is using the table they would not need the glove because their shot directions would appear on the display next to the table showing where on the ball they should hit. But for the vision impaired person this becomes a much harder task, but in this case we would eliminate shots that would involve spin, so we would focus on hitting the ball as centered as possible, this would allow for the most consistent results, also for the vision impaired shots they will be more direct on for proof of concept so we do not have to worry about changing left and right position as well as height.

Overview: We need to determine the height of the hand when in the position to take a shot, this is necessary because in order to have an accurate shot you must have the stick hit the correct spot on the ball. Based on the required best shot we would be able to determine how the user would have to hit the ball. To obtain the height that the user's hand is at we can use a flex sensor along the glove.

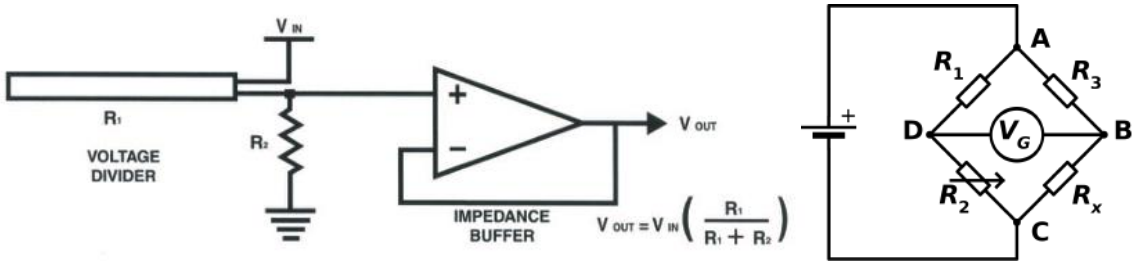


Figure 3.21: Schematic of Flex Sensor Implementation

The principle of flex sensors is that when it is straight it has a set resistance but when you flex it or apply strain the resistance would increase, so we could put this inside the glove and as the user would flex their hand to set up a correct shot it would increase the resistance, so we could put this to use by creating a simple circuit such as a voltage divider or a wheatstone bridge that would allow us to measure the change in voltage and therefore assign a value to how much it flexes. The issue with this is that it is not directly measuring the height of the hand but instead the resistance which we would then have to correlate back to the height. A way that we could make this work would be by doing continuous testing to determine what values would pertain to what height and since within our range of testing the flex sensor would have a linear trend with resistance we could approximate our height relatively accurately.

Options:

1) *SEN-08606*

   This is the linear flex sensor pictured above which we could use. This has a life cycle of > 1 million which means it will not decay or change performance in the time we are using it. It is 4.419 inches long which will fit on our glove. When flat it has a resistance of 10 kΩ and the bend resistance change is 60 kΩ to 110 kΩ. This particular unit is used in other glove equipment where they wanted to measure whether the glove was flexed or not. We could very easily set this up with a circuit that will allow it to send an analog value back to the microcontroller which would then let us know what the current height is and compare this to the desired height of the shot.

2) *SparkFun Qwiic Flex Glove Controller (SEN-14666)*

   This flex sensor has an ADS1015 which is a 12-bit ADC-to-I2C chip; we would be able to get multiple analog inputs without needing to use the microcontrollers ADC pins. Another plus to this sensor is the fact that it is meant for a glove. Some

of the features are operating voltage: 2.0 V - 5.5 V, an operating temperature: -40°C - 125°C, a resolution of 12 bits, a sample rate: 128 Hz - 3.3 kHz, a current consumption: 150 µA (Typ.), and finally an I2C address: 0x48 (default), 0x49, 0x4A, 0x4B. We compared this to just using a flex sensor it would be much better on its own but since we would already need the microcontroller to transmit a Bluetooth signal we would most likely not need all the features that this device offers.

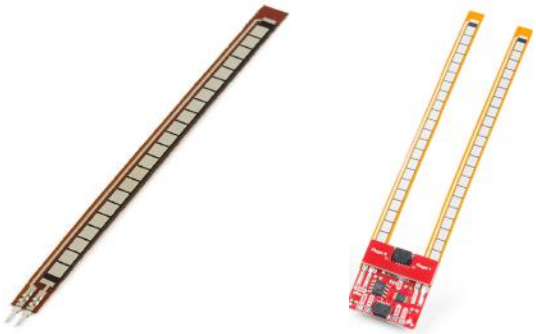


Figure 3.22: SEN-08606 (left) and SEN-14666 (right). Reproduction permission requested from SparkFun.

Due to simplification of the design for the visually impaired use, we used a fixed point of contact at the center of the ball in order to make the shot. This was achieved by creating a mechanical housing for the components that has a indent for the placement of the cuestick that keeps it level with the center of the ball.

Accelerometer And Gyroscope Options: The accelerometer and gyroscope were required for this project to track the cue stick's orientation and speed when striking the cue ball. The data from these sensors is sent to the central computer where the path and speed of the pool cue are then compared to the ideal “shot” provided by the table team’s AI algorithm. The system then provides the user with feedback as to how accurate their attempt was to the ideal shot. After multiple shots and sessions, it is expected that the user’s billiard stroke would become more consistent and accurate, specifically their path of motion and speed control. The following section discusses the aspects taken into consideration when developing this sensor system.

Placing these sensors on the cue stick itself results in several considerations that must be taken into account. First, the sensors need to be as small and lightweight as possible. Violating this principle could result in the user’s shot being altered, making any corrections to their shot meaningless as our training system would no longer be accurate to a standard billiards cue. Another factor that must be addressed is the power consumed by the sensor. As all of the components mounted to the pool stick will be powered by the same battery system, it is critical that each component draws as little power as possible.

Independent Accelerometers:

1) *ADXL343*

The ADXL343 is a triple-axis accelerometer manufactured by Analog Devices. This sensor can use either I2C or SPI to communicate with an MCU. It has a wide sensitivity range, allowing for sensitivity levels of ±2g, ±4g, ±8g, or ±16g. This sensor has a configurable resolution, operating with either a 10 or 13-bit internal ADC. Additional features include built-in motion detection and two interrupt pins. The ADXL343 is 26.0mm x 17.8mm x 4.6mm in size. The operating voltage ranges from 2.0 to 3.6V, with a typical value of 2.5V. The supply current is 140μA. This results in a typical power draw of 0.35mW. This breakout board typically retails for $5.95.

2) *MMA8451*

The MMA8451 is an accelerometer from Freescale. The breakout board itself is manufactured by Adafruit. This sensor communicates over I2C and has an address selection pin that allows a system to have multiple sensors on the same bus. This sensor has three sensitivity levels, ±2g, ±4g, and ±8g, and comes equipped with a 14-bit ADC. The accelerometer also has built-in tilt and orientation detection. The MMA8451 is 21.0mm x 18.0mm x 2.0mm. The operating voltage ranges from 1.95 to 2.6V, with a typical value of 2.5V. The supply current is 165μA. This results in the sensor having a typical power draw of 0.41mW. This sensor retails for $7.95.

3) *LIS3DH*

The LIS3DH is a low-power accelerometer made by STMicroelectronics. The breakout board itself is produced by Adafruit. This board can use either I2C or SPI and includes an address selection pin. It has the same sensitivity options as the ADXL343 and has a 10-bit ADC. The dimensions of the LIS3DH are 20.6mm x 20.3mm x 2.6mm. The supply voltage ranges from 1.71 to 3.6V, with a typical voltage of 2.5V. The current consumption in normal mode is 11μA. This results in a normal power consumption of 0.3mW. This breakout board retails for $4.95.

Independent Gyroscopes:

1) *L3GD20H*

The L3GD20H is a three-axis gyroscope manufactured by STMicroelectronics. The breakout board itself is provided by Adafruit. This sensor supports both I2C and SPI for communication. The chip has three sensitivity ranges, ±250, ±500, and ±2000°/sec. This breakout board is 30.65mm x 19.11mm x 3mm. The supply voltage ranges from 2.2 to 3.6V; the typical voltage is 3.0V. The supply current is 5.0mA. Therefore, this sensor is 45mW. This gyroscope sells for $12.50.

Inertial Measurement Units (IMUs):

1) *MPU-6050*

The MPU-6050 is a six-axis MotionTracking device manufactured by Adafruit. The sensors themselves are made by Invensense. The MPU-6050 features 16-bit ADCs. The gyroscope has four scale ranges, ±250, ±500, ±1000, and ±2000°/sec. The accelerometer also has four scale ranges, ±2g, ±4g, ±8g, or ±16g. The dimensions of the board are 26.0mm x 17.8mm x 4.6mm. The operating voltage of the board ranges from 2.375 to 3.46V. The normal operating current is 3.9mA, resulting in an average power consumption of 11.3mW. This IMU normally retails for $6.95.

2) *BNO055*

The BNO055 is an IMU made by Adafruit that uses sensors from Bosch. It contains a MEMS accelerometer, magnetometer, and gyroscope, as well as an ARM Cortex-M0-based processor. This 9-DOF sensor is special as it uses "sensor fusion algorithms" to blend the sensor data into a stable three-axis orientation output. The accelerometer has four sensitivity ranges (2g, ±4g, ±8g, or ±16g) and the gyroscope has five sensitivity ranges (±125, ±250, ±500, ±1000, and ±2000°/sec). The dimensions of the IMU are 20.0mm x 27.0mm x 4.0mm. The supply voltage ranges from 2.4 to 3.6V; the total supply current is 12.3mA. This results in a typical power draw of 36.9mW. This board retails for $34.95.

3) *LSM6DSOX*

The LSM6DSOX is an IMU sensor made by Adafruit whose sensor comes from STMicroelectronics. The board itself uses either I2C or SPI to communicate and has two interrupt pins. The accelerometer has four output ranges, 2g, ±4g, ±8g, or ±16g, and the gyroscope has five sensitivity ranges, ±125, ±250, ±500, ±1000, and ±2000°/sec. The dimensions of this sensor are 25.6mm x 17.8mm x 4.6mm. The supply voltage of this sensor ranges from 1.71 to 3.6V, with a typical voltage of 1.8V. The current consumption of this device is 0.55mA, resulting in a power draw of 0.99mW. This IMU is sold for $11.95.

Summary of Accelerometer and Gyroscope Options: The accelerometers and gyroscope that have been considered, along with their technical specifications, are compared in Table 3.7.

Table 3.7: Summary of Accelerometer and Gyroscope Options

Board	ADXL343	MMA8451	LIS3DH	L3GD20H
Type	Accelerometer			Gyroscope
Accelerometer Range	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g	±2 g, ±4 g, ±8 g, ±16 g	N/A
Gyroscope Range	N/A	N/A	N/A	±250, ±500, ±2000°/sec
Connectivity	I2C/SPI	I2C/SPI	I2C/SPI	I2C/SPI
Dimensions	26.0 mm x 17.8 mm x 4.6 mm	21.0 mm x 18.0 mm x 2.0 mm	20.6 mm x 20.3 mm x 2.6 mm	30.65 mm x 19.11 mm x 3 mm
Power Consumption	0.35 mW	0.41 mW	0.3 mW	45 mW
Price	$5.95	$7.95	$4.95	$12.50

Summary of Inertial Measurement Unit Options: The IMUs that have been considered, along with their technical specifications, are compared in Table 3.8.

Table 3.8: Summary of IMU Options

Board	MPU-6050	BNO055	LSM6DSOX
Accelerometer Range	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g, ±16 g	±2 g, ±4 g, ±8 g, ±16 g
Gyroscope Range	±250, ±500, ±1000, ±2000°/sec	±250, ±500, ±1000, ±2000°/sec	±250, ±500, ±1000, ±2000°/sec
Connectivity	I2C/SPI	I2C/SPI	I2C/SPI
Dimensions	26.0 mm x 17.8 mm x 4.6 mm	20.0 mm x 27.0 mm x 4.0 mm	25.6 mm x 17.8 mm x 4.6 mm
Power Consumption	11.3 mW	36.9 mW	0.99 mW
Price	$6.95	$34.95	$11.95

Accelerometer, Gyroscope, and IMU Selection: After viewing the various accelerometers, gyroscopes, and IMUs that could be used to track the motion of the cue stick, we decided that using an IMU to do so would be the optimal method. Using an independent accelerometer and gyroscope would require our microcontroller to power and interface with two separate devices instead of one. Doing so would add complexity as connecting

an extra device would require at least another four wires. It would also put an extra load on the microcontroller as it would need to constantly switch between the two devices, increasing latency and reducing battery life. As for which IMU to use, we decided to use the Adafruit BNO055 due to the additional processing power that is integrated in the sensor. This sensor is used on both the cue stick and the glove to implement functionality.

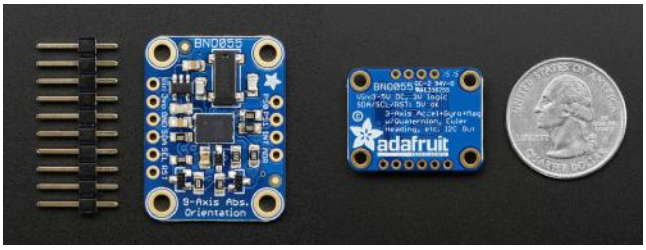


Figure 3.23: Adafruit BNO055. Reproduction permission requested from Adafruit.

3-Axis Magnetometer: In order for a visually-impaired person to hit the cue ball correctly to make a shot, they would need to be angled correctly one possible way that this could be done would be to use a compass and map where they would need to be and what angle with respect to north they would need to face towards. This would be able to reliably face our user towards the ball and then we can use this and the other features mentioned above to accurately hit the ball. One major issue that comes with this is the fact that if we want to use a hall sensor to detect the ball with a magnetic core that would impede on the effectiveness of the digital compass. So in order for this to be used to effectively we would need to have this either not placed directly on the glove or we would have to set up a system where we would first check angle then once the user is in the correct position the digital compass readings would be shut off and then we would use the hall sensor to read where they are in relation to the cue ball.

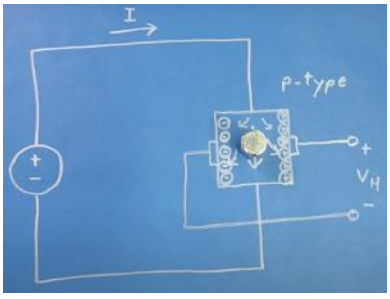


Figure 3.24: Drawing of Magnetometer Functionality.

This system works very similarly to a hall sensor in the way that it detects magnetic fields. As shown in the picture above this is done by applying a source to a p-type semiconductor and then a magnetic field will polarize the charge carriers to each side which will then get you a voltage that you can measure and then based off of that you could get a larger or smaller voltage by orientation therefore giving the principle mechanic of a compass.

1) *BMM150*

This is a 3-axis Geomagnetic sensor that can be used to determine the angle of the user in relation to north. Performance and features of BMM150 are carefully tuned and perfectly match the demanding requirements of all 3-axis mobile applications such as electronic compass, navigation or augmented reality. An evaluation circuitry (ASIC) converts the output of the geomagnetic sensor to digital results which can be read out over the industry standard digital interfaces (SPI and I2C). As the sensor features an ultra-small footprint and a flat package, it is ingeniously suited for mobile applications. The wafer level chip scale package (WLCSP) with dimensions of only 1.56 mm x 1.56 mm x 0.6 mm ensures high flexibility in PCB placement. VDD voltage range from 1.62V to 3.6V, VDDIO voltage range 1.2V to 3.6V) and can be programmed to optimize functionality, performance and power consumption in customer specific applications. The programmable interrupt engine gives design flexibility to the developer. The BMM150 senses the three axes of the terrestrial field in cell phones, handhelds, computer peripherals, man-machine interfaces, virtual reality features and game controllers. Another application that could be useful to put the user in the correct location is that this device is able to count steps so we would know how many steps the user has moved.

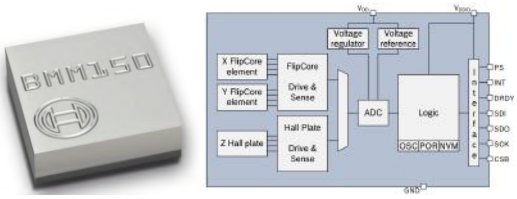


Figure 3.25: BMM150. Reproduction permission requested from Bosh.

2) MAG3110

The MAG3110 is another option for a digital compass, it has very similar functionalities as the BMM150. The MAG3110 is capable of measuring magnetic fields with an output data rate (ODR) up to 80 Hz; these output data rates correspond to sample intervals from 12.5 ms to several seconds. It has a 1.95 V to 3.6 V supply voltage (VDD), 1.62 V to VDD IO voltage (VDDIO). Ultra small 2 mm x 2 mm x 0.85 mm, 0.4 mm pitch, 10-pin package this will be good for a small PCB design. Full-scale range ±1000 T with a sensitivity of 0.10 T. The device will be able to give virtually live feed back with Output Data Rates (ODR) up to 80 Hz, as well as 400 kHz Fast Mode compatible I2C interface. This has slightly less functions as the BMM150 but has all the necessary applications needed for our functionality. This device has been used in this application many times and has many resources including articles as well as video examples for obtaining this functionality.

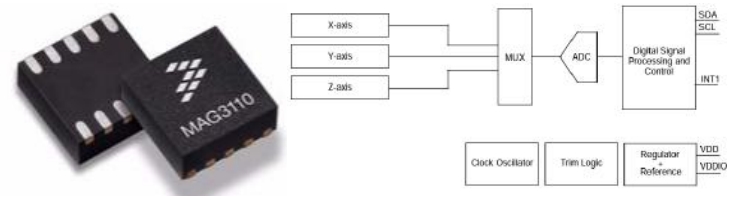


Figure 3.26: MAG3110. Reproduction permission requested from Freescale.

3) HMC5883L

The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low cost compassing and magnetometry. The HMC5883L includes a state of the art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I2C serial bus allows for easy interfacing. The HMC5883L is a 3.0 mm x 3.0 mm x 0.9 mm surface mount 16-pin leadless chip carrier (LCC). Applications for the HMC5883L include Mobile Phones, Netbooks, Consumer Electronics, Auto Navigation Systems, and Personal Navigation Devices. This chip also allows for a built-in self test. But the main draw to this chip is the 12 bit ADC which would free up the ADC on the ESP32 which would be the likely option for transmitting this output.

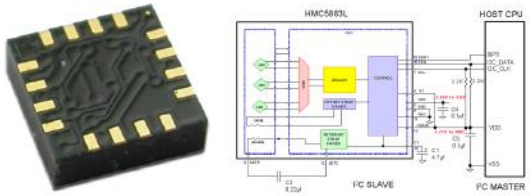


Figure 3.27: HMC5883L. Reproduction permission requested from Honeywell.

Table 3.9: Summary of Digital Compass Options

Chip	BMM150	MAG3110	HMC5883L
Communication	SPI/I2C	I2C	I2C
Magnetic Field Range	±1300µT (x, y-axis), ±2500µT (z-axis)	Full-scale range ±1000u T	2 milli-gauss Field Resolution in ±8 Gauss Fields
Connectivity	I2C/SPI	I2C/SPI	I2C/SPI
Dimensions	1.56 x 1.56 x 0.6 mm3	1.56 x 1.56 x 0.6 mm3	3.0 x 3.0 x 0.9 mm3
ADC	None	None	12-Bit
Price	$8.90	$11.98	$6.59

Angle Detection Sensor Selection: Due to testing of these components in the lab during senior design 1 we decided that using the BNO055 sensor would be significantly better at finding the angle of the glove due to the ability to use the accelerometer and gyroscope to determine the yaw which if we keep the sensor parallel to the table this will match our desired angle.

Metal Detector Circuitry: Since one of the possibilities of cue balls is a metal core we could use a device that can detect metal objects in front of the user's hand and therefore could detect the cue ball.

1) *TIDA0161*

These monolithic integrated circuits are designed for metallic body detection by sensing variations in high frequency Eddy current losses. Using an externally-tuned circuit, they act as oscillators. The output signal level is altered by an approaching metallic object. The output signal is determined by supply current changes. Independent of supply voltage, this current is high or low, according to the presence or absence of a closely located metallic object. This device has an oscillator frequency of 10MHz and a supply voltage of 4V to 35V. A large part of this would be the connection circuit which actually allows for the detection. The TDA0161 acts as an oscillator with the help of this externally tuned circuit. When the LC circuit that is L1 and C1 has got any resonating frequency from any metal which is near to it, electric field will be created which will lead to induces current in the coil and changes in the signal flow through the coil. The metal object has to be at a distance of 10mm for the detector to detect it. This also poses the issue of just being on or off, we would not be able to detect how close to the system, also 10mm might not be close enough to the core of the ball. A small concern of this circuit as well would be the size of the inductor needed. All of this would need to fit on a glove and most inductors would be

rather large thus making the glove more bulky and less functional for shooting a pool ball. There are also other designs posted online using this device with a lot of copper looped in order to detect at a further range.

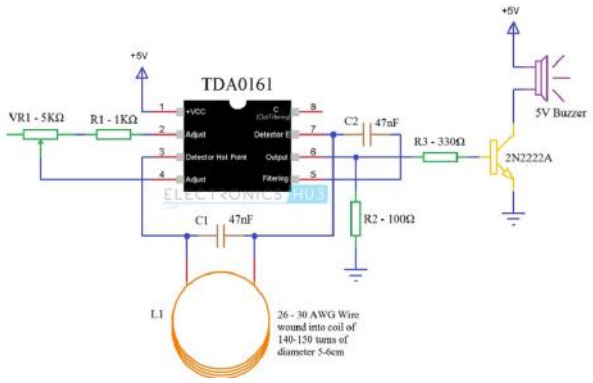


Figure 3.28: TIDA0161. Reproduction permission requested from ElectronicsHub.

2) Ximimark Metal Detector

This is a metal detector kit that is made to help with soldering practice for building the components. This detector is stated to have a range of 60mm which would be reasonable to use for our case of detecting the metal inside the pool ball. It has a working voltage of 3V to 5V DC, this device also comes in a very small package relative to other metal detectors. The main downside associated with this is it seems to have unreliable detection due to it having a very small coil of wires, this is due to its limitations of being on a PCB and using the traces of the PCB and these can not be overlapped. But if this could be made reliable then it would be a very small way to reliably check if the user's hand is in front of the ball, but since this and other metal detectors would generate a magnetic field this may affect the digital compass that would be placed on the glove to detect the angle of the visually impaired user.

Figure 3.29: Ximimark Metal Detector. Reproduction permission requested from Ximimark.

Selection of the Distance Sensor: We selected the RCWL-1601 due to its input voltage level being 3V-5.5V which will allow us to use the power off of the ESP32, this also has a superior range compared to the other two ultrasonic sensors. But also we chose ultrasonic sensor because it had the best ability to track if something is infront of it as

well as it gives us the ability to be precise about our distance we stop infront of the cue ball.

3.2.4 CAMERA

A camera is required for this project in order to detect the actual impact location between the cue stick and the cue ball. The purpose of this feature is to provide valuable feedback to the user, via the display or audio, about his accuracy in matching the impact location provided by the table team's AI algorithm. It is expected that the user, upon receiving enough comparisons over multiple shots, will be able to make changes to his play style to further match the impact location calculated by the table team's AI. This will eventually lead the user to be a better player and make more shots. In order to provide this information, a laser is inserted in the cue stick, and the camera will detect the point of impact based on the location of the laser's reflection on the cue ball. Utilizing a laser facilitates image detection and reduces the amount of processing that needs to be done. This section focuses more on the camera aspect of this feedback feature.

As this is the user side of the project, most of our components will be placed on the user. Based on that, the best location for the camera is on the wearable HUD. This allows the camera to be in an elevated position and to also be directly behind the cue ball, which is optimal for impact location detection. It is true that the camera location could be enhanced by increasing its elevation and reducing distance between it and the cue ball; however, this results in practical issues that cannot be addressed. Thus, we have decided to place the camera on the HUD.

Placing the camera on the HUD results in multiple challenges. First, the camera has to be small and lightweight in order to maintain the desired size and weight of the HUD. If breached, these can then cause practical issues and make the HUD inconvenient for the user. Second, the camera has to be low power, as the HUD Power System must be limited. This is to reduce heat generation and increase both the safety and convenience of the HUD. While the quality of the camera is relatively important, this will not be an issue due to two reasons: first, modern cameras are very powerful and provide more than enough resolution. Secondly, the laser reflection should facilitate the location of the point of impact. With this, we will now present a few camera models that we have found suitable and provide an analysis on each. Images are included for size comparisons.

Physics: All cameras have an image sensor. This is the component required to convert visual or optical input into a digital format that is understandable by CPUs and electronics. Traditionally, image sensors used CCD technology. Most modern cameras however employ CMOS image sensing. The following is a summary of both CCD and CMOS image sensing.

CCD Image Sensing: CCD stands for Charged Coupled Device. The sensor is comprised of a grid of pixels, each with a device that produces an electric charge that is proportional to the light intensity on the device. The basic principle is to design devices that can detect this charge and thus assign a light intensity value based on the amount of charge detected to that particular pixel. This usually utilizes capacitors, followed by amplifiers and then

Analogue to Digital Converters (ADCs). When done on an array with millions of pixels, an image is formed in digital format. Charge generation is based on the photoelectric effect.

For each pixel, a silicon device with a photodiode is present. As light incidents on the diode, negative charge is generated. Within the CCD, there are well and barrier regions. Note that these are not physical wells and barriers; rather, these are regions with different applied voltages that affect the motion of electrons. A positive potential is applied to areas designated as wells. This repels holes and attracts electrons. The opposite is true for barriers: a negative potential is applied to attract holes and repel electrons. By applying specific clock signals to the CCD, wells and barriers can be generated sequentially to allow charge transport via drift.

From a top level perspective, the charges move from the diode horizontally to a vertical analogue shift register. From there, they are moved vertically down to another horizontal shift register, where then they go to an amplifier. Charge transport is done through the method outlined above.

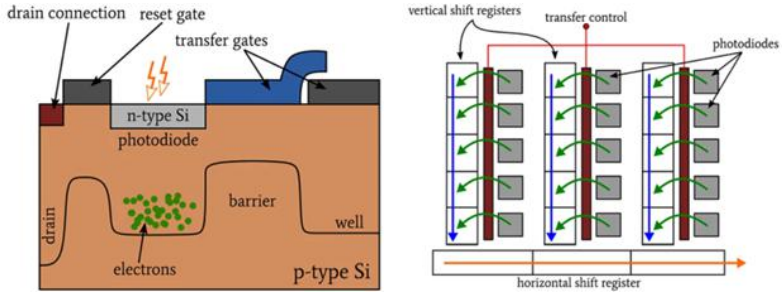


Figure 3.30: On the left, a silicon device with a photodiode representing the structure of an individual pixel. On the right, network of shift registers to move pixel data to the amplifier. Reproduction permission requested from AllAboutCircuits.com.

CMOS Image Sensing: CMOS sensors use rows of photodiodes coupled with amplifiers to amplify the outputs of the photodiodes. One advantage of CMOS sensing is that their manufacturing is rather uncomplicated, and the usual processes employed in the fabrication of computer chips can be modified to produce CMOS image sensors. This is opposed to the complex and special processes required to produce CCD image sensors. CMOS image sensors also use less power and are significantly faster than CCD sensors.

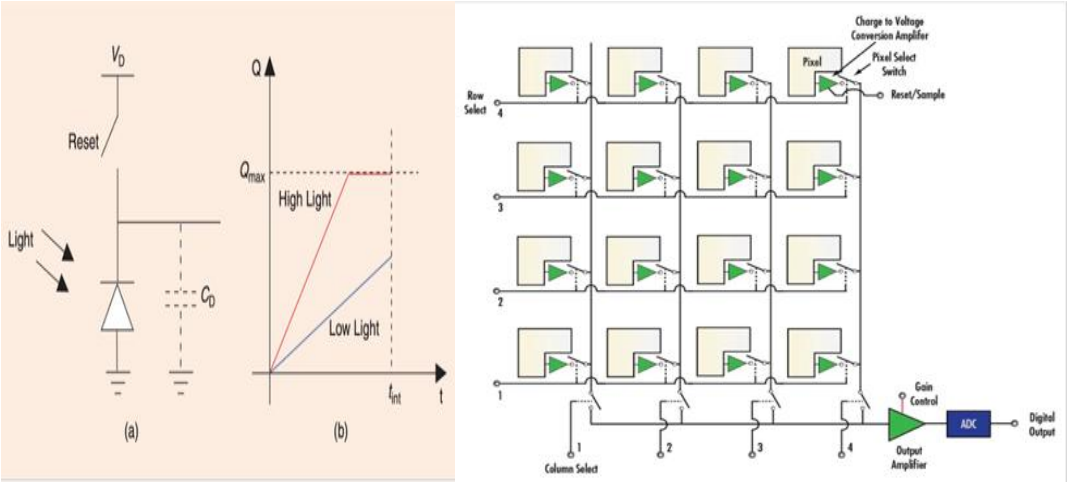


Figure 3.31: On the left, a photodiode circuit for the pixel in CMOS image sensing. On the right, the network moves data from CMOS sensors to the amplifier. Reproduction permission requested from AllAboutCircuits.com.

When a photodiode is reverse biased, the amount of reverse bias current is proportional to the light intensity of the light falling on the diode. The diode has some internal capacitance, Cd. As more light falls on the diode, reverse bias current, also called photocurrent, continues to flow, charging up Cd. In modern applications, a photodiode is exposed to light for a period of time. This helps increase the charge buildup for ease of detection, as well as reduce the noise in the image. Once the photodiodes build up enough charge, this charge is converted to a voltage and then amplified. The output voltage that goes through an ADC to digitize the measurement. The process is iterated through the array of pixels via different combinations of the row and column select switches (which model transistors) shown in the figure above [3.8].

Camera Options:

Microcontroller-Oriented Cameras:

1) *ESP32-CAM*

The ESP32 microprocessor comes in different varieties and on different development boards. One of these is the ESP32-CAM. This development board comes with a slot for a camera, as well as a microSD card slot. The video or images from the camera can be saved on the SD card or streamed to a web server. While this is not ideal, further measures could be taken to extract the image and send it to our central processor.

The camera used for this board is the OV2640. It has a maximum power consumption of 140 mW, and a standby power consumption of 600uW, which makes it ideal for our low power and heat generation requirements. Additionally, it comes at the very small size of 36 mm x 24 mm. The video or images can be streamed in UXGA format, which comes at a resolution of 1600x1200 pixels.

While this is not ideal, combined with the laser, this should be sufficient for detecting the point of impact with reasonable accuracy. The cost of the ESP32-CAM module with the camera ranges from $6 to $18, depending on the distributor.

Figure 3.32: ESP32-CAM. Reproduction permission requested from Phil Schatzmann.

2) *ArduCAM and ArduCAM Mini*

This is a MCU oriented camera module that actually uses the 2 MP and 5 MP cameras below. It thus will have similar power consumption and quality; however, it uses SPI to communicate with the microcontroller, and allows the utilization of a variety of libraries, including Arduino and Raspberry Pi libraries, that can significantly simplify programming and usage. To be precise, SPI is used for camera commands and streaming, whereas I2C is used for configuration. The board size is 34 mm x24 mm. Normally, the camera is operated at 5 V and 70-390 mA depending on the resolution (2 MP vs 5 MP), but the low power mode can bring the current down to 20 mA. Thus, we can say that in low power mode, the camera module only uses 100mW of power, whereas it could use 350 mw – 2 W of power in normal operation. Note that low power mode only works when the camera is not actively capturing inputs. While this module uses significantly more power than the base camera modules, it enables easier programming and configuration, as well as ensures compatibility. It is important to note that the camera will be used to take snapshots only when the user is about to shoot, which means that it can be left in low power mode most of the time. This can reduce average power consumption and reduce heat generation. The disadvantage that remains is that a power source capable of providing maximum power for short periods of time must be used. This will create size and safety challenges. If this module is to be used, the 2 MP version would be much more suitable to our application. The price of the ArduCAM 2MP Mini board is $26.

Figure 3.33: ArduCAM. Reproduction permission requested from ArduCAM.

3) *OpenMV Cam H7 R2*

This product is actually a board containing a camera and a programmable microcontroller. The overall solution is actually a low power computer vision system that can do simple image processing. This module uses 3.3 V input and a maximum of 170 mA, which results in a maximum power consumption of 561 mW. It can be programmed in high level python. It supports RGB and JPEG, with a maximum resolution of 640x480. The size of the board is 36 mm x45 mm. Due to its onboard ARM processor, this module can do the image processing itself, reducing the latency required to show the user the point of impact on the display. The disadvantage of this module is that it is more expensive at $65.

Figure 3.34: OpenMV Cam H7 R2. Reproduction permission requested from OpenMV.

Individual Cameras:

1) *OV5640*

The maximum power consumption for this camera is 3 V * 140 mA which is 420 mW. While this is higher than the OV2640, this camera supports better resolution as well as more image formats and a higher image quality. This camera can support 1080p as well as a resolution of 2592x1944 (5 MP). Image formats such as JPEG and RAW RGB are supported. The price is around $32.

Figure 3.35: OV5640. Reproduction permission requested from Omnivision.

2) MT9D111

This camera is more of the middle option between the previous two. It supports a 2MP image quality, and uses a maximum of 348mW of power. It supports JPEG, RGB, and RAW image formats. The camera itself and its packaging is only 30 mm x30 mm.

Figure 3.36: MT9D111. Reproduction permission requested from ArduCAM.

Table 3.10: Camera Option Comparison

Camera	Cost	Power	Size	Ease of Programming
MT9D111	$17	0.348 W	30x30mm	Hard
OV5640	$32	0.42 W	8.5x8.5x6mm	Hard
OpenMV	$65	0.56 W	36x45	Medium
ArduCAM (2MP)	$26	0.35 W (5V * 70mA)	34x24mm	Easy
ESP32 CAM	$18 + ESP cost	0.14 W	36x24mm	Medium

Interfacing

Interfacing with OV5640: There are 8 main registers that control the OV5640: CMD_MAIN, CMD_ACK, CMD_PARA0, CMD_PARA1, CMD_PARA2, CMD_PARA3, CMD_PARA4, FW_STATUS. Different commands can be written to the CMD_MAIN register (with optional parameters for some commands in the CMD_PARA_* registers). Upon receipt of the command, the CMD_MAIN register will be cleared by the on-board microcontroller unit. Once the command is completed, the CMD_ACK register is cleared to alert the user that their request has been processed. The

following is a table with the possible commands that can be written to CMD_MAIN, along with their parameter requirements:

Table 3.11: Possible commands for CMD_MAIN

Value	Command	Parameters
0x03	Trigger auto focus	No
0x07	Get Focus results	Need
0x08	Release Focus	No
0x12	Re launch zone config	No
0x80	Launch default zone config	No
0x81	Set and launch touch mode zone config	Need
0x8f	Enable custom zone config	No
0x90	VVF zone 0	Need
0x91	VVF zone 1	Need
0x92	VVF zone 2	Need
0x93	VVF zone 3	Need
0x94	VVF zone 4	Need
0x98	Set weight of zones	Need
0x9f	Launch Custom Zone config	No
0xEC	Encrypt command	Need

The following is a table with possible FW_STATUS values (note that these are very important when configuring the camera):

Table 3.12: FW_STATUS values

Status Code(s)	Meaning
0x7F	Firmware downloaded but not run
0x7E	Firmware initializing
0x70	Idle state
0x10	Focused
0x16	Config zone mode
0x00-0x0F, 0x80-0x8F	Auto focus running

Note that once the camera is initialized and firmware is downloaded, the user must check the value of FW_STATUS and confirm that it is idle. From there, the user can configure the zone and trigger the autofocus using the commands shown above.

