RevMetrix Project Proposal

Fall 2025

Team Cellular

Members

Josh Byers, Charles Carroll, Zach Cox, Emmet Larson, Jakeb Nielsen, Andrew Olvera

Github Repo

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

Current State

The mobile applications main use is to save and track shot data for a game of bowling. Currently, users are able create an account and sign in to access the features of the app. Once logged in, users can navigate to a few other pages. The ball arsenal page allows users to add balls to their arsenal and input the balls name, serial number, weight, and core type. The session list page is used to create sessions. After creating a session, games can be created for each session. To track a game, users will navigate to an instance of the shot page. This page is the core data input for the app. It displays a view of all frames under the specified game and shows pins knocked down, as well as pins left standing. The page also has buttons representing 10 pins, foul, gutter, spare, strike, comment, and next. These buttons are used to track data and progress through a game. The Bluetooth page is currently just a placeholder. The Data page allows users to create and manage establishments and events. The API Test Page is a proof of concept for connecting the mobile app to the database hosted through Digital Ocean. Finally, the Account page lets users view and edit their account information.

Problem

The problem being solved is to be able to accurately track a bowler's shot data on their phone while also tracking the balls metrics using the smart dot module. The information will be saved so that it can be queried by bowlers to see statistics on how they bowl.

Major Technical Challenges

The first major challenge for this semester will be connecting the smart dot module to the mobile app. This will require us to port a Java API to C#. The second major challenge will be fully connecting the cloud database to the app. The third major challenge will be connecting to and communicating with a smart watch.

Technologies & Why

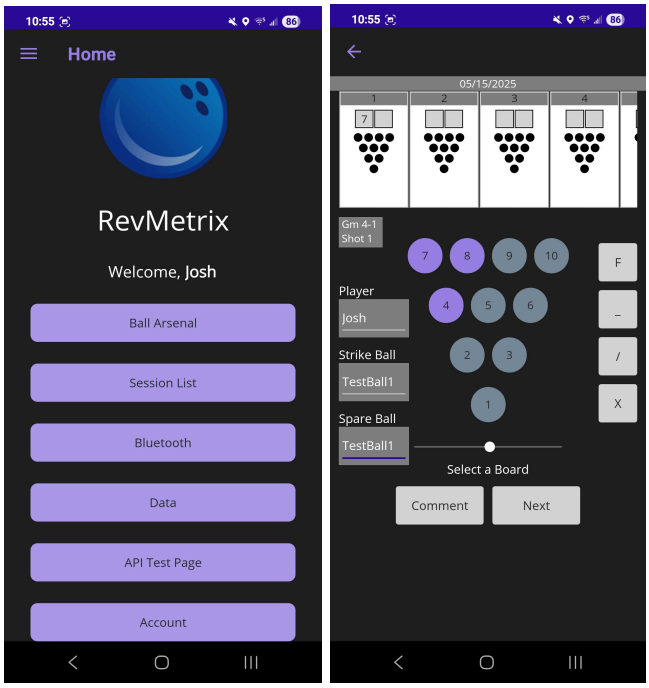
The mobile app is being developed using .NET MAUI (.NET 9) because it allows us to develop across multiple platforms. It is also being used for the ball spinner application.

C# is being used as the main programming language as .NET MAUI primarily uses it for functionality.

XAML is being used for the UI as .NET MAUI uses it.

SQLite is used for the local database because it’s used in many applications, including mobile operating systems, desktop applications, and web browsers.

UI



Team Smart Watch

Members

Josh Byers, Charles Carroll, Zach Cox, Emmet Larson, Jakeb Nielsen, Andrew Olvera

Github Repo

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

Current State

This is a new part of the project and there are currently no active prototypes for the smart watch application. The phone application will be used for functionality, but the actual watch app

will need to be built from the ground up based off of the phone application due to the need of a specialized development environment.

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

We will be developing the watch application in Android Studio which is a new environment to the project. This will require the entire team to learn Kotlin, the primary language used in android apps, as well as how to use the Android Studio environment.

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.

