PongPal

Design Review | Team A8
Michael Bahner, Alex Kireeff, Seung Yun Lee

Use Case

Problem: People might not be able to play water pong for different reasons like sickness, lack of proximity, or disabilities

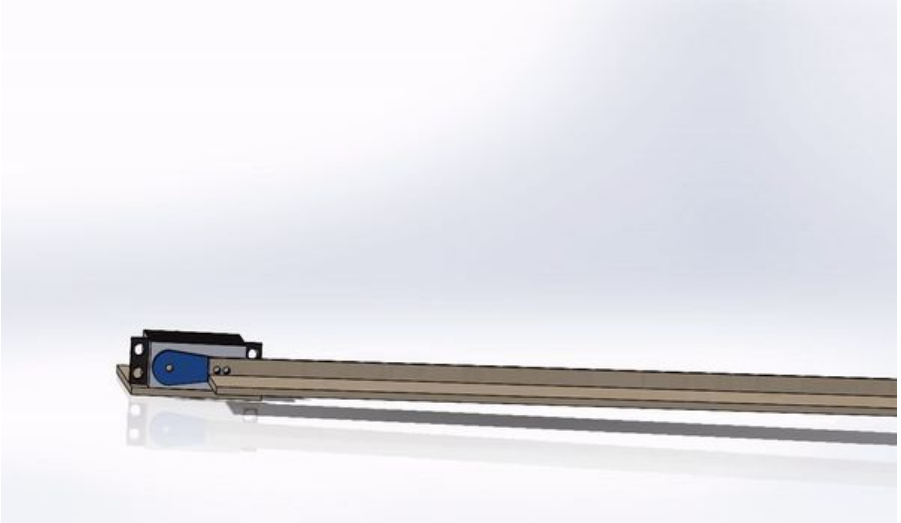
Solution: PongPal™ makes water pong more accessible by allowing one to remotely control the robot and get feedback

ECE Areas: Software Systems, Signals and Systems