Interfacing with MT9D111: The MT9D111 has a two wire serial interface that is used for JPEG configuration and camera control. This two wire interface is capable of accessing any configuration register in the MT9D111. In order to access it, the host processor (which is the main HUD MCU in our case) needs to write the SADDRESS of the

MT9D111, the register address, as well as the data to be written. Output data is available on an 8 pin parallel interface with additional FRAME_VALID, LINE_VALID, and CLK signals.

Operation modes are organized in contexts. Each context has reconfigurable settings such as image size, frame rate and resolution. To switch modes, the host processor must send a certain command on the two wire interface to switch contexts. For our use case, we shall mainly use Context B, which can be configured for snapshot mode by specifying image size and enabling JPEG compression. When the context switch is made, the camera automatically goes through the following sequence of events:

1) If an LED flash is present and required, it is turned on.
2) White balance settings are automatically adjusted based on the flash configuration.
3) Autofocusing is attempted, followed by enabling JPEG compression
4) Lastly, the specified number of frames is captured.

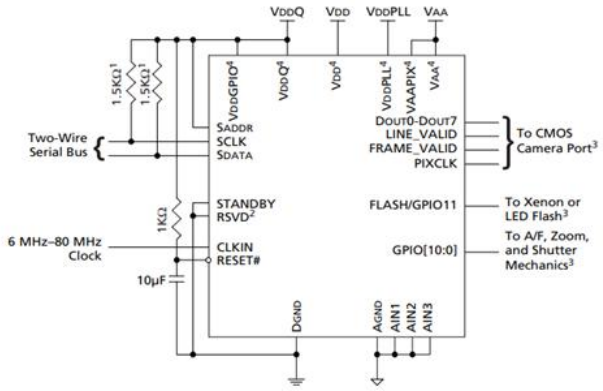


Figure 3.37: Camera controller schematics. Reproduction permission requested from ArduCAM.

Interfacing with ArduCAM/Mini: As mentioned above, the ArduCAM may use more power than the raw camera modules, however, it will significantly simplify the interfacing process. System configuration is achieved through I2C communication, whereas data is passed through SPI protocol. This imposes a requirement on the MCU of our choice: it must have both an SPI and an I2C interface. This should not be an issue as most modern microcontrollers have this capability.

Arducam actually utilizes a custom coprocessor named Arduchip. This is an FPGA based camera controller that is capable of handling complex and rapid camera signals as well as communication with other MCUs. The camera connects to a camera interface, such as DCMI described above, which then packetizes the data and stores it (through a memory interface such as DMA) in an off-chip frame buffer. Data can then be read from the frame buffer through Arduchip's SPI slave whenever the MCU requests it and at a speed that suits the MCU. In effect, all the MCU needs to do is to configure the image sensor through I2C, issue read/capture commands, and then read the data through SPI module.

Arducam uses an image sensor with JPEG compression capabilities to ensure that the image can fit in the off-chip frame buffer.

Programming the MCU to communicate with ArduCAM/Arduchip is facilitated by software libraries provided by the manufacturer. These include functions that can initialize the camera, program the configuration registers, set image settings, and send commands to capture and read images.

The table 3.13 shows the register addresses that the user can read or write from:

Table 3.13: Summary of User-Accessible Registers

Register Address bit[6:0]	Register Type	Description
0x00	RW	Test Register
0x01	RW	Capture Control Register Bit[2:0]: Number of frames to be captured The value in this register + 1 equal to the number of frames to be captured. The value=7 means capture continuous frames until the frame buffer is full, it is used for short video clip recording.
0x02	RW	Bus Mode Determine who is owner of the data bus, only one owner is allowed. Bit[7:2]: Reserved Bit[1]: Camera write LCD bus Bit[0]: MCU write LCD bus
0x03	RW	Sensor Interface Timing Register Bit[0]: Sensor Hsync Polarity, 0 = active high, 1 = active low Bit[1]: Sensor Vsync Polarity 0 = active high, 1 = active low Bit[2]: LCD backlight enable 0 = enable, 1 = disable Bit[3]: Sensor PCLK reverse 0 = normal, 1= reversed PCLK
0x04	RW	FIFO control Register Bit[0]: write '1' to clear FIFO write done flag Bit[1]: write '1' to start capture Bit[4]: write '1' to reset FIFO write pointer Bit[5]: write '1' to reset FIFO read pointer
0x05	RW	GPIO Direction Register Bit[0]: Sensor reset IO direction Bit[1]: Sensor power down IO direction

From this table, we can see that the addresses of the registers controlling the FIFO reads (reading the image) are 0x3C and 0x3D. The user, in this case the microcontroller would need to send a byte of data through SPI to Arduchip in order to read the FIFO. This will be in the format {0,cmd}, where cmd is 0x3C or 0x3D. In order to communicate with the Arduchip registers, we send the first bit, which is 1 for writing and 0 for reading, followed by 7 bits which make up the address to be read or written. Presumably, if it was

a write operation, write data would follow, whereas in a read operation we expect to receive the data via SPI interface on the microcontroller.

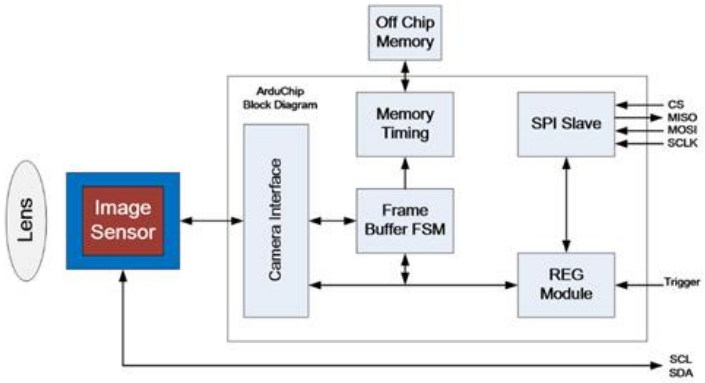


Figure 3.38: Block level diagram of ArduCAM chip. Reproduction permission requested from ArduCAM.

Data transfer between MCU/Camera and CPU box

1) *Camera to MCU to Bluetooth module transmitter*

The ESP32 comes with four SPI interfaces that can be used in a user program. Our proposed scheme is to read the image data from the FIFO in Arduchip byte by byte by sending {1,3C} from the ESP32 to Arduchip through the SPI interface. After each byte is read, it would be sent to the Bluetooth module to be transmitted to the CPU box. Effectively, we are setting up a pipeline with 3 stages: Arduchip's FIFO, an SPI rx buffer on the ESP32, and a Bluetooth tx buffer also on the ESP32. The CPU box shall be responsible for keeping the number of bytes it received from the HUD processor, and it will be responsible to determine when the image has been fully received, according to a predetermined number of bytes per each image. While this process may seem slow, recall that our application does not demand high performance. Only a single frame should be taken right before the user's cue stick impacts the ball, and minimal image processing needs to be done on it in the CPU box by the time the user finishes taking his shot. The time window that we are aiming for is on the scale of a few seconds, and that is to process a single image. Thus, slow models can be used. The reason for this compromise is the limited memory on microcontrollers, which may not be enough to hold a full JPEG image (recall that this is the reason why Arduchip implements its own image data FIFO).

On the ESP32, SPI communication can be achieved through setting certain data flags and structures. We will give a brief discussion about those.

There are 4 SPI interfaces in the ESP32. SPI0 and SPI1 are used to communicate with the internal flash memory on the module. SPI2 and SPI3 are general purpose SPI interfaces that can be used to communicate with external devices. These will be used to communicate with Aruchip. In the ESP32 world, SPI communication is

divided into five phases, most of them are optional: command, address, write, dummy, and read. In the command phase, a 0-16 bit command is written to the bus by the host. In the address phase, a 0-64 bit address is transmitted over the bus by the host. In the write phase, the host can send data to the device. The dummy phase is an optional configurable phase for user purposes and timing requirements. The read phase involves the device sending data to the host. As mentioned earlier, these phases can be configured or skipped based on certain flags, values and structures that can be set in the program. These include:

- spi_transaction_t::length determines length of write phase
- spi_transaction_t::rx_length determines length of read phase
- spi_device_interface_config_t::command_bits determines length of command phase
- spi_device_interface_config_t::address_bits determines length of address phase

If the length of any of the above parameters is set to 0, that phase is skipped. To specifically skip the read and write phases however, we also need to set spi_transaction_t::rx/tx_buffer to NULL, and deassert SPI_TRANS_USE_RX /TXDATA. SPI transactions in the ESP32 can be marked as interrupt based or polling based, with the former freeing up the CPU to do other tasks during transmission or receipt of data, while the latter reduces transaction time due to the lack of queue and ISR overhead. Data from SPI is stored in spi_transaction_t::tx_buffer (tx data) and spi_transaction_t::rx_buffer (rx data).

The ESP32 comes with an onboard Bluetooth and wifi module. Two types of Bluetooth are supported: Classic Bluetooth and Low Energy Bluetooth. Those have been discussed in the protocols section. In order to communicate with the Bluetooth module within the ESP32, either UART or SPI communication can be used, with UART being the default mode. We extensively use the Bluetooth module throughout this project. The following is a list of communications that the Bluetooth module in the HUD's ESP32 processor will undertake:

- Receiving a signal from the cue stick indicating the user is attempting to shoot, which will trigger the camera to take a snapshot
- Transmitting camera data to the CPU box for processing
- Receiving display data from the CPU box and/or other devices to be transferred to the HUD's display system
- Receiving orientation/location data from the glove in order to properly drive the audio system

To use the Classic Bluetooth option, the ESP32 uses the BluetoothSerial library, which simplifies communication between ESP32's processor and the internal Bluetooth controller. Internally, this is done over a serial interface. This library provides functions such as begin(), available(), write(), and read() to easily check if data is available, write to the Bluetooth interface, and read from it respectively.

It may be profitable to use Bluetooth in BLE mode, as we can get the power savings without sacrificing performance. This is because the data that we need to receive or transmit is not very large, and also not continuously present. BLE usage is a bit different compared to Classic Bluetooth. Although a very brief summary was given about BLE in the protocols section, we need a deeper understanding to continue.

BLE is more complex than classic Bluetooth. Communication channels have GATTs, which are generic attribute profiles. These define specs for data transfer, and include services and characteristics. A characteristic is a group of attributes (such as value, description, and properties), while a service is a collection of characteristics and has a 16 bit/128 bit UUID. This can be generated online easily. BLE devices have 5 states: scanning, advertising, initiating, connection, and standby. A BLE device can either be a server or a client. Data moves from server to client. It is possible for the same device to alternate between being a server and a client; however, a device cannot be both simultaneously (though time slicing techniques can be applied). ESP32 provides some libraries like BLEDevice, BLEUtils, and BLEServer that provide useful API's for BLE communications.

2) *Bluetooth Module Receiver to MCU in CPU box*

Assuming that we use the Raspberry Pi for our central CPU box processor, no external Bluetooth module is required. This is because, like the ESP32, the Raspberry Pi comes with an integrated Bluetooth and Wifi chipset. Whereas most of the model 3 Raspberry Pi's (with the exception of 3B) come with a Broadcomm solution, the 3B and the 4 models come with Cypress solutions. Regardless, both groups of versions support BLE and Classing Bluetooth natively. The BlueZ Bluetooth stack is used with all modules.

BlueZ is the official Linux Bluetooth stack. It supports core Bluetooth layers and protocols, and takes care of all the low level details. It consists of a multitude of modules:

- Bluetooth Kernel
- Audio Kernel layers
- RFCOMM, BNEP, and SCO kernels
- UART and USB virtual device drivers
- Bluetooth libraries
- Configuration and testing utilities
- Protocol decoding tools

Using BLE Bluetooth on Raspberry Pi: To start using BLE on the Raspberry Pi, we must first install a clean version of Linux, followed by BlueZ, which we have given a brief introduction about above. It is also useful to install hcidump, which is an addition to the BlueZ stack. While the Raspberry Pi does indeed support Bluetooth, It is recommended that we connect a BLE USB adapter to one of the Raspberry Pi's ports. The reason is that Raspberry Pi can reliably support Classic Bluetooth. However, when using BLE, the

Raspberry Pi has been known to be quite unreliable. Utilizing a BLE USB device should eliminate or at least significantly reduce this problem. From there we can use the object nodes ‘noble’ and ‘bleno’ in our code. These are part of bluez and are used to construct objects that can act as BLE centers (noble) or BLE peripherals (bleno). After that, gatttool, another tool that is part of the bluez stack can be used to connect to nearby BLE devices. Recall from our discussion on BLE protocol that peripheral devices advertise services which contain data fields called characteristics; whereas center devices scan and receive data from peripherals. This is a one way simplex communication channel, and so it is necessary to setup both objects and alternate between their use to be able to both send and receive data from the same device.

Alternatively, we can use bluepy, without the BLE USB device dongle

Using Classic Bluetooth on Raspberry Pi: BLE Bluetooth is much more complicated to work with on the Raspberry Pi than Classic Bluetooth. There have been many observed errors, dependencies, and communication problems that make debugging a much more intensive project. Thus, it may be beneficial to use classic Bluetooth on our central CPU unit. The only challenge that comes with that is interfacing the CPU’s classic Bluetooth with the HUD, glove, and stick’s BLE units. While this may be a more involved process, it provides the best power efficiency as it allows the battery powered, wearable devices to use the low energy BLE, while also allowing us ease of programming on the CPU’s classic Bluetooth module. More exploration needs to be done on whether this is a possibility. We must also consider the hardware and protocols supported by the Raspberry Pi. Upon further research it appears that Bluetooth 4.0 and beyond is only available in BLE; however, the built-in Bluetooth module in the Raspberry Pi, which uses Bluetooth 4.1, and thus only BLE, is backwards compatible with Bluetooth classic devices. Thus, using Bluetooth classic depends on the other devices, which, in our case, are the ESP32’s present on the HUD, glove, and stick.

3.2.5 POWER SYSTEM

Microcontroller Power: For the glove, we clearly wanted our power system to be wireless so our best option would be battery power. We had to determine what would be the best type of battery power to go with. As we decided to use the ESP32, we needed to power it with approximately 2.55 to 3.6 V. We could then just use the regulated 3.3 V from the ESP32 to power our sensor circuits [3.9].

Possible Options:

1) *Power Bank*

   Power Banks internally uses a 3.7 V lithium battery, then transforms this voltage to 5 volts with loss, a connected ESP32 then uses an LDO (low drop-out voltage regulator), This steps down the 5 volts into 3.3 volts. This can largely affect powering our ESP32 due to the multiple step downs in power which cause large losses, our system would require much more efficient power supply.

Figure 3.39: Power Bank. Reproduction permission requested from Radio Shuttle.

2) NiMH Batteries or Standard Batteries

Direct operation with two NiMH batteries does not work, as one battery only supplies approx. 1.2 volts, i.e. 2.4 volts with two batteries. This is too little for the required 2.55 volts, which the ESP32 needs at least. Three NiMH batteries connected in series are also not an option, as the maximum voltage of 3.6V for the ESP32 is exceeded with full batteries.

Figure 3.40: NiMH Batteries. Reproduction permission requested from Radio Shuttle.

3) Lithium Batteries

Whether two 1.5V lithium batteries in series or one CR123 3V lithium battery, everything works perfectly with lithium batteries. These keep a voltage of 3V quite constant, at less than 2.7 volts more than 90% of the capacity of a lithium battery is used, at 2.55 volts it is practically empty. Lithium batteries also provide the high short-term power requirements of WiFi operation without any problems.

Figure 3.41: Lithium Batteries. Reproduction permission requested from Radio Shuttle.

4) *LiFePO4*

Modern LiFePO4 batteries also work excellently, but deliver approx. 70% less energy than a lithium battery of the same size. However, LiFePO4 batteries can be recharged or replaced with a charged battery. LiFePO4 batteries also deliver unproblematically high performance for WiFi operation, but do not have the disadvantage like lithium polymer batteries, which can catch fire if used incorrectly or if quality is poor.

Figure 3.42: LiFePO4 Battery. Reproduction permission requested from Radio Shuttle.

5) *Lithium Polymer or Lithium Batteries*

Needless to say, lithium polymer or lithium batteries work because they provide enough power for the ESP32. However, the voltage of 3.7 to 4.2V is much too high for the ESP32, depending on the state of charge, and must therefore be reduced. This has the disadvantage that a large part of the energy is permanently used to reduce the voltage to 3.3V. Simple LDO controllers require approximately 2000 times more power for standby than the ESP32 in deep sleep mode, every second, 24 hours a day, 365 days a year. Even better controllers still need a lot of power. Of course, lithium polymer batteries work for one day or a few days, but for weeks and years this is almost impossible.

Figure 3.43: Lithium Polymer Battery. Reproduction permission requested from Radio Shuttle.

Power Supply Selection: The best possible choice for powering the ESP32 was be the lithium batteries or LiFePO4 batteries so that we couldreach the desired voltage to power the device with high WiFi performance. This was necessary as we wanted to have an easy connection.

Connection to the ESP32: So after we chose the power supply, we needed to have a way to connect this to our ESP32, in some cases you would believe this to be a simple connection of your power supply directly to the microcontroller but this would actually cause a lot of problems. When connecting power to any device it is best practice to use bypass capacitors. The main reason for the use of bypass capacitors is for noise, this noise can be from many different things such as from the IC or even other IC's but in this case we would want to use bypass capacitors for the noise coming from the power supply in this case the multiple batteries. These bypass capacitors are most commonly connected between the VDD and the GND pins of the device and they would eliminate the effect of voltage spikes from the power supply as well as reducing noise. In most cases we tend to use two bypass capacitors one at the power supply and one at the device.

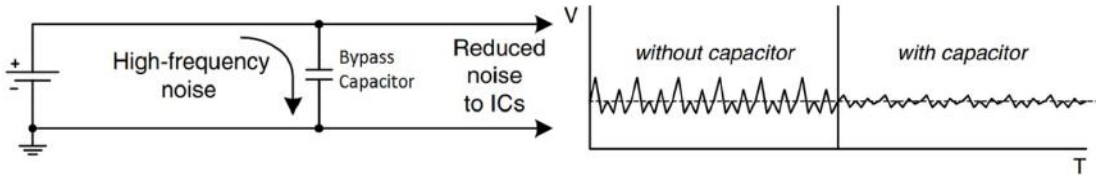


Figure 3.44: Image showing bypass capacitors effect on noise, reproduction permission obtained from Electronics Hub.

When selecting what bypass capacitor to use there are a couple things to take into account. One is the value of the capacitor for lower bandwidths of the device, such as 1 MHz, we would use a value of 1 uF; for higher frequency bandwidths, such as 10 MHz, we would use 0.1 uF capacitors. Another aspect that we take into account when using bypass capacitors is the typing we would want to use Multilayer Ceramic Chip Capacitors for bypassing a power supply. The last thing that we would take into account for bypass capacitors would be the placement. It is very crucial to place these capacitors to be as close to the device as possible because the extra track on the PCB or wire can translate into an inductor and resistor which would lower the useful bandwidth.

HUD Power System: Because each sub-system is operating wirelessly from each other, each one needed its own battery that takes into account the peripherals of that system. For the HUD, these peripherals are the display, camera, speaker, and ESP32 microcontroller. The possible options of batteries remain the same as those from the microcontroller shown previously, so we will begin by simply exploring the power requirements.

Power Requirements for Components: Based on the engineering and marketing requirements, the HUD requires a power supply that keeps the HUD operational at an absolute minimum of at least 30 minutes before needing to be charged. This requirement and the voltage requirements of each individual component are the driving factors that determine what size and array of batteries need to be utilized in this design. The table below lists the voltage and power draw requirements for each component. The power draw requirements will have a range from maximum to minimum draw and be listed in mA.

Table 3.14: HUD Battery Requirements

Battery Requirements			
Component	**Voltage Req**	**Power Draw (Peak)**	**Power Draw (Typical)**
ESP32	3.3 or 5	240	80
Display	3.3	0.78	0.43
Camera	5	70	20
DAC/Speaker	3 to 6	40	16
Total		351	117
Total (With 1.25 Tolerance)		439	146

Layout: On the HUD, there will be two sides. On the left side of the device, will be the speaker and camera, and on the right side will be the ESP32 and display. In the interest of balancing weight management, ergonomics, and overall size, it is in the best interest of the design to have the two batteries, one for each side. This keeps the overall size of each battery much smaller and prevents any potential imbalance of the device due to the relatively heavy batteries.

Battery Consideration: After voltage, the next most important parameter for the battery supply selection is size. This size is given most commonly in mAh. Milliamp-hours (Or mAh) is the measure of how many hours a battery can sustain a constant draw of current. For instance, if a device requires 100 mA of current, a battery with a 1000 mAh capacity could supply those 100 mAs of current for 10 hours.

This equation of Battery Capacity / Average Current Consumption = Maximum Lifetime will be used to calculate the battery capacity required.

Battery Comparisons:

Table 3.15: Battery Option Comparison

Part Number	**Chemical Makeup**	**mAhs**	**Output Voltage**	**Approximate Dimensions (inches)**
US-702528-500 0PCBJST	Li-Ion Polymer	500	3.7	1.18 x 1 x 0.28
9V Rechargeable Battery	Li-Ion Polymer	600	9	1.9 x 1 x 0.65
Adafruit 328	Li-Ion Polymer	2500	3.7	2 x 2.55 x 0.3

These three options shown above are indicative of much of what is on the market that would work in this application. Due to battery cell configurations, there are almost no direct 5V batteries on the market, and anything smaller will need additional batteries to power the camera, which requires a 5V input. With this in mind, the widely used, 9V battery used in conjunction with a buck converter was chosen. The chosen battery to be used with all peripherals had a 9V output and a 600 mAH lifespan, meaning it couldsupply the components at peak draw for over an hour, meeting out requirements. Additionally, with its rechargeability, this battery would keep operating costs lower.

3.2.6 OPTICS AND PHOTONICS

The technique chosen to display the information to the user is to create a Head's Up Display (HUD). The inspiration for this design stems mostly from the HUDs created in the car industry. Car makers implement a system that reflects a light source off of two separate mirrors, through a lens, then onto a specially coated portion of the windshield to display important telemetry about the vehicle right in front of the driver's eyes.

This project took a similar approach to placing the information in front of the players eyes. The light source was in the form of the display chosen in the previous section, which is then reflected off of one mirror and then onto a specially coated clear surface for the user to see.

In this case, the display is placed perpendicular and facing forward when related to the user's face. The only way for the user to see the information was to reflect it off a mirror. The mirror diverts the display's light out from the device and onto the desired location at an angle of approximately 45 degrees.

This all will required concepts revolving around the effect that mirrors have on light source to ensure proper selection of components. These two topics will be covered below and then the selection of the proper mirror and lens will be made.

Mirrors: The physics of a mirror are somewhat similar to those of a lens, except obviously rather than bending the light that passes through the mirror, it bends and manipulates the light that hits its surface and then reflects the manipulated light away. [3.10] The three types of mirrors that are most widely used are the plane mirror, the convex mirror, and the concave mirror. A plane mirror reflects the image with normal proportions but reversed from left to right. This is what is found in a bathroom and is why your left arm appears on the left side of your body in a mirrored image, even though if one were to imagine the perspective of the mirror, it should appear on your right. A convex mirror is curved towards the incoming light and appears to "expand" the image as it brings the light waves farther apart and creates a wider field of view. These are often found in parking garages to enable drivers to see clearly around the corner. Finally, a concave mirror is curved inward and brings the light waves together towards the center causing the reflected light to appear larger and "magnified". An example of this is seen in makeup mirrors and spotlights. These three types and their effects on an incoming light wave can be seen in the below Figure 3.45.

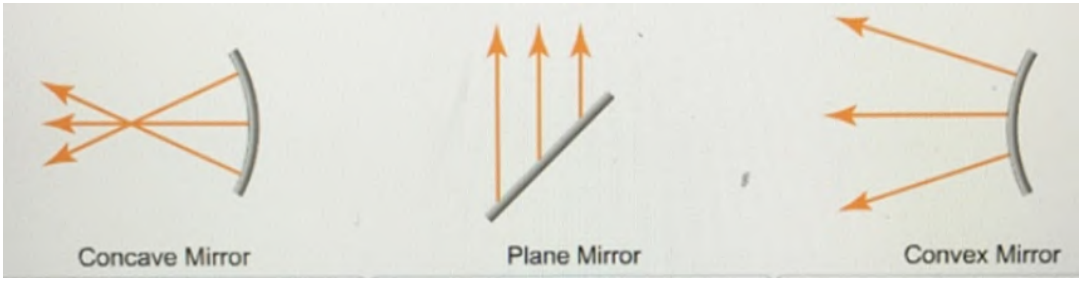


Figure 3.45: A view of how reflected light waves behave coming from the three most common mirror geometries. Reproduction permission requested from Physics LibreText

In this project, the focus is on simply changing the direction of the light rays coming from the display. Therefore, the plane mirror is the best choice as if either a convex or concave mirror was used to manipulate the waves in any way, the image would be inevitably distorted. As the plane mirror is the best choice, further investigation will be done.

The way physicians have determined to explain the image distortion of mirrors is to talk in terms of virtual and real images. What this means, is that the image seen in the plane mirror, that appears to be located behind the physical location of the mirror, is a virtual image. On the other hand, a real image is formed by curved mirrors, as both the image and the light appear to be on the same side of the mirror. Since the focus of this design relies on the physics of a plane mirror, only the virtual image will be explained.

When looking into a mirror, one observes the left-right image reversal that occurs with plane mirrors. [3.11] This phenomenon is extremely important with respect to this application, as image clarity is imperative to the users feedback and ease of use. The most obvious example that displays both the relevance to this project and visualizes the issue is seen with shirts containing lettering on them. If the person is wearing a shirt that says “UCF” on it, when observed from the plane mirror, it will appear as “FCU” (Note both the letters order and orientation will be reversed). This is a direct result of the lack of manipulation of the incoming light rays.

Imagining the word of an image traveling straight into the mirror from where the person is standing, it makes intuitive sense that that is exactly what will come back at the user. The image shown in Figure 3.46 visualizes this even better. What is occurring to the word, is also occurring to will also occur to any light source. This is why the image must be reversed again before being placed on the display. To accomplish this, a lens will be used.

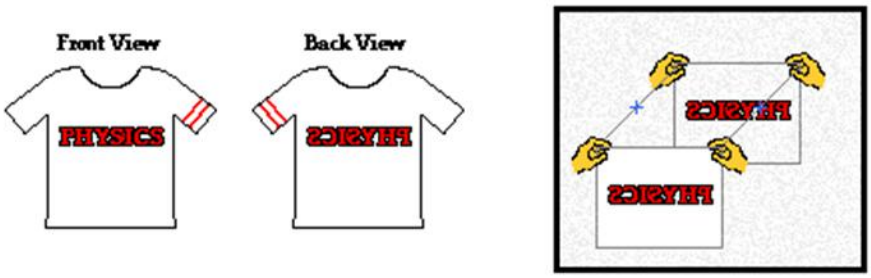


Figure 3.46: Showing the left-right reversal effect of a plane mirror. Reproduction permission obtained from Physics Classroom.

Another ideal characteristic of plane mirrors is the relative size and distance of the virtual image. Unlike concave or convex mirrors, plane mirrors do not distort the light rays that hit it. This means that if a light source is 3ft away from a plane mirror, its reflection will appear as if it is 3ft away. This means there does not need to be any consideration into the focal distance. This lack of distortion also means that the size of the image will be exactly the same size after being reflected.

Now that the plane mirror has been determined as the ideal technique to reflect the image, the last thing to consider is the importance of quality in mirror selection. This is simple, as one can easily understand that any imperfections in the mirror will lead to distortion or blockage of the light rays, creating an unclear image.

Lenses: A lens is a transparent piece of material that is shaped in such a way to cause light to bend in specific directions as it passes through. [3.12] The bend of the lens can cause the light to either converge or diverge away from the center point. If the lens is in a convex shape with respect to the incoming light waves, it will converge, while if it is in a concave shape, it will diverge. This effect is shown in Figure 3.47 below.

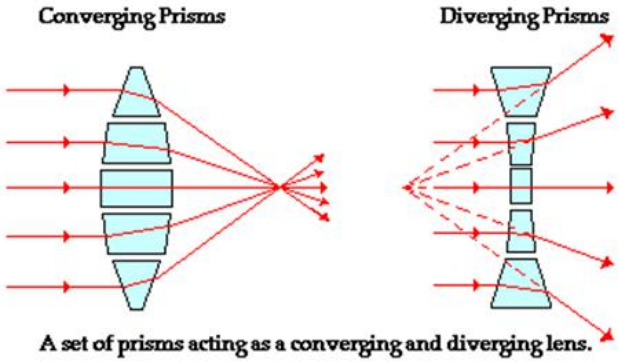


Figure 3.47: Diagram of parallel rays being converged or diverged. Reproduction permission requested from BYJUS

Two important terms that are used often when describing a lens are its focal length and focal point. Focal point refers to the place that all the incoming parallel light rays will converge after passing through the lens, while the focal length is the distance from the

center of the physical lens, to the place where the rays converge (The Focal Point). The focal point can be clearly seen in the above Figure 3.47 as the point where all the lines intersect.

As discussed in the above section on mirrors, the chosen type is the plane mirror. The major effect of the plane mirror that the lens will be used to counteract is the left-to-right reversal of the image. As one can see from the above Figure 3.47, only one of the two lenses will be useful in counteracting this effect, and it is the converging prism, which corresponds to a convex lens.

This means that to achieve a clear image, the light will have to pass through the lens, the rays will intersect and invert at the focal point, and the surface the image is placed on must be at the location where all the rays are the same width as they were when they first entered the lens. The math to accomplish this, while seemingly complex, is surprisingly simple. The lens must be in the near perfect center between the mirror and reflection surface. If this is achieved, the image will be approximately the same size as it was when the image entered the lens, enabling good clarity.

These convex lenses can be found and obtained from any magnifying glass or set of reading glasses. Simply cut the desired size out of the glass and it will behave as desired. Much like the mirror, quality is important as any imperfections will lead to distortion of the overall image.

Waveguides: Waveguides work exactly as the name suggests. [3.13] Whether it be extremely high frequency light waves, or relatively low frequency (by comparison) RF communication, a waveguide's purpose is to control and move the waves to a desired location. Additionally, these waveguides can, while moving the waves, change certain characteristics as it occurs. For instance, in RF communications, impedance matching must be done in the waveguide to minimize the overall loss. For our purposes however, and in the context of augmented reality glasses, waveguides work by bending a projected light beam in specific ways and then onto the eye of the user or the surface in front of the user's eye. There are a couple of different types, the most common of which are diffractive waveguides, holographic waveguides, reflective waveguides, or virtual retinal displays. For the purpose of AR glasses specifically, the one most commonly used is the diffractive waveguide.

Diffractive waveguides get their name from the use of diffractive grating within the waveguide. These ridges that can be seen in Figure 3.48 break the light waves up at specific wavelengths across the visible light spectrum. The distance between each point is known as the grating period. When the light encounters a ridgeline, the beam is split into multiple diffraction orders that can be either further split up, or have an incident angle such that they exit the waveguide and get projected.

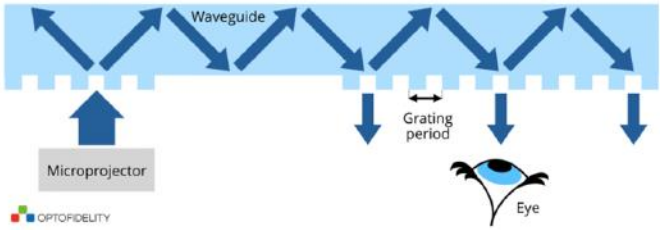


Figure 3.48: Optical Waveguide. Reproduction permission requested from OptoFidelity

Projection Surface: For the surface that the image will be reflected onto, it is most desirable for it to be clear. This allows the information to be obtained without obstructing the user's vision. The only issue with clear surfaces, is that it creates two images when things are reflected onto it. Since light is allowed to pass through the first layer of glass or plastic, it will also interact with every layer past that. This creates an undesirable, and distorted image. Therefore, the plastic or glass must be selected specifically to avoid this or coated with a special film that prevents this from occurring.

3.2.7 AUDIO AND HAPTIC FEEDBACK SYSTEMS

HUD Audio System: One of the core tenants of this project is to assist the visually impaired in playing and training to become competent pool players. As such, conventional methods that have been used to start the game and display data cannot be utilized for this purpose. For example, actual shot power, compared to the desired power, cannot be displayed on the HUD for obvious reasons. It was thus up to us to develop a thorough reporting and communication system that is suitable for the needs of the visually impaired. We have decided to use the HUD's audio system for this purpose. This system will be comprised of a variety of modules and submodules, including a microSD card, speakers, and various boards and circuitry. Other options will also be explored.

Text to Speech:

1) *Background*

   This subsection aims to deliver a high level understanding of the basic technology behind Text-to-Speech devices and programs. While low level hardware and code will not be discussed, the topic will be dissected from a block level perspective. Before diving in, we must first understand the definitions of a few terms relating to audio and sound:

   - Phoneme: A phoneme is the smallest unit of sound in a word. The phrase "hello, have a good day" can be broken down into the following phonemes, separated by spaces: HH AH0 L OW1, HH AE1 V AH0 G UH1 D D EY1.
   - Mel Spectrogram: a transformation to the short time Fourier Transform of an audio signal. The high level idea of this spectrogram is that it emphasizes small details in low frequencies and also reduces the

protrusion of high frequency components. This is important since typically, speech is in the lower frequencies, whereas the higher frequencies are usually noise signals which are unwanted.

A Text-to-Speech system takes text as input and produces audio as output. There are four main components of a TTS structure: the preprocessor, the encoder, the decoder, and the vocoder. Here, we provide a high level overview of the functionality of each block.

The preprocessor takes the text input and tokenizes the sentences into individual words. After that, words are broken down into phonemes. The output of the preprocessor is called a Linguistic Feature.