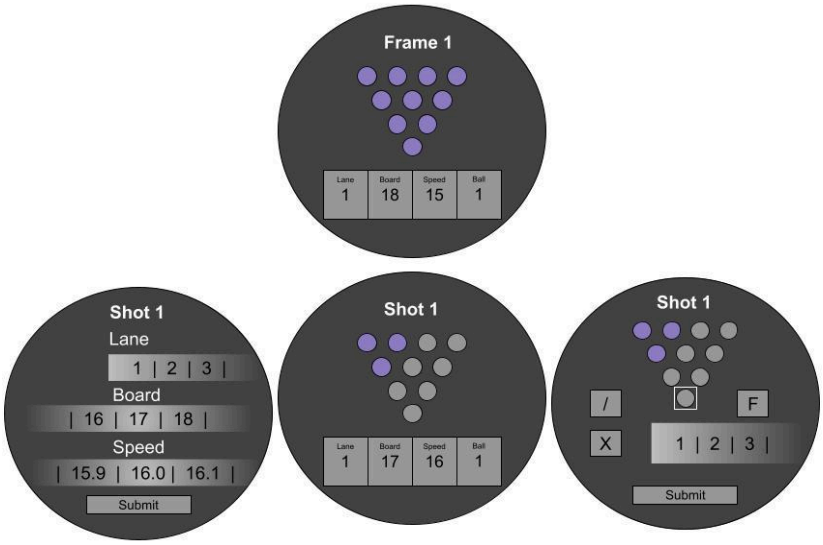
Technologies & Why

The watch application will be developed using Kotlin. This is because that is the main language used to develop many android applications and it has a close relation to java which should help with compatibility.

The IDE of choice is Android Studio. This is because it has a lot of features that help build and package watch applications. It is the most frequently used IDE for developing android watch applications.

On the hardware side, we need a separate smart watch for testing and development purposes. For this project the Samsung Galaxy Watch 7 would be the best option because of the size and affordability. This watch can be found at Best Buy for around 200 dollars.

UI



Ciclopes

Members

Andrew Olvera, Carson Mack, Zach Cox

Github Repo

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

Current State

Previous Research

- Using YOLO models pretrained on Ultralytics object detection with round bounding "box" allows detecting the ball in the frame
- Fine tuning on sim data allows fitting and evaluating fast

Problem

Moving from ball detection to metric calculations

- Requires detection of lane, 2 options
    - Segmentation, meaning the polygon of the lane visible to the camera is found in the frame and math can be done to get the relative positions of the detected ball to the lane. More "visual" heavy meaning sim to real is more complicated
    - Ball detection and planar homography, only DL application is detecting the ball using the round bounding "box" and using open cv2 calibrating the homography equation with the lane to find the points. This is then able to be used to go to a

BEV (Birds Eye View) representation for more robust calculations over the multiple frame dynamics. Much more robust to the visuals meaning sim to real is much easier

Major Technical Challenges

- Clean training and evaluation pipeline with sim data including domain randomization for more generalization
- IsaacSim implementation with single lane and multi lanes with noise (other balls / moving objects other than your lane), as well as labeled data export
- Compiling PyTorch model to ONNX for “production”
- Calibration of homography and lane point detection using cv2
- Setup in FastAPI and torchserve mocking production on a GPU cluster
- Streaming video from phone to microservice to start processing right away to minimize latency
- Phone aspect ratio is odd for CV which may cause some issues

Technologies & Why

- PyTorch for model training and evaluation
- IsaacSim for sim, synthetic data generation, labeling, and domain randomization for sim to real
- YOLOv8 derivatives as a pretrained model to break off from, small fast and robust, trained on COCO ball data allowing circular bounding “boxes”
- Core and util functions for cv2 implementation, calibrated and well tested
- FastAPI for implementation in application stack

