RevMetrix Project Proposal

Spring 2026

GANTT

https://ycp325394.monday.com/boards/18398298457

Team Cellular

Members

Josh Byers, Zach Cox

Github Repo

https://github.com/YCP-Rev-Metrix/Cellular.git

Proposed Additions

- Update Ball Arsenal
- Create Syncing Feature
- Creating Create pages for Session and Game
- Updating UI and UX
- Improved Stats implementation
- Ciclopes Integration
    - Integrate MMS and Video to work together. The video is going to be able to start when the MMS light sensor is high enough. Then, the phone will save the data and take the derivative of the acceleration to see when the ball hits the lanes to send it over to the Ciclopes system.
- Email Verification

- Smart dot Saving
    - Save the data Locally
        - Save the mac address

Team Backend

- Fixing phone API Points
- Get Video being sent and stored via API point and Space Object Storage
- Get pushed data on main to digital Ocean
- Update Sensor Data for Pi Team. Spin_Instruction_Points, Tilt_Instruction_Points, Angle_Instruction_Points. maxvar.
- Deprecating old features

Team Smart Watch

Members

Charles Carroll, Jakeb Nielsen

Github Repo

https://github.com/YCP-Rev-Metrix/Android-Smart-Watch.git

Current State

Last semester the watch app got to a minimal working system. The user can enter shot data in the revised two phase shot system which is then associated to a frame, then a game, then a session. This included full backend object passing and manipulation, as well as local storage. The two phase shot system included the first phase where the bowler enters information before the shot is bowled and the second phase where the remaining data is entered after the shot is complete. The user interface can be used to navigate between shots, frames, and games as well as some test screens like the bluetooth settings page. The watch can be paired to the phone, as well as send and receive a single hardcoded packet to prove out BLE data transmission. Some pages, like the game page(s), still have placeholder data that simply identifies the page.

Problem

The bowler may not have their phone available for quick data collection or reference due to one of two reasons:

1) Phone is not on the person when bowling
2) Ciclopes is being used on the phone

This requires an alternative method to track and view data. The solution is the smart watch application. The smart watch app will need to take the functionality of the existing phone app and strip it down to be a user friendly interface to be used on a smart watch screen which will presumably be on the bowler's non-dominant hand at all times during the session.

Major Technical Challenges

A technically difficult task is the initial handshake between the watch and the phone. This initial handshake will transfer existing session data, if there is any, and the session ID when prompted to start or restore the session. The session ID is needed to store all recorded data into the database. The challenge is when that handshake happens, when the data is going to be sent, and how this event is triggered.

Overall our product from last semester worked very smoothly and could function correctly, but that does not mean there weren't still bugs. A challenge this semester will be fixing the ones already found and on top of that fixing the ones that will evidently pop up along the way. Our product needs to be polished for the capstone expo so that is why having these bugs worked is so important.

Coming in from last semester we had a two phase shot functionality which left the watch screen feeling very cluttered and made it difficult to see and/or select certain buttons. A new design proposed by our client will help to fix this problem by organizing each data input onto their own screens. This will be a challenge because it will take some time to completely overhaul our previous setup, and it will take time away from the other necessary things that need to be done for a fully working product. The sacrifice is worth it for a more easily usable product.

The goal of this semester is to have a somewhat polished product to show at the capstone expo. One of the biggest challenges is getting everything done from now until the capstone expo, that due date cannot be pushed back. Most bugs must be fixed and the features that are promised must be done.

Another big challenge is the watch itself. When making the UI, screen size is extremely limited making it increasingly important to have an intuitive UI. On the backend of the watch,

storage, computing power, and data transmission will all be limitations to be aware of during development.

Working in sync with the cellular team will also be a challenge mainly due to the need for complete correlation between the mobile app and smart watch app. The smart watch application should act as an add on, so the ui, functionality, and general appearance of the app should act a lot like the mobile app. Keeping everyone in sync and on the same page will be a challenge.

Technologies & Why

The framework and language we decided to go with are Flutter and Dart. The reasoning behind this is Flutter allows a way to create a single codebase for compilation of wearOS and watchOS, this will allow for easier integration and upkeep later down the line when the IOS development has begun, as we will not have to maintain two codebases. In Flutter you can create a single project and then add extensions for watchOS and wearOS. Another reason as to why we chose this framework is because of its standalone capabilities. With other frameworks like React Native and Kotlin Multiplatform, you have to have a companion application built side by side with the watch application. In our case the companion app was already created in .NET MAUI, so we needed a framework that allows us to build the application solely standalone. Lastly, it has bluetooth packages available like flutter_blue_plus and flutter_reactive_ble for data transfer from the watch to the phone over Bluetooth Low Energy.

Another language being used is Kotlin. This is being used to implement Android specific features such as bluetooth permissions, foreground services, and background processes. We are focusing mainly on the Android side and then expanding to IOS later down the line, so Kotlin