The linguistic feature is then sent to the encoder. The encoder then figures out properties such as pitch, energy, and duration, and concatenates those to the phonemes to create what is called a Latent (Processed) Feature.

In the decoder, the Latent Processed Feature is converted to an Acoustic feature which is our Mel Spectrogram. The reason for this intermediate step is because there is less variance in the Mel Spectrogram than in the audio file, which would cause a bigger information gap.

Lastly, the Vocoder converts the Mel spectrogram into an output waveform, which is audio. In modern day TTS, this is typically done by a trained neural network.

2) *Possible Implementation Methodology*

There are a variety of ways to implement text to speech on a microcontroller like the ESP32. One popular way is to utilize a web server. Thus, programming is only needed to connect the ESP32 to the server and acquire the ability to send and receive data. From there, the webserver does all the heavy lifting, and sends the output to the ESP32, which can then drive the speaker module. It is also advised to use an amplifier analogue circuit in order to reduce noise and increase sound quality. The advantages of this methodology is that it can be very flexible, meaning that we can convert any text that we have to speech. Web Servers that offer this service typically use a highly trained deep neural network that can produce high quality sound for any input text (or at least a large subset of text). The disadvantage of this method is that it creates a lot of latency, due to the connection and data transfer between the microcontroller and the webserver, as well as the time it takes for the web server to process requests. This can severely hinder our system, especially since audio data is needed to help guide the visually impaired in the hand placement and aiming process, meaning that the speech needs to continuously change.

Another approach, which is more suitable for our application is to not use actual text to speech. The proposition is to store the audio files for certain predefined commands, such as “aim higher”, “aim lower”, “move hand to the right”, “move

hand to the left", "ready to shoot", "shoot at half strength". These can be stored on an SD card, provided we are able to procure an ESP32 board capable of hosting one. Depending on data imported from IMUs and other sensors on the glove and stick, the microcontroller will choose which command to output. Once chosen, the audio file will be loaded from the SD card memory and outputted to a speaker interface. This eliminates the need for complex processing (which can be quite costly in terms of power), longer than required latency, and potential failures due to , say, errors connecting to the server. The disadvantage of this system is that it is severely limited in terms of output speech. Only a limited number of commands can be stored on the SD card. In addition, the commands themselves cannot be long, otherwise we run into issues with interfacing with memory and outputting to the speaker at a fast enough rate. However, because our application is limited with regards to voice commands, these disadvantages can be endured for the pros of the system. One necessary aspect to investigate is the relative performance and throughput of memory access protocols such as DMA (and potentially even SPI, depending on the access mechanism) compared to the performance and throughput required of the IO pins to the speaker (or whatever protocol the speaker follows). We must ensure that data transfer from memory is faster than data transfer to the speaker. This is needed to avoid 'cutting' in the sound. A buffer may be needed if this condition is not inherently satisfied.

Interfacing with an SD Card:

1) *Introduction*

Interfacing with an SD card using microcontrollers such as the ESP32 is a useful feature that can be utilized to read audio files (as mentioned above) required to produce audio commands to aid the visually impaired in placing their hands correctly on the billiards table and aiming the cue stick at the correct position. It is also useful in providing hints about the required strength of the shot. SD cards can be interfaced using SPI protocol. This is the basic way which most microcontrollers use. Another way to interface with an SD or microSD card using the ESP32 is to use MMC. Both methods will be outlined in the following two sections. Note that a discussion of SPI will not be provided as it is available in another section in this document.

2) *SPI Communication*

Most SD card slots/readers come with an IO SPI interface. This abstracts interfacing with memory to interfacing with a simple SPI slave. The end user does not need to know information regarding internal memory protocols such as DMA, which has been discussed earlier in this document. File data is read via SPI to the ESP32. Audio data can then be routed to the I2S driver, which can drive an onboard or offboard DAC, connected to an amplifier and speakers. There are a plethora of available libraries that can carry the process of reading the SD card via SPI as well as routing the data to the I2S driver. More information about this will be discussed in future sections

3) *SDMMC mode*

Instead of using SPI, the ESP32 can also use MMC, which provides a significant speedup in file IO. SD cards (or microSD cards) typically also support MMC configurations. Similar to the SPI interface, there are also a variety of libraries available to interface with MMC.

MMC, or MultiMedia Card, is a standard for data storage and communication developed by SanDisk and Siemens in 1997. The main motivators behind it are low cost and low pin number.

Amplifier Boards with DACs:

1) *Background information on DACs*

Digital to Analogue converters (DACs) are circuits that are used to convert digital data (which is usually a stream of bits, that is, 1's and 0's) into analogue data (which are continuous numbers, for example 5.648302). DACs are strongly needed in today's world because although most computations and data manipulation happens in the digital world (for example inside registers in CPUs and MCUs), the real world remains, for better or for worse, an analogue world. Thus, conversion between the two datatypes is necessary in order to collect or inject data into the real world, which has been digitally manipulated. The following is a discussion on how two common types of DACs work: the Binary Weighted DAC and the R-2R Ladder DAC.

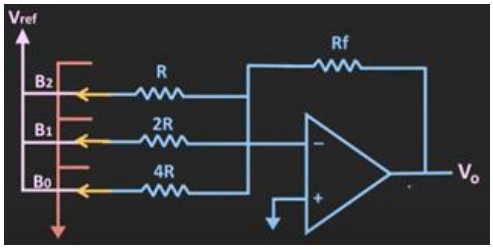


Figure 3.49: Binary Weighted DAC Schematic (Basic)

The image above shows a Binary Weighted DAC. It is clear that the main component of this type of DAC is a summing amplifier. It can be shown that the output Vo of a summing amplifier is simply a weighted average (or weighted sum) of the inputs. However, we can see that in the circuit, all inputs are connected to either the reference voltage or ground, depending on the digital value of each input bit (though it is represented by a switch in the image, it is not hard to reconstruct the same circuit using MOSFETs). If an input terminal is connected to ground, then since Vin is 0v, it is not present in the output. Using this fact, it can be shown algebraically that

$$Vo = -K * Vref * \Sigma(\frac{B}{2^{N-1-i}})$$

where K is some gain, N is the number of input bits, and B is the ith binary digit input. We can see now where the name comes from. The output is always a fraction of Vref (since B is either 1 or 0), and the weight of that fraction depends on the position of the bit. If we examine this output function, we see that it implies a stepping transfer function, i.e the output values are still at discrete levels, not the smooth graph that we associate with analogue signals. That is correct at first; however, the output of this circuit is typically passed through a low pass filter, which then reduces the output signal to a smooth graph. The problem with this type of DAC is that better resolution requires more resistors, which are hard to obtain at an exact value in real life, increasing the error in the final output.

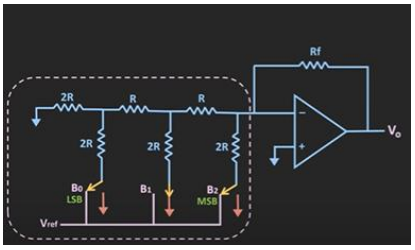


Figure 3.50: Ladder DAC Schematic (Basic)

The above is an image of the R-2R Ladder DAC. Again, while the circuit above employs simple switches, it is not hard to imagine MOSFETs in place of those switches. This type of DAC is more scalable. Using some algebra and clever circuit theory (which includes Thevenin's theorem and Superposition theorem), it can be shown that the output voltage is represented by:

$$Vo = -K * Vref * (\frac{B0}{2^n} + \frac{B1}{2^{n-1}} + \cdots + \frac{B(n-1)}{2^1})$$

That is, the output is again a weighted sum (which is always a fraction less than one) times the gain times the reference voltage, where n represents the number of input bits. Again, Bx is either 0 or 1, and we can see that the most significant bit, Bn-1, contributes the most to the result. This shows that this is also a DAC. Once again, while the output is a stepping function, it can be passed through a filter to smoothen it out.

2) *Introduction and Motivation to Use Amplifiers/DAC boards*

There are two main questions to be answered here: first, why do we need to use a separate DAC, and second, why do we need to use an amplifier board. We will discuss these two questions in this section. Before that, let us briefly examine the I2S capabilities of the ESP32.

The ESP32 contains two I2S modules (I2S0 and I2S1) which also have DMA controllers, meaning they can directly access memory without using the main processor, improving performance. The I2S modules can also route their output

data to either the onboard 8-bit DAC, or can transmit their data in PDM form. PDM (Pulse Density Modulation) is similar to pulse width modulation (PWM). Typically, the ESP32 is used to output raw I2S though, and data is fed to an off-chip board with a DAC and an amplifier (which can then be connected to a speaker).

External boards are used for better sound volume and quality. Simply put, the 8-bit DAC present on the ESP32 does not have a satisfactory resolution, especially in today's world where sound quality has been improving exponentially.

Amplifiers are needed to produce audible output. The output of the DAC is usually not powerful enough to power any speaker. An amplifier is needed to produce higher peak to peak voltages capable of driving the speaker. It is important to note that while presence of an amplifier is always required, purchase of an off-chip amplifier may or may not be needed, depending on the type of speaker that is being utilized. Active speakers come with built-in amplifiers. Those can be connected directly to DACs. Passive speakers; however, require an external amplifier.

3) *PCM5102*

The PCM5102 is a stereo decoder. It takes in a stereo I2S stream, and outputs analogue audio on two pins (one pin per channel). These two pins cannot drive a speaker independently, i.e another standalone amplifier must be used. Alternatively, some PCM5102 boards come with a headphone jack. Thus, the PCM5102 may be useful for certain applications.

4) *MAX98357A*

The MAX98357A is a class D, 3W audio amplifier with an onboard DAC. It takes I2S signals as input and outputs an analogue signal which can directly drive a standard speaker. Note that this output cannot be connected to another amplifier (or an active speaker) because the final output is a 300KHz PWM signal. Operating voltage for this board is between 3 and 6 volts. The output power is 3.2W for 4 ohm impedance, and 1.8W for 8 ohm impedance. Efficiency is 92% if load resistance is 8 ohms and output power is 1W. Maximum quiescent current is 2.9mA.

Table 3.16: DAC Option Comparison

DAC	Cost	Power
MAX98357A	$5.95	1.8W-3.2W
PCM5102	$9.00	3.3V, undisclosed current draw or power

5) *DAC Selection*

For our DAC, SCRATCH utilized the MAX98357A DAC. This was due to its significantly lower cost, as well as its power requirement that met our performance and sizing/heating constraints. Additionally, the sound quality mes our expectations.

Speakers:

1) *Introduction*

In our project, the HUD serves as the main guide for a user throughout the game of pool. It provides visual guidance regarding the pocket to aim for, the most amortizing point of impact on the cue ball, as well as the optimum strength of the shot. However, one category of users that our product aims to serve is the visually impaired user. Due to obvious reasons, visual guidance systems will clearly fail in assisting the visually impaired throughout the game. As a result, our product will use auditory guidance to provide valuable information regarding hand placement, orientation, general aim, and strength of the shot to our visually impaired user. In order to do so, a speaker is required. To avoid unnecessary complexity and also provide a transparent environment, use of headphones will be avoided. Instead, we shall use a single speaker that will output audio commands to help guide the user in the aspects mentioned above. Before discussing possible speaker options, we will present a brief discussion on the fundamentals of how speakers operate, and the brilliant engineering behind them.

2) *Physics and Types*

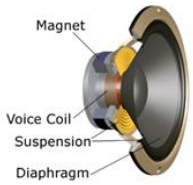


Figure 3.51: Diagram of a basic speaker. Reproduction permission requested from SoundGuys.

Speakers operate on Faraday's law. The main component that produces the sound is called the speaker driver. This comprises of a coil of wire suspended between two poles of a permanent magnet. On one end of the coil, a speaker cone is attached. This speaker cone is held by an airtight surround. When the coil of wire is driven by an AC signal, due to Faraday's law a force is generated which moves the coil back and forth. This in turn moves the cone, and therefore the air in the surround, back and forth, creating the sound wave. Speaker drivers are typically housed in box-like fixtures to minimize opposite waves that are created along with the desired soundwaves. This occurs because of the geometry of the box, and allows for a louder and more audible sound, especially in the lower bass frequencies.

As a reminder to the reader, Faraday's law is the principle behind both motors and generators. It stipulates a relationship between electric current, magnetic fields, and motion of conductors relative to magnetic field lines. The presence of two of those elements creates the third. In motors, magnetic fields and current produce motion, whereas in generators, motion and magnetic fields produce electric current. Faraday's law can be summarized in the following equation:

$$\varepsilon = -N\frac{d\phi}{dt}$$

Where epsilon is the induced voltage (producing electric current), N is the number of loops in the coil of wire, and the Greek letter Phi is the magnetic flux. Notice that here we are looking at the change in magnetic flux which, even though it can be produced by changing the power of the magnets, is usually caused by relative motion of the coil.

From an electrical standpoint, there are two types of speakers: active speakers and passive speakers. Active speakers have their own built in amplifier, whereas passive speakers do not have their own built in amplifier. When choosing a speaker, we must first carefully examine the input to that speaker, most often a DAC board. Depending on the output of such a board, the type of speaker has to be chosen appropriately. If we choose a board with an on-chip amplifier, such as the MAX98357A, then a passive speaker must be paired to it. However, if we choose a board without an amplifier, such as the PCM5102, then an active speaker must be chosen. Another important aspect of a speaker is power. This must be taken into consideration in certain applications, such as our application, where power and heat are constrained for user safety and to conserve battery power.

Another factor to consider when purchasing speakers is the speaker impedance. This is particularly important when examining the connection between the amplifier and the speaker. Basic knowledge on amplifiers tells us that each amplifier has a certain output impedance and input impedance. Typically, the lower the output impedance, the higher the power transferred to the load. This is because the load and output impedance act like two resistors that form a voltage divider. However, it is important to remember that current draw will also differ. Decreasing the output impedance will change current draw. From Thevenin's theorem, we get that the maximum power transfer to the load occurs when the impedance of the load matches the Thevenin impedance of the amplifier, which is the output impedance. Thus, impedance matching is a significant factor and if ignored can result in distorted or quiet sound. Generally, lower impedances (as long as they match the amplifier's output impedance) produce higher quality sound [3.14].

3) *Possible Options*

As mentioned above, choosing a speaker depends on the DAC board that we choose to use, if any. This determines whether an active or passive speaker will be utilized. Here, we will discuss some options for each type of speaker.

If we use the MAX98357A board, we can simply use the Adafruit 3" Speaker. It is a 3W passive speaker. The impedance is 4 ohms, and the dimensions are 77.8mm x 77.8 mm x 25.49 mm. It costs about $1.95 and can be purchased from Adafruit (product ID 1314).

If we use the PCM5102 board, then we have two options. One option is to purchase a breakout version of the PCM5102 which includes a headphone jack socket. That way, we can easily connect any headphones to the board. These usually have their own amplifiers. However, we do not use headphones. This is for safety, convenience, and testing purposes. The other option is to first connect the PCM5102 output to an external amplifier board, and then connect that to a passive speaker like the one mentioned above.

Filter Circuits: Typically, amplifier boards also contain some filtering components. Thus, we provide a brief discussion on the types of filers that can be used. While some of these might be familiar, a few may be new to the reader. We will briefly discuss high pass, low pass, band pass, bandgap, notch, all pass, and equalizer filters. These, or more complex variants of them, may be present within the amplifier boards such as the MAX 98357A board.

As the name suggests, high pass filters allow high frequency inputs to pass with little attenuation, whereas low frequency components are strongly attenuated. While the frequency response is a smooth curve, the boundary is usually thought of at the cutoff frequency, where the input signal loses more than half of its power. This cutoff frequency depends on the resistor and capacitor values of the circuit, as well as the topology used.

Low pass filters perform the opposite job of a high pass filter. They attenuate high frequencies and let low frequencies pass to the output. The curve is also smooth, but the boundary is again thought of at the cutoff frequency, which is obtained similar to the high pass filter discussed above.

A Bandpass filter attenuates components with both high and low frequencies, and ideally does not attenuate components within a certain range of frequencies. This range is defined by the low and high cutoff frequencies, and are obtained based on the circuit topology and RC values.

The bandgap filter is the opposite of the Bandpass filter. It allows all frequencies to pass through with the exception of frequencies in a certain range which are strongly attenuated. The difference between these two frequencies, both in bandpass and band gap filters, is known as the bandwidth of the filter.

Notch filters are similar to band gap filters, however their bandwidth is very narrow. They are almost ideal band gap filters.

All pass filters are filters that do not necessarily attenuate any frequency components, i.e they let all frequencies pass through, but they can modify the phase relationship between the frequencies. The circuit can be used as a frequency selective phase shifter circuit.

An equalizer filter can selectively attenuate, amplify, or let pass inputs with certain frequency components. This allows the designer to correct for certain losses incurred during passage through the channel.

All these filter types enable circuit designers to produce sound with phenomenal quality despite channel losses and other distortions and disturbances that occur in the real world.

3.2.8 COMMUNICATION PROTOCOLS

HUD: The HUD communicates with the central control unit (CCU) in a half duplex manner. Image data is sent from the HUD to the CCU for further processing to determine the point of impact between the cue stick and the ball. Other than that, post- and pre- shot data, including ball strength, and audio guidance information will be sent from the CCU to the HUD for output purposes. Because the HUD is intended to be a low-power device (it will be battery powered and needs to last an entire game. Additionally, it is wearable and will make direct contact with the user's skin, so temperature is a constraint), low power communication protocols should be used. A suitable candidate is Bluetooth Low Energy (BLE). If complexities arise from BLE (due to the need to switch between client and server roles), Wi-Fi may be considered.

The HUD microprocessor requires wired communication with several subsystems, mainly the audio, camera and display systems. Regarding audio, the HUD microprocessor provides I2S audio data output to a DAC board. This is then connected to a speaker to output auditory guidance information for the visually impaired user. In addition to that, the HUD microprocessor needs to maintain an SPI connection to an onboard SD card. The SD card contains the audio files from which the microprocessor will output auditory guidance. Regarding the Camera system, the microcontroller needs to maintain both an I2C and an SPI channel with the camera. SPI is used for the transmission of actual image data, and I2C for programming the camera and passing instructions. The display is also run using the I2C protocol. To keep it simple, rather than utilizing a single I2C bus and working with the protocol's multiple slave device procedure, we use the ESP32's second optional I2C bus by reconfiguring the GPIO pins.

Cue Stick: The cue stick needs to transmit its orientation and force data to the central control unit. It needs to send the force data to the HUD in order to enable the camera in order to capture the point of contact. The ESP32 has three methods of wireless communication: Wi-Fi, BT, and BLE. As not much data is going to be sent, all of these protocols have more than enough bandwidth. The main consideration for choosing a wireless communications protocol would be power consumption. As BLE consumes the

least power and has a latency that is low enough for our purposes, it will be the protocol that the cue stick will use.

On the cue stick, the microcontroller was required to have at most two wired data connections. As our development boards have built-in wireless communications, one of those connections is not needed. It is important to note that if we were to switch to a board without a built-in adapter we would need to use UART as that is the protocol most wireless cards use. The remaining connection would be to the sensors in charge of orientation and force data. As we see in the following section, the majority of these sensors can either use I2C or SPI for communication. Due to I2C using two wires, as opposed to SPI's four, I2C will be the communication protocol to interface with the sensors.

Glove: The glove transmits its angle as well as proximity data to the central control unit. The ESP32 has three methods of wireless communication: Wi-Fi, BT, and BLE. As not much data is going to be sent, all of these protocols have more than enough bandwidth. The main consideration for choosing a wireless communications protocol would be power consumption. As BLE consumes the least power and has a latency that is low enough for our purposes, it is the protocol that the glove uses.

On the glove, the microcontroller was required to have at most two wired data connections. As our development boards have built-in wireless communications, one of those connections is not needed. It is important to note that if we were to switch to a board without a built-in adapter we would need to use UART as that is the protocol most wireless cards use. The remaining connection would be to the sensors in charge of determining angle as well as proximity. As we see in the following section, the majority of these sensors will end up using I2C to communicate with the microcontroller.

3.2.9 PROGRAMMING LANGUAGES

A programming language is the means by which programmers communicate with computers. They consist of a set of instructions that are designed to perform a task. In the next two paragraphs, the terminology that is required to discuss programming languages will be explained.

There are two methods of converting human-readable code into machine code: compilation and interpretation. Compiled languages are converted directly into machine code. This results in faster and more efficient execution when compared to interpreted languages. Compiled languages also give the programmer more control over the hardware. There are a few drawbacks to using a compiled language. The first is that compiled languages must be manually compiled before execution; recompilation is required every time a change is made. Another drawback of using a compiled language is that the machine code generated is platform dependent, meaning that running the program on a different platform would require recompilation. In interpreted languages, the instructions are not directly executed by the process, rather the instructions are read and executed line-by-line by another program known as an interpreter. Interpreted languages are usually more flexible; the use of an interpreter makes the code platform

independent. The main drawback of using an interpreted language is that runtime is slightly worse.

Another attribute of a language is its readability, the ease with which a language is read and understood without special training. This is determined by how close the programming language is to the human language. Programming languages have varying levels, how close they are to communicating directly to the silicon of the processor. The lower the level of the language, the closer it is to the hardware. The various levels of programming languages will be explained below. Afterwards, the languages available for the embedded and central control applications will be discussed.

Binary/Machine Code: The lowest level programming language is binary known as machine code. The language consists of ones and zeros; the task performed by the processor depends on the configuration of these ones and zeros. These series of numbers are easy for the processor to understand, however, it is hard for the average human to interpret. Therefore, machine code is only used in very specific applications and requires special training.

Assembly: The next level above binary is assembly. Assembly is essentially just binary that humans can easily understand; an instruction in assembly can be directly translated into binary and vice versa. Since assembly is chip-specific, programmers must be familiar with that chip's instruction set. As assembly is very low-level, performing a task takes more lines of code when compared to higher-level languages. This also means that developing a program in assembly takes considerably more effort and time for most programmers. Even though time may be lost during development, assembly makes up for it in execution time. The greatness of this advantage has been lessened, however, due to modern compilers being much more optimized.

Low-Level: Above assembly are low-level languages. These languages are very versatile and can be used in many environments and on various platforms. As was alluded to in the previous paragraph, low-level languages need compilers to translate the code into instructions that the processor can understand. This comes with the downside of the program not being able to dynamically change its operation. On the other hand, the readability of the language and an instruction set that is not chip-specific make low-level languages an optimal choice for many applications.

High-Level: At the very top are high-level languages. These languages have the highest level of abstraction from the mechanisms of the processors. This makes the languages much more understandable to programmers, allowing for faster development times. This convenience comes at a cost as high-level languages tend to be less memory efficient. Another advantage of using a high-level language is that the programs that are written are portable, meaning that they can be run on multiple systems.

Embedded Application: There are over three hundred languages that programmers use worldwide, however, due to the hardware limitations of embedded systems, the number of languages at our disposal becomes much smaller. The applications running on our

embedded systems will be required to be able to send the data from the peripherals attached to the microcontroller to the main controller.

Embedded Language Options: As was mentioned in the section discussing the ESP32, we have multiple methods for programming the boards: the ESP-IDF extension in VSCode, the Arduino IDE with the ESP32 Arduino Core, and MicroPython. In the following paragraphs, we will look at the pros and cons of each platform and decide which to use moving forward.

The Espressif IoT Development Framework (ESP-IDF) is the lowest level option for programming the ESP32. Using ESP-IDF would give us the most control over the board, however, in order to take advantage of the hardware, one would need to be rather familiar with the API. Fortunately, all of the sample code provided by Espressif in the documentation is targeted to ESP-IDF so learning how to use the API would not be that difficult.

The Arduino IDE with the ESP32 Arduino Core is the mid-level option for programming our development boards. Setting up the Arduino IDE for ESP32 developments is much simpler than the setup process for ESP-IDF. By using the Arduino IDE, we also would have access to a large catalog of libraries and documentation that would assist in the development process.

MicroPython is the highest level option for programming the ESP32. Using MicroPython would allow us to write simple code to control the hardware as opposed to the complex low-level ESP-IDF. This makes MicroPython very beginner-friendly and would allow for faster prototyping. However, being beginner-friendly comes at a cost as MicroPython is slower than other options as the instructions would first need to be translated into instructions that the hardware can understand. This makes MicroPython a poor choice for battery-powered devices.

Embedded Language Selection: The three languages that have been considered, along with their features, are compared in Table 3.17.

Table 3.17: Embedded Application Language Comparison

Language	ESP-IDF	ESP32 Arduino Core	MicroPython
Level	Low	High	High
Compiled/Interpreted	Compiled	Compiled	Interpreted
Execution Time	Fast	Fast	Slow
Production Time	Long	Short	Longest
Readability	Low	High	Very High
Experience	No	Yes	No

Using ESP-IDF would allow us to have maximum control over the hardware. However, that would require us to become very familiar with ESP-IDF's instruction set and its idiosyncrasies, resulting in a considerably long production time. Since Arduino uses C++, using the ESP32 Arduino Core should be relatively easy as our team is already familiar

with C++. As the ESP32 Arduino Core is simply a translation layer on top of ESP-IDF, there is significantly less overhead when compared to MicroPython; using MicroPython would result in the shortest battery life. Taking all these factors into account, the decision was made to use the ESP32 Arduino Core to write our embedded application.

Central Controller/Desktop Application Language Options: The application that will run on our central control unit (CCU) does not have the same restraints as the applications that will run our embedded systems. The program will be running on a computer with a full Operating System (OS) and multiple IDEs are available to help in writing, compiling, and debugging the program. This application will require the following features: the rendering of an intuitive graphical user interface (GUI) that will be displayed on the HUD, the sending/receiving of data to/from the various microcontrollers via wireless communications, and the ability to interpret data received in order to conclude shot strength and accuracy in order to make corrective suggestions. The languages that will make the most sense for our purposes will be discussed below and compared in the following paragraphs.

C is the most popular language for use in embedded systems. However, it is also widely used in other applications as C is one of the first languages most programmers learn. It is possible to create a graphical user interface (GUI) in C using libraries such as GTK+ and IUP. C is the most low-level programming language that we discuss in this section.

C++ is an object-oriented programming (OOP) language. This means that the language is built around the use of objects and the manipulation of properties. Object-based languages allow for many convenient features such as abstraction, polymorphism, inheritance, and data encapsulation. All of these are features that would make writing our program easier. As C++ is based on C, C++ has all the features of C, but it also has other features that make memory management much simpler.

Java is another object-oriented programming (OOP) language developed by Sun Microsystems. Like C++, Java is based on objects and has the benefits that come with using objects. The main benefit of using Java compared to other object-oriented languages is that the code runs in a Java Virtual Machine (JVM). This special runtime environment loads the Java class files, verifies them, and converts them to byte code that the processor can understand. This means that Java can run on any platform that supports the JVM. Unlike the previous languages discussed, memory management is done automatically. This makes programming more convenient but makes runtime less consistent as the timing of tasks, such as garbage collection, is not controlled by the programmer.

Python is an open-source high-level language. Its design philosophy emphasizes the readability of code. This results in the language being easy for beginners to learn and allows for fast development. Python is an interpreted language. This makes Python a very portable language, however, it comes at the cost of runtime. Another benefit of using Python is that it is open-source. This essentially means that if something in computer science exists, someone has more than likely made a Python library for it.

Central Control Application Language Selection: The four languages that have been considered, along with their features, are compared in Table 3.18.

Table 3.18: Central Control Application Language Comparison

Language	C	C++	Java	Python
Compiled/Interpreted	Compiled	Compiled	Compiled	Interpreted
Execution Time	Fast	Fast	Slow	Slowest
Production Time	Longest	Long	Short	Shortest
Readability	Low	Moderate	High	High
Experience	Moderate	Low	High	Low

All of the languages discussed previously each have their own ideal applications, and all could be used to write the central controller application. However, one language must be chosen, so that will be discussed here. C, while being the language of choice for embedded applications and has the necessary features to create a GUI, will not be our preferred language as we will not need to be able to access specific memory locations. Having to manually manage memory and only having access to lower-level instructions would also make development harder. Using C++ would help as memory allocation can happen automatically and higher-level instructions would be available. However, as the central controller does not have the same processing power limitations as our embedded systems, the time difference in execution time between C++ and Java or Python would not make up for extra production time. Therefore C++ will not be used for our desktop application.

Our two remaining options were Java and Python. As was mentioned, both languages are very powerful and either could be used to develop the application that will run on the central control unit (CCU). The members of our team that would be responsible for writing this application were already relatively well versed in Java. However, due to the popularity and versatility of Python, as well as our team's desire to learn the language as it is an important skill to possess, our team decided to use Python to develop the CCU program.

4. RELATED STANDARDS & DESIGN CONSTRAINTS

In this section, we discuss the various standards and design constraints to which SCRATCH must adhere. We begin by discussing the various communications standards that can be used to communicate between components as well as entire devices. Then, we discuss various data format standards. Lastly, we discuss the design constraints that have been imposed on our team via our design specifications.

4.1 WIRELESS COMMUNICATIONS

Bluetooth Classic (BT): Bluetooth Classic, or, simply, Bluetooth (BT) is a short-range wireless technology that is used to transmit data across short distances. It does so by using UHF radio waves in the ISM bands. It was introduced in 1998 and is developed by the Bluetooth Special Interest Group (Bluetooth SIG) [4.1]. There are five major revisions of the Bluetooth standard; all versions of Bluetooth standards support backward compatibility. There are also multiple classes of Bluetooth radios. Each class has its own maximum permitted power and typical range of operation. The data rate of Bluetooth Classic ranges from 1 to 3Mbps. The protocol has a total latency of approximately 200 ms.

For our project, Bluetooth would be used to transmit and receive data between the subsystems and the central controller as well as between subsystems.

Bluetooth Low Energy (BLE): Bluetooth Low Energy (BLE) is a wireless PAN technology designed and marketed by Bluetooth SIG. Nokia developed the original specification in 2006 under the name Wibree. It was then integrated into Bluetooth 4.0 in December 2009. It is independent of classic Bluetooth and therefore has no compatibility, however, both protocols can be supported by one device. BLE uses the same radio frequencies as classic Bluetooth and is marketed for low-energy technology [4.2]. BLE is ideal for applications that do not need to exchange large amounts of data; the data rate of BLE ranges from 125kbps to 2 Mbps. The protocol has a total latency of approximately 9 ms.

BLE is more complex than classic Bluetooth. Communication channels have GATTs, which are generic attribute profiles. These define specs for data transfer, and include services and characteristics. A characteristic is a group of attributes (such as value, description, and properties), while a service is a collection of characteristics and has a 16-bit/128-bit UUID. This can be generated online easily. BLE devices have 5 states: scanning, advertising, initiating, connection, and standby. A BLE device can either be a server or a client. Data moves from server to client. It is possible for the same device to alternate between being a server and a client; however, a device cannot be both simultaneously (though time slicing techniques can be applied).

Like BT, BLE would be used in our project to allow the various subsystems to communicate. A clear upside of using BLE as opposed to classic Bluetooth is the energy savings, which would be important as one of the goals of the project is to optimize power usage.

Wi-Fi: Wi-Fi is a family of wireless network protocols based on the IEEE 802.11 family of standards. It was first introduced in 1997 and is a trademark of the Wi-Fi Alliance. Wi-Fi uses the UHF and SHF radio bands to allow nearby devices to exchange data; the bands are subdivided into multiple channels [4.3]. Wi-Fi's wavebands work best for line-of-sight use. The various standards of Wi-Fi allow for max data rates from 11 Mbps to 9.2 Gbps.

For our project, Wi-Fi would either be used in place of or supplement Bluetooth in communicating between subsystems. One downside to using Wi-Fi as opposed to Bluetooth is that Wi-Fi is less energy efficient. One IEEE paper found that Wi-Fi was thirty percent less efficient than Bluetooth for transmitting the same data [4.4].

Zigbee: Zigbee is a specification for high-level communication protocols. It is based on the IEEE 802.15.4 standard [4.5]. Zigbee was initially developed in 1998, standardized in 2003, and revised in 2006. It is currently developed by the Connectivity Standards Alliance. This standard is used to create PANs and was designed for low-power, low data rate transmissions. Zigbee has a data rate of 250 kbps and therefore is used for intermittent transmissions. For single-hop transmissions, latency is less than 60 ms.

Like the other wireless communication protocols defined above, Zigbee would be used in our project for communication between the various subsystems.

4.2 WIRED COMMUNICATIONS

Universal Asynchronous Receiver-Transmitter (UART): A universal asynchronous receiver-transmitter (UART) is a communications protocol for asynchronous serial communication [4.6]. It is one of the earliest and most commonly used hardware protocols. UART transmits individual bits sequentially using two wires. Communication using UART can be simplex, full-duplex, or half-duplex. UART is also configurable in the fact that the number of data bits, stop bits, parity, and baud rate can be specified by the user. For our project, UART would be used by the microcontrollers to interface with a wireless interface adapter if the board we selected did not have an integrated adapter.

Serial Peripheral Interface (SPI): SPI is a serial communication protocol that involves one master and a theoretically unlimited amount of slaves. In order to explain SPI, let us assume one master and one slave. Each slave has MOSI (master out slave in), MISO (master in slave out), SCLK (serial clock) and a CS (chip select). Due to the presence of a clock, SPI is considered a synchronous communication protocol. The CS line allows the master to choose which slave its communicating with. If there are four slaves, only the slave to which the master is communicating will have its CS line asserted. This shows a limitation of the SPI interface. While in theory, an SPI master could have as many slaves as is needed, each slave needs a CS wire. Adding new slaves to an existing setup then becomes challenging as it one cannot just add a wire to a chip; as such, there is typically a maximum number of SPI slaves for each SPI master.

An easy way to visualize how the MOSI and MISO lines are connected is to assume there is a shift register on each end of the two wires. In reality, there is more complex circuitry

(for example, the user can choose which edge of the clock to sample/send data on, and whether or not to ignore the first edge (clock polarity and phase) by programming a register. This requires more digital logic to implement), but the top level model can be simplified to that. When a master wants to communicate to a slave, it enables its CS line, and on each clock of the SCLK, data is shifted serially on the MOSI line. If a slave needs to respond, the response is carried in the MISO line.

SPI can support high data transfer rates, and is also a full duplex communication channel, which means that data can be going both ways at the same time. However, SPI lacks error checking and acknowledge handshakes, which facilitate undetected errors during transmission [4.7].

Inter-Integrated Circuit (I2C): I2C, IIC, Two-Wired Interface or Inter-Integrated Circuit are all names for this common serial communication protocol that is used over a wide variety of short distance applications. [4.8] This technique operates using only two bi-directional lines that are referred to as Serial Data (SDA), which transfers the actual data bits, and Serial Clock (SCL), which carries the clock signal. I2C allows one to connect either only one or multiple slaves that can be controlled by a single master or even multiple masters. This all allows for relatively simple expansion of the communication network.