BackEnd

Members

Zach Cox, Josh Byers, Emmet Larson

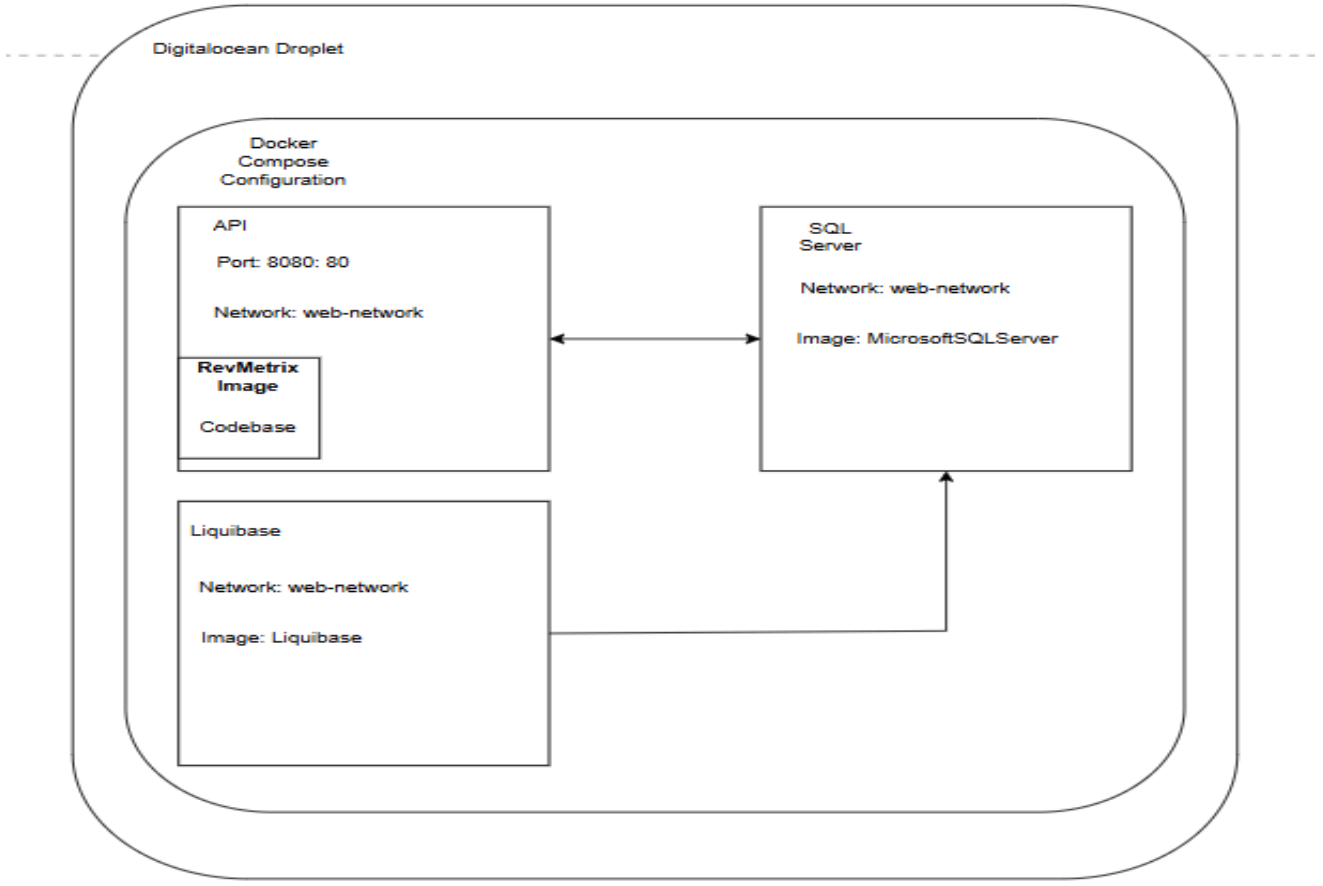
Github Repos

https://github.com/YCP-Rev-Metrix/BallSpinner-Cloud

Current State

The current state of the Backend is that we use Digital Ocean and Docker to run our cloud network. This cloud network makes it possible to connect the Ball Spinner App to connect to our Microsoft SQL database. We have a droplet on Digital Ocean that hosts a Linux-based virtual machine capable of running many services. We use Docker on the droplet to run our API, SQL server, and Liquidbase. The picture below shows a high-level overview of our server configuration. The droplet runs the Docker service containing the API, SQL servers, and Liquibase containers. The image defined for the API also contains the code for the API. The port definition maps port 8080 on the droplet to port 80 running on the API container. In order to perform database transactions, the API container needs to access the SQL server container. It does this through the web network managed by Docker, where the two containers can communicate with each other through their container names. This setup also enables the API server to communicate with the SQL Server over a private network, ensuring that the SQL Server remains isolated from public access. The Liquibase container is responsible for managing and executing database migrations within the SQL Server environment. By running as a separate

service, it ensures that schema changes, version control, and updates can be applied in a consistent and automated manner. Like the API server, it communicates with the SQL Server through the same private Docker network, maintaining a secure and isolated environment where database access is restricted to trusted internal services only. In essence, Docker is the tool used by the RevMetrix team for server configuration on the Digital Ocean droplet due to its vast array of useful features.