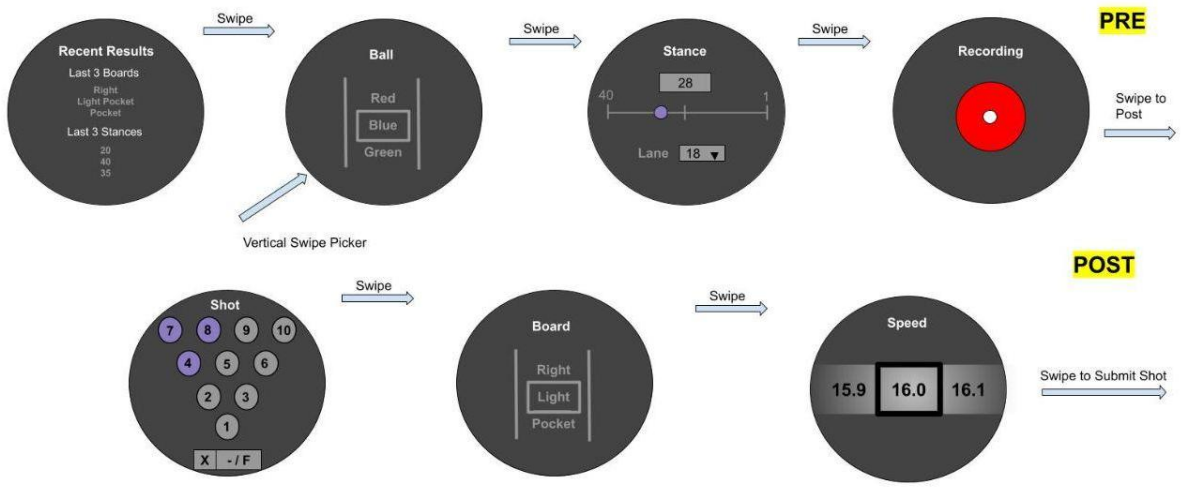
integrates seamlessly with the Android ecosystem within the Flutter project, providing access to platform APIs and certain services and libraries.

The IDE we are using is Visual Studio Code. This is because in VScode you can work in the native languages for Android and IOS without having to use separate environments. Another main advantage of using VScode is the extensions. We will be using an extension called Android/IOS launcher, this allows us to run our code through an emulator for Galaxy watches and Apple watches straight from VScode. Otherwise we would have to launch Xcode or Android Studio to test the different platforms.

For connection and data passing between phone and watch, we will be using Bluetooth Low Energy. Bluetooth Low Energy is designed for communication with low-power, resource-constrained devices like a smart watch. The advantage of using this is its ability to maintain the wireless connection while preserving battery life of the watch, and its less overhead for a smart watch with little cpu power. The limited packet size for transferring data is a constraint when transferring large amounts of session data.

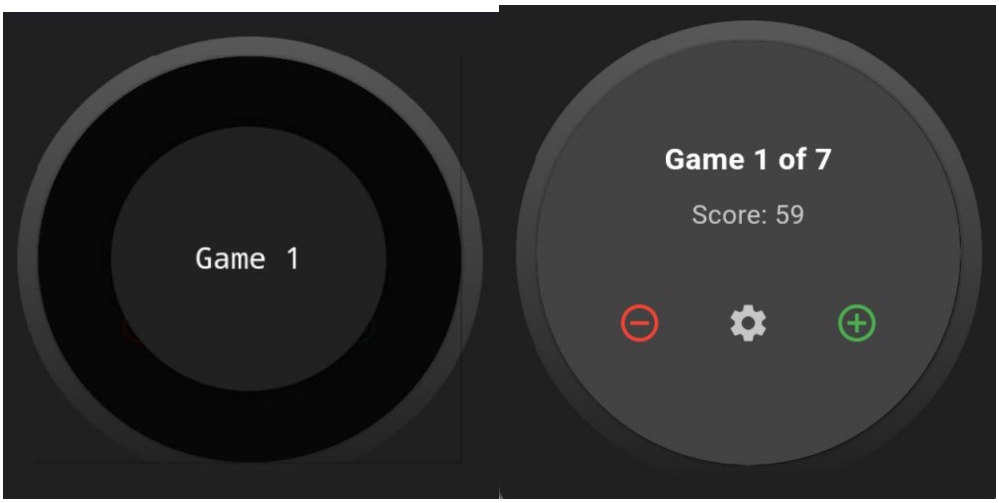
UI

New Shot UI

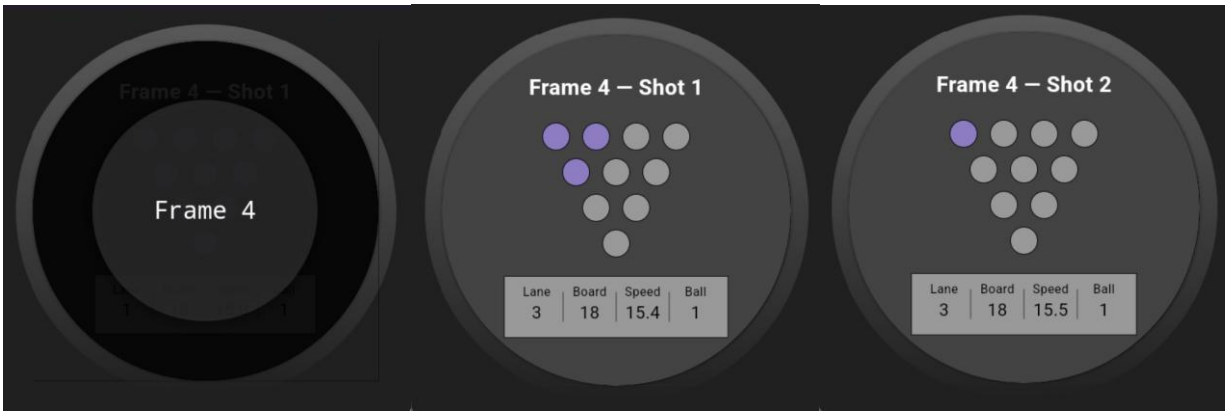


Existing Watch UI

Game



Frame



Shot and Bluetooth

Ciclopes

Members

Andrew Olvera, Zach Cox

Github Repo

https://github.com/YCP-Rev-Metrix/Ciclopes.git

Current State

Previous Research

- YOLO segmentation models can be fine tuned to segment both the ball and the lane
- The lane segmentation allows for computation of a homography transformation matrix which can be applied to the ground contact point of the ball to get (x,y) coordinates of the ball on the lane per frame that the ball is visible
- Interprolation of the ball points allows robust rendering and calculations even with bad frames

Proposed Additions

Ciclopes is picking up from last semester with the majority of the problems identified previously solved and tested in simulation. From this point we must implement existing functionalities into the cellular application and backend/cloud stack. Also, the proposed pose rendering feature is ready to be implemented adjacent to the Ciclopes features, this involves only inference setup and rendering on the frontend as the model is open weight and ready to use.