According to I2C protocols, the data line will only change when the clock line is high. Additionally, because of the ability to have multiple devices, each device must have a unique slave address. The communication process is as follows. With the SDA and SCL lines starting high, the master will first send a start condition in the form of turning the SDA line low. This will be followed by the target slave's 7- or 10-bit address and the read/write bit. Each slave device will compare the address sent by the master with their own address, and if the two match, a return "acknowledge" (ACK) bit will be sent back to the master and the SDA line will be left high and ready for data transfer. The data transfer can begin and an 8 bit block, with an acknowledge bit in between every 8 bits until the stop condition is sent to or from the slave. The stop condition is done by switching the SCL high before the SDA is also switched to high.

This process is done in a synchronous manner with a half-duplex layout. What this means is that while the two lines are bi-directional, and data can be sent both to and from the slave or master, the communication can only occur in one direction at a time and must be done in the form of "frames" or blocks. An example of these frames, as well as a general layout of the data transfer protocol is shown below in Figure 4.1.

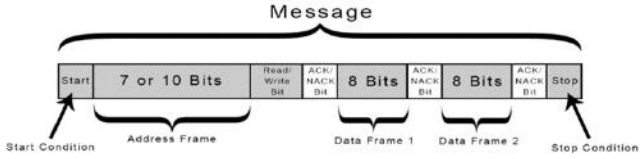


Figure 4.1: Display of I2C message protocol. Reproduction permission requested from Circuit Basics

Universal Serial Bus (USB): USB, or Universal Serial Bus, is an extremely common communication protocol used in a large variety of applications. [4.9] With the utilization of additional drivers and appropriate hardware, it can even transmit enough information to support audio or video. Like I2C and UART, the data is framed into various packets that can be used to initiate or end communication, transmit data, or acknowledge/reject data. Modern day USB 3.0 can handle data rates of up to 5 Gbps and power levels of up to 4.5W. This communication standard allows for both transmitting data as well as power to devices, making it an extremely enticing option for wired applications.

High-Definition Multimedia Interface (HDMI): High-Definition Multimedia Interface, or as its more commonly known, HDMI, provides both audio and digital video communication between devices. [4.10] Of the connectors 19 pins, most are dedicated towards the Transition Minimized Differential Signaling (TMDS) protocol. This encoding provides protection to the signal in one part by having each of the twisted pairs include the data itself and an inverse copy of the data. The bandwidth of HDMI is 4.9 to 10.2 Gbps and can support up to 120 frames per second. This communication interface would allow for an extremely nice display to be used on the HUD, the major downside however, is because of the extremely high data rates, it must be wired directly to the system.

Serial Camera Control Bus (SCBB): Serial Camera Control Bus is a serial, synchronous communication protocol developed by OmniVision (OV) for use on their cameras. SCBB is a two-wire interface with a master-slave combination.

There are two pins: SIO-C and SIO-D. SIO-C acts similar to a clock. When the bus is idle, SIO-C is driven by a logical 1. When a transmission is to start, SIO-C is pulled low by the master. After the transmission of 1 bit, SIO-C is driven high again. A bit can only be transmitted when SIO-C is low, and SIO-C goes high to indicate bit transmission. As we can see, SIO-C behaves very similarly to a clock; however, transmission is level triggered rather than edge triggered. SIO-D is a bidirectional data line that can be driven (only when SCIO-C is low) by either the master or slave. When the bus is idle, SIO-D is floating (utilizes a tri-state buffer circuit).

Data is organized into phases (similar to packets). Each phase has 9 bits. There are three possible types of transmission in SCBB: three phase write, two phase write, and two phase read. In the three phase write cycle, the first phase has the address of the slave (similar to the first packet of an I2C transmission). The second phase has the subaddress of the register to be written within the slave, and the last phase has the data to be written. The other two cycles are used for reading. First, a two-phase write cycle containing the address and subaddress of the slave and register respectively is sent to the slave. The slave replies with a two phase read cycle containing the address of the slave and the read data [4.11].

SCBB is used to configure the camera, and is basically a modified version of I2C.

Digital Camera Media Interface (DCMI): DCMI stands for Digital Camera Media Interface. This is a communication protocol or standard that is used by many image

sensors and their onboard MCU's (such as the OV5640 and its onboard STM processor) to pack the image data into a desirable format and then send the data via DMA to a memory device. It is an intermediate layer between the image sensor and memory. DCMI interface has 17 pins: a pixel clock, horizontal sync, vertical sync, and up to 14 data pins. The horizontal and vertical sync carry information about the pixel data position, while the data pins carry the actual pixel data [4.12].

Before continuing on DCMI, we must first examine the basic concept of horizontal and vertical sync as this constitutes the basis of image processing and control. If we think of the image as a rectangular frame, with x and y coordinates, we can imagine that as a processor attempts to display an image, it displays the data pixel by pixel, going from left to right in a horizontal line. Once the processor reaches the end of the screen, the pointer moves to a new line and resets to the left. This is derived from the operation of cathode ray screens that were popular in the past. The horizontal sync signal is activated when the pointer reaches the end of a horizontal line, essentially indicating the end of a line. Once the pointer reaches the end of the final line of the screen, the image is complete, and this marks the end of the frame. At that point, the vertical sync signal is triggered. Thus, we can see that the horizontal and vertical sync signals are used to indicate the end of a line and the end of a frame respectively, and thus carries valuable information regarding the positioning of the pixels.

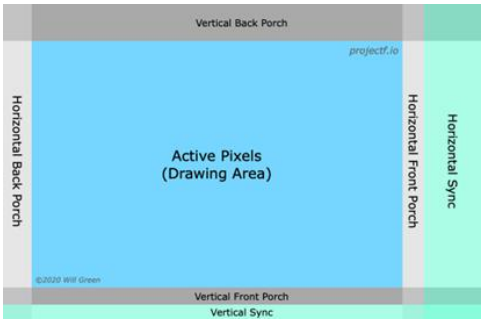


Figure 4.2: Display protocol timing regions. Reproduction permission requested from Will Green.

Returning to DCMI, the interface consists of four main stages: a synchronizer, a data extractor, a FIFO, and a final 32-bit register to hold image data to be sent over DMA.

The synchronizer is responsible for controlling data flow throughout the DCMI block. It controls the other three blocks.

Data extractor simply extracts the data in the 14 data pins and passes it to the FIFO.

The FIFO is implemented to adapt to different data rate transfers.

The 32-bit register contains the final formatted data to be transferred through DMA. The frequency of valid data depends on the DCMI width used. If 8 bit DCMI is used, each byte is packed in the 32 bit register, from LSB to MSB. Thus, the 32 bit register is ready in 4 cycles. If 10 bit DCMI is used, the 32 bit register is divided into two words, with the

10 LSB's of each word being meaningful. Thus, data is ready every two clock cycles. This is the same for 12 and 14 bit DCMI usage.

To reiterate, data from the image sensor is passed to the processor through the DCMI interface, which utilizes the synchronizer, FIFO, data extractor, and the 32 bit register. From there, image data is sent to memory through DMA. DCMI can also be used to format the image data into compressed JPEG format.

Direct Memory Access (DMA): DMA, or direct memory access, is a way to transfer data to and from memory without involving the CPU. This can free the cpu while the memory access is taking place. From a high level perspective, a CPU initially configures the DMA controller (it gives the DMA controller the address to be written, address of the IO bloc, type of operation, the number of bytes, and some other control signals) and then requests data to be placed into a buffer in the IO controller. From there, the CPU can continue doing its tasks, while the DMA completes the data. The DMA sends a request to the IO controller to write the buffered data to memory. When the first byte is complete, the IO controller sends an acknowledgement to the DMA controller. DMA then decrements the number of bytes to be sent. If the number is still not zero, the DMA repeats the process, i.e it sends a request to the IO controller to write buffered data to memory and waits for another acknowledgement. The process repeats until the number of bytes to be sent becomes zero. From there, the DMA controller generates an interrupt that is sent to the CPU to inform it of the data transfer.

Note that the DMA, CPU, memory and IO are all on the same bus. In order to prevent bus contention, DMA goes through a process called cycle stealing. In this process, the DMA requests hold on the bus. CPU then acknowledges the request by tri-stating the bus. During this process, the output from the CPU to the bus goes through a tri-state buffer in its OFF or disconnected state, which means that the CPU is effectively disconnected from the bus. The DMA can then freely control the bus without contention. All this is done through hardware, and so effects on CPU performance are minimal [4.13].

Inter-IC Sound (I2S): I2S is a communication protocol designed by Philips Semiconductors, now NXP Semiconductors) in the 1980s as a standard interface to facilitate the transport of data relating to digitized audio between different integrated circuits, analogue to digital converters and digital to analogue converters. Inherently, I2S is a two-channel protocol, which makes it ideal for communicating stereo sound data. Typically, there is an I2S transmitter and an I2S receiver. There are three lines between the two devices:

- SCK: the clock, for synchronization.
- WS: word select, which determines which audio channel, right or left, is currently being transmitted. WS transitions one clock cycle before the completion of the transmission of one data word
- SD: serial data, where the actual data is driven. This clock runs continuously

Note that for the SD line:

- There is no agreed upon word length between the transmitter and receiver. While this increases the hardware complexity, it allows I2S to be much more efficient and thus versatile
- Although new data can be sent on any clock edge, new data can be read only on the positive edge of the clock
- There are no unused clock periods between words, which increases efficiency

I2S emphasizes high transfer speeds for clear and high quality audio. It is a point to point protocol, not intended for use in large networks of devices [4.14].

4.3 OTHER STANDARDS

IEEE 754: IEEE Standard for Floating-Point Arithmetic, or IEEE 754, is a technical standard for floating-point arithmetic set in 1985 by IEEE. The most current version of this standard was published in July 2019 [4.15]. The purpose of this standard was to address multiple issues that came from having a diverse floating-point implementation which resulted in less reliability and portability. This standard defines arithmetic formats, interchange formats, rounding rules, operations, and exception handling.

A floating-point format is specified by a base, also called a radix, *b*, which can either be represented in binary or decimal, a precision value, *p*, and an exponent range from *emin* to *emax* with *emin* = 1 - *emax* for all IEEE 754. A format defines finite numbers, positive and negative infinity, and quiet and signaling NaNs.

For our project, this standard should be followed as the data that will be sent from the cue stick to the CCU is likely going to consist of floating-point numbers. Therefore, it is vital that we choose some standard format to follow in order to ensure the data is encoded and decoded properly.

MPEG-1 Audio Layer III (MP3): MP3 (more officially MPEG-1 Audio Layer III) is a format used to store and transmit digital audio. It was established by the Fraunhofer Society in Germany, but received wide support from entities within the United States and also internationally. MP3 actually supports multiple data rates and a variety of channels. One of the main advantages of MP3 is data compression. With regards to this topic, MP3 utilizes lossy data compression to encode the data. Data is partially discarded and approximations are made, which is the basis of lossy compression. The main principle behind lossy compression is to discard information that may not necessarily make a conspicuous, observable difference for the end user. For example, lossy compression of an image may discard certain colors that are not distinguishable by the human eye. Lossy compression of audio may involve discarding certain frequencies that are above the user's hearing range, or frequencies that cannot be distinguished by the human ear. In fact this is precisely how MP3 compression works. It discards information that are beyond the hearing capabilities of the majority of users. Besides that, MP3 also utilizes the fast four transform (FFT) and the modified discrete cosine transform (MDCT) to store the relevant auditory information.

For our project, the MP3 file format is one of the options that we can use to store the audio clips that will be used as guidance and feedback for the visually impaired user. Having compressed files would be useful as it would save space on whatever media we decide to use; however, it would not make that big of a difference as our audio files will not be that large to begin with [4.16].

Waveform Audio File Format (WAV): Waveform Audio File Format, or WAV, is an audio file format developed by IBM and Microsoft; its initial release was in August 1991 [4.17]. It is the primary format used for uncompressed audio on Microsoft Windows. It is an application of the Resource Interchange File Format for storing data in chunks. Files can be tagged with metadata; this metadata is inserted into the INFO chunk. The main limitation of WAV is that files are limited to 4 GiB. This is due to the files using a 32-bit unsigned integer for the file size header.

For our project, the WAV file format is the other option that we can use to store the audio clips that will be used as guidance and feedback for the visually impaired user. Having uncompressed files most likely would not make a difference in our project as the sound system likely will not be sophisticated enough to play the clips accurately enough to hear the difference.

4.4 DESIGN CONSTRAINTS

Ergonomics of the Glove: The glove needs to be limited in size due to the ease of use. The glove is only used for the visually impaired user to help direct them so they are in position to take a shot. One main reason for the need of a small light weight design of the glove is to limit the difficulty of maneuvering the hand. If this glove is too heavy due to the electrical components on it, it will make it harder to move accurately. This directly hinders our performance of needing to have accurate movement in order to match the angle that is necessary to take an accurate shot. We have defined our angle of error to be five degrees off of the given angle needed for the shot, if the glove is too large we can not guarantee that this design constraint can be met. Another reason we need to have a compact size for the glove is due to impacting the user's shot. Since the visually impaired person can not guarantee that they can place the cue stick in a position that won't impact the shot, we must take some design precautions. If we are able to limit the size of the glove electronics, we would be able to minimize the percentage of times that the user would be impacted when taking a shot. We also wanted to make sure these electronics do not get damaged by a shot taken.

Ergonomics of the Cue Stick: In order to minimize the effects that our hardware has on the user’s shot, the components that will be on the cue stick must be as small and lightweight as possible. The additional weight added on the stick must also be balanced as to not influence the user’s shot in a certain direction. If either of these values is not followed, it cannot be guaranteed that the user’s shot will be recorded properly. It is also important that the electronics and the enclosure housing them do not extend too far from the shaft of cue for that same reason. We want this product to mimic a realistic game of billiards to the best of our ability since this will be used to help improve the skills of a player. If this cue stick is significantly different than a regular stick, this may actually

cause the user to perform worse when playing in a regular game of pool without SCRATCH. Therefore, we have specified that the sum of the masses of all components on the cue stick must be less than 1000 grams.

Ergonomics of the HUD: For the users comfort and safety, the HUD must be designed with size and ergonomics in mind. It must be balanced and secured enough to be able to stay on the user's head as they walk around the billiards table and play the game. This may have to be done using straps and other structural supports, but the device must stay secure. Additionally, the design must remain as small as possible to remain non-bulky. The device should weigh less than 1500 grams and not protrude more than 2" out from the users head. These two constraints will assist in guiding and providing goals to reach when creating the mechanical structures and designs for the HUD. If the HUD exceeds 1500 grams, prolonged use (which is expected by users as they will use our SCRATCH system for both training as well as playing the game) can cause neck damage over long periods of time. This must be avoided, and can be easily achieved, as mentioned, by the limit on weight and protrusion.

Performance of the HUD: The performance constraints of the HUD revolve mostly on the speed of feedback and battery capacity. With regards to the battery capacity, the HUD should remain charged for a minimum of 30 minutes without needing to be recharged. Once recharging is required, it should be relatively simple to change out the rechargeable battery. With regard to the feedback speed, it must interrupt the flow of the game of billiards as little as possible. To achieve this, the feedback should be obtained within a few seconds of impact with the ball. This includes the time to send the image back to the main computational unit, process the image, and send the feedback back to the HUD to be displayed. This also includes the time to process the speed at which the shot was taken at, send it to the main computational unit via Bluetooth, and send the feedback to the HUD to be displayed.

Visibility of HUD Display: The HUD's display must be able to clearly display the information required in focus in front of the user's eye. This information includes both pre and post-shot information such as cue ball impact point, shot selection, and shot strength. As a result of the reflections and other optical characteristics of the HUD design, measures must be taken to ensure that the reflected image does not have double images, is in focus, and is visible in a standardly lit room. These constraints of not having a double image, maintaining focus, and being visible are all heavily important towards the maintaining image clarity, which is extremely important towards this project's success as this is how the information will be conveyed to the user.

Temperature: For the user's safety, the temperature of the components must be considered and maintained to a safe level. While the computations done on the HUD's peripherals are very low and should not require high processing power, with these components being placed in a structure on the user's head, this must be ensured. This is an equally important decision for the glove as it is also worn, and directly touching the user's hand. Prolonged exposure to excessive heat can cause both short term and long term damage. Even if temperature is not excessive enough to cause permanent or temporary damage, small increases in temperature are enough to cause discomfort to the

user, which will likely limit user training and usage time. This defies the whole purpose of the project, which is to provide modern, convenient, and effective ways to train both visually impaired and non visually impaired users in the game of billiards.

Economic Constraints: This project is being funded entirely by the members of our senior design team. As this project also has humanitarian and philanthropic purposes (introducing the visually impaired user to the billiards community and providing more options for them in regards to activities and potentially professions), it is implied that the cost per unit should be within the budget of the average American, more specifically the average visually impaired person. According to recent studies, the median income of the visually impaired user is $37, 195 annually (American Foundation for the Blind). This is an objectively small amount, and thus our design must be more cost effective. Due to not having any outside funding we need to choose design options not only based on performance but also based off of price. These restrictions cause the project to not be as robust as possible since we would be unable to get the most expensive or top of the line components. This means when creating our design we must take into account the actual price and weigh our options accordingly.

Design Time Constraints: As this project will only occur over two semesters worth of time, it is important that the goals and end product of our project be set realistically. Under ideal circumstances, we would be able to create multiple iterations of our design, incrementally improving our design based on any undesirable behaviors that reveal themselves during testing and normal use. However under our current circumstances, this is not possible. The solutions to the problems that we look to solve may not be ideal or completely refined, but they should be sufficient and well-thought out enough to solve the problem.

Computational Time Constraints: It is important to realize that our system operates in real time. SCRATCH provides aiming guidance and assistance to the visually impaired user as well as the non-visually impaired user. In professional billiards, a shot must be taken within thirty five seconds, however, taking a shot in a reasonable amount of time is required regardless of the level of professionality of the games. This is due to clear and obvious practical reasons. If a player takes more than two minutes per shot, the game is considered moot. It is practically a waste of time for all parties involved. As a result of this problem, it is necessary to ensure that all the computer vision, data communication, data processing, and Input/Output (IO) is completed within two to four seconds, allowing both the visually impaired and non visually impaired user to make professional, time efficient and accurate shots. This will contribute to the overall purpose and goal of this project which is to make the game of billiards more accessible and enjoyable to the non visually impaired user, and more importantly, the visually impaired user. It will also assist in the user's development in the game as training would take significantly less time and effort.

5. HARDWARE DESIGN AND MODELING

For each peripheral of the system, with the exception of the central control unit, a 3D printed enclosure will be used to house and protect the components. These systems include the ones on the user's head with the HUD, on the cue stick, and on the glove. Each system has different components, uses, and constraints, and these will be carefully considered when designing them.

Fabrication Comparisons: There are many different options to consider when choosing how to build and what to build the enclosure for the HUD and related components out of. Ideally, if one was to do this how tech giants such as Google or Apple does it, the enclosure would be made out of high quality plastics and carbon fibers. These, while they allow for lost cost mass production, have a large initial cost and are difficult to use when designing prototypes or one-off products. For these purposes, 3D printing polymers or machined metals are the preferred materials of choice and as such are the ones that will be considered for this project.

Below a variety of materials used in both 3D printing and machining will be compared to determine which is best for this application. The most important parameters for the HUD are weight, ease of use, and price. The first section will talk about machining with aluminum. Then the following sections will discuss only 3D printed options.

Machining: The tried-and-true technique for manufacturing, prototyping, and designing components to parts as old as time has been metal. Aluminum is most popular for its very low weight while maintaining high strength when compared to other materials. Additionally, it is very easy to obtain and affordable. Its drawback however for the purposes of making the HUD is the requirement of machining. Machining uses expensive lathes and mills that can perform many tasks, however, as the details become finer and finer, one has to make important design considerations to ensure that the machine's tooling can access the various points. The most common metal used for machining is Aluminum 6061-T6.

Plastic 3D Printing Materials: By far the most common material type used in 3D printing today is plastics. It is extremely diverse, easy to use, and generally very affordable. The plastic products are used with FDM (Fused Deposition Modeling) printers, which are also some of the most common and affordable printers available as well. These printers rely on heat to melt the plastic, allowing the material to flow and fuse together with itself layer by layer into the 3D design of your choice. The most common types of plastics are Polylactic Acid (PLA), Acrylonitrile butadiene styrene (ABS), Polyvinyl Alcohol Plastic (PVA), Polyethylene Terephthalate (PET/PETG) and Polycarbonate (PC). New technologies have enabled carbon fiber weaving into the plastic filament as well, enhancing strength without increasing the weight of the product at all, and in some cases, even decreasing said weight, this of course comes at a more expensive price point however. [5.1]

Powdered 3D Printing Materials: Powdered 3D printers are becoming more and more common in the industry as a result of their high strength, detail, and finish qualities when

compared to some of the other options. The way these printers work is by printing the entire model one layer at a time. Much like with plastics, heat is used, but rather than a single extruder laying out each individual layer, lasers can be used with nylon. This greatly increases the accuracy and speed of this material. For powdered materials, the most common are Polyamide (Nylon) and Alumide. [5.1]

Resin Based 3D Printing Materials: The last 3D printing style discussed here, and the least common of the three, is the use of resin. Resins come in a large variety of options, ranging in transparency, color, and finish. When compared to other materials, resins are often less flexible and weaker. They are often also much harder and messier to work with, requiring alcohol baths and exposure to UV light to reach their final mechanical and physical properties. They do however, produce extremely smooth surfaces and are desirable in cases where intricate detail is required and strength is not the highest priority. [5.1]

Comparisons and Selection: In the table below, the physical properties of some of the most popular and readily available options are listed. The properties that are most interesting for this application are the density, the tensile strength, the flexural strength, the cost, and for some 3D printing applications, the minimum recommended layer height. The density is obviously important due to the fact that the user will be wearing all of this on their head and it needs to be supported by glasses. The lower the weight of the overall model, the better. The tensile and flexural strength are both measures of how easily the material will break after it had been printed and cured. Tensile strength, break refers specifically to the maximum stress that a material can be stretched or pulled before it breaks. Flexural strength is the force required to break the object when a force is applied perpendicular to its longitudinal axis, which is a fancy way to say, bent. For this application, these two strength parameters are not that important as there should be no large forces applied to any models used in this project. Leaving the two most important parameters being the density and the cost.

The last possibly unclear property shown on the table is the minimum recommended layer height, which applies specifically to the plastic materials that would be dispensed on an FDM printer. This measurement is exactly what it sounds like, and the main reason it is interesting to this application is the smaller the layer height, the cleaner the print will appear. Each of these materials cost and properties measurements were obtained from one website and may vary across suppliers, however the trends in comparison would remain the same.[5.2] Additionally, this pricing is based on a quantity of 1 kg, which should be more than enough for this application.

Table 5.1: Comparison of Building Materials

Material	Density (lb/in^3)	Tensile Strength, Break (PSI)	Flexural Strength (PSI)	Minimum Rec Layer Height (mm)	Cost
Aluminum 6061-T6	0.0975	45000	47000	-	6
PLA	0.045	8100	16679.3	0.25	32
PLA (CF)	0.046	6961.81	12908.4	0.25	64
ABS	0.038	6091.58	11022.9	0.25	32
PETG	0.045	6526.7	10442.7	0.25	32
PETG (CF)	0.048	8122.11	11603	0.25	48
Nylon	-	10007.6	15954.2	-	100
Resin	0.043	7432	9427.45	-	34

Of the above options, the only ones that will be considered are the plastic, FDM based ones. This is entirely based on the fact that FDM printers are much easier to use and more readily available than machining, powder based, or resin based techniques. Machining requires heavy equipment, requires much more waste, and is generally much more dangerous and as a result expensive than any of the 3D printing options. Nylon and resin based systems are also much more expensive than their plastic counterparts without offering enough of a considerable increase in performance.

This leaves only the plastic options of PLA, PLA (CF), ABS, PETG, and PETG (CF). For this application, the extra cost of the carbon fiber weaved reels is not worth it. Of the remaining three, PLA is the strongest and has the same density as PETG. This leaves PLA and ABS as the only contenders and since PLA is more readily available and strong, it will be chosen.

Threaded Inserts: One major aspect that will affect the overall fabrication of the housings is how everything will be held together. For machined parts, one can simply use a tap to install threads into drilled out holes. This, however, does not work with 3D printed models. The primary reason is that the forces required to cleanly tap a hole are too large for 3D printed materials to hold up to, and then further, when tightening screws/bolts into 3D printed material, because their material is stronger than the models, it can lead to damage to the threads or entire model. To combat this, threaded brass inserts can be utilized and are very simple to install [5.3]. First, when modeling the device, one needs to include the desired fastener locations. Then once printed, one can use a soldering iron and a pair of pliers, and then using 3D printing materials low heat deflection they can press the insert into the desired location by holding the insert steady with the pliers and pressing straight down with the soldering iron set to high heat. This process will allow for much cleaner and more robust models as glue or other cheap adhesives will not have to

be used. These adhesives, while easy, are a much lazier, messier, and less effective way to design a part and should never be relied upon, especially in cases such as this where the housing will likely need to be assembled and reassembled many times during the testing phase.

Figure 5.1*:* Example of a threaded insert. Reproduction permission requested from CNC Kitchen**.**

Acetone Smoothing: If the chosen 3D printing material used in the project is ABS, the process of acetone smoothing can be implemented. [5.4] This creates an extremely high quality finish to the final product, resembling the look and feel of a high quality plastic injection molded part. This is extremely desirable especially when compared to FDM printers typical rough, matte finish with visible layer lines. Acetone is a solvent that can be used to dissolve only the topmost layers when used carefully, completely eliminating the roughness of the part. This chemical is quite dangerous and needs to be handled with care since it is very flammable, and the fumes can cause symptoms. For this reason, nitrile gloves, goggles, and a well ventilated area will be required. Two of the most common methods involve vapor smoothing or simply applying the acetone with a brush. The latter method is exactly as one would expect, simply dip the brush into the acetone and apply evenly onto the surface. The downsides however, are the risk of uneven application leading to uneven and lower quality finish than found with vapor smoothing. This technique is easier to perform than it initially sounds. One simply has to place the 3D print in an enclosed space surrounded by paper towels dampened with acetone. The acetone will evaporate very quickly from the paper towel and the vapor that forms will even break down the top layer of the 3D print, leaving a very glossy and smooth final product. This method is the one that will be utilized for this project.

HUD Modeling: When starting a model to design, the way that works best for a lot of people is to entirely plan out and imagine the final product on paper or in one's head. In this case, most of the designs are relatively simple. The simple portions of the designs are associated with the physical housing that will contain all the peripherals. The more complicated portion of the designs involves securing the housing to its location, whether this be on the glasses for the HUD, or onto the pool stick. There will be four different

modes to create, these being both the pool stick's and the HUD's multiple housings and latching mechanisms.

Solidworks: Solidworks is a Computer-Aided Design and Engineering (CAD and CAE) software that is used across the world by engineers to model, simulate, and assemble parts ranging from a single bolt to an entire car assembly. [5.5] For this project, almost none of its large array of capabilities will be utilized, however, this software or at the minimum a software like this is crucial to every engineer's arsenal regardless of their specific line of work. Its purpose in this application will be to initially design the housings for each component. Then, test fits can be done utilizing the assembly function. Manufacturers often will provide 3D models for engineers to use in a CAD software of their choice to ensure that the parts all fit together. After all the modeling and test fitting is done, the file can be exported and sent to the 3D printer as an STL file.

Component Weights: Another driving design characteristic is the overall weights of all the components. Depending on this total, additional straps or other supports may be required to maintain the ergonomics of the system. Additionally, if the weight is low enough, magnets or other simply mounting structures could be employed. The weights of each component, 3D structure (Including non electric support hardware ie. mirrors), and the side of the head that the component will be mounted on is summarized in the table below.

Table 5.2: HUD Component Weights

Component Weights		
Component	**Weight**	**Location**
ESP32	9.6g	Right
Display	4g	Right
Right Side Structure	40g (Estimated)	Right
Speaker/DAC	15g	Left
Camera	6g	Left
Left Side Structure	40g (Estimated)	Left

HUD Housing: For the HUD, there will be two very similar housings, one on each side. On the right side of the glasses, will be the housing for the display and related components. On the left side of the glasses will be the camera. Both housings will be initially designed as boxes with the covers being separate printed components. This ensures a couple of things, firstly being structural rigidity. By modeling the design to print the base and all the walls in one piece, one can avoid unnecessary fasteners. The cover will need to be designed to be removable, and such adhesives will be avoided at all costs because this team consists of good engineers who do not take shortcuts with things such as glue. The alternative of course, is fasteners such as screws and nuts. The layout of the housings are dictated almost entirely by the component sizes. Therefore, in the final SolidWORKS assembly, 3D models of each component will be used to test fit before

printing. In the front of the housing, there will only be the mirror set to a 45 degree angle, while at the rear will be all of the components behind the display.

Latching/Mounting Mechanism HUD: The most difficult 3D model to design will be the mounting mechanism for the HUD's housing onto the glasses, whether this is a separate component of the assembly, or built directly into the housing. One major source of inspiration is the system used by the VUFINE+ Wearable Displays.[5.6] This, shown below in Figure 5.2, uses a two-part system that makes it compatible with any pair of glasses. The way this mount works is by sliding a rail over the arm of any pair of glasses, and then tightening it down. Next, using a magnet, the rest of the housing is secured. Depending on the total weight of the HUD system, a strong magnet could be a great option, however, as the magnet gets stronger, they may also interfere with the electronics within the device.

Figure 5.2: Universal HUD mount example. Reproduction permission received from Vufine.

Cue Stick: With regard to the cue stick, the IMU and related positional and strength finding components will need to be housed directly on the stick for maximum accuracy.. These components, like the ones found on the HUD, are relatively sensitive and need to be fully enclosed to be protected. A box will be designed and mounted to the opposite end of the cue sticks striking point and this is where all the cue sticks peripherals will be. Additional design considerations will be listed as they are encountered, but for now, the key points remain identical to the HUD. This will be interacted directly with the user and button placement and overall ergonomics need to be considered when doing the design.

Glove: The glove will consist of a microcontroller and a positional component. This component, working in conjunction with those found on the cue stick, will also need a housing to be protected from incidental ball/cue stick contact as well as just general usage by the user. The design will be similar to that of a low profile computer mouse, and fit in the palm of the user's hand to ensure accuracy and ease of use while playing the game of billiards.

6. SOFTWARE DESIGN

Since a significant portion of SCRATCH's scope was software-based, it is important that our team put some forethought into the decisions we made. This section begins with a discussion of the various IDEs that were at our disposal. Choosing the correct IDE was important as the majority of our development time was spent using the IDE. We then discuss different version control methods that were used to ensure the security of our files. Last, we outline how the software in each individual subsystem functions.

6.1 DEVELOPMENT ENVIRONMENT

IDE: An integrated development environment (IDE) is an application that combines many developer tools into a single program. Most IDEs contain a source code editor which is used to write software and a debugger which is used to test and run the program. IDEs vary in complexity and feature sets; they range from simple text editors that have color-coding that allows for easier code organization to full IDEs that have code completion and extensions. It is important to choose a code editor/IDE that matches the needs of the project.

Embedded Application IDE: After selecting the ESP-WROOM-32 as our microcontroller of choice and deciding to use the ESP32 Arduino Core to program it, our choice of IDE became very obvious. While it is possible to use these Arduino libraries in other IDEs, the Arduino IDE already has native libraries and processes for connecting the computer to the microcontroller development board. This will make it much easier to communicate with the board and start developing our programs. In addition to this, there are many example projects, a large Arduino community, and various external libraries that can be used that would assist in application development.

Central Control Application IDE Options: After selecting Python to be our language of choice, we then had to decide on what development environment to use. Unlike our embedded application, we have numerous options for tools that could be used for production. In the following paragraphs, we will discuss and compare the software available to us, then we will choose one to use for developing the central control application.

Notepad is a simple text editor that is built into Windows. Its primary use is to create and edit plain text documents. If Notepad were to be used, it would supplement our primary development software as Notepad lacks basic features that would aid in development.

IDLE, short for Integrated Development and Learning Environment, is an IDE that is installed by default when installing Python. IDLE is intended for beginners and is cross-platform. Its main features include a multi-window text editor, syntax highlighting, code autocompletion, a built-in Python shell, and an integrated debugger. IDLE is lightweight and simple to use, but it is not optimal for larger projects due to a limited feature set.

Atom is a free, open-source code editor developed by GitHub. Like other IDEs, it has basic features such as syntax highlighting, code auto-completion, and a debugger. Some advantages of using Atom are that its user interface is fully customizable and the program itself is very well documented. Some disadvantages of using Atom are that it is not fully optimized, uses excessive amounts of system memory, and has comparatively high latency. Like IDLE, it is not suitable for handling large projects.

PyCharm is an IDE developed by JetBrains. Among its many features, PyCharm has syntax highlighting, live code verification, and auto-completion. JetBrains provides three versions of PyCharm: Community (open-source and free), Educational (open-source and free), and Professional (proprietary and paid). The advantages of using PyCharm are that there is an active community that provides support and that it can execute, edit, and debug Python code without any additional installations. The disadvantages of using PyCharm are that projects have a long loading time and that settings may need to be adjusted to open existing projects. Another possible issue with using PyCharm is that tools developed by JetBrains, which is based in the Czech Republic, had been used by hackers to access confidential systems in the United States.

Visual Studio Code (VS Code) is an IDE developed by Microsoft. VS Code is built on the same framework as Atom, Electron. This means that VS Code includes the same features as Atom. Additional features include code refactoring and embedded Git. Support for Python is done through the simple installation of a plugin. The main advantage of using VS Code is that it is extension-based, meaning that features can be added to the IDE with a click of a button. Keeping this in mind, the main disadvantage of using VS Code is finding the extension you need due to the vast library of extensions.

Eclipse is an IDE developed by the Eclipse Foundation. Eclipse was designed for the Java language, but Python support can be added through the installation of Pydev. Like the other IDEs discussed, Eclipse features a debugger and basic syntax highlighting. Eclipse is designed for working with large projects, however, the main disadvantage of using it is that Eclipse has a tendency to be slower than other IDEs.

Central Control Application IDE Choice: The code editors/IDEs that have been considered, along with their features, are compared in Table 6.1.