Problem

The Ball Spinner application and the Mobile app both need functionality to save and transfer user data between devices. Additionally, future plans require the ability to export this data from the applications and store it in an external system.

Major Technical Challenges

The main priority is rebuilding the database by combining the Mobile App and Ball Spinner App databases. Ryan and Zach have already created a database schema that will guide this process. Once the new database is in place, some methods in both the Mobile App and Ball Spinner App will need to be rewritten to accommodate the updated data structure. Currently, the Ball Spinner App has a build server that automatically compiles the application and runs tests. However, the Mobile App, the Watch, and Ciclopes do not yet have this capability. At the moment, no tests exist for these applications, so we will need to create test suites and ensure proper build automation. Finally, we will continue to provide support for Ciclopes, the Ball Spinner team, and the Mobile team for any additional needs moving forward.

Technologies & Why

The current backend is built using the ASP.NET server architecture. We're continuing to use the existing platform from the previous semester as it integrates nicely with the MAUI app (both using the .NET platform) and has lots of functionality that comes along with it in Microsoft's ASP.NET development environment. This includes debugging tools, package management, and configuration/deployment tools. ASP.NET also comes with an integrated API that the project is already taking advantage of. This service integrates nicely with the rest of our system and is specifically created for use with a .NET server. We're also using Docker for containerization so we can instance our database and code deployment. This will also allow us to have automated code testing and build checking whenever we make changes. For database management, we're using Liquidbase for integration and managing rollbacks. Many of these

technologies have already existed in the project from previous semesters, and we have chosen to continue using them as they still support all of the features we intended to implement.

UI

Though the backend doesn't have UI outside of the frontend part of the project, we do want to do a better job with test creation and management. The structure is already in place to have automated testing/building/deployment but not much has been done with it. A goal this semester is to improve the existing systems to have a wider range of test coverage, and ensure that code we push will always compile and deploy properly. This is possible through the use of Docker to create automated build tests whenever commits are pushed.

Team Pi

Members

Hunter Wolfe, Gabe Manero, Matt Brown, Brandon Woodward, Carson Mack

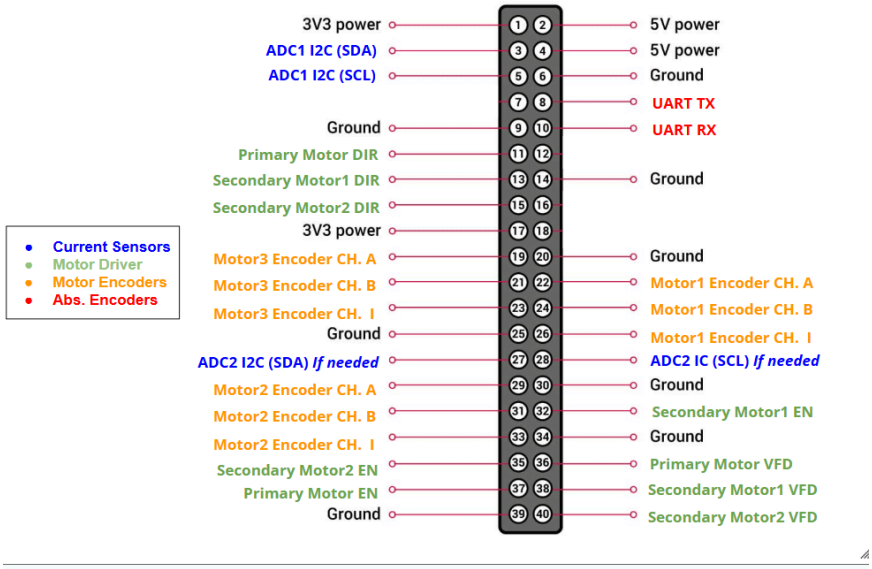
Github Repos

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

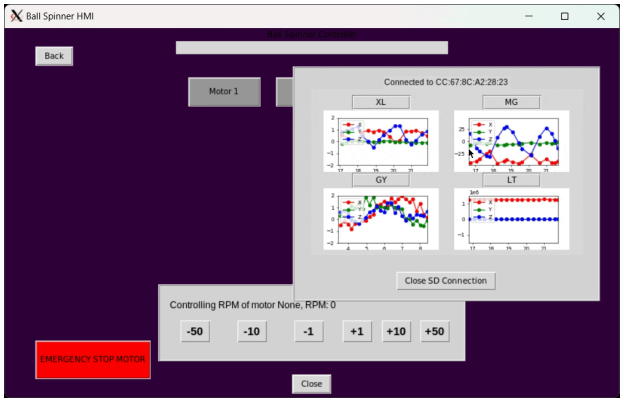
https://github.com/YCP-Rev-Metrix/BallSpinner-Application

Current State

The current state of the Ball Spinner includes a physical housing where a Raspberry Pi 4 and PCB reside. The GPIO of the Pi is utilized by the PCB and the motors. Below is a figure displaying the connections.