As a stretch goal we would like to implement on device inference of the computer vision model. This would allow a pre-recording workflow where the user gets direct feedback if the camera is in a good position or not, and to choose the lane that they are using which will lower the probability of user error. Another stretch goal that is an open problem is allowing the user to use Ciclopes without a smart dot module. Currently the smart dot module is used to identify when the ball hits the ground which is a requirement for the planar homography, competing applications specifically LaneTrax does not have this requirement, allowing a larger user base.

We would like to identify how to accomplish this and implement it in our stack to allow use of Ciclopes without the smart dot.

Problem

- Edge cases:
    - Multiple balls and lanes in view of the camera
    - Homography requires the ball to be in contact with the lane
    - Ball exits the lane into the gutters mid video
    - Frame corruption and frames with missing ball / lane
    - The distance of each pixel at the beginning of the lane distorts as the ball moves down the lane introducing variance in calculations toward the end of the shot
    - Real world noise and variance
- SAM3D Body integration and reduction of jitter in pose rendering over multiple frames
    - Jitter comes from inaccuracy in processing of each frame causing watching the render as a video corresponding to the frames to seem jittery

Major Technical Challenges

When using computer vision for applications such as the one we are doing. The machine learning work and base computations are the easy part. The hard part is developing an algorithm which is sustainable in a noisy environment when segmentations may be bad, frames may be corrupted, there are multiple detected objects in the frame, and endless edge cases that may come up in distributed real world use.

Also, a core problem facing the integration into the cellular application is inference of the model on the backend and not on device. The decision to not do model inference on device comes from the requirement to target the different architectures based on platform, as well as older models do not have tensor cores or neural network acceleration hardware making on device inference slow and costly to battery life. This introduces the requirement of saving recorded videos in a bucket in the cloud and querying to the GPU containing instance (Ciclopes-API). Data traveling to the different nodes before inference and processing even happens introduces latency that could break the user experience. Therefore efficiency in processing on the Ciclopes-API instance is of utmost importance, and will be a focal point of development.

Technologies & Why

- PyTorch for model training, inference, and evaluation
    - Ciclopes segmentation model + SAM3D Body
- YOLOv26 derivatives as a pretrained model to break off from, small fast and robust
- Core util functions with OpenCV2 implementation, calibrated and well tested
- FastAPI for implementation in application stack

UI

This semester we will need to integrate statistics and rendering UI functionality into the application. This will be done in the form of components that can be brought up as a modal and has side swipe functionality to view different features. This will include a tabular view of shot statistics, a 2D render of the shot trajectory down a rendered lane, and a 3D rendering component showing the pose of the bowler in a draggable space with the camera orbiting the rendered pose. The 3D component will also allow for dragging a slider across the frames, as well as playing at

video speed to view the shot mechanics in action. The visual design and UX integration into the existing application workflow is still to be decided.