Table 6.1: Comparison of Python Development Environments

IDE	Notepad	IDLE	Atom	PyCharm	VS Code	Eclipse
Syntax Highlight	No	Yes	Yes	Yes	Yes	Yes
Code Completion	No	Yes	Yes	Yes	Yes	Yes
Execute Code	No	Yes	Yes	Yes	No (Yes w/ Plugin)	No (Yes w/ Plugin)
Debug Code	No	Yes	Yes	Yes	No (Yes w/ Plugin)	No (Yes w/ Plugin)
Save Code	Yes	Yes	Yes	Yes	Yes	Yes

Notepad was immediately discarded as an option due to its lack of basic development features. As for the rest of the IDEs, the decision came down to personal preference as each had the same basic features. Visual Studio Code has been chosen to be our team's development environment of choice. The main reasons for choosing VS Code is the ability to install extensions that will assist in developing the central control application, the integration of Git that will help with software version control, and our team's familiarity with VS Code.

6.2 SOFTWARE VERSION CONTROL

Version control is the practice of tracking and managing the changes made to code. Keeping track of all the changes is important as it allows developers to go back to a previous version of the software if a mistake is made. Some types of version control allow for multiple team members to work on the same project.

Local Version Control: Local version control is the simplest and first form of version control. It consists of copying files into a new directory in order to denote software versions. This approach is very common due to its simplicity, however it is very prone to errors. It is easy to forget which folder is being edited, resulting in changes being made to the wrong files. Local version control is a viable option when a single person is editing code, but it is not suited for development teams as changes are difficult to track.

Centralized Version Control: The next form of version control is centralized version control. Centralized version control systems (CVCSs) were developed in order to allow for collaboration between multiple systems [6.1]. These systems have a central server that contains all the versions of the program and clients check out files from that central server. Clients then check the modified files back into the server. The advantage of CVCs is that clients can see what everyone else is doing to a certain degree and administrators have control over client permissions. The main disadvantage is the lack of data redundancy. Having a single point of failure means that if access to the central server is lost, no one can check out or check in files. If the project is lost on the central server, the entirety of the project is lost except whatever files were saved on local machines.

Distributed Version Control: The final form of version control was designed to solve this data redundancy issue [6.1]. In distributed version control systems (DVCSs), clients mirror the full repository whenever checking files out as opposed to the snapshots checked out with CVCSs; this means that every client has the full history of the project. If any of the servers with the project dies, any of the client repositories can be used to restore the server. Due to our team's desire for multiple developers and the desire for data security, we are using a DVCS. Some popular options are Git, Mercurial, Bazaar, and Fossil, all of which will be discussed and compared.

Git is a free distributed control system (DVC). It was originally created by Linus Torvalds in 2005 for the development of the Linux kernel. Git's use in the development of Linux led to the implementation of the following features [6.2]: support for non-linear development, distributed development, compatibility with existing systems, support for large projects, authentication of history, multiple merge strategies, and garbage

accumulation. The main feature of Git is that it uses snapshots for directory trees. These features have helped make Git the most popular version control system [6.3]. When used in combination with GitHub, developers can collaborate anywhere in the world.

Mercurial is an open-source distributed control system. Mercurial is ideal for beginners as the syntax for commands is simpler and the documentation is easier to understand. By default, a user cannot change history in Mercurial. Mercurial used to be widely used, but now it only holds two percent of the VCS market [6.3].

Bazaar is a free distributed control system. Bazaar's main feature is its cross-platform GUI. This GUI makes Bazaar beginner-friendly. Bazaar has support for renaming files and directories, making merging less problematic. The development of Bazaar appears to have ceased as the last stable release is from 2012.

Fossil is a free and open-source distributed control system. It has similar features to Git and Mercurial as well as support for bug tracking, forums, and email alerts. Fossil has a built-in web interface that provides "everything" needed for software development [6.4]. One unique aspect of Fossil is that it supports an "auto sync" mode that reduces the forking and merging found in distributed projects.

Version Control System Selection: As shown above, there are a variety of distributed version control systems that are available to our team for the development of SCRATCH. Due to our team only consisting of four members and the relatively small size of the software that will be managed, local control systems would suffice. However, it would be important to have cloud backups of our project in the case of a catastrophic occurrence. Therefore, Git has been chosen to be our primary method of version control. When used with GitHub or other repository managers, our team would be able to have a public repository on the cloud that can be pulled and pushed to at will. Our team's previous experience with using Git/GitHub for other academic projects was a significant contributing factor in our decision.

6.3 EMBEDDED SOFTWARE DESIGN

6.3.1 HUD SOFTWARE

The following is a flowchart of how the HUD software operates:

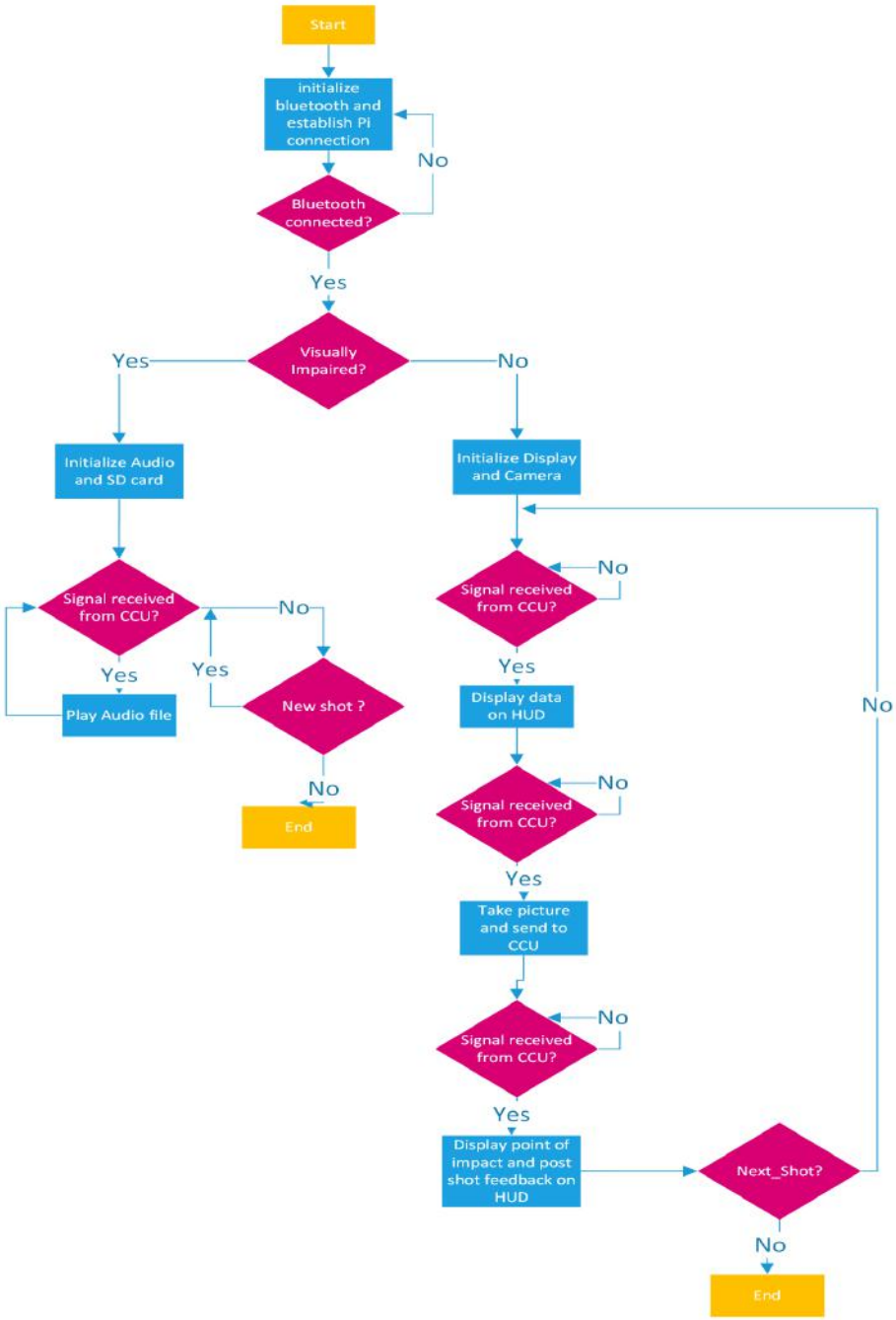


Figure 6.1: Flowchart of HUD Operation

As can be seen from the diagram above, the HUD and CCU form a master/slave configuration, with the CCU being the master, and the HUD being the slave. Connection

is established via Bluetooth Low Energy (BLE), and data is sent in notification mode to save power (this allows the bluetooth link to sleep in between data transfer). Initially, the CCU informs the HUD whether or not we are in visually impaired mode. This allows the HUD to initialize the proper subsystems.

After that, in visually impaired mode, the CCU repeatedly sends trigger signals to the HUD, which will force the HUD to play specific audio files to guide the user in aiming. The audio files play corresponding to the numerical input received from the CCU. This continues as long as the user attempts new shots, otherwise the program ends.

In non-visually impaired mode, the CCU sends data to the HUD to tell it what to display. These involve the aiming point and shot power data. The CCU then identifies the time at which the user attempts to shoot and immediately send a trigger signal to the HUD, which prompts the HUD to take a picture and sends it to the CCU via bluetooth. From there the CCU performs some image processing on the received image to determine the point of impact of the cue with the ball, and send coordinates to the HUD so the HUD can display the feedback data. This process repeats until the user stops attempting to make new shots, and the program ends.

Image processing involves simple color filtering and transformations, circle detection, and glare detection.

6.3.2 CUE STICK SOFTWARE

The microcontroller and the accompanying hardware on the cue stick has two main functions. The primary function of this subsystem is to collect and transmit the orientation and acceleration of the cue to the central control unit (CCU). The secondary function of this subsystem is to allow the user to control the central control application without the use of any other external peripherals.

Initializing the System: The first step of the initialization process is to start the connection to the central control unit. After the connection to the CCU is made, the next step is to initialize the peripherals connected to the cue stick's ESP32, which in this case is simply the BNO055. This starts by setting the baud rate at which the IMU will be read. The software then checks whether the IMU is connected. If the BNO055 is not connected, the cue stick's microcontroller will send an error message to the CCU and the cue stick's program will stop. If the IMU is connected, the program will continue.

After the IMU is verified to be connected, the program will then set the correct ranges for both the accelerometer and gyroscope. The filter bandwidth for the gyroscope will also be set at this time. After these parameters are set, the stick will enter calibration mode. Calibration requires the user, or in the case of the visually impaired, their assistant, to place the cue stick on the surface of the billiards table. After a predefined time period, the cue stick is leveled out with respect to the table.

After the calibration process is complete, the cue stick enters standby mode. In standby mode, the cue sends button data to the CCU.

Button Data: As was mentioned in the overview of this section, the cue stick will be the primary means by which the user will control the central control application. In order to conserve battery life, the cue stick's ESP32 will enter low-power mode. Whenever a button is pressed, an interrupt is raised to wake up the ESP32. The ESP32 then waits for a very short period of time to check if that button was released. During this period of time, the ESP32 also checks if other buttons are being pressed. After that period of time passes, if the initial button was determined to no longer be pressed, the cue stick's ESP32 sends a packet to the CCU containing the button data.

If the initial button is still being held, the ESP32 will recognize it as being held and will begin a timer. This timer will be set to run for approximately three seconds. If the button is held for the entire three seconds, the ESP32 will send the CCU a special message that signifies that the cue stick will now enter "shot mode". The contents of this message begins with the header of standby mode (00), followed by the remaining bits set to '1'.

0	**0**	**↑**	**↓**	**←**	**→**	**A**	**B**	**0x0**		
31	30	29	28	27	26	25	24	23	…	0

Figure 6.2: Format of Standard Standby Message

0	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0xFFFFFF16**		
31	30	29	28	27	26	25	24	23	…	0

Figure 6.3: Format of Special Switching Message

Motion Data: When in shot mode, the cue stick's ESP32 alternates between sending orientation data and force data with status codes 01 and 10, respectively. The microcontroller receives this data from the attached MPU-6050 by requesting each angle and the force sequentially.

The ESP32 first sends the roll, pitch, and yaw to the CCU as a series of 9-bit unsigned integers. This message begins and ends with the status code "01". The microcontroller then sends the force applied to the Z-axis in the IEEE 754 "binary16" standard. This message is denoted by beginning and ending with the status code "10".

0	**1**	**Roll**			**Pitch**			**Yaw**			**0x1**		
31	30	29	…	21	20	…	12	11	…	3	2	1	0

Figure 6.4: Format of Orientation Data

1	**0**	**±**	**Exponent**			**Fraction**			**0x2**		
31	30	29	28	..	24	23	…	14	13	…	0

Figure 6.5: Format of Force Data (binary16)

At the end of every cycle of reading from the IMU and transmitting to the CCU, the microcontroller checks whether any button has been held for three seconds. As was the case in standby mode, a button being held for three seconds is used to switch operating modes; the same special message is sent to the CCU in order to exit shot mode and enter standby mode.

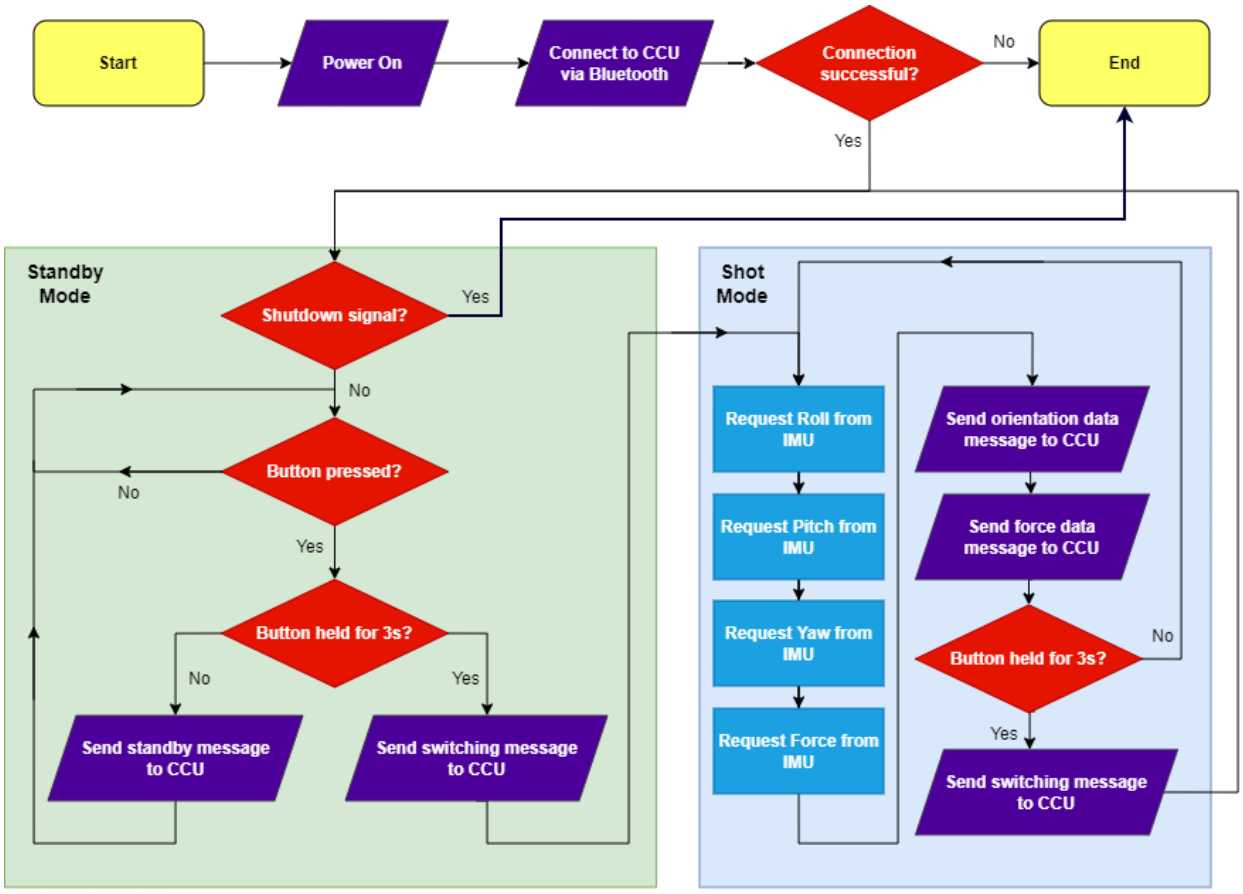


Figure 6.6: Flowchart of Proposed Cue Stick Operation

Final Implementation of Cue Stick Software: The final method for having the cue stick send data was much simpler than was originally planned. Instead of performing bitwise operations, multiple BLE characteristics were used. The data for the orientation, acceleration, and buttons are sent one after the other to the CCU for processing. The buttons were consolidated to a single characteristic each button being mapped to an integer value. The finite state machine for the cue stick was also made more complex, with the final FSM having five key states, “NOT_READY”, “READY”, “WAITING”, “TAKING_SHOT”, and “SHOT_TAKEN”.

6.3.3 GLOVE SOFTWARE

The following is a flowchart on how the glove software will work.

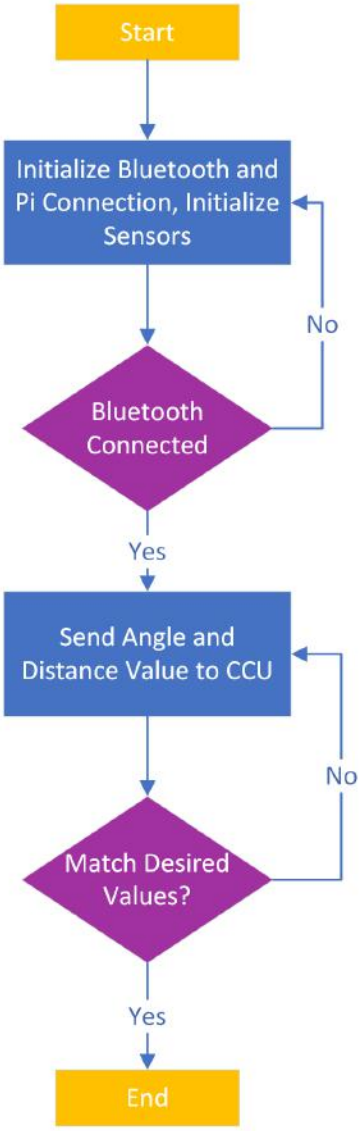


Figure 6.7: Flowchart of Glove Operation

The glove software is the most simple out of all of the subsystems due to it being very sequential. The glove takes the data that is received from the different sensors then compare this with a preset specific value and then if this matches it moves to the next step, the main interaction that this has with the CCU is to match the angle that is received from the VISION team to see if the reading of the current users angle matches, the calculation of the whether the angle is correct will be done on the CCU. Once the angle is matched we then check the distance to see if this is the desired value on the CCU but the glove constantly just sends these two integers for the CCU to read.

6.4 CENTRAL CONTROL APPLICATION DESIGN

Overview: As the name suggests, the application running on the central control unit (CCU) will be responsible for communicating and controlling the various subsystems in SCRATCH. It will also be responsible for receiving and interpreting the data received from the VISION team. Among the various features that this application has are the ability to maintain user profiles, training modes in which the user can improve specific components of their shot, and a game mode in which the user is guided through a real game of billiards.

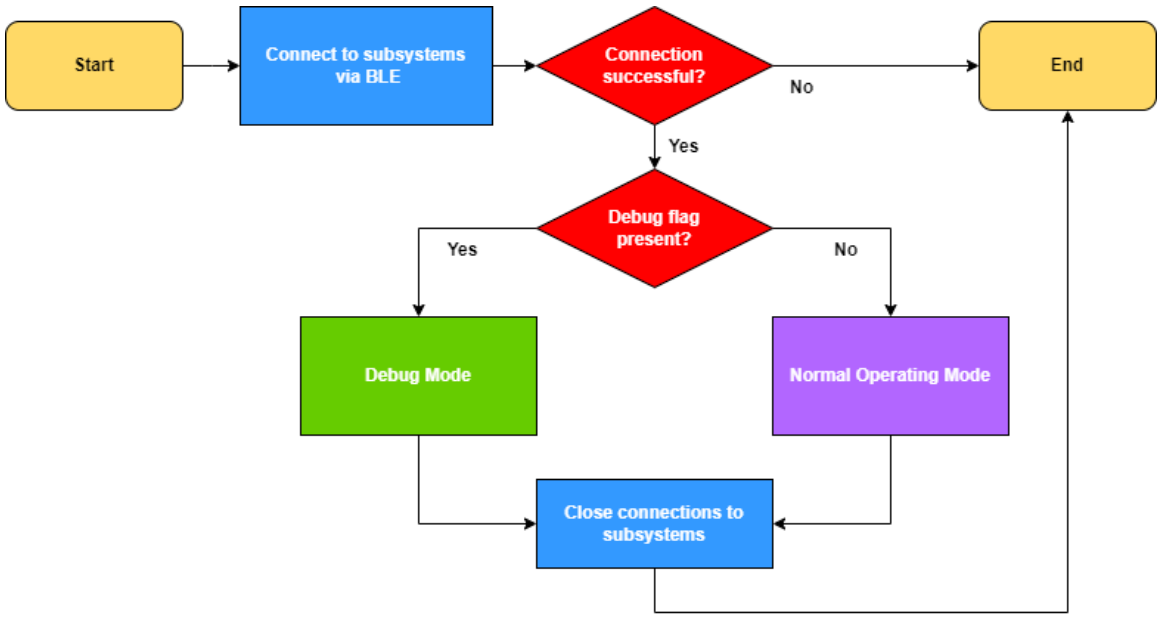


Figure 6.8: Abstracted Flowchart of Central Control Unit Operation

Communicating with SCRATCH Subsystems

Cue Stick: For most operations, the CCU acts as a master device to the cue stick subsystem. The CCU will either receive button data or motion data from the cue stick depending on which mode the cue stick is in. There are only two occasions in which the CCU would send data to the cue stick, both of which will be explained in the following sections.

When the cue stick is in standby mode, the CCU receives a message whenever buttons are pressed. The central control application parses through this message and checks if any of the buttons pressed matches an action for the current state of the program. If it does, then the corresponding action is taken; if not, the message and its contents are discarded. When the CCU receives a switching message when the cue stick is in standby mode, the central application prepares to begin receiving and storing shot data. This process includes clearing the memory in which the shot data will be stored.

When the cue stick is in shot mode, the CCU continuously receives messages about the current orientation of and force applied to the cue stick. For efficiency purposes, the central control application does not store all this data; the program is designed to only hold five seconds of shot data. Rather, the program only begins to store data once it has determined that the user is about to attempt a shot and stops storing once an attempt has been made. The processes by which the central application identifies a shot being taken are explained in depth in a later section. After a shot is taken, the CCU sends the cue stick a switching message so that it will transition to standby mode. If the user were to have cancelled their shot and prematurely switched back to standby mode, the CCU would have received a switching message signaling that change.

Determining a Shot Attempt: To optimize memory use and maximize computing power, the CCU's application only stores data for what it considers a shot attempt. The beginning of a shot attempt is recognized by application when both the orientation and forward force applied to the cue stick remains relatively constant for two seconds. The system determines that a shot attempt has been completed after the cue stick has decelerated after a spike in force.

Determining Shot Accuracy

Contact Point: The CCU will send a capture trigger to the HUD in order to capture an image right before the user makes contact with the ball. After that, the HUD will send the image to the CCU. The CCU will perform image processing to determine the point of contact between the stick and the ball. This will involve two steps: the first is to crop the image to focus on the ball. The second is to detect the laser pointer on the ball and determine its coordinates. The coordinates will then be sent to the HUD for display, and the user can see where he hit the ball versus where he was supposed to hit.

Strength: Finding the strength of the shot attempt is one of the most tricky aspects of determining the shot accuracy. While raw speed and acceleration values can be recorded, these values are relatively useless to the average user. Therefore, some abstraction was necessary, resulting in seven discrete strength values. These strength values are mapped to exact speeds in the training mode. During gameplay, the current values will be compared to these stored values in order to determine how hard the user hit the cue ball.

Path: Determining the errors within the path of shot is relatively straightforward. An ideal shot is one that begins with the cue stick in a certain orientation and that orientation is kept throughout the duration of the shot. In other words, an ideal shot attempt is completely straight, with no variation on any axis. Therefore, the process for determining how accurate the path of the shot is recording the orientation of the stick during the shot and comparing it to the orientation at the beginning of the shot attempt. The CCU will then calculate the amount of error the player's shot attempt had compared to the idealized shot attempt.

Operating Modes: The SCRATCH system will have two main operating modes: Game Mode and Training Mode. SCRATCH's Game Mode will be primarily used by the visually-impaired and will use shot information from the VISION team. SCRATCH's

Training Mode, on the other hand, is intended for use by the non-impaired to hone their billiards skills.

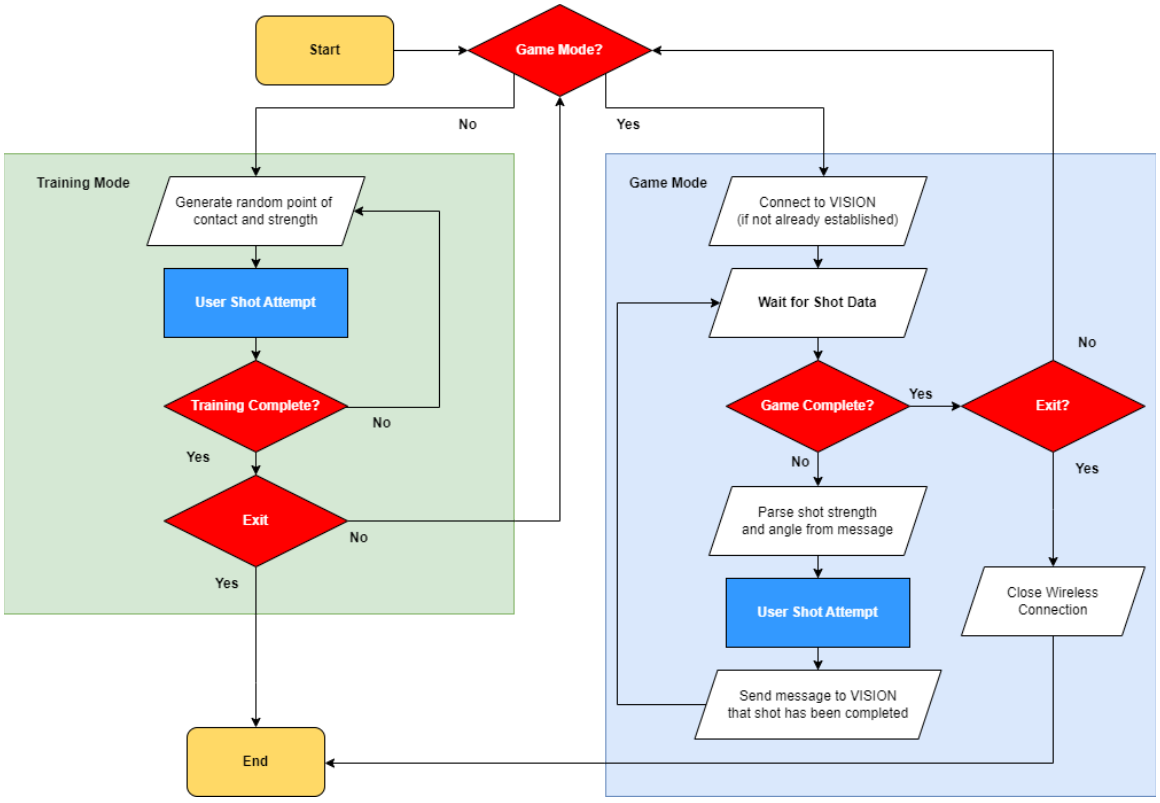


Figure 6.9: Flowchart of SCRATCH's normal operating mode

Training Modes: When training, there are two different modes: aim training and strength training. When in aim training, the user can practice hitting the cue ball on various locations, resulting in different spins. By using this training mode, the user can learn to hit the cue ball in the correct spot more accurately, resulting in better performance in game mode and in real-life billiards. When in strength training, the user can practice hitting the ball at the various strength levels that we have predetermined. The near instant feedback on the shot strength will allow the player to become more familiar with the various speeds that they will be asked to hit the ball while in game mode. This mode also has the added benefit of allowing the player to become more familiar with the way the cue ball reacts when hit at various speeds.

Game Mode: Game Mode is the operating mode in which the SCRATCH system will cooperate with the VISION system in order to guide a player through an entire game of pool. The VISION team's system will scan the table and identify the best shot for the player. Their system will then guide the player to roughly the correct position and direction to stand with respect to the table. Lastly, their system will send shot data to the CCU of the SCRATCH system and hand over control. After this, it is the responsibility of the SCRATCH system to guide the user through the process of completing a shot attempt, the process for which is outlined below.

Guiding a User Shot Attempt - Impaired Mode: The process for guiding an impaired player through a shot is as follows.

1. Have the user "zero-out" the heading of the glove and cue stick

   After VISION gets the user in the correct position, the CCU will trigger the HUD to play the audio cue instructing the user to place the glove along the edge of the pool table, such that the back edge of the glove fits into the curvature of the table. After the heading of the glove remains constant for an extended period of time, the CCU assumes that the glove is in the correct position and then triggers the HUD to play the audio cue instructing the player to place the cue in the bridge of the glove. Once the cue's orientation remains constant for an extended period of time, the CCU makes the assumption that the cue is in the correct position and sends the cue for the HUD to play the audio instructing the user to press the button on the glove. Doing so sets the current heading for both the stick and the glove to 0°, resulting in the heading of both being measured relative to the current side of the table.

2. Check if the user's hand is at the correct heading

   This check verifies that the user is facing the correct direction. This is the most important step as doing this step incorrectly would result in the user missing the cue ball entirely. If the user's hand/glove is not facing in the correct direction, the CCU will send the HUD audio system a message with a number corresponding to the proper correction audio feedback. The CCU then waits for the glove to send updated heading data. This process continues until the heading of the glove matches the desired heading.

3. Check if the user's hand is the correct distance from the cue ball

   This next check verifies that the user's hand is the correct distance from the cue ball. The CCU will receive distance data from the sensors on the glove. If the user is too far from the ball, the user will be instructed to move their hand closer via the audio cue that will be sent to the HUD; the opposite will happen when the user's hand is too close. This process repeats in five second intervals and will repeat until the user's hand is the correct distance from the ball.

4. Check if the cue stick's pitch is equal to zero

   This last check verifies that the user is holding the cue stick level to the table. The glove's bridge has been designed such that holding the cue level results in the cue being aimed at the center of the cue ball. If the received pitch value is greater than 0, the CCU will signal the HUD to play the audio instructing the user to aim lower; the opposite occurs when the pitch value is less than 0. This process repeats in five second intervals and will repeat until the cue stick is level.

5. Send audio cue for user to take the shot

Once all of these checks have been passed, the user should be aimed at the correct spot to take the shot. Therefore, the CCU will send the cue to the HUD that instructs the user to take the shot. This cue will be sent to the HUD every five seconds until a shot attempt is recognized.

6. Provide "feedback" to the user

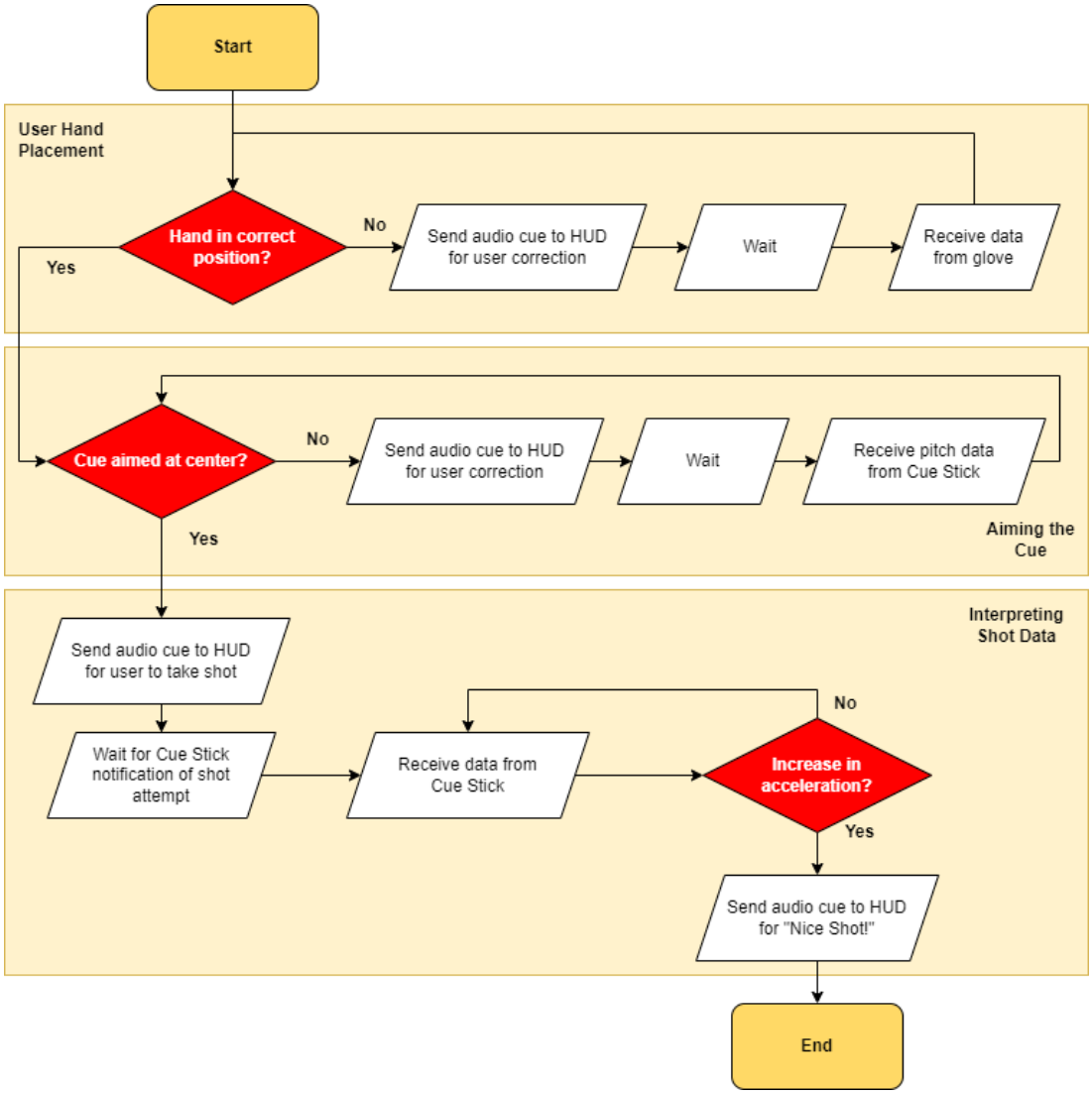


Figure 6.10: Flowchart for Guiding an Impaired User through a Shot

Guiding a User Shot Attempt - Non-Impaired Mode: When guiding a non-impaired user through a shot attempt, it is expected that the user can position themselves correctly, therefore communication with the glove is unnecessary. The CCU will receive the required force from the VISION team and the current shot will be displayed on their

display. The purpose of VISION in this case is to provide meaningful corrective feedback regarding one's shot.