The Ball Spinner's software consists of two main processes that exist inside of the BallSpinner-Controller repository, the Ball Spinner Controller (BSC) has two modes. A server mode and local mode. The BSC Server Mode on the HMI and allows the user to connect to the BSA (Ball Spinner Application) and communicate via a TCP protocol. The BSA can then send messages requesting the BSC to control the hardware and collect data with the SmartDot. Local mode uses the touchscreen interface to control the motors, connect to the SmartDot, graph real time data, and display status information about the hardware. The BSC currently only controls two motors, both taking the exact same input.



HMI's live data graphs

https://drive.google.com/file/d/1pT5uPfhIw47HnuuPYa8zBCgIjMIihH2S/view?usp=sharing

The link above is a video of the entire working system

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 scope of the BSC increasing exponentially, the scope is intersecting with work on the Ball Spinner Application (BSA). To avoid performing duplicate work, as well as to

streamline the process, a combination of the BSA and BSC is in order. The BSC will adopt the responsibilities of the BSA by connecting to the Digital Ocean Cloud in order to send and receive data to our cloud database. The BSC will pick up where the BSA left off regarding bowling shot control. Revising and porting the existing (BSA) primary axis bezier curve input into a sine-like curve of rpm (primary axis of rotation). As well as going further to implement the remaining detailed shot control inputs, the secondary axis control graph and tertiary axis control graph. These shot control inputs could be stored as scripts and be easily run multiple times to replicate the same shot quickly. Preset scripts can be saved to the cloud and the data with their generated data sets connected via script foreign key and run count.

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

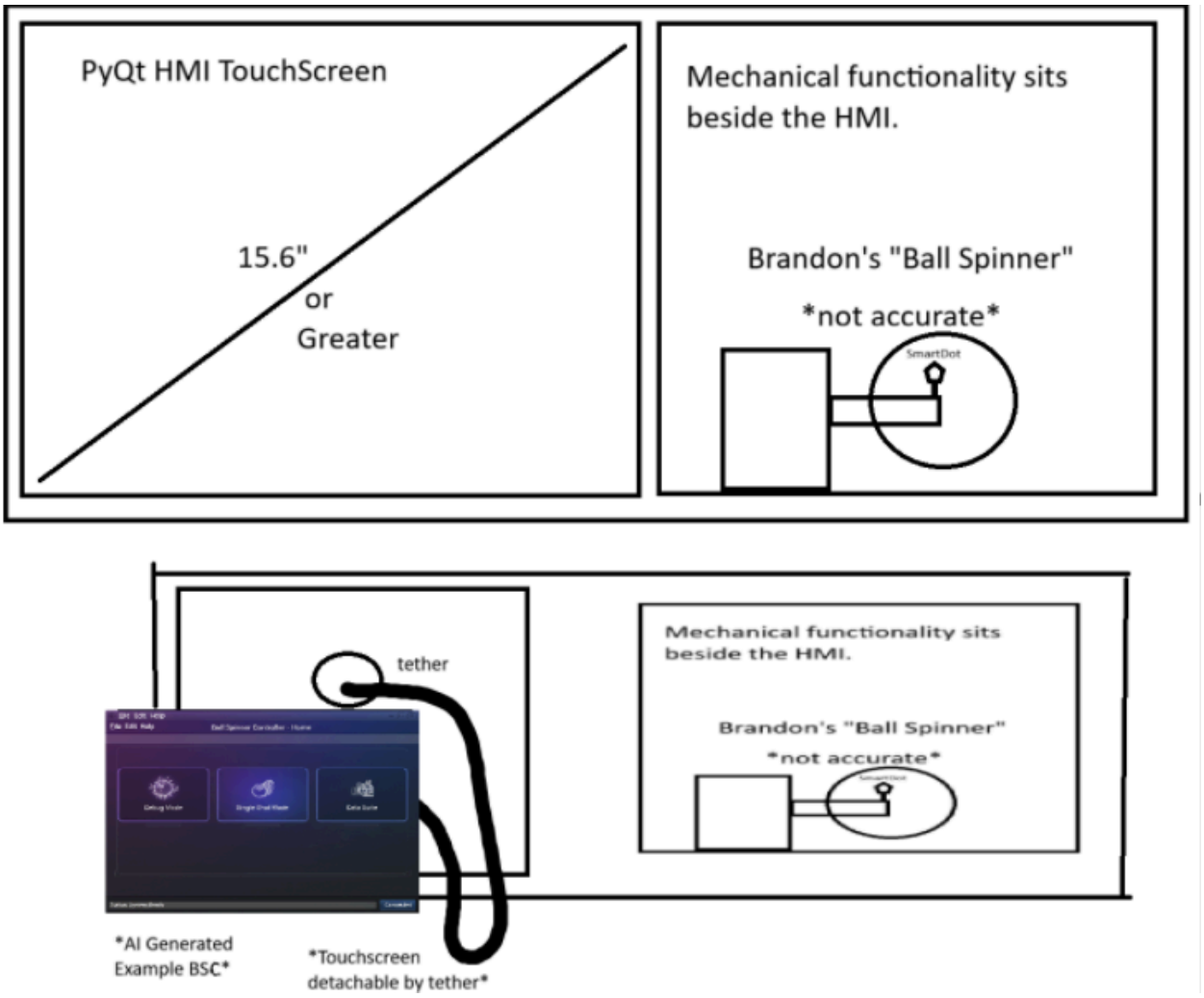
- Having functional communications between the Pi and the cloud
- Optimizing Local Storage vs Cloud Storage
- Transmitting data from smartdot module and motor sensors to the HMI display
- Having the ball spinner emulate a human shot accurately
- Allowing input for the shot's start, peak, and final RPM, time for all three motors and their special constraints.