Team Pi

Members

Matt, Brown, Gabe Manero, Hunter Wolfe

Github Repo

https://github.com/YCP-Rev-Metrix/BallSpinner-Controller-v2

Current State

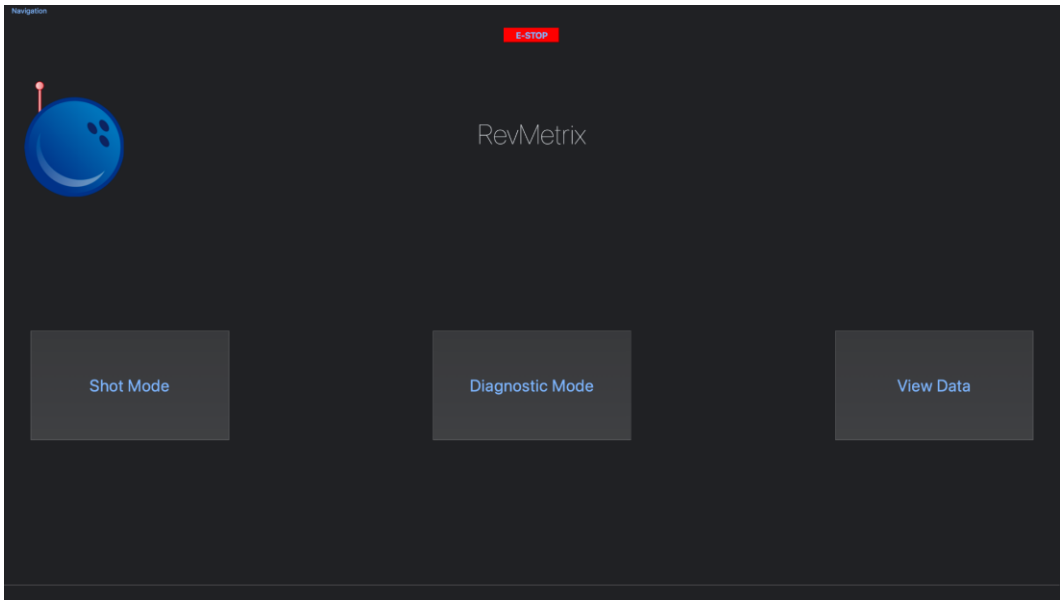


Figure 1: The current Ball Spinner Controller's home page

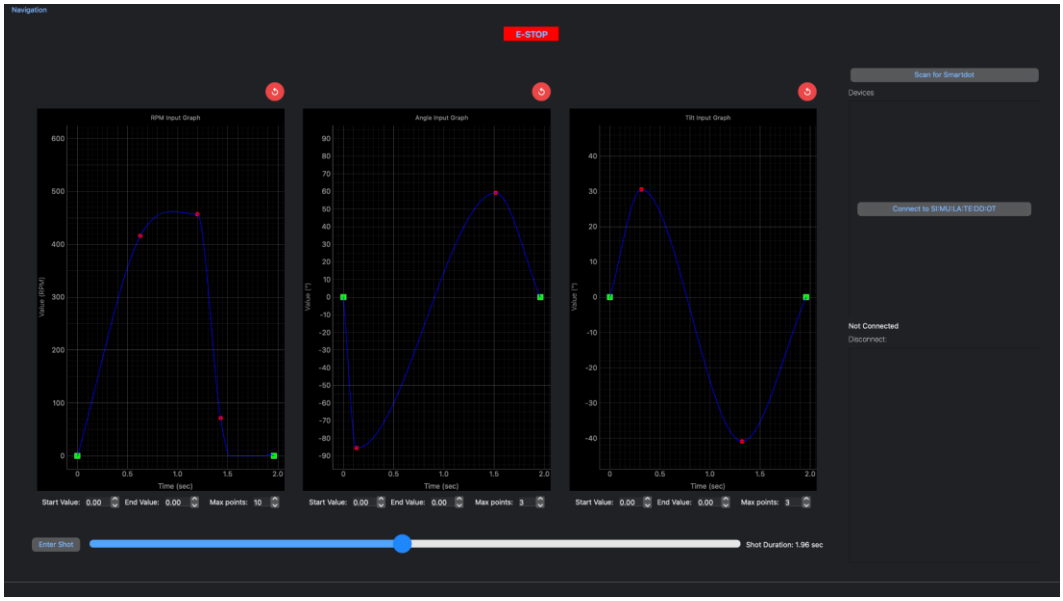


Figure 2: The current Ball Spinner Controller's shot page with a shot entered

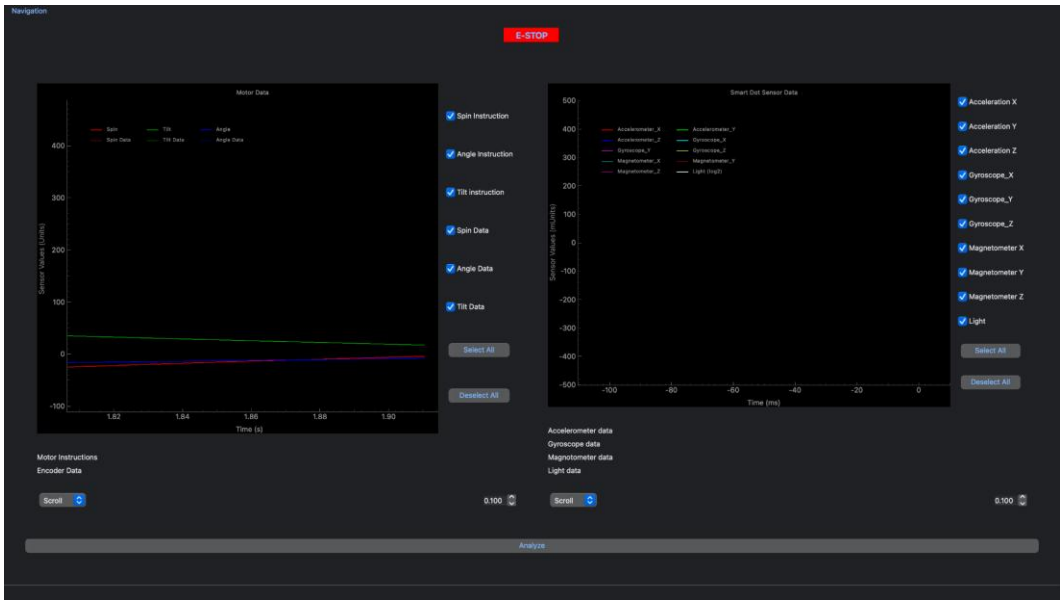


Figure 3: Ball Spinner Controller's shot page running the shot shown in Figure 2 with simulated motors