1. Have the user "zero-out" the cue stick

   In order for the shot information to be accurate, it is important that the acceleration and orientation values of the cue stick be zeroed out. This is done by placing

2. Wait for user to be ready to take a shot

   In order to signal to the rest of the system that the user is about to take the shot, the user must press one of the buttons on the cue stick. This signal is sent to the CCU and primes it to receive shot data.

3. Determine whether a shot attempt is occuring:

   The process for this has been outlined previously in the subsection titled "Determining a Shot Attempt"

4. Signal HUD camera to take pictures and determine the point of contact:

   Once the CCU determines that a shot attempt is occuring, it sends a signal to the HUD to initiate its camera to take a burst of pictures. The pictures are sent back to the CCU where image processing will be used to determine the point of contact between the cue ball and the cue stick.

5. Compare desired shot with ideal shot

   The process for this has been explained previously in the subsection titled "Determining Shot Accuracy"

6. Display shot feedback

   After the CCU has determined how well the user shot the ball, it will send feedback data for the HUD to display. For contact point, the CCU will send coordinates that HUD will use to display the point of contact. For strength, the CCU will send an integer that maps to one of the strength values.

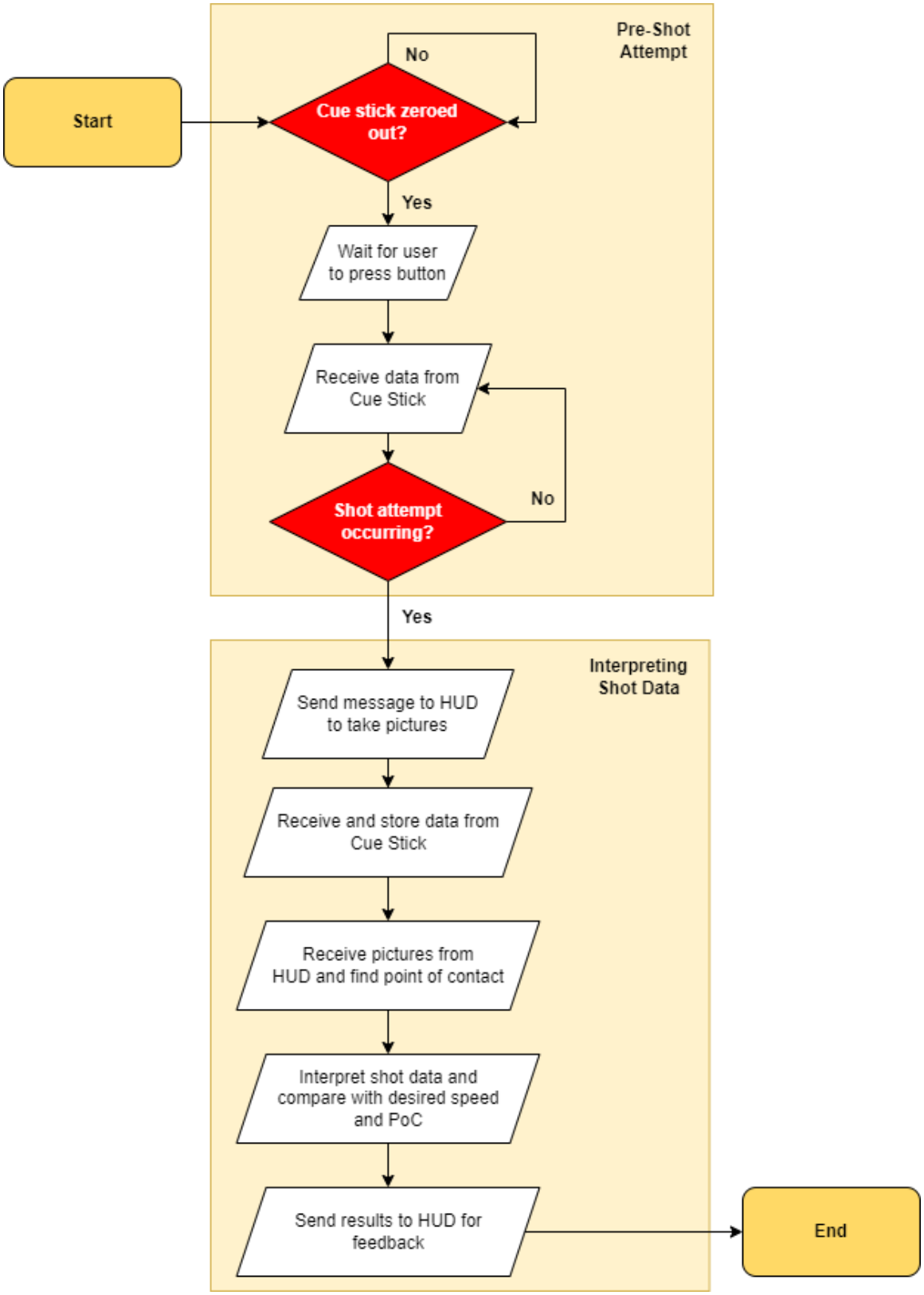


Figure 6.11: Flowchart for Guiding an Non-Impaired User through a Shot

Maintaining User Profiles: For players to track their progress, it is important that the application maintains statistics on the users' habits and past performance. As the application's save data will be stored on a microSD card and not much data will be stored, there should not be any storage capacity problems. Therefore, our application will not have a limit as to how many user profiles can be created and stored.

Within these profiles, key data regarding the player and their performance will be stored. This includes the player's name, their shot tendencies, and play history.

Shot Tendencies: Under shot tendencies, the usual contact point and strength deviations will be stored. For contact point, the direction of the deviation in which the player hits the ball, e.g., high, low, left, or right, as well as the distance of the deviation, e.g., 5 mm. For strength, the deviation, while easy to quantify, would be difficult to represent to the user in a meaningful manner. Therefore, the exact deviations will be stored in the save file, however, the displayed tendency would be more abstract, e.g., slightly too hard or soft.

Play History: The application will store various statistics about the user's play history. This includes the last time they played, the number of hours that they have played, the number of games they have played, and the number of each type of shots that the user has taken. For example, the number of shots that the user has taken with backspin will be tracked. These numbers will be shown to the player both directly and indirectly through skill levels.

The different skill levels serve two main purposes. The first is that it gives the player an indication of what skills they have been working on. The second is that the player is given goals to reach for, resulting in them using the system more and becoming better billiards players in the process.

The different skill levels are shown below in Table 6.2.

Table 6.2: Skill Levels and Requirements

Skill Level	Shots Needed	Play Time (hours)
Novice	10	0.5
Rookie	20	1
Pro	40	2
All-Star	80	5
Superstar	160	10
Hall of Fame	320	25
God	640	50

The different skills that will be tracked are shown below in Table 6.3.

Table 6.3: Skills Tracked

Skill Tracked	Name Shown	Method of Tracking
Top Spin	Off the Top	Shots
Back Spin	Bring it Back	Shots
Side Spin	Side to Side	Shots
No Spin	Front & Center	Shots
Game Mode	Running the Table	Time
Training Mode	Get It Right	Time

Save Format: The save data for each user will be saved in individual text files. While it would be possible to simply save the data as a series of attributes, it would be ideal for the save data to be readable to the average person. The beginning of each section will be denoted by brackets. Then each field will be listed, followed by an equal's sign and its respective value. An incomplete example of a save file is shown in figure 6.12.

```
[user]
name = John
total = 6
aim_dir = "left"
aim_mag = 5
strength_ab = 1.2
strength_es = "too strong"

[stats]
top = 12
back = 38
games = 12
g_mode = 2.4

[skills]
OtT = "novice"
BiB = "rookie"
```

Figure 6.12: Example of save file format

Debug Menu: To ensure that all our equipment is working properly and sending the correct data to the CCU, it is important that we have a portion of our application dedicated to testing the various subsystems. Our program will have a debug menu that can only be accessed from the title screen. To access it, the flag "--debug" must be provided when starting the program. Within this debug menu, the user is given the option to test the various subsystems.

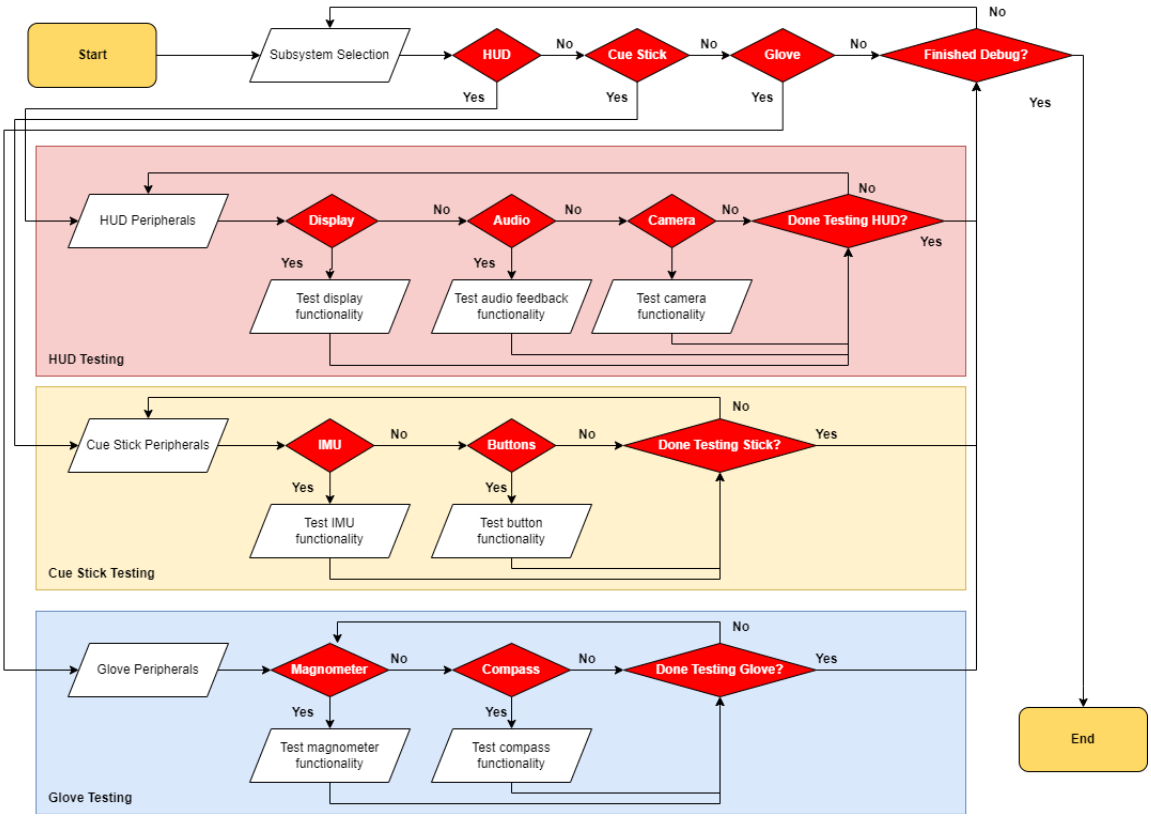


Figure 6.13: Flowchart of SCRATCH's debug mode

Changes Made to Final CCU Software: The feature set of the final version of the CCU's software is reduced compared to what we originally envisioned. Due to time constraints and issues that arrived with other subsystems, we were not able to implement user profiles. Another minor change was made in the debug menu concerning the glove's options as we replaced the magnetometer with an ultrasonic sensor and the compass with an IMU.

7. SYSTEM FABRICATION AND INTEGRATION

This section will cover the integration of the PCB components and the overall schematic of the cue stick subsystem since this was the subsystem that was made into a PCB. Before creating the schematic we must decide which PCB design software we want to use to develop this project as well as understand what to do when creating a PCB. After we establish this we can begin creating the schematic and eventually move on to building the layout of the board.

7.1 PCB DESIGN SOFTWARE

Altium: Altium is one of the more expensive PCB softwares due to its very high functionality, these prices tend to average around $4000 a year which is why this software tends to be used by companies with much larger budgets, but Altium offers an educational license which would be free as long as you are able to provide proof of being a student. Altium does include a very good management system such as being able to make adjustments to BOMs without having to regenerate every time as well being very easy to export all the required files to get a quote and board printed. Altium also offers a very understandable user interface whether that is for schematic design or layout. This tool seems to be the most powerful out of all the other PCB softwares having access to many libraries that would most likely contain the components needed for this design. Also one of the best features that Altium has to offer is the 3-D rendering of your board so that you can see the layout and how this would actually look in real life with the ability to rotate and check if everything looks as it was designed.

Eagle: Eagle is one of the most recognizable PCB softwares due to it being heavily used in academic settings. Eagle is one of the more affordable PCB design softwares coming in at around $65 a month and another benefit is that it also has an educational version that can be used by students as long as you use a university email there are still some limitations on the educational version such as board size and number of layers but these limitations should not affect our boards. A large benefit of using Eagle is its simplicity, it has a very user friendly system allowing for simple schematic designs and board layouts. Another large benefit is that eagle is very easy to import schematics and footprints of devices through ultra librarian or snapeda, and if there isn't one available for the device you would want you can technically do it yourself or have a distributor company do it for you.

EasyEDA: EasyEDA is different from the other two programs due to it being a web-based electrical design automation (EDA) tool. A very convenient feature of EasyEDA is that since it is a web browser design tool you can use it from any source, Mac, Linux, Windows. EasyEDA is similar to the other programs because it is also seen as a very user friendly program, this would allow for a smaller learning curve to design devices. Another large benefit would be the numerous libraries that are available and if a device you need is not included then you have the option to create your own, also EasyEDA also has the ability to support Altium and Eagle libraries.

PCB Design Software Selection: The best option for our PCB design would most likely either be between Eagle or Altium just based on functionality. Since both programs are free for students, SCRATCH will likely use Altium due to being more comfortable with the software as well as its extensive use in the professional environment so many resources for troubleshooting issues would be available.

7.2 PCB DESIGN PHILOSOPHY AND BEST PRACTICES

Since PCBs have become such an important role in creating electrical designs, it is best to create reliable long lasting designs[7.1]. When most people start developing a PCB, they tend to focus on the schematic first which is correct since you need to have a design in order to make a functional circuit, but it is always necessary to keep in mind how the layout will look so that future errors do not occur. No matter what design you are creating it is best to follow these five steps:

- Defining board design rules with the goal of ensuring fabrication and assembly yield
- Component placement, where the goal is to ensure solvability and ease of routing
- Grouping components by type to prevent the need to route all over the board
- Location of power and ground in the PCB stackup, including some points for mixed signal PCB layout designs
- Obeying mechanical constraints, such as connector locations and enclosure limitations

Defining Board Design Rules: When starting a new printed circuit board design, it is sometimes easy to forget about the important design rules that will govern your project. The first place to start is to talk to your PCB fabrication house. A good fabricator will usually post their capabilities online or will supply this information in a document. Once you've found their list of capabilities, you should compare these to whatever industry reliability standard you'll work with (Class 2 vs. Class 3, or a specialty standard). Once these points are determined, you should select the more conservative design layout limits needed to ensure manufacturability and reliability, and you can encode these into your board design rules. You can also adjust your design rules in order to meet certain specifications and can check if your design meets these rules by using Electrical Rules Check (ERC) and Design Rules Check (DRC).

Electrical Rules Check (ERC): The Electrical Rules check to be sure that proper connections are made to power and ground planes, signal transition times, capacitive loads and fanouts have the appropriate bounds. The ERC determines if there are any unconnected inputs or shorted outputs. Also, checks to make sure that gates are not directly connected to supplies. The Electrical Rule Checks are based upon the normal operating conditions of the application specific integrated circuit.

Design Rules Check (DRC): The Design Rules Check is a program that automatically checks to determine if a particular design corresponds and is in accordance with a set of predefined technology rules that have been given by the foundry in order to show that a design can be manufactured successfully.

Component Placement: The goal in component placement is to create a board that can be easily routed, ideally with as few layer transitions as possible. In addition, the design has to comply with the design rules and satisfy must-have component placements. These points can be difficult to balance, but a simple process can help a board designer place components that meet these requirements. First, there are often components that must be placed in specific locations, sometimes due to mechanical enclosure constraints or due to their size. it is best to place these components first and lock-in their position before proceeding to the rest of the layout. Second, components like high pin count ICs or processors generally need to make connections to multiple components in the design. Locating these components centrally makes trace routing easier in the PCB layout. Third, when components are placed in the PCB layout, the unrouted nets are normally visible. it is best to try and minimize the number of crossing nets. Each net intersection will require a layer transition through vias. If you can eliminate net crossings with creative component placement, it will be easier to implement the best routing guidelines for a PCB layout. Fourth, it is recommended to place all surface mount device (SMD) components on the same side of the board. The main reason for this arises during assembly; each side of the board will require its own pass down the SMD soldering line, so placing all SMDs on one side will help you avoid some extra assembly costs. Lastly, it is okay to rotate components to try and eliminate net intersections. Try to orient connected pads so that they face each other as this can help simplify routing.

Grouping Components: Some components are best placed in the PCB layout design by grouping them together in one area. The reason is that they might be part of a circuit and they may only connect to each other, so there would be no need to place the components on different sides or areas of the board. PCB layout then becomes an exercise in designing and laying out individual groups of circuitry so that they can be easily connected together with traces.In many layouts, you'll have some analog and some digital components, and you should prevent the digital components from interfering with the analog components. The way this was done decades ago was to split up ground and power planes into different regions, but this is not a valid design choice in modern board designs. Unfortunately, this is still communicated in many board layout guidelines and it leads to many bad routing practices that create EMI. Best practice would be to use a complete ground plane below your components and do not physically break the ground plane up into sections. Keep the analog components with other analog components operating at the same frequency. Also, keep the digital components with other digital components. You can visualize this as having each type of component occupying a different region above the ground plane in the PCB layout design, but the ground plane should stay uniform in the majority of board designs.

Location of Power and Ground/Routing Recommendations: It is generally the case that power and ground are placed on two internal layers. For a 2-layer board, this might not be so easy, so you would want to place a large ground plane on one layer, then route signals and power traces on the other layer. With 4-layer circuit board stackups and higher layer counts, you should use ground planes instead of trying to route ground traces. For components that need direct connections to power, it is recommended to use common rails for each supply if a power plane is not used; ensure you have wide enough traces

(100 mils is fine for 5 to 10 A). A good rule to follow for routing is connecting your signal traces to match the nets in your schematic. PCB layout best practices recommend that you always place short, direct traces between components when possible, although this may not always be practical in larger boards. If your component placement forces horizontal trace routing on one side of the board, then always route traces vertically on the opposite side. Although in very complex boards for specialized applications, many of the commonly-touted PCB best practices may no longer apply, and you'll need to follow PCB board design guidelines that are particular to your application.

The required trace width for different nets depends on three possible factors:

1) Manufacturability: Traces cannot be too thin, otherwise they cannot be reliably manufactured. In the majority of cases, you'll be working with trace widths that are much larger than the minimum value your fabricator can produce.

2) Current: The current carried in a trace will determine the minimum required width to prevent the trace from overheating. When the current is higher, the trace will need to be wider.

3) Impedance: High speed digital signals or RF signals will need to have a specific trace width to hit a required impedance value. This doesn't apply to all signals or nets, so you do not need to enforce impedance control on every net in your board design rules.

For traces that do not need specific impedance or high current, a 10 mil trace width is fine for the vast majority of low current analog and digital signals. Printed circuit board traces that carry more than 0.3 A may need to be wider. To check this, you can use the IPC-2152 nomograph to determine your PCB trace width for a required current capacity and temperature rise limit.

7.3 PCB SCHEMATICS

The following schematic is of the electronics that will be used for the cue stick. Among these components are the ESP32 and BNO055. For power regulation, we made a switching regulator using a TPS40303DRCR. This also includes the circuitry to include the input buttons that can be triggered by the user. This meets the design requirements of the project since the PCB is required to have a microcontroller on it to control some system such as the sensor shown below as well as the control for the push buttons, it is also required to have some sort of powering circuitry which is shown using a switching regulator circuit.

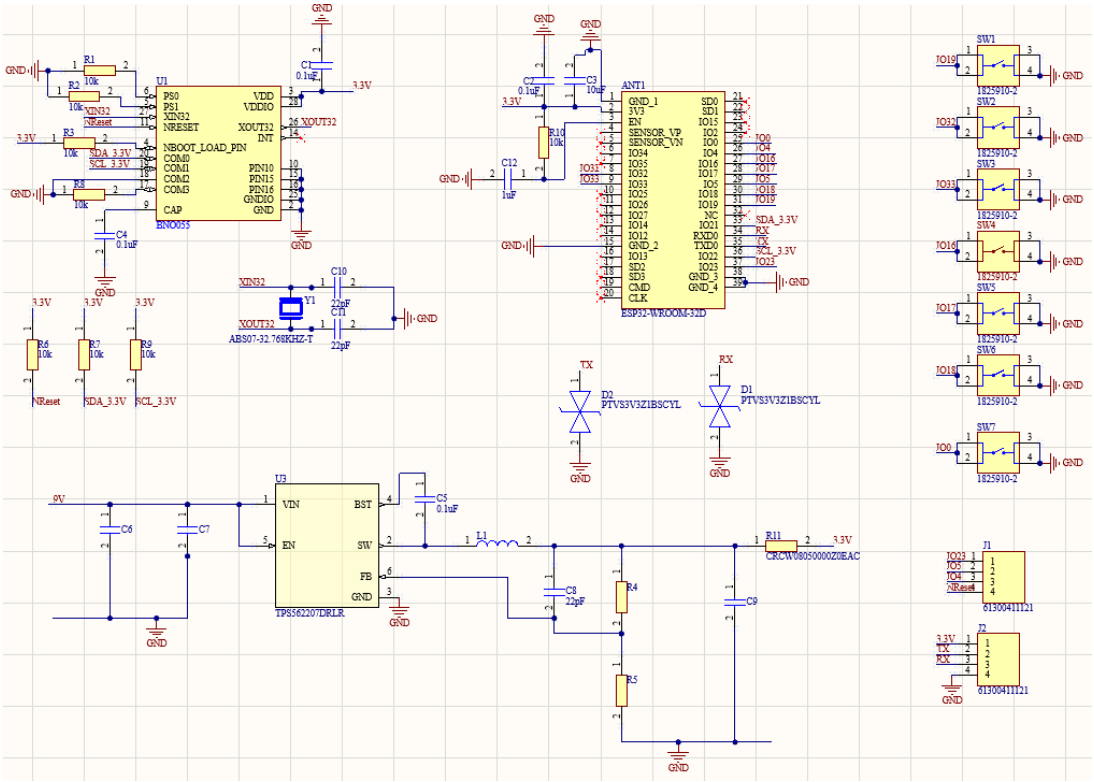


Figure 7.1: Cue Stick Schematic

8. PROTOTYPE SYSTEM TESTING

In this section, we outline the procedures to ensure that all of the hardware in SCRATCH is functioning as expected. The procedures for testing each major component and each subsystem as a whole are outlined below.

8.1 HUD SUBSYSTEM TESTING

Testing of the HUD subsystem will involve four main stages:

- Audio Subsystem Testing
- Camera Subsystem Testing
- Display Subsystem Testing
- Integration and Power Testing

Audio Subsystem Testing: The purpose of this testing is to ensure that the HUD is capable of receiving numerical input via bluetooth, and based on that input decide on a particular mp3 file stored on the SD card used, and successfully play that through the attached speakers in an audible and clear fashion. Testing should also involve reliability checks.

1. Set up the HUD ESP32 as a BLE client, and either use another ESP32 or the Raspberry Pi as a BLE Server
2. Set up the server to send random numbers to the client via notification mode
3. Setup the client to receive the random numbers and based on the number play a certain mp3 file from the SD card
4. Start both microcontrollers, and turn on the serial monitor for the ESP32 client
5. Observe the messages on the serial monitor to ensure that a connection has been established, and confirm by waiting for data receipt
6. Confirm that the the correct audio files are being played (i.e they correlate to the random number received)
7. Restart the server and check to see if the client can establish connection, then repeat steps (5) and (6)
8. Repeat step (7) 10 times
9. Restart the client and check to see if reconnection occurs, then repeat steps (5) and (6)
10. Repeat step (9) 10 times

Camera Subsystem Testing: The purpose of this testing is to ensure that the HUD is capable of immediately taking a picture based on some trigger received by bluetooth, and that the HUD can send that image to the Pi for processing, while preserving data integrity.

Basic Camera Hardware Test (this is a simple test to ensure hardware is working, and will utilize provided example code):

1. Connect the camera to power and the ESP32 as necessary
2. Follow the example code to capture and upload live video data to a server, and view from laptop
3. Ensure that image is clear

Actual Subsystem Test:

1. Properly connect ESP32 and ArduCAM and ensure sufficient power is provided
2. Open the Serial Monitor
3. Connect the ESP32 to a bluetooth server (could be another ESP32 or the Pi)
4. Through the bluetooth server, send a signal (anything that is previously agreed upon. Can be a numerical input for testing purposes) to the ESP32 connected to the Camera
5. Print the sending time in microseconds on the serial monitor
6. Print the signal receipt time in microseconds on the serial monitor
7. The ESP32 should now be preparing to take a picture using the camera. Print the time right before sending the capture command on the serial monitor
8. After the image is captured, print the time on the serial monitor
9. Begin sending the image from the ESP32 to the Pi for processing
10. In the Pi, print the receipt time for processing
11. Save the received image in human readable format, then begin processing the image
12. Ensure minimal delay between trigger signal receipt on the ESP32 and image capture
13. Ensure that the received image on the Pi is not corrupted (although a small number of pixels may be corrupted, if the image is impeccable to the human eye, it is sufficient for the purposes of this project)
14. Repeat the process ten times to:
    a. Ensure image is never corrupted
    b. Get the average delay between receipt of the trigger signal and the capturing of the image

Display Subsystem Testing: For the display, the testing initially started with ensuring that the quality of the device and the layouts of the information maintained high readability and clarity. Additionally, the power consumption was tested to ensure it matched the manufacturers specification in both the active and idle mode. Finally, the display was set to turn all the pixels on to ensure no pixels were dead.

After the display's complete functionality was confirmed. The reflected image had to be viewed to ensure that it maintained its clarity and readability. With the assistance of the first-surface mirror and reflective film on the glass, the image maintained its quality and readability.

Integration and Power Testing: The purpose of this testing is to ensure that our power system can provide sufficient energy output to the HUD, especially when the HUD is operating at the most demanding level. Additionally, testing is needed to ensure that the overall system is functionally correct, and that certain parameters, such as delay, satisfies the engineering requirements mentioned in this report.

Power Testing:

1. Ensure all the connections required in the HUD are properly maintained
2. Power the HUD using the regular system that will be used in the finished product
3. Ensure the following has been established (while these systems will never be on at the same time in the finished product, it is important to test whether our power system can safely handle it in case of glitches or bugs):
    a. Bluetooth connection with the Pi
    b. The display is on
    c. The audio system is on
    d. The camera is on and ready to capture
4. Send trigger signals from the Pi to the ESP32 in order to continuously capture image data (the camera supports burst mode)
5. Send trigger signals from the Pi to the ESP32 in order to display an image of the cue ball and a triangle showing where the user hit the ball (this is not based on an actual shot, rather just random test data)
6. Configure the display such that the triangle randomly changes location every 0.5s (this is to ensure the display draws more power)
7. Send the trigger signals for the ESP32 to start playing random audio files from the SD card
8. Ensure that steps 4-7 are done simultaneously to push the HUD to draw as much power as possible
9. Continue doing steps 4-7 for 45 minutes, monitoring the HUD and power system for any abnormalities
10. Continuously monitor the temperature of the HUD and ensure it doesn't break the 120 degrees fahrenheit threshold

Integration Testing (integrating HUD subsystems together):

1. Do steps 1-2 from power testing
2. Assume we are in the non visually impaired mode
3. Turn on the display, bluetooth and camera
4. Turn off the audio subsystem
5. On the Pi, send coordinates of where the cue stick is supposed to impact the cue ball
6. Ensure that the display correctly shows the sent position
7. On the Pi, send the required power of the shot
8. Ensure the display correctly shows it
9. On the Pi, send the trigger to take a picture
10. Ensure that picture is taken and sent to Pi for processing

11. Ensure the above steps are done with delays within the specified engineering requirements
12. Now assume visually impaired mode
13. Turn on Audio System , bluetooth, and camera and turn off the display
14. On the Pi, send random signals to the ESP32 and ensure ESP32 plays the correct audio files based on signal received
15. Repeat steps 9-11.

8.2 CUE STICK SUBSYSTEM TESTING

The following section contains the procedures for testing the various portions of the cue stick subsystem hardware.

Equipment:

- SCRATCH (Cue Stick and CCU)
- Monitor with HDMI output
- Protractor
- Level

Push Buttons: The push buttons are tested by verifying that all of the button press responses are printed to the console.

1. When starting the SCRATCH software on the CCU, use the command line argument "--debug" to enter debug mode.
2. From the available options, select "Cue Stick".
3. From the available options, select "Button Input"
4. Press each button and ensure that each button push results in the correct statement being printed.

IMU: The IMU's functionality is tested by verifying that the angle of orientation in each axis and the acceleration on the x-axis are being reported within an acceptable margin of error.

Orientation:

1. When starting the SCRATCH software on the CCU, use the command line argument "--debug" to enter debug mode.
2. From the available options, select "Cue Stick".
3. From the available options, select "IMU"
4. From the available options, select "Orientation"
5. Zero out the cue stick's orientation.
6. Use the protractor to verify the roll, pitch, and relative yaw are reported within the tolerances specified in the design specifications.

Acceleration:

1. Repeat steps 1-3 from orientation.

2. From the available options, select “Acceleration”
3. Zero out the cue stick’s acceleration
4. Using the level, point the cue stick downwards 90 degrees and verify acceleration due to gravity (9.81 m/s^2) is being reported within the tolerances specified in the design specifications.
5. Repeat step 4 with the cue stick facing upwards 90 degrees and verify the acceleration due to gravity is negated (-9.81 m/s^2).

8.3 GLOVE SUBSYSTEM TESTING

Testing for the Glove involves three main systems:

- Ultrasonic Sensor Detection of the Ball
- IMU Angle Detection

IMU: This component will be used to get the data regarding angle of the shot. In order to test this we wrote the code so that the IMU outputs the angle of the glove relative to the side of the table

1. Properly connect the IMU to ESP32, to ensure proper use of I2C communication
2. Use both IMU and protractor to measure the angle to ensure proper heading as well proper change in the angle as you rotate the device
3. Use CCU to determine if the data is being correctly sent in order to be compared to the desired value

Ultrasonic Sensor: This device will be used to determine the proximity to the cue ball, we are under the assumption that the magnetometer has worked and we are correctly aligned.

1. Set up the ultrasonic sensor on the ESP32 so that we can detect a distance to objects
2. This will trigger a flag that says to the user to stop, we can then use a ruler to measure the distance from the ball
3. Finally we test if this distance sends correctly to the CCU so it can progress in the software flow

8.4 CENTRAL CONTROL APPLICATION TESTING

The following section contains the procedures for testing the various portions of the central control application.

Equipment:

1. SCRATCH
2. Monitor with HDMI output
3. Protractor
4. Ruler

Check Connection with Subsystems and VISION: It is important to check that the communications between the CCU and the various subsystems as well as the VISION hardware. This can be done by entering the debug menu and verifying that all systems are connected

1. When starting the SCRATCH software on the CCU, use the command line argument "--debug" to enter debug mode.
2. From the available options, select "CCU".
3. The console will now list all of the systems connected to the CCU.
4. Verify that all the subsystems are present.

Ensuring Validity of Data Received/Sent: As the CCU is the main processor in SCRATCH, it is important that the data that passes through is handled properly.

Cue Stick: For testing the validity of the data being received from the cue stick, the procedures outlined in section 8.2 will be used.

Glove: When communicating with gloves, various signals will be going back and forth between the two subsystems, namely the angle data, the distance data, and a zero-out signal.

1. When starting the SCRATCH software on the CCU, use the command line argument "--debug" to enter debug mode.
2. From the available options, select "Glove".
3. Transmission between the glove and CCU will now begin. The heading of the glove, the distance from the glove to the ball, and the status of the zero-out signal will be printed to the console.
4. To verify that the heading is being reported correctly, rotate the glove 360 degrees and ensure that the values being reported change accordingly. Ensure that the change in angle is correct by using the protractor
5. To verify that the distance is being reported correctly, place the ruler between the cue ball and the glove. Move the glove along the ruler and ensure that the reported values change accordingly.
6. To verify that the zero out signal is being received properly, press the zero-out button on the glove. When the button is pressed, the console should show the signal as true; likewise, when the button is not pressed, the console should show the signal as false.

HUD: The communications between the HUD and CCU are largely one-way. The CCU will either send the HUD a number that corresponds with the correct audio file to play or coordinates that tell the HUD what to draw. The only data that will be received from the HUD is the pictures that will be used to determine the point of contact.

Communication with Audio System:

1. When starting the SCRATCH software on the CCU, use the command line argument "--debug" to enter debug mode.

2. From the available options, select “HUD”.
3. From the available options, select “Audio”.
4. The CCU will now begin sending numbers to the HUD that correspond with various audio clips. Ensure that the audio clip being played matches up with the number that is printed to the console.

Communication with Display System:

1. When starting the SCRATCH software on the CCU, use the command line argument “--debug” to enter debug mode.
2. From the available options, select “HUD”.
3. From the available options, select “Display”.
4. The CCU will start testing the display by sending coordinates to the HUD. To ensure that this communication is happening correctly, ensure that the point of contact is being displayed properly on the HUD.
5. After the CCU has cycled through all of the possible coordinates, the CCU will cycle through the seven different strength values. Ensure that the HUD displays these strength values properly.

Communication with Camera System:

1. When starting the SCRATCH software on the CCU, use the command line argument “--debug” to enter debug mode.
2. From the available options, select “HUD”.
3. From the available options, select “Camera”.
4. The CCU will send a signal to the HUD which tells the camera to take a picture and send it back to the CCU.
5. Ensure that the CCU receives the image from the camera.

9. ADMINISTRATIVE CONTENT

Below, all of SCRATCH's administrative content will be discussed in detail. These key logistical points such as timeline, milestones and budget, will be imperative to keep the team on track both timewise, and monetarily.

9.1 TIMELINE AND MAJOR MILESTONES

The SCRATCH team formed at the beginning of the summer and started collaborating and brainstorming ideas throughout the summer to prepare for Senior Design I. Therefore, the milestones for our team have been scheduled according to the progress made thus far.