- Finding out the correct amount of torque the axis and rotational motors need to create with out becoming destructive to the system
- Creating a “Hard Stop” button that is able to disperse all the kinetic energy that is produced during the shot.
- Robust Application with proper **testing**.
- Clean User Interface

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.
- PyQt - Moving away from the outdated tkinter, PyQt provides much greater support for more complex projects. Tkinter is a great tool for small scale projects and quick prototypes, but for a better user facing product and more robust, manageable system, PyQt is leagues ahead.
- DSP: For FFT, NumPy is a common data processing library that includes many math functions. This is not included in the project right now, but is a simple package inclusion. https://numpy.org/
- DSP: For wavelets, PyWavelets is a feasible library. This library is formed on C and Cython, improving performance for lower powered machines. https://pywavelets.readthedocs.io/en/latest/

UI



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

James Devine, Joseph Downey, Kyle Fankhauser, Gavin Wentz

Current State

The ball spinner currently is able to spin a foam bowling ball analog using a stepper motor and chain drive supported by a wooden frame. This assembly collects data from a SmartDot module attached to a separate apparatus that simulates the module's position within a bowling ball and mimics the foam ball's rotational speed through use of an encoder connected to a separate stepper motor. The ball spinner assembly is housed within a plexiglass protective frame to prevent interaction with spinning components during testing.

Problem

The Physical Design Team faces a number of issues with the current ball spinner model. The main concern being the current model is made out of scrap wood from the shop, resulting in an unbalanced, aesthetically displeasing model that is liable to break under certain circumstances. Another issue that the team faces is that of safety and heat dissipation, being a concern as the current model has no true safety features and doesn't allow for fluids to dissipate heat generated by the motors. These are all problems that the Physical Design Team must address to improve the 1st degree of freedom as well as when making the 2nd and 3rd to ensure the same standard of performance.

Major Technical Challenges

When iterating on the current design, finding ways to add more to the design while reducing weight and size will be a challenge. But these are major factors that the Physical Design team will need to contend with in order to add safety features as well as proper ventilation

Technologies & Why

The Physical Design Team will be switching to SolidWorks over from Onshape. This is being done to make use of the system's superior design features, such as FEA, system assemblies, and dynamic assembly testing. These are all functions that will be greatly beneficial for the Physical Design Team when it comes to designs and redesigns of digital models.