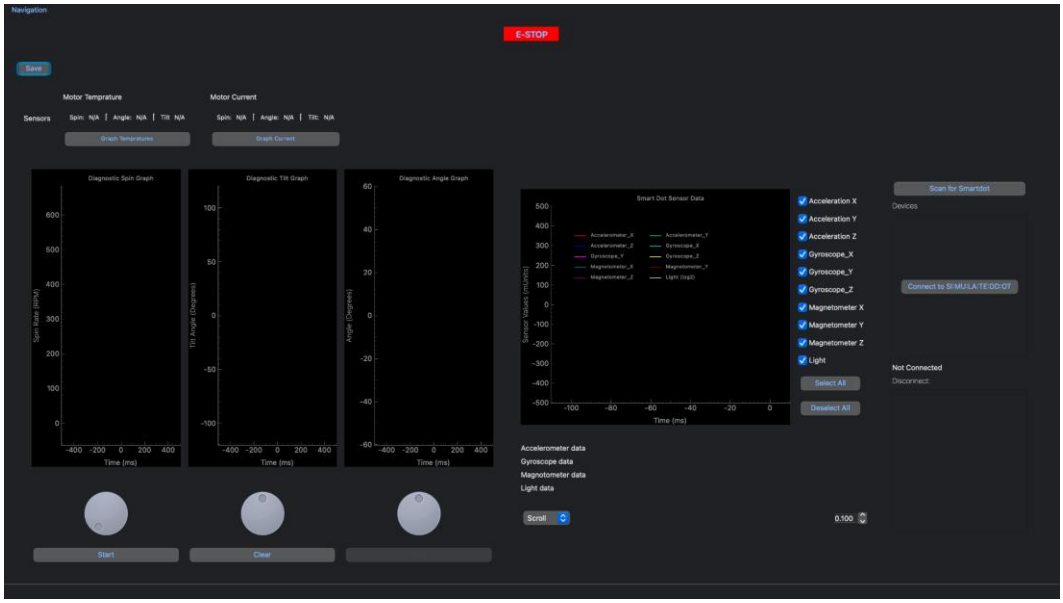


Figure 4: The Ball Spinner Controller's diagnostic mode page

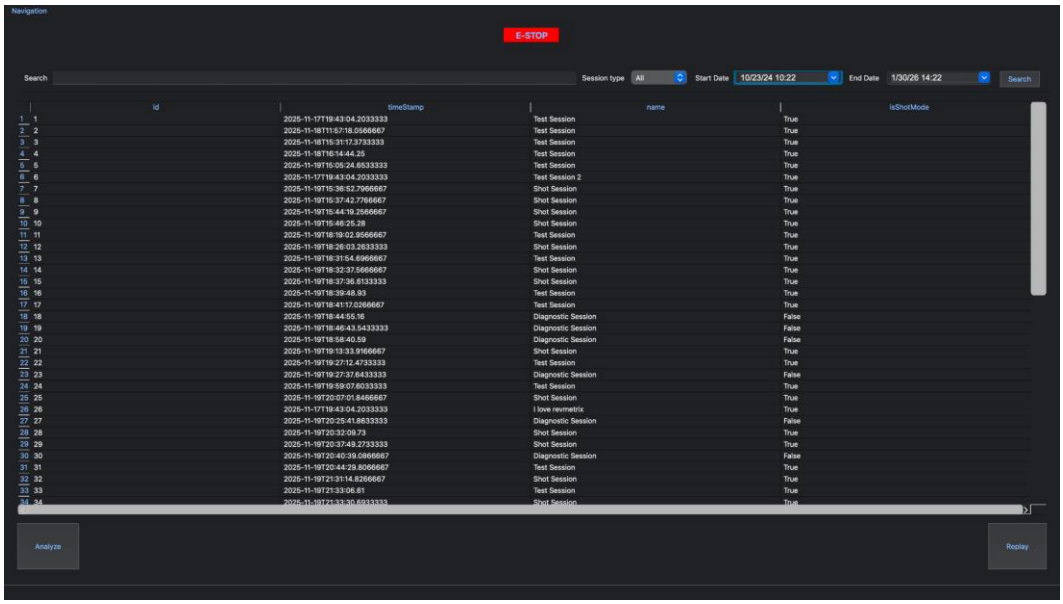


Figure 5: The Ball Spinner Controller's data view page

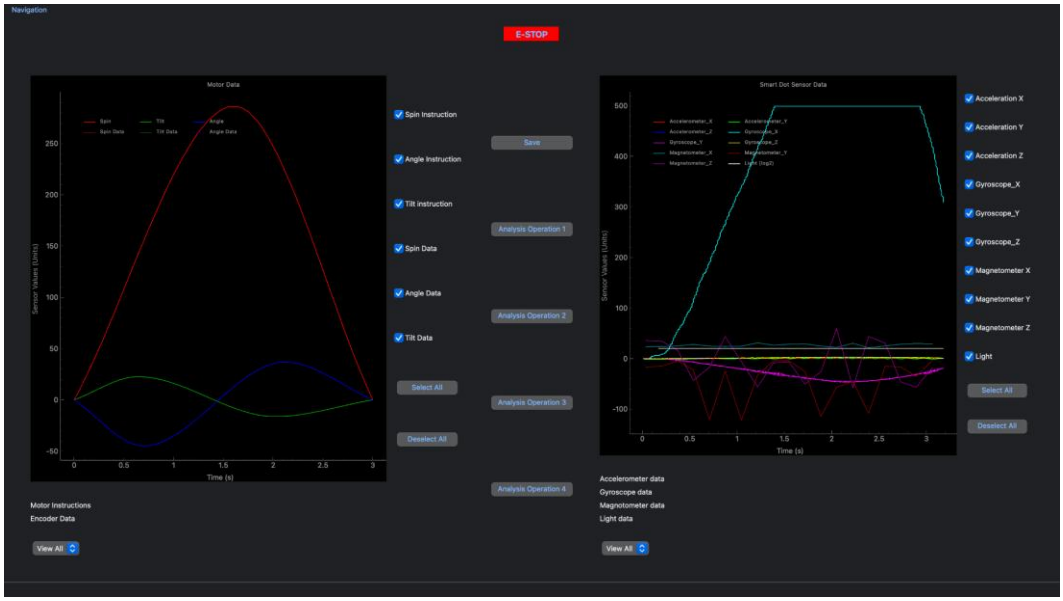


Figure 6: The Ball Spinner Controller's analysis page, analysing a shot collected from the system.