Table 9.1 Senior Design I Project Documentation Milestones

Item	Start Date	Anticipated End Date	Duration
Project Brainstorming	Summer	Summer	0 weeks
Project Scope Finalized	08/22/2022	08/26/2022	1 week
Individual Research	08/22/2022	09/02/2022	2 weeks
Initial Design Document	08/22/2022	09/05/2022	1.5 weeks
30-Page Milestone	08/22/2022	09/09/2022	3 weeks
60-Page Milestone	09/10/2022	09/30/2022	3 weeks
90-Page Milestone	10/01/2022	10/21/2022	3 weeks
120-Page Milestone	10/22/2022	11/25/2022	4 weeks
Final Draft	11/26/2022	12/02/2022	1 week

Table 9.2 Senior Design I Project Design Milestones

Item	Start Date	Anticipated End Date	Duration
Individual System Design	09/05/2022	10/02/2022	4 weeks
Individual System Testing	10/03/2022	10/30/2022	4 weeks
Breadboard Prototyping	10/31/2022	11/21/2022	3 weeks
PCB Design/Ordering	11/22/2022	12/12/2022	3 weeks

Table 9.3 Senior Design II Project Design Milestones

Item	Start Date	Anticipated End Date	Duration
PCB Testing	01/09/2023	01/23/2023	2 weeks
System Integration/Testing	01/24/2023	02/16/2023	4 weeks
Practice Project Demo	02/14/2023	02/27/2023	2 weeks
Finalize Documentation	02/28/2023	03/13/2023	2 weeks
Practice Final Presentation	03/14/2023	03/20/2023	1 weeks
Final Presentation	04/19/2023	04/19/2023	30 minutes

9.2 BUDGET

The table below shows SCRATCH's current bill of materials. The quantities listed are for a completed prototype, however, extra quantities of materials may be purchased as backups.

Table 9.4 SCRATCH's Bill of Materials

Component	Subsystem	Quantity	Unit Cost	Total
ESP-32 Development Boards	All	3	$5.96	$17.88
ESP-32 Module	All	3	$8.95	$26.85
Raspberry Pi 4 8GB	CCU	1	$174.90	$174.90
Raspberry Pi PSU	CCU	1	$7.99	$7.99
Raspberry Pi Case	CCU	1	$11.59	$11.59
Ultrasonic Sensor	Glove	1	$4.95	$4.95
Display	HUD	1	$7.30	$7.30
3D Printing Materials	HUD	1	$25.00	$25.00
Mirror	HUD	1	$5.00	$5.00
Lens	HUD	1	$5.00	$5.00
ArduCAM camera	HUD	1	$25.99	$25.99
microSD card (512MB)	HUD	1	$4.95	$4.95
microSD card breakout board	HUD	1	$2.95	$2.95
MAX98357A DAC	HUD	1	$5.95	$5.95
Speaker	HUD	1	$1.95	$1.95
PCB	Stick	1	$40.00	$40.00
Laser	Stick	1	$6.99	$6.99
Adafruit BNO055	Stick/Glove	2	$29.95	$59.90
Pool Cue	Stick	1	$13.97	$13.97
Total				**$462.11**

9.3 PROJECT SUMMARY , CONCLUSION and FINAL REMARKS

SCRATCH started as a billiards training tool which evolved to a tool that assists both visually impaired and non-visually impaired users. This is done to accommodate people who would otherwise never have been given the opportunity to play billiards. This part of the project works with another senior design project called VISION which handles all the tasks regarding the pool table and the game of billiards. SCRATCH handles all tasks regarding guiding the user as well as providing feedback on performance. These projects were designed to have minimal connections between them so they can act as separate systems. In this paper we cover the overall design of the SCRATCH system and how each smaller subsystem works and interacts with each other.

To start out this paper we had to first define our goals for the project and lay out a clear path to be able to achieve these goals. This included researching different technologies that have been used in the past which relate to this project. This allowed us to much more quickly learn how these technologies function and what aspects of each we can borrow from in order to develop our design. From this research we learned that we needed to create 3 different subsystems that interact with a central control unit in order to connect all these systems.

The first one of these subsystems is the cue stick. This is the system that we used to create our PCB, the goal of the cue stick is to determine when the user took a shot and what power they used to achieve this. We chose a variety of sensors (shown in the document above) that will be on the cue stick that will implement these goals. The cue stick also relays live aim information to assist with aim guidance for the visually impaired. Lastly, the stick also includes the controls for starting our system.

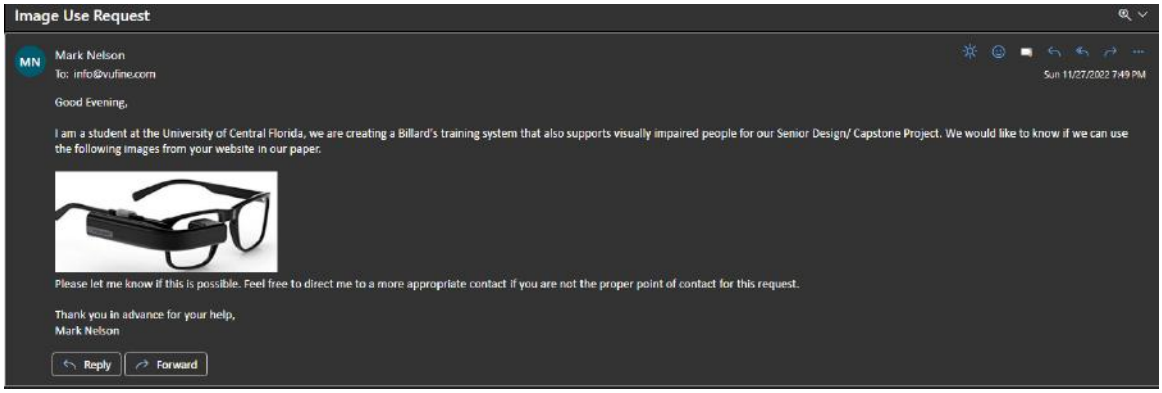
The next subsystem is the HUD. This is the system that most directly interacts with the user. The HUD's subsystems consist of a few different peripherals. A camera, which is used for cue aiming feedback, a display, which is used to convey both pre- and post-shot information to the unimpaired user, and a speaker, which is used to communicate information auditorily to the impaired user.

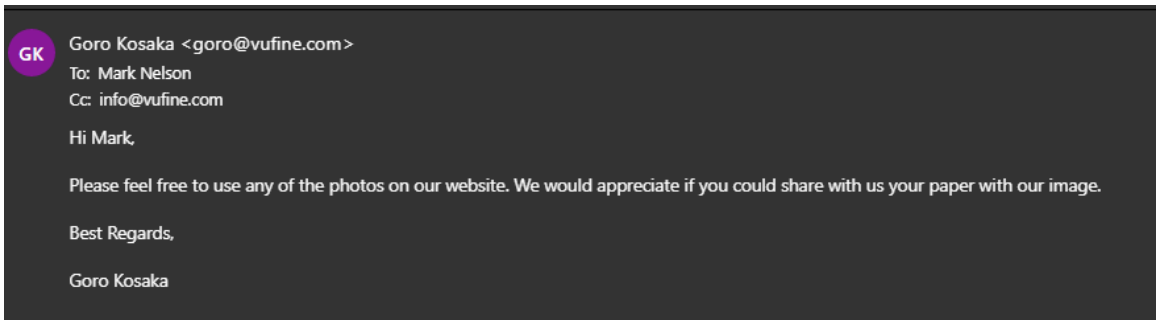
The final subsystem we used to implement this design is the glove. The glove's main purpose is to assist the visually impaired user with setting up a shot. This is done by determining the correct orientation and distance and then sending this data to be communicated to the user.

Overall, the researching, designing and implementation of this project has been a large learning experience for all the members involved. With these systems all working together properly, SCRATCH allows people both to improve their skill at a game, and experience a game for the first time in a way never imagined before. It is a magnificent feat of engineering that facilitates diversity and inclusion. Although our project meets all specifications, we believe that improvements could be made. Given more time, user profiles can be added to further enhance and tailor user experience. Additionally, angle feedback could be provided as the visually impaired user moves towards the ball for more accuracy.

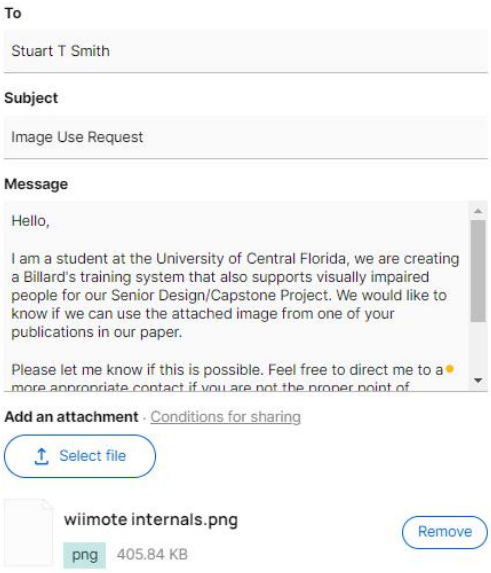
APPENDIX A - COPYRIGHT PERMISSIONS

Request for Figure 3.1:





Request for Figure 3.2:



Request for Figure 3.3:

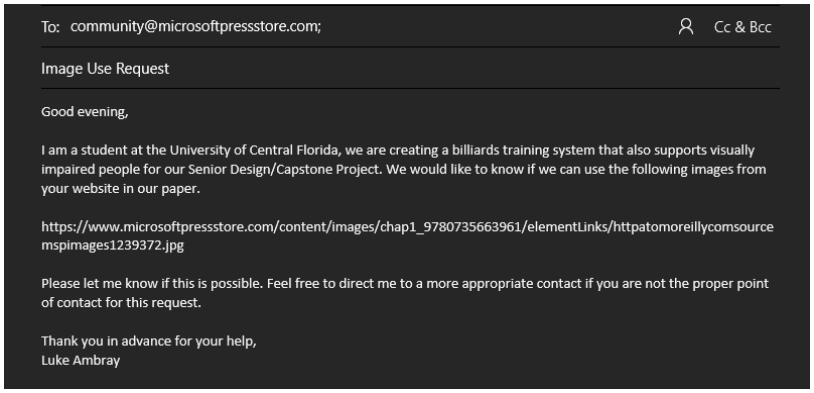

To: community@microsoftpressstore.com; Cc & Bcc

Image Use Request

Good evening,

I am a student at the University of Central Florida, we are creating a billiards training system that also supports visually impaired people for our Senior Design/Capstone Project. We would like to know if we can use the following images from your website in our paper.

https://www.microsoftpressstore.com/content/images/chap1_9780735663961/elementLinks/httpatomoreillycomsourcemspimages1239372.jpg

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Luke Ambray

Request for Figure 3.4:

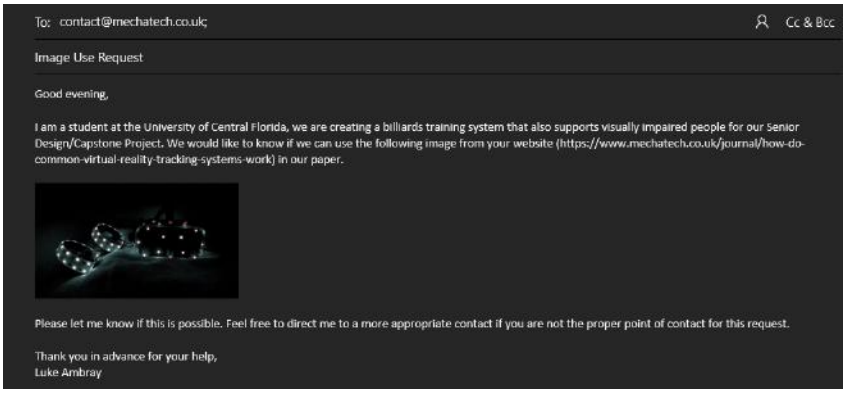

To: contact@mechatech.co.uk; Cc & Bcc

Image Use Request

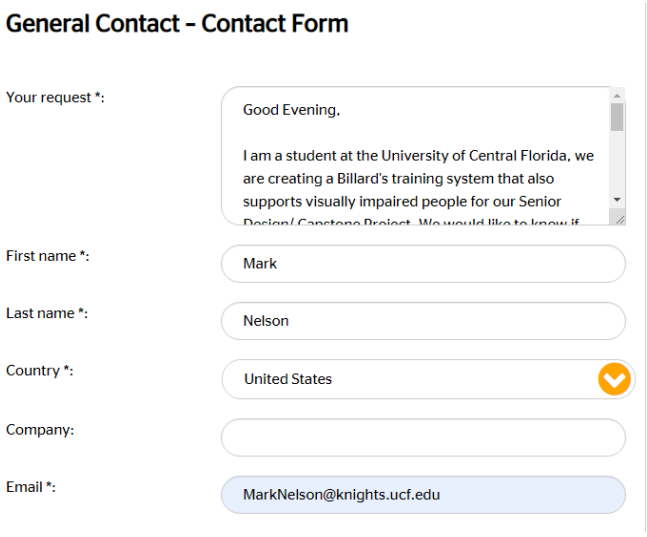
Good evening,

I am a student at the University of Central Florida, we are creating a billiards training system that also supports visually impaired people for our Senior Design/Capstone Project. We would like to know if we can use the following image from your website (https://www.mechatech.co.uk/journal/how-do-common-virtual-reality-tracking-systems-work) in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Luke Ambray

Request for Figure 3.5:


General Contact - Contact Form

Your request *: Good Evening, I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior

First name *: Mark

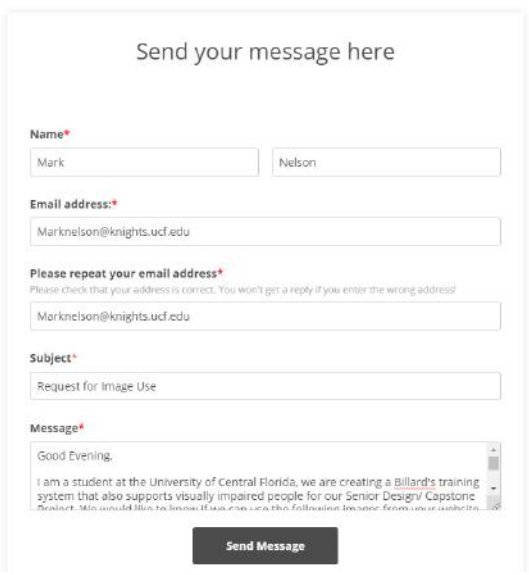
Last name *: Nelson

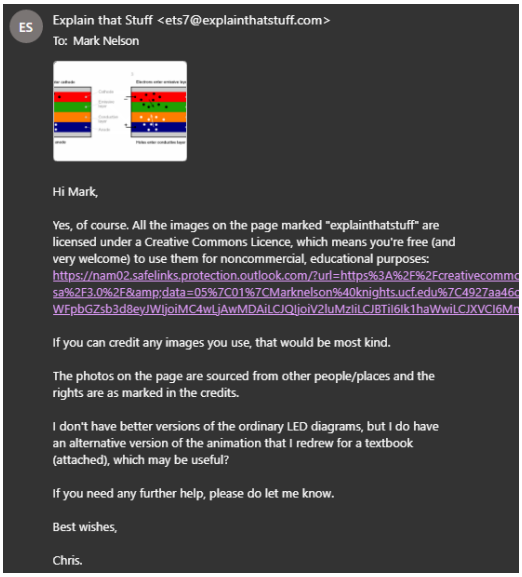
Country *: United States

Company:

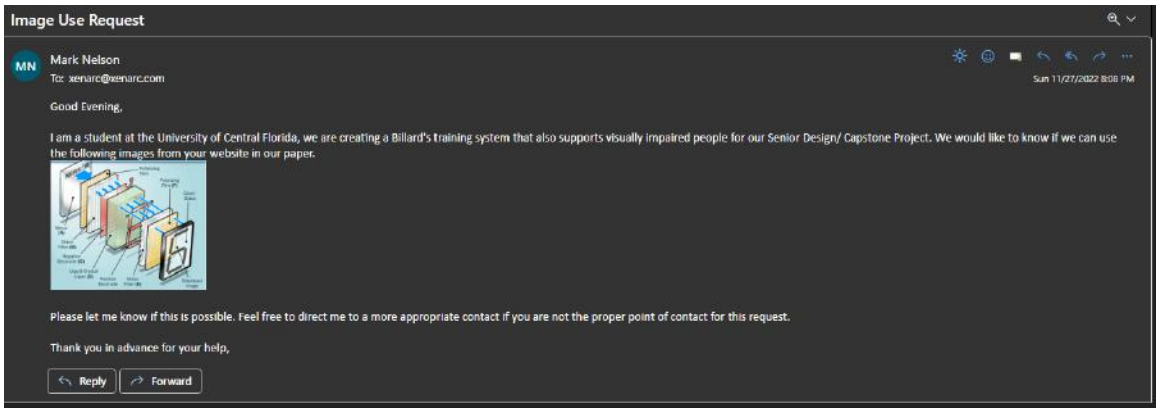
Email *: MarkNelson@knights.ucf.edu

Request for Figure 3.6:

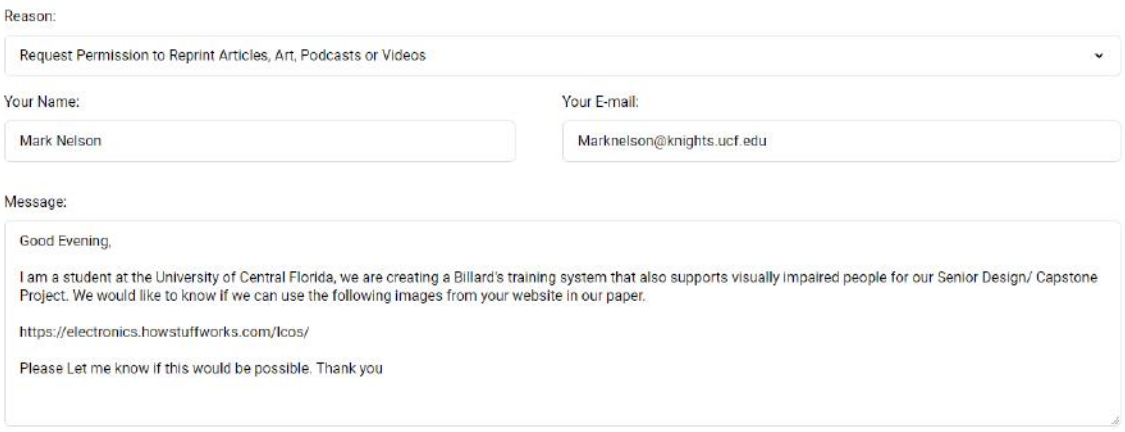




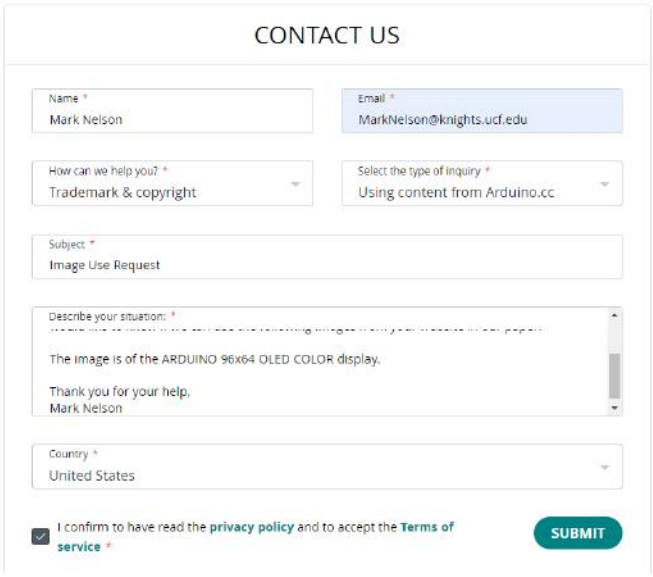
Request for Figure 3.7:



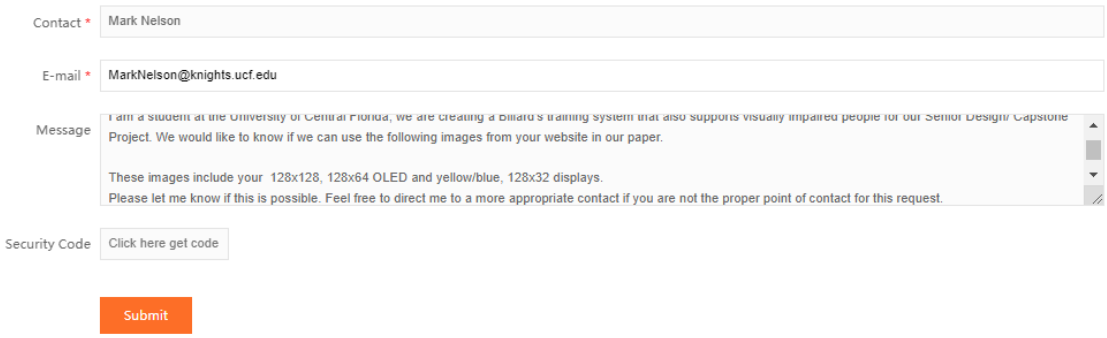
Request for Figure 3.8:



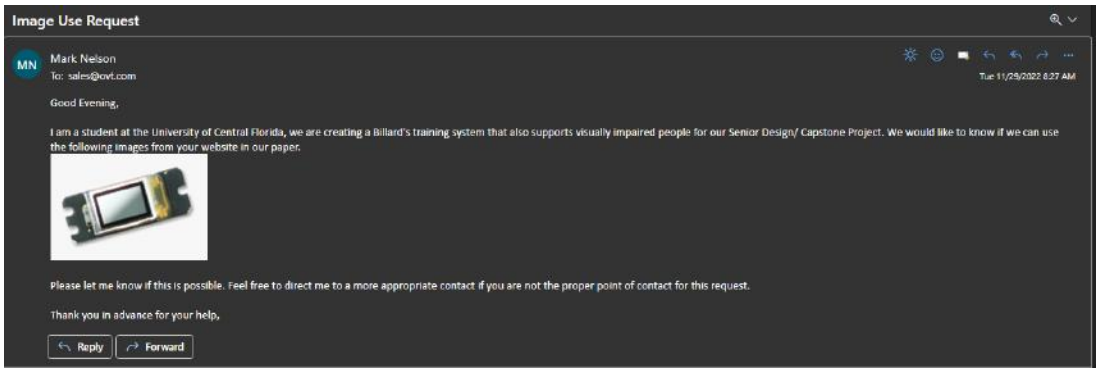
Request for Figure 3.9



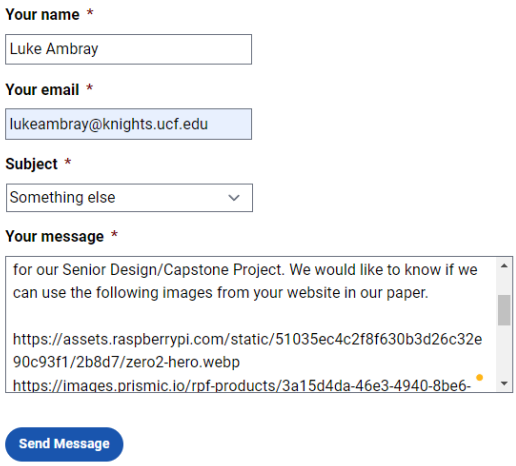
Request for Figure 3.10 - .12:



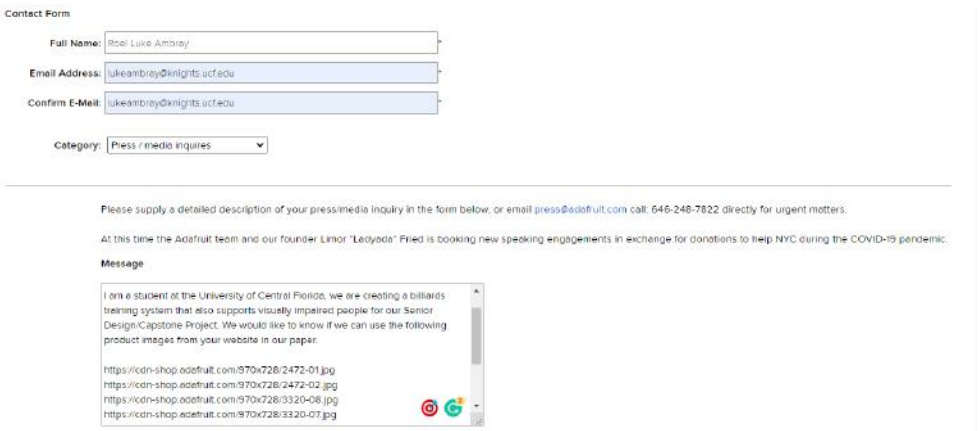
Request for Figure 3.13:



Request for Figure 3.14:



Request for Figure 3.15 & 3.23



Request for Figure 3.16:

Request and Response for Figure 3.17:

GL

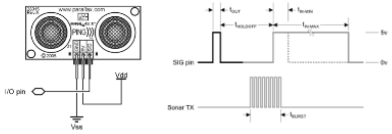
Goran Lalich

To: support@parallax.com

Sun 11/27/2022 7:15 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Goran Lalich

PI

Parallax Inc. <support@parallaxinc.zohodesk.com>

To: Goran Lalich

Cc: support@parallax.com

Mon 11/28/2022 4:48 AM

Good morning Goran,

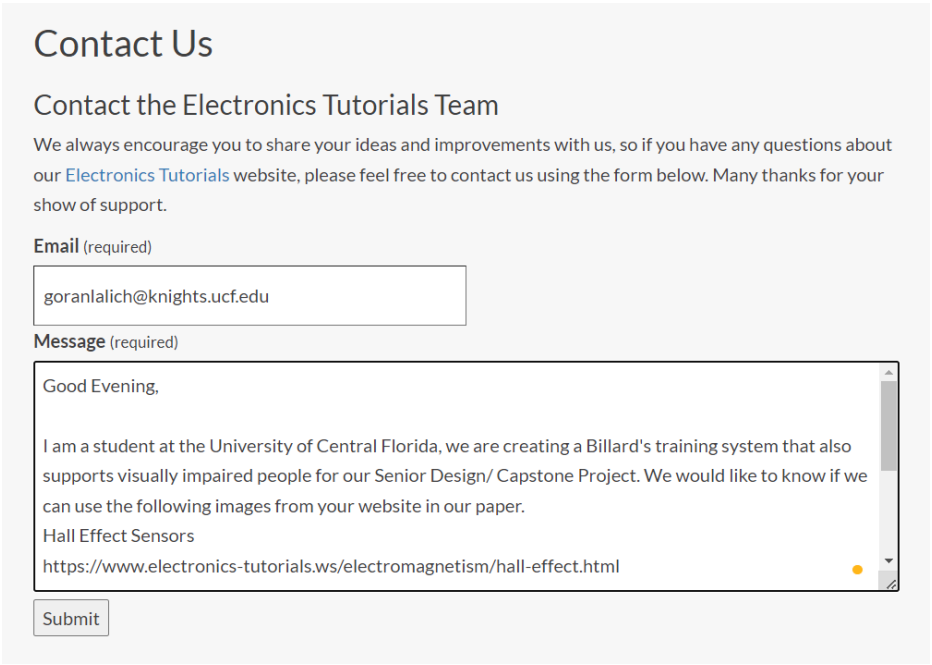
Thank you for asking, and **yes** you are free to use those images in your paper.

Best regards,

Michael Mulholland

-Parallax Support Team

Request and Response for Figure 3.18:



Contact Us

Contact the Electronics Tutorials Team

We always encourage you to share your ideas and improvements with us, so if you have any questions about our Electronics Tutorials website, please feel free to contact us using the form below. Many thanks for your show of support.

Email (required)

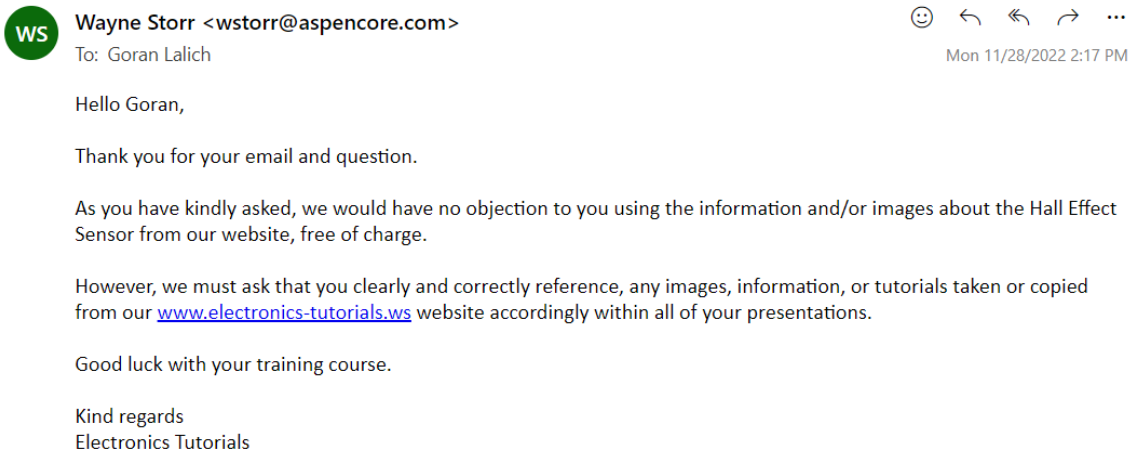
goranlalich@knights.ucf.edu

Message (required)

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.
Hall Effect Sensors
https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html

Submit



Wayne Storr <wstorr@aspencore.com>
To: Goran Lalich
Mon 11/28/2022 2:17 PM

Hello Goran,

Thank you for your email and question.

As you have kindly asked, we would have no objection to you using the information and/or images about the Hall Effect Sensor from our website, free of charge.

However, we must ask that you clearly and correctly reference, any images, information, or tutorials taken or copied from our www.electronics-tutorials.ws website accordingly within all of your presentations.

Good luck with your training course.