UI

The only UI integrated with the ball spinner will be the HMI from Team Pi

Designs (3DOF)

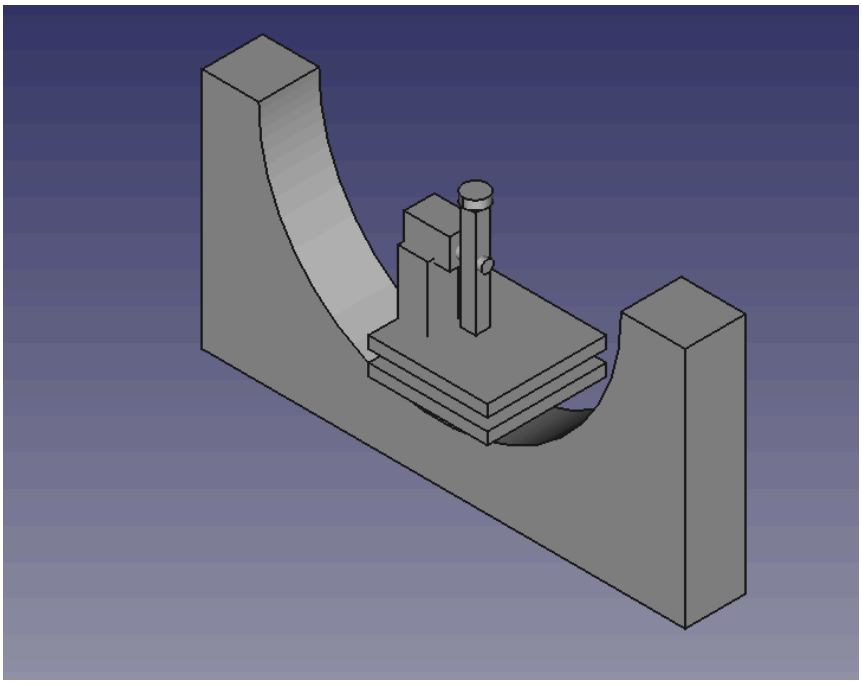


Figure 1: Three degrees of freedom with circular track

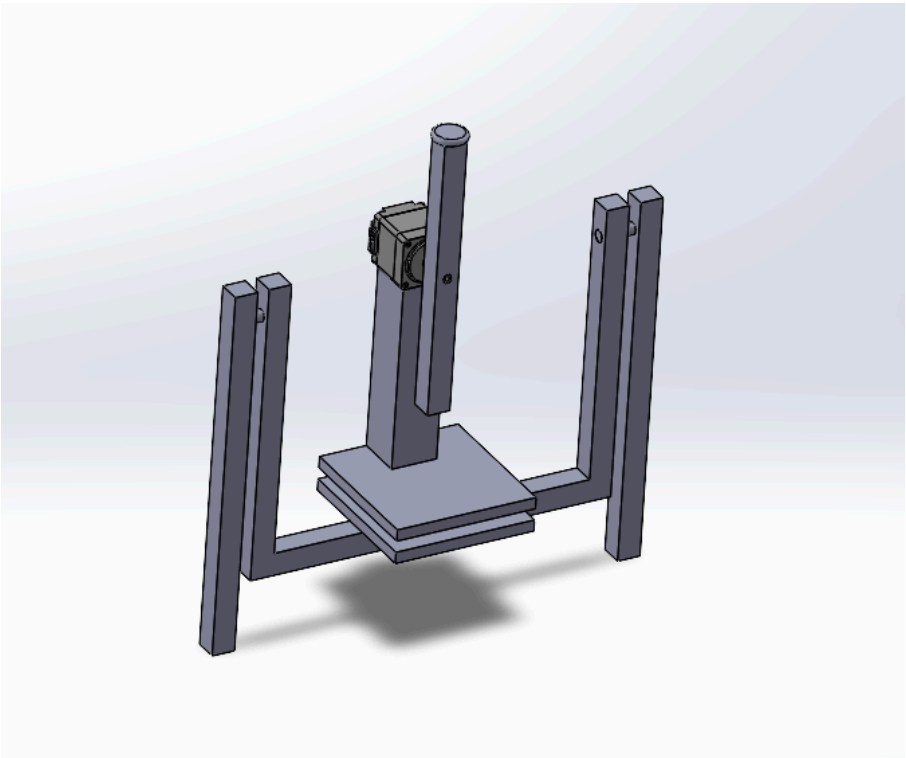


Figure 2: Three degrees of freedom using 3-axis gimbal