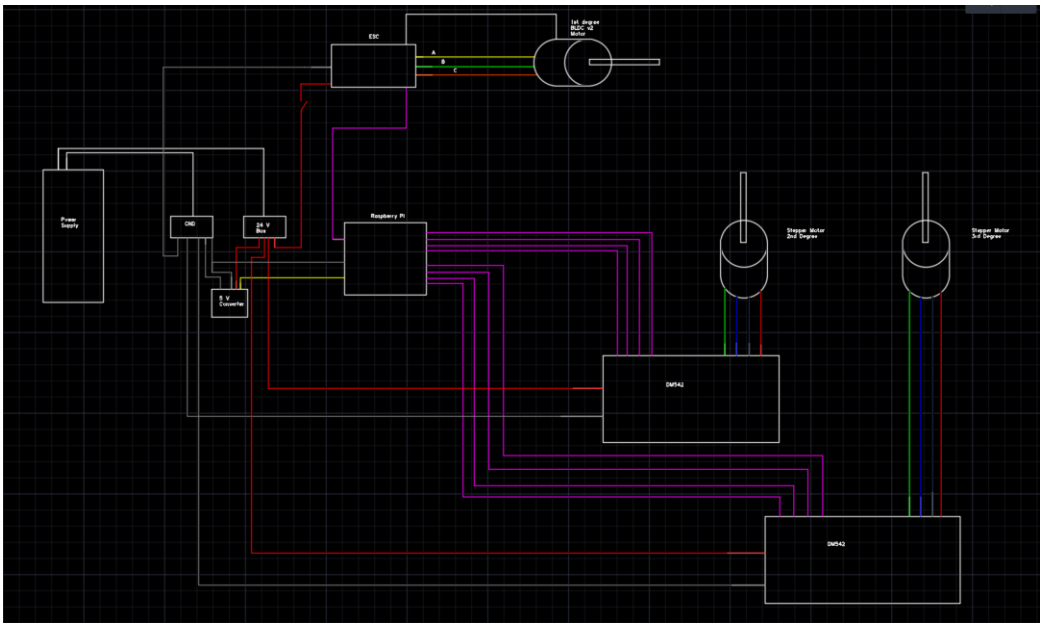


Figure 7: The Ball Spinner Controller's Electrical System Overview

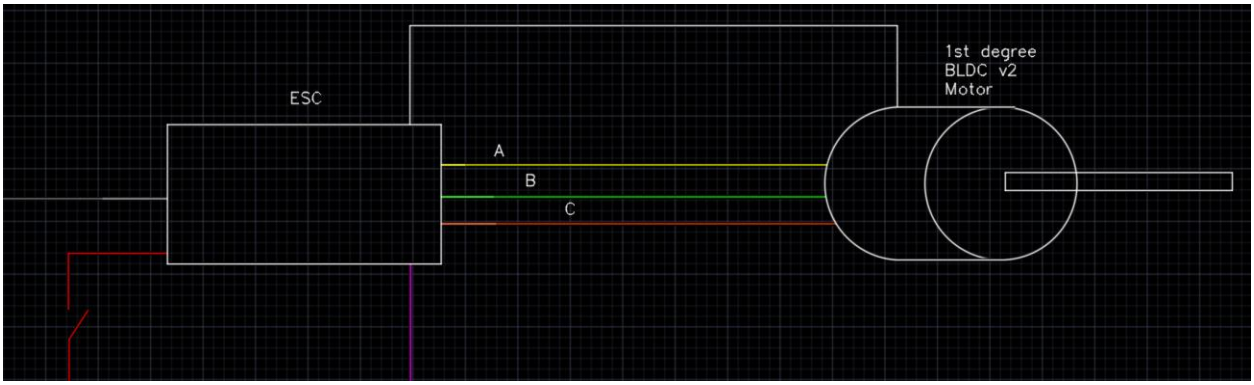


Figure 8: 1st Degree Bushless DC Motor and VESC

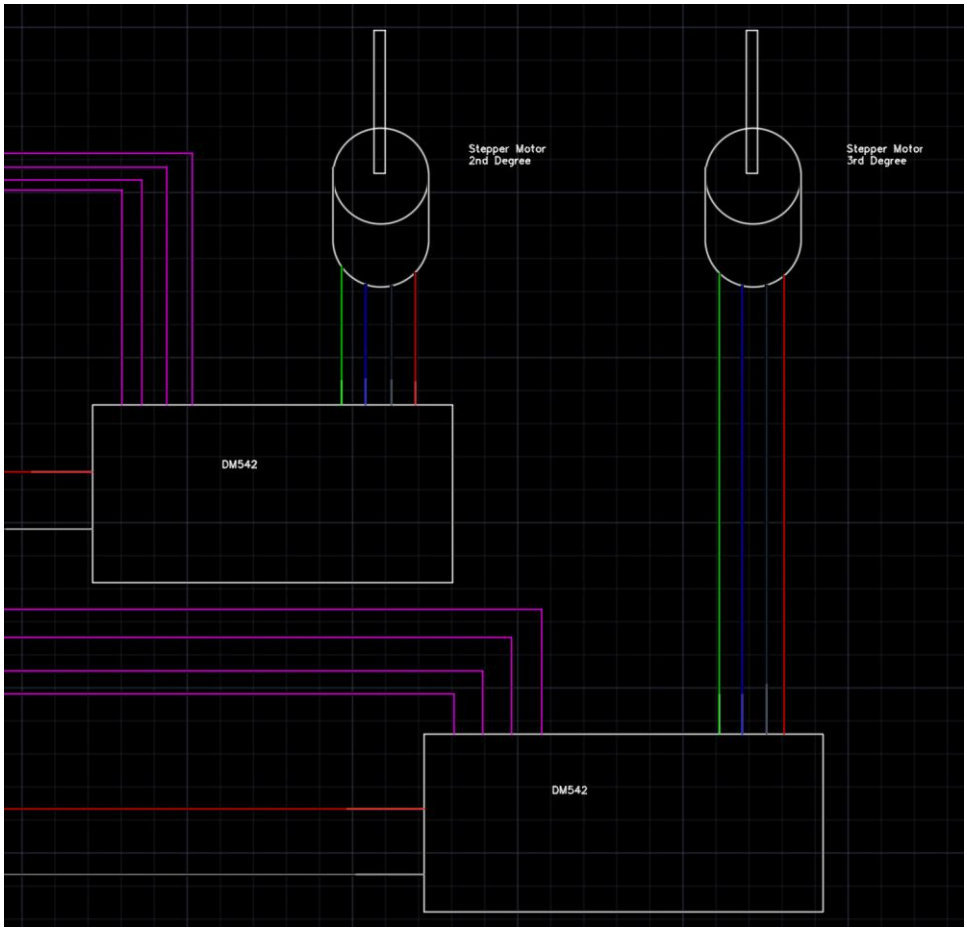


Figure 9: DM542 (x2) and Stepper Motors

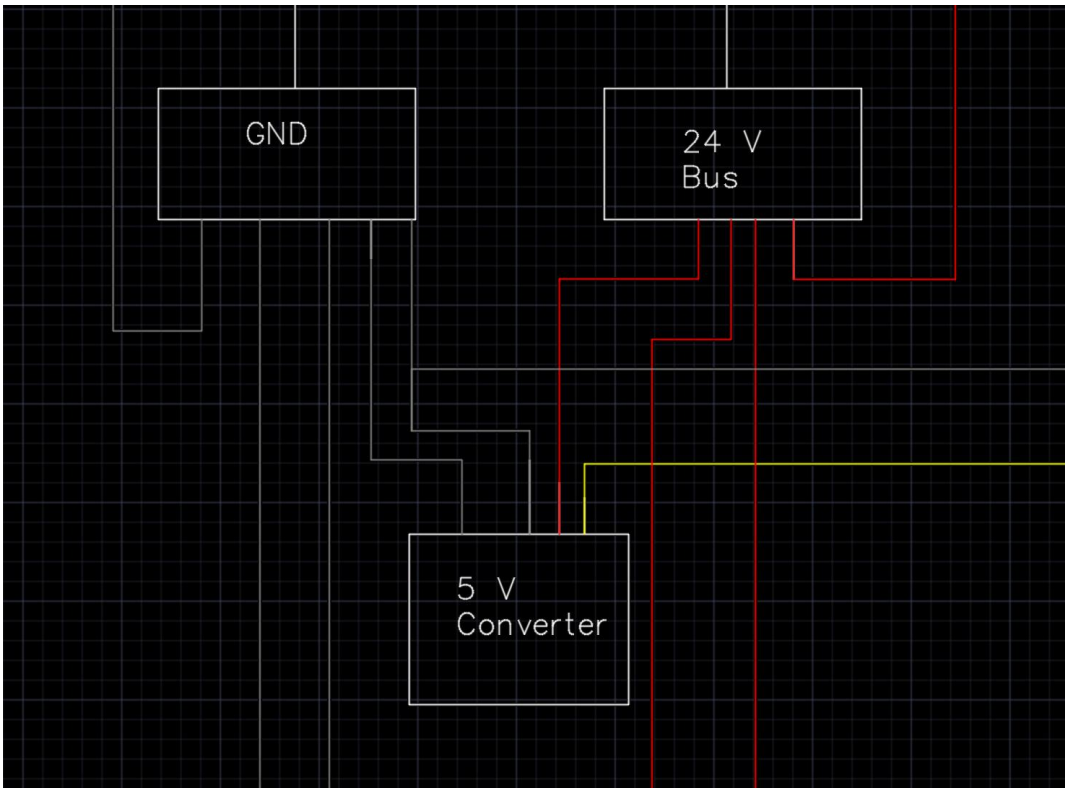


Figure 10: 24 Volt bus and 5 Volt Converter

The current state of the Ball Spinner is a functional system with several subsystems defined and partially implemented

The electrical architecture is centered around a 24 V DC power supply capable of supplying up to 6 A, with an expected maximum load of approximately 5 A during operation. Power distribution has been designed to supply a Flipsky Mini VESC for the brushless motor control as well as two external DM542 step motor drivers. These components include their own internal protection features, allowing for more focus on upstream protection and improving system performances.

We have the Smartdot connected to the Raspberry Pi 5 via BLE. The Pi is able to store and display the Smart dot data as part of the session. Through the Pi the user can control all 3 motors. The Pi uses a PyQt6 user interface that can be interacted with via the touchscreen attached to the

Pi. The user can create a shot session that is meant to simulate a real bowing shot. The user can adjust the curves and duration using the controls shown in *Figure 2* to determine the characteristics of the simulated shot. Additionally the user can use the controls shown in *Figure 4* to run a diagnostic session that allows the user to directly control the motors speeds and positions. Both of these session types can be sent to the cloud and retrieved for analysis. Additionally both forms of sessions are able to be run with either real or simulated motors and Smartdots.

Problem

We need to create a Ball Spinner Controller (BSC) that allows the ball spinner to rotate a Smart Dot Module along 3 degrees of freedom in order to generate reliable and known data for algorithmic research. With the Smart Dot Module two inches from the axis of rotation, we need to control the rotation of the Smart Dot Module up to 600rpm while controlling the axle tilt up to 45° vertically and rotation up to 90° horizontally. We will need to create a single shot mode which is able to accept inputs to control the three motors responsible for these degrees of freedom in order to emulate a bowling shot, with an adjustable length of approximately 2-4 seconds. Additionally, a continuous testing mode must exist which will allow real time individual control over the three motors. The SmartDot should be able to connect and display live data in both modes.

With the new program being fully run on a Raspberry Pi 5, a larger touch screen is necessary for smooth operation, as well as a bluetooth keyboard/mouse for control. After

performing a shot in the single shot mode, or by selecting a previous shot from the cloud, the user should be able to review the data recorded and perform analysis in the form of simple digital signal processing on their simulated shot. Examples of DSP that will be considered are FFT and wavelet analysis.

Major Technical Challenges

- Having the ball spinner emulate a human shot accurately
- Creating a “Hard Stop” button that is able to disperse all the kinetic energy that is produced during the shot.
- Having comprehensive and accurate testing
- Optimizing the user interface for touch input.

Technologies & Why

- Raspberry Pi 5 - with a 2.4GHz processor and up to 16 GB of ram, it will be able to handle our DSP, display our user interface, and control our ball spinner hardware.
- Large Touchscreen Monitor - Provides the space for a cleaner, easier to use user interface.
- PyQt6 - This is the library we are using to make a clean user interface.
- Qt Designer - This application enables us to create the user interface in a graphical format that can then be displayed by PyQt6.
- NumPy - A common data processing library that includes many math functions, including fast fourier transforms as well as first and second derivatives.
- PyWavelets - A library used to perform wavelets on the collected data for data analysis.
- KiCAD - Open-source Electronic Design Automation (EDA) software suite used to design electronic circuits, create schematic diagrams, and lay out Printed Circuit Boards (PCBs).
- AutoCAD - a 2D and 3D computer aided design software application for document and planning for implementation .

UX

- Improved touch controls: Make the buttons larger and easier to use on the touch screen.
- Clarity: Update names and add labels to explain items better.
- Ease of use: add additional buttons for common navigation.

Notes:

Ball spinner app run on startup

Update housing on controller

Temperature & current sensors on motors

Shot initial values and new graphs

Physical Design Team

Members

Joseph Downey, Gavin Wentz

Current State

The ball spinner consists of three independently operating motors representing the three degrees of freedom of a spinning bowling ball, as well as a SmartDot mounting arm which replicates the dimensions of a bowling ball. This assembly collects data from a pair of SmartDot modules attached to the mounting arm. The first degree of freedom is housed within a protective plexiglass frame to prevent interaction with spinning components during testing.

Problem

The main challenge facing the Physical Design Team is the merging of the three degrees of freedom into a single assembly. This will require the manufacturing of a system of metal brackets and supports capable of mounting the motors and some secondary components at the correct positions relative to each other. Another issue that the team faces is that of safety and heat dissipation, being a concern as the current model has limited safety features and doesn't allow for airflow to dissipate heat generated by the motors. These are all problems that the Physical Design Team must address to ensure an operational and robust system for accurate data collection and analysis.

Major Technical Challenges

The design for the ball spinner, currently only a 3D model, must be manufactured as a physical object. Issues with the current design may only become clear once it has begun to be constructed. The implementation of ventilation will be a unique challenge, requiring concepts extending beyond the solid mechanics of the ball spinner design. As the ball spinner approaches a more final layout, more aspects of the Ball Spinner Controller and ECE team's designs will need to be integrated into the physical ball spinner design.

Technologies & Why

The Physical Design Team will continue using SolidWorks as a primary design software due to its robust 3D modeling capabilities. Once beginning fabrication, the team will use Drill Presses, Band Saws, CNC routers, and other tools to facilitate efficient assembly. 3D printing software

and machines will be used to prototype new designs and produce non-critical parts with complex geometries.

UI

The only UI integrated with the ball spinner will be the HMI from Team Pi. Other UI considerations are ease of access to the electrical hardware, motor mount modularity, and repairability. Due to the nature of the physical system, all other UI components are handled by the other teams.

Designs (3DOF)

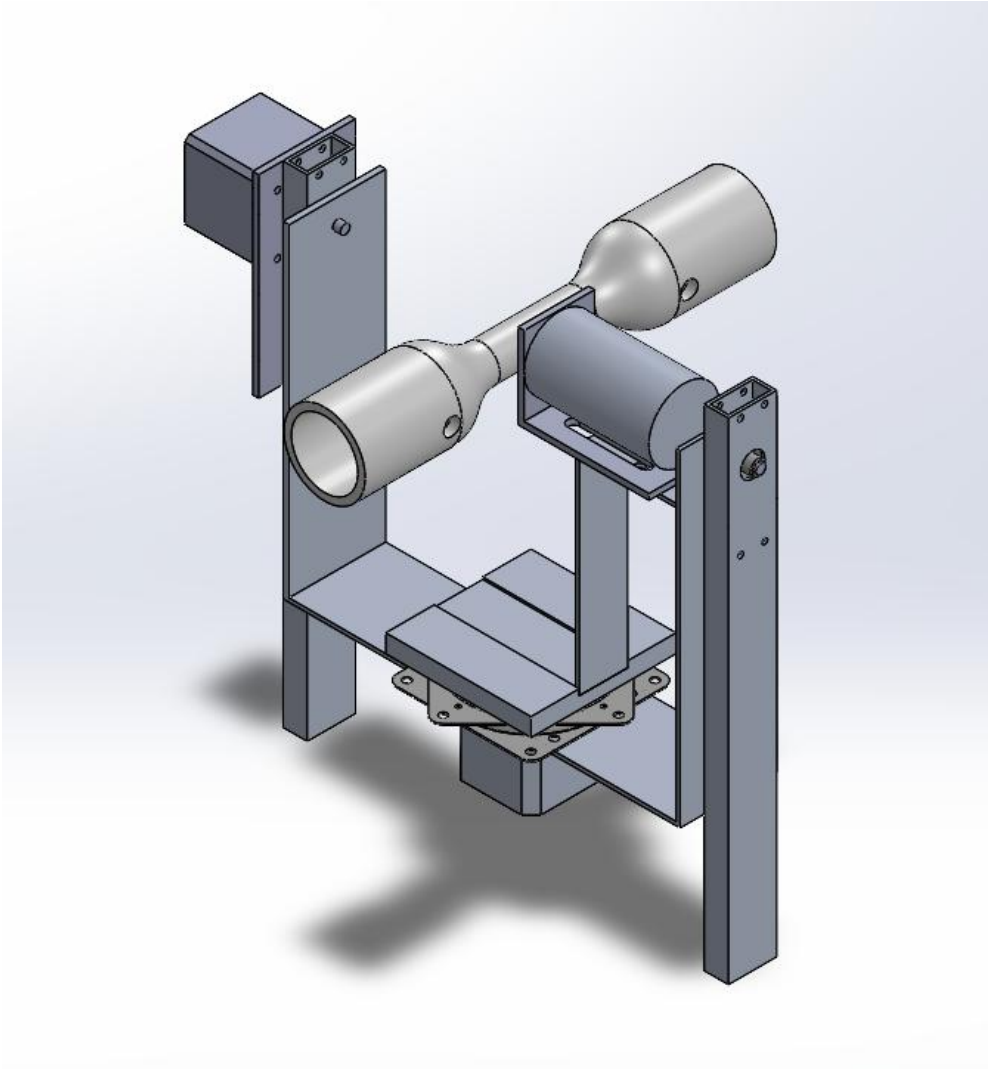


Figure 1: Three degrees of freedom using 3-axis gimbal