Kind regards
Electronics Tutorials

Request for Figure 3.19:

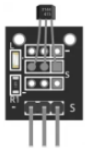
GL Goran Lalich

To: support@arduinomodules.com

Sun 11/27/2022 7:50 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Goran Lalich

Request for Figure 3.20:

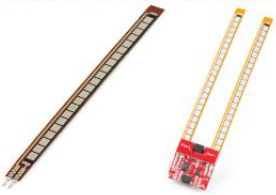
GL Goran Lalich

To: services@sparkfun.com

Sun 11/27/2022 8:16 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Goran Lalich

Request for Figure 3.25:

GL

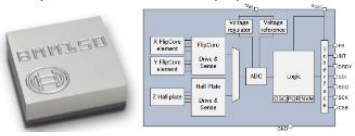
Goran Lalich

To: contact@bosh-sensortec.com

Mon 11/28/2022 2:48 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Goran Lalich

Request for Figure 3.26:

GL

Goran Lalich

To: support@freescale.com

Mon 11/28/2022 2:54 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Goran Lalich

Request for Figure 3.27:

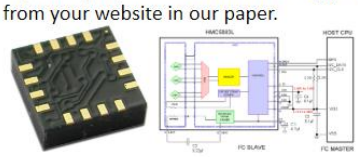
GL Goran Lalich

To: support@honeywell.com

Mon 11/28/2022 2:59 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Goran Lalich

Request for Figure 3.28:

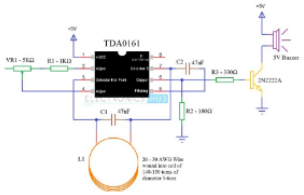
GL Goran Lalich

To: admin@electronicshub.org

Mon 11/28/2022 3:03 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Goran Lalich

Request for Figure 3.29:

GL

Goran Lalich

To: support@Ximimark.com

Mon 11/28/2022 3:12 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Goran Lalich

Request for Figure 3.30:

MM

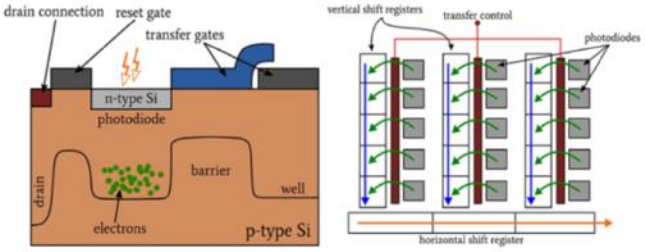
Mena Mishriky

To: editorial@allaboutcircuits.com

Sun 11/27/2022 7:18 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

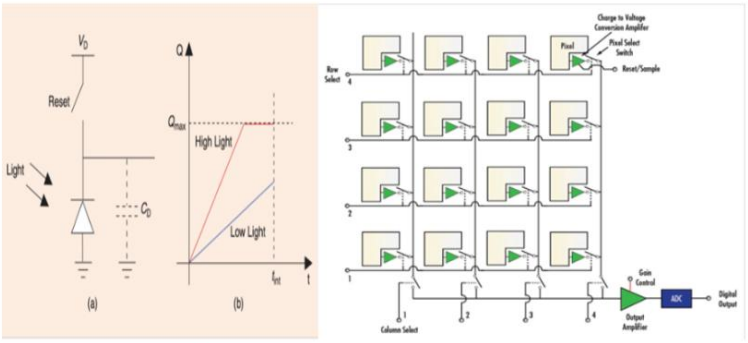
Mena Mishriky

Request for Figure 3.31

MM Mena Mishriky
To: editorial@allaboutcircuits.com
Sun 11/27/2022 7:24 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Mena Mishriky

Request for Figure 3.32

MM Mena Mishriky
To: phil.schatzmann@gmail.com
Sun 11/27/2022 7:31 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

Reply Forward

Request for Figure 3.33

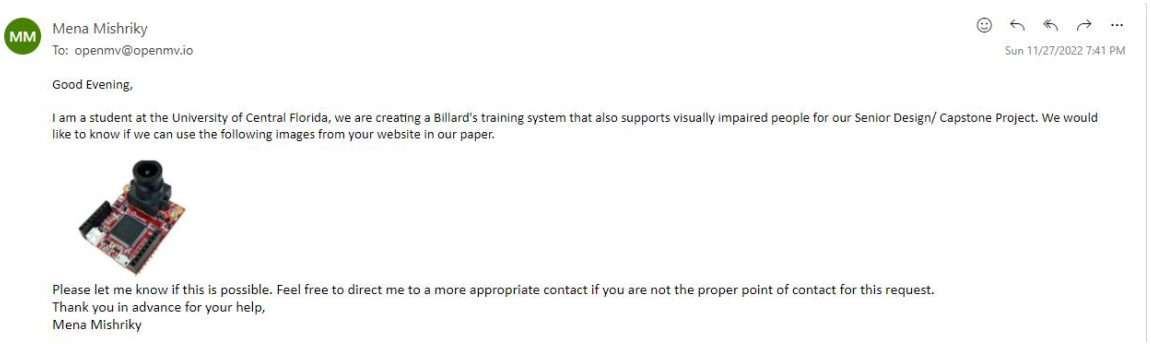
MM Mena Mishriky
To: support@arducam.com
Sun 11/27/2022 7:36 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.
Thank you in advance for your help,
Mena Mishriky

Request for Figure 3.34



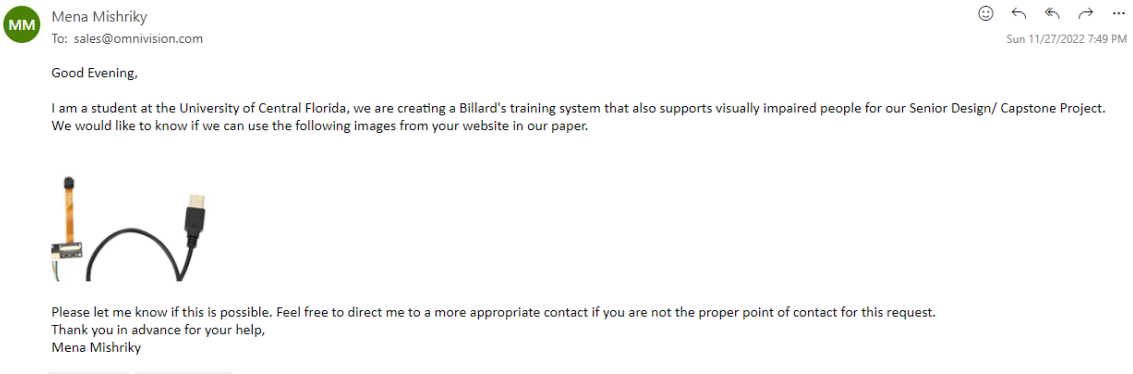
MM Mena Mishriky
To: openmv@openmv.io
Sun 11/27/2022 7:41 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.
Thank you in advance for your help,
Mena Mishriky

Request for Figure 3.35



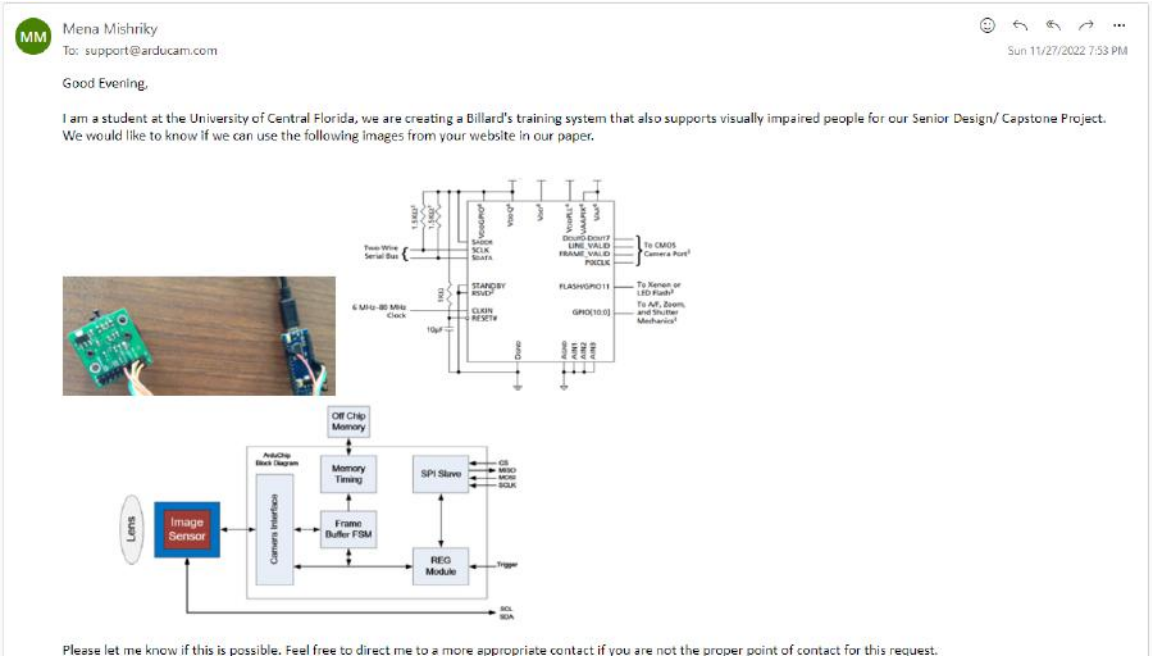
MM Mena Mishriky
To: sales@omnivision.com
Sun 11/27/2022 7:49 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.
Thank you in advance for your help,
Mena Mishriky

Request for Figures 3.36, 3.37, 3.38



MM Mena Mishriky
To: support@arducam.com
Sun 11/27/2022 7:53 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Request for Figure 3.39-3.43:

GL Goran Lalich
To: laura@radioshuttle.de
Mon 11/28/2022 3:25 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

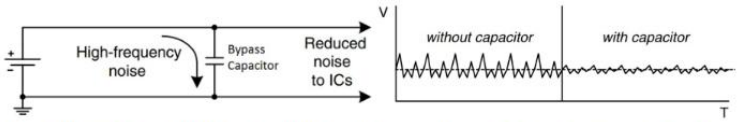
Thank you in advance for your help,
Goran Lalich

Request for Figure 3.44:

GL Goran Lalich
To: admin@electronicshub.org
Mon 11/28/2022 3:29 PM

Good Evening,

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.



Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,
Goran Lalich

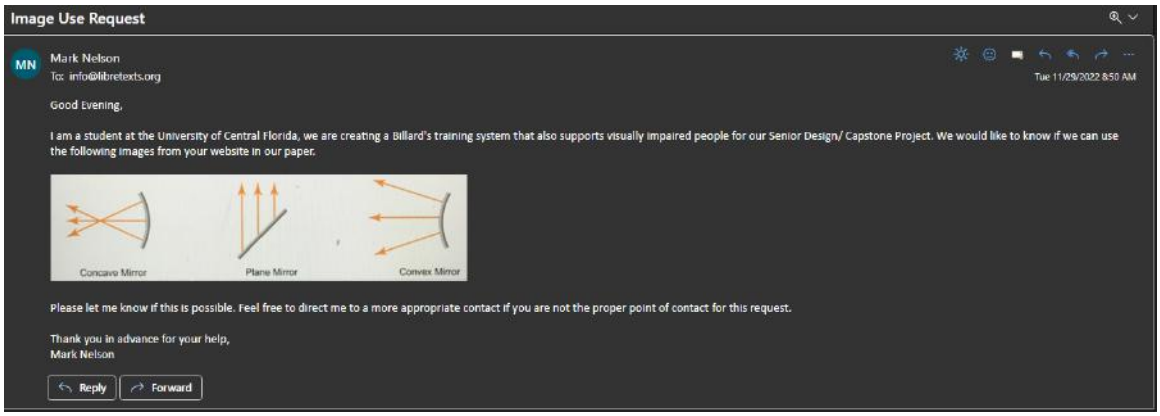
E ElectronicsHub.org <admin@electronicshub.org>
To: Goran Lalich
Tue 11/29/2022 1:06 AM

Hi,

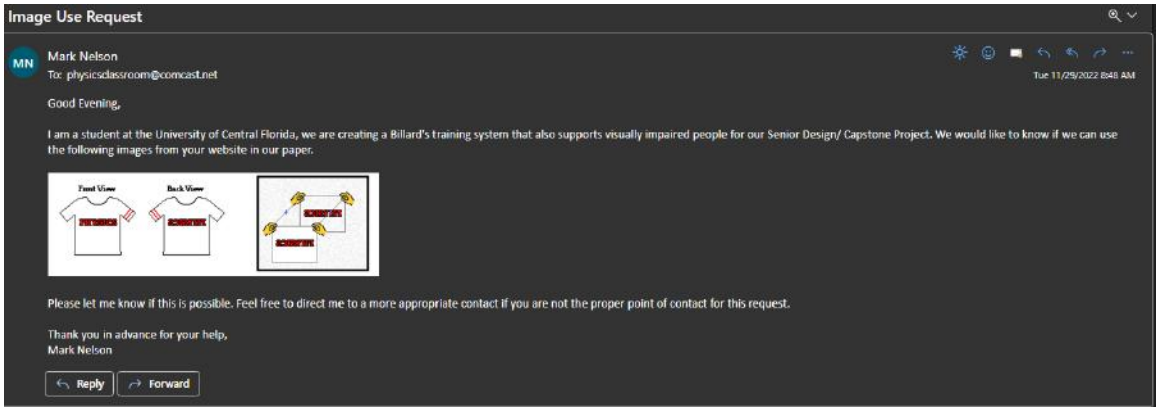
You can use those images, but please credit back to our website:**electronicshub.org**.

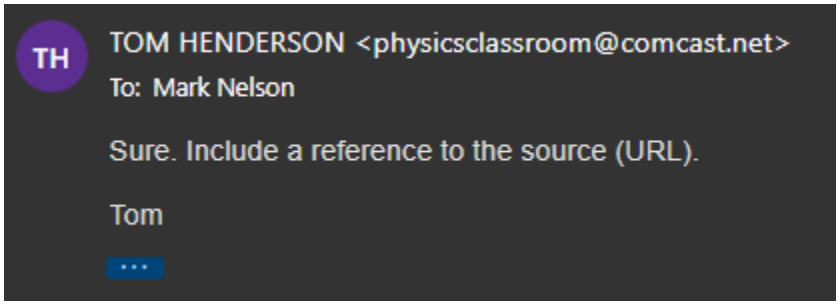
Thanks you
Leona Lee

Request for Figure 3.45:

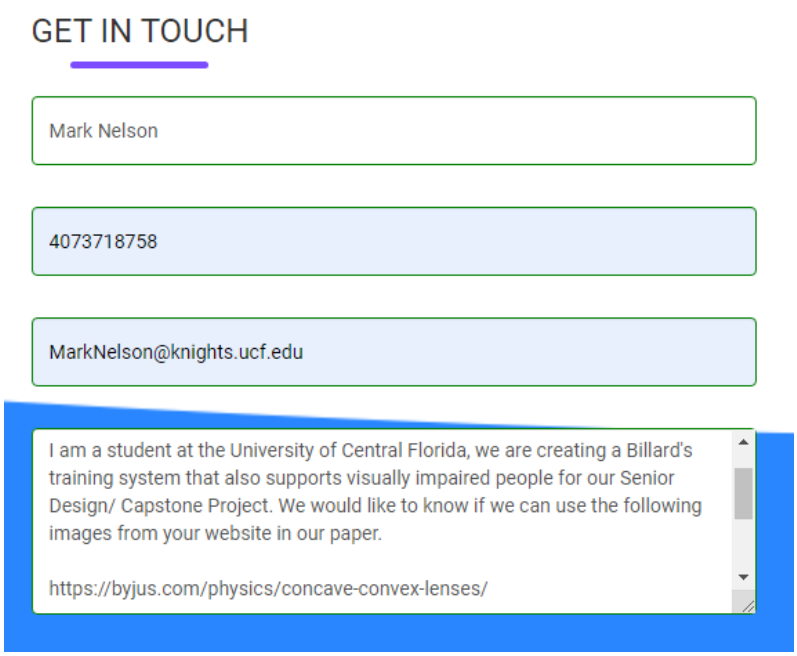


Request for Figure 3.46:

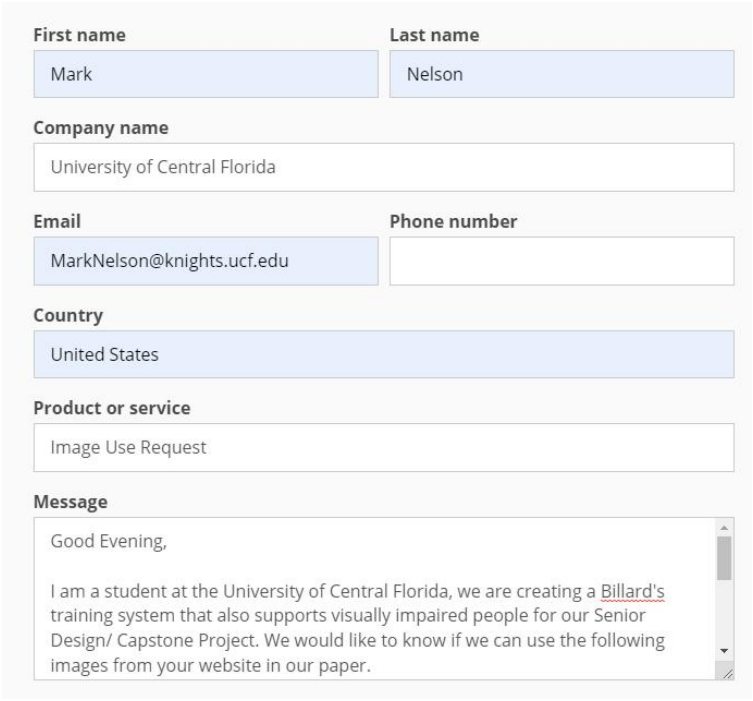




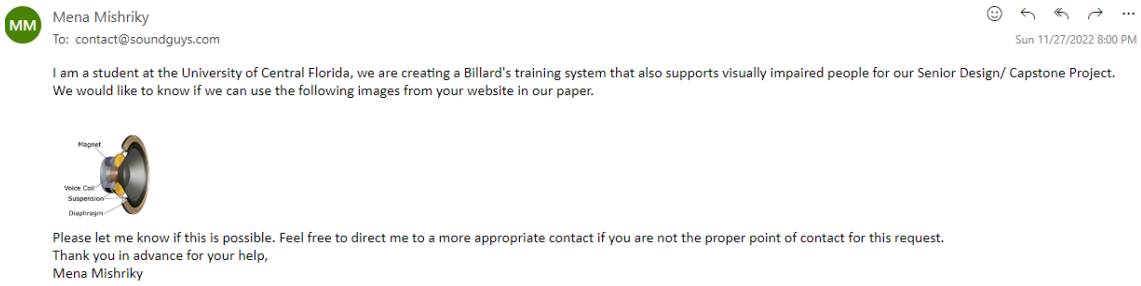
Request for Figure 3.47:



Request for Figure 3.48



Request for Figure 3.51

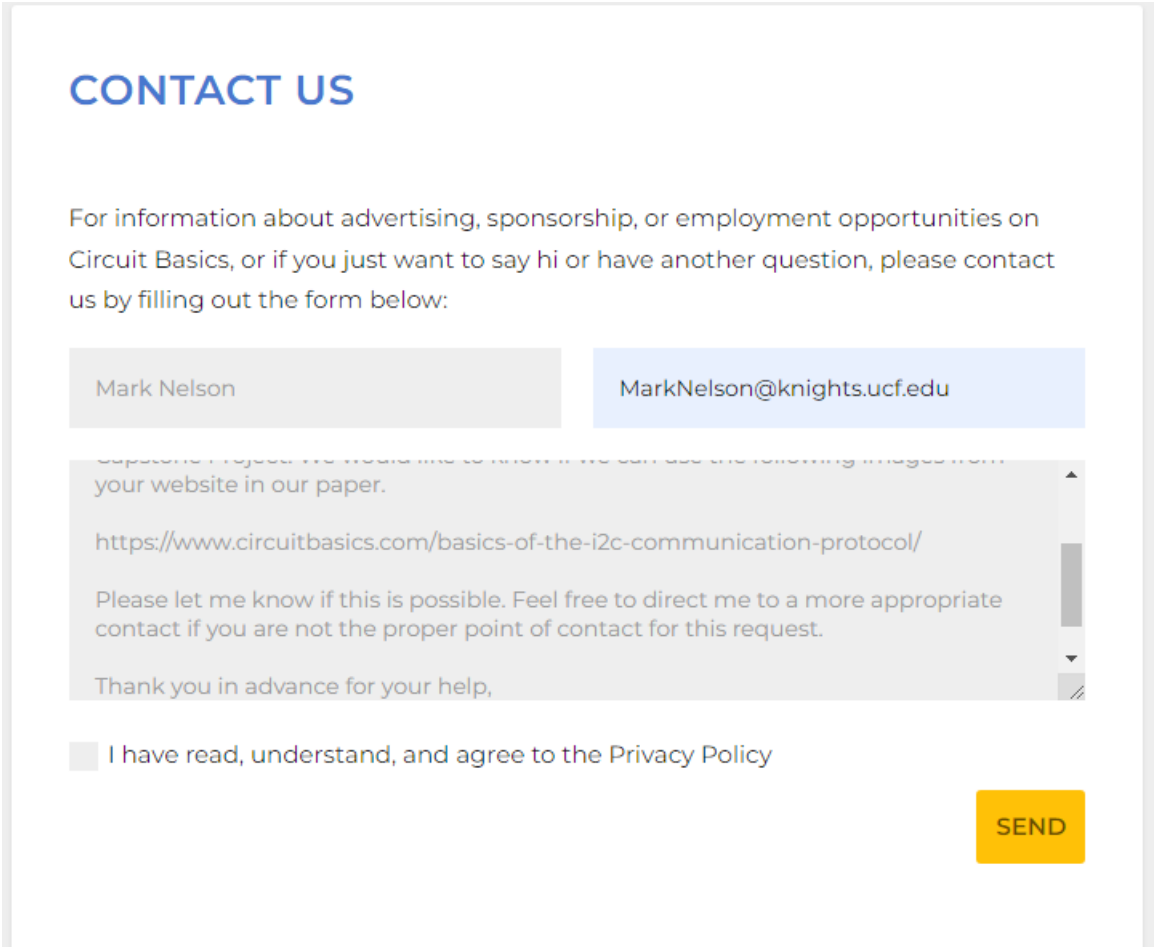


Mena Mishriky
To: contact@soundguys.com
Sun 11/27/2022 8:00 PM

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.
Thank you in advance for your help,
Mena Mishriky

Request for Figure 4.1



CONTACT US

For information about advertising, sponsorship, or employment opportunities on Circuit Basics, or if you just want to say hi or have another question, please contact us by filling out the form below:

Mark Nelson

MarkNelson@knights.ucf.edu

your website in our paper.

https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

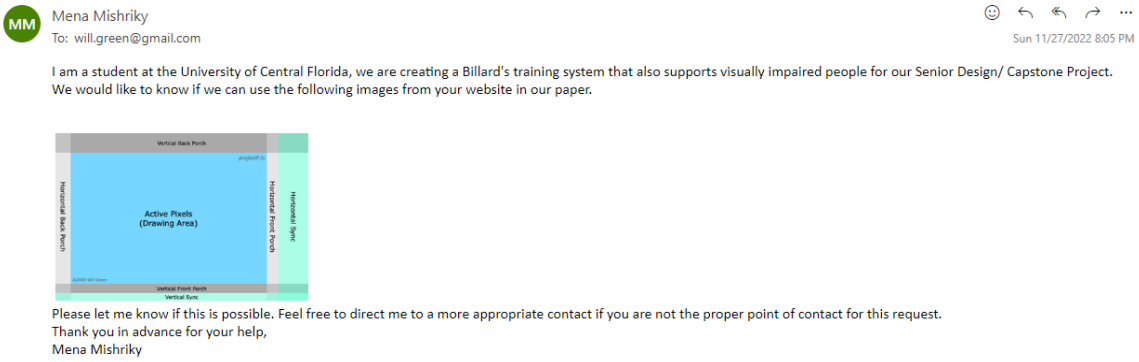
Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.

Thank you in advance for your help,

I have read, understand, and agree to the Privacy Policy

SEND

Request for Figure 4.2

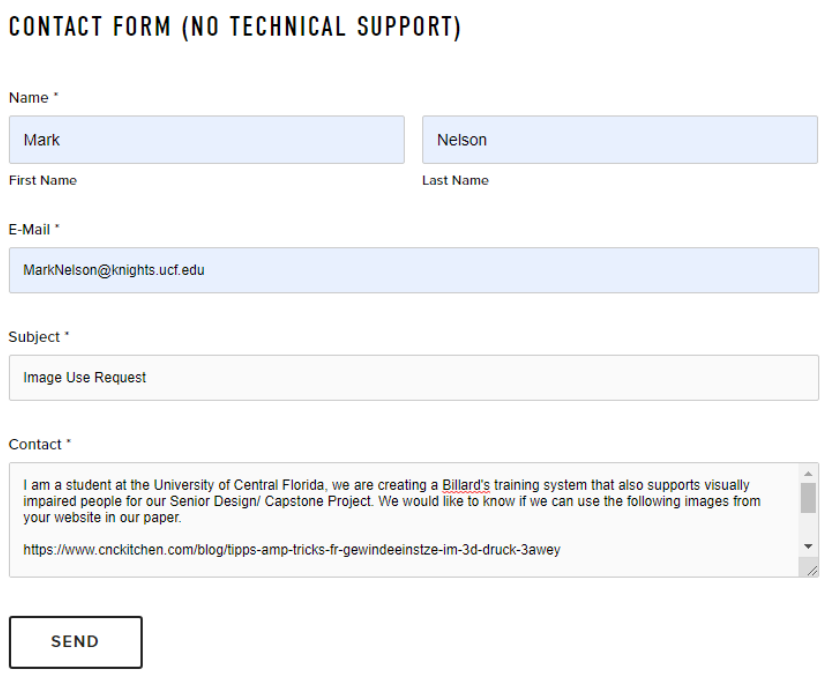


MM Mena Mishriky
To: will.green@gmail.com
Sun 11/27/2022 8:05 PM

I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

Please let me know if this is possible. Feel free to direct me to a more appropriate contact if you are not the proper point of contact for this request.
Thank you in advance for your help,
Mena Mishriky

Request for Figure 5.1



CONTACT FORM (NO TECHNICAL SUPPORT)

Name *

Mark — First Name

Nelson — Last Name

E-Mail *

MarkNelson@knights.ucf.edu

Subject *

Image Use Request

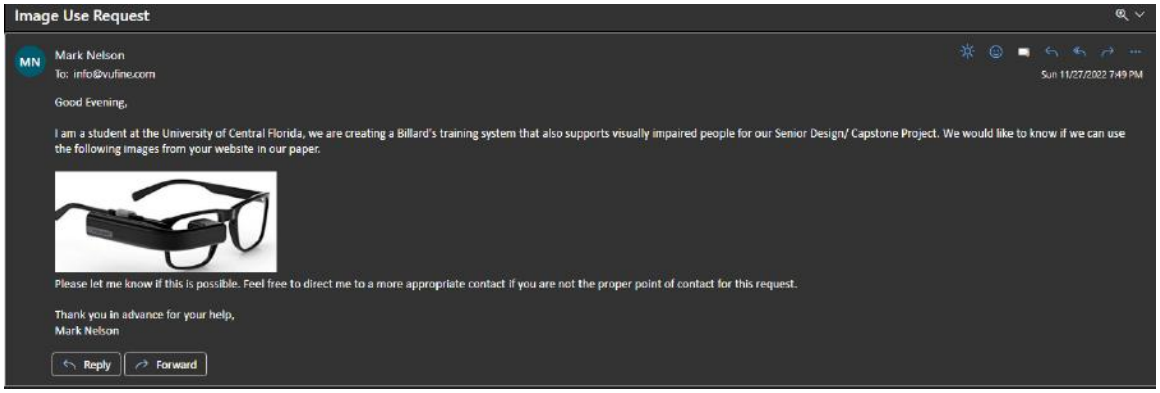
Contact *

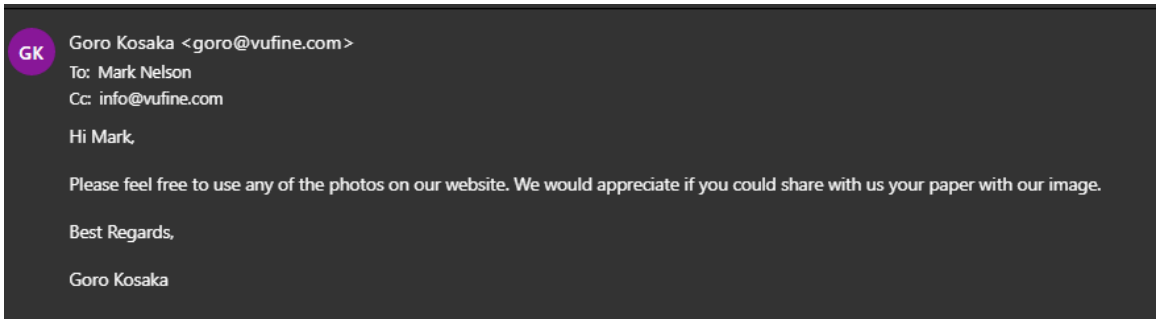
I am a student at the University of Central Florida, we are creating a Billard's training system that also supports visually impaired people for our Senior Design/ Capstone Project. We would like to know if we can use the following images from your website in our paper.

https://www.cnckitchen.com/blog/tipps-amp-tricks-fr-gewindeeinstze-im-3d-druck-3awey

SEND

Request for Figure 5.2





APPENDIX B - REFERENCES

[3.1] "VUFINE+ Wearable Display," *Vufine*. [Online]. Available: https://store.vufine.com/products/vufine-wearable-display-1. [Accessed: 27-Nov-2022].

[3.2] "Novel optics: Plastic micro-optics cater to automotive HUD design ..." [Online]. Available: https://www.laserfocusworld.com/optics/article/16550165/novel-optics-plastic-microoptics-cater-to-automotive-hud-design. [Accessed: 28-Nov-2022].

[3.3] Chris Woodford. Last updated: September 1, "How oleds (organic leds) work," *Explain that Stuff*. [Online]. Available: https://www.explainthatstuff.com/how-oleds-and-leps-work.html#led. [Accessed: 27-Nov-2022].

[3.4] "LCD technology," *How The Technology of LCD Displays Works - Xenarc Technologies Blog*. [Online]. Available: https://www.xenarc.com/lcd-technology.html. [Accessed: 27-Nov-2022].

[3.5] T. V. Wilson, "How lcos works," *HowStuffWorks*, 12-Jan-2006. [Online]. Available: https://electronics.howstuffworks.com/lcos.htm. [Accessed: 27-Nov-2022].

[3.6] R. Burnett, “Understanding how ultrasonic sensors work,” *MaxBotix Inc.*, 04-Mar-2021. [Online]. Available: https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm#:~:text=Ultrasonic%20sensors%20work%20by%20sending,and%20to%20receive%20the%20echo. [Accessed: 29-Nov-2022].

[3.7] “The Guide to Hall Effect Sensors,” *RS*. [Online]. Available: https://se.rs-online.com/web/generalDisplay.html?id=ideas-and-advice%2Fhall-effect-sensors-guide. [Accessed: 29-Nov-2022].

[3.8] “Understanding the structure and functionality of ccds - technical articles,” *All About Circuits*. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/understanding-the-structure-and-functionality-of-ccds/. [Accessed: 27-Nov-2022].

[3.9] *Radioshuttle Network protocol*. [Online]. Available: https://www.radioshuttle.de/en/media-en/tech-infos-en/battery-powered-esp32/. [Accessed: 29-Nov-2022].

[3.10] Libretexts, “Mirrors,” *Physics LibreTexts*, 21-Jun-2021. [Online]. Available: https://phys.libretexts.org/Bookshelves/Optics/Supplemental_Modules_(Components)/Mirrors. [Accessed: 27-Nov-2022].

[3.11] “Physics tutorial: Image characteristics of plane mirrors,” *The Physics Classroom*. [Online]. Available: https://www.physicsclassroom.com/class/refln/Lesson-2/Image-Characteristics. [Accessed: 27-Nov-2022].

[3.12] Admin, “Concave and convex lenses - image formation: Curvature & focus,” *BYJUS*, 13-May-2022. [Online]. Available: https://byjus.com/physics/concave-convex-lenses/. [Accessed: 27-Nov-2022].

[3.13] J. Simonen, “Characterizing augmented reality displays based on waveguide gratings,” *Characterizing Augmented Reality Displays Based On Waveguide Gratings*. [Online]. Available: https://www.optofidelity.com/blog/characterizing-augmented-reality-displays-based-on-waveguide-gratings. [Accessed: 27-Nov-2022].

[3.14] “How do speakers work?,” SoundGuys, 05-Oct-2022. [Online]. Available: https://www.soundguys.com/how-speakers-work-29860/. [Accessed: 27-Nov-2022].

Bluetooth Classic:

[4.1] “Introduction to Bluetooth Classic,” *Argenox*. [Online]. Available: https://www.argenox.com/library/bluetooth-classic/introduction-to-bluetooth-classic/. [Accessed: 17-Nov-2022].

Bluetooth Low Energy:

[4.2] "Bluetooth Technology Overview," *Bluetooth® Technology Website*. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/. [Accessed: 17-Nov-2022].

Wi-Fi:

[4.3] M. Brain and T. Homer, "How WiFi Works," *HowStuffWorks*, 17-Aug-2021. [Online]. Available: https://computer.howstuffworks.com/wireless-network.htm. [Accessed: 02-Jun-2022].

[4.4] G. D. Putra, A. R. Pratama, A. Lazovik and M. Aiello, "Comparison of energy consumption in Wi-Fi and bluetooth communication in a Smart Building," *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017, pp. 1-6, doi: 10.1109/CCWC.2017.7868425.* [Accessed: 17-Nov-2022].

Zigbee:

[4.5] "Zigbee FAQ," *Connectivity Standards Alliance*. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/zigbee-faq/. [Accessed: 02-Jun-2022].

UART:

[4.6] E. Peňa and M. G. Legaspi, "UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter," *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices*, Dec-2020. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Accessed: 02-Jun-2022].

SPI:

[4.7] "Serial peripheral interface," Wikipedia, 27-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface. [Accessed: 27-Nov-2022].

I2C:

[4.8] S. Campbell, "Basics of the I2C communication protocol," *Circuit Basics*. [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed: 27-Nov-2022].

USB:

[4.9] "About the USB protocol, common USB bus errors, and how to troubleshoot them," *Total Phase Blog*, 22-Dec-2021. [Online]. Available: https://www.totalphase.com/blog/2020/07/about-the-usb-protocol-common-usb-bus-errors-and-how-to-troubleshoot-them/#:~:text=The%20USB%20protocol%2C%20also%20known,a%20multitude%20of%20different%20devices. [Accessed: 27-Nov-2022].

HDMI:

[4.10] T. V. Wilson, "How HDMI works," *HowStuffWorks*, 08-Oct-2007. [Online]. Available: https://electronics.howstuffworks.com/hdmi.htm. [Accessed: 27-Nov-2022].

SCBB:

[4.11] Omnivision, "Serial Camera Control Bus Functional Specification", 26-Feb-2003. [Online].Available:https://www.waveshare.com/w/upload/1/14/OmniVision_Technologies_Seril_Camera_Control_Bus%28SCCB%29_Specification.pdf [Accessed: 27-Nov-2022].

DCMI:

[4.12] STMicroelectronics, "Digital Camera Interface for STM32 MCUs", 02-Oct-2022. [Online].Available:https://www.st.com/resource/en/application_note/an5020-digital-camera-interface-dcmi-on-stm32-mcus-stmicroelectronics.pdf [Accessed: 27-Nov-2022].

DMA:

[4.13] "Direct memory access," *Wikipedia*, 23-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/Direct_memory_access. [Accessed: 27-Nov-2022].

I2S:

[4.14] "Introduction to the I2S interface - technical articles," *All About Circuits*. [Online]. Available:https://www.allaboutcircuits.com/technical-articles/introduction-to-the-i2s-interface/. [Accessed: 27-Nov-2022].

IEEE 754:

[4.15] "IEEE Standard for Floating-Point Arithmetic," in IEEE *Std 754-2019 (Revision of IEEE 754-2008)* , vol., no., pp.1-84, 22 July 2019, doi: 10.1109/IEEESTD.2019.8766229.

MP3:

[4.16] "MP3,"Wikipedia,02-Nov-2022.[Online]. Available: https://en.wikipedia.org/wiki/MP3. [Accessed: 27-Nov-2022].

WAV:

[4.17] "Multimedia Programming Interface and Data Specifications 1.0," 1991. [Online]. Available: https://www.aelius.com/njh/wavemetatools/doc/riffmci.pdf. [Accessed: 17-Nov-2022].

[5.1] Sharretts Plating Company, "What materials are used in the 3D printing process?: SPC," *Sharretts Plating Company*, 28-Sep-2022. [Online]. Available: https://www.sharrettsplating.com/blog/materials-used-3d-printing/.[Accessed:27-Nov-2022].

[5.2] "PETG 3D printing filament," 3DXTECH. [Online]. Available: https://www.3dxtech.com/product/max-g-petg/. [Accessed: 27-Nov-2022].

[5.3] "Ultimate Guide to threaded inserts and 3D prints: MakerBot 3D printers," *MakerBot*.[Online].Available:https://www.makerbot.com/professional/post-processing/inserts/.[Accessed: 27-Nov-2022].

[5.4] "How to Smooth Abs 3D prints: Vapor Smoothing Step by step," *The 3D Printer Bee*. [Online]. Available: https://the3dprinterbee.com/how-to-smooth-abs-3d-prints-vapor-smoothing-step-by-step/. [Accessed: 27-Nov-2022].

[5.5] "3D CAD Design Software," *Solidworks*. [Online]. Available: https://www.solidworks.com/. [Accessed: 27-Nov-2022].

[5.6] "VUFINE+ Wearable Display," *Vufine*. [Online]. Available: https://store.vufine.com/products/vufine-wearable-display-1. [Accessed: 27-Nov-2022].

Software Version Control:

[6.1] "1.1 Getting Started - About Version Control," *Git*. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. [Accessed: 17-Nov-2022].

[6.2] "1.2 Getting Started - A Short History Of Git," *Git*. [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git. [Accessed: 17-Nov-2022].

[6.3] C. Gehman, "Mercurial vs. Git: How are they different?," *Perforce Software*, 09-Jan-2019. [Online]. Available: https://www.perforce.com/blog/vcs/mercurial-vs-git-how-are-they-different#:~:te

xt=Mercurial%20Is%20Safer%20For%20Less,%E2%80%9Chg%20commit%20%E2%80%93%20amend%E2%80%9D. [Accessed: 17-Nov-2022].

[6.4] "Fossil," *RSS*. [Online]. Available: https://www2.fossil-scm.org/home/doc/trunk/www/index.wiki. [Accessed: 17-Nov-2022].

System Fabrication and Integration:

[7.1] "Top 5 PCB design rules you need to know," *Altium*, 21-Nov-2022. [Online]. Available: https://resources.altium.com/p/pcb-layout-guidelines. [Accessed: 29-Nov-2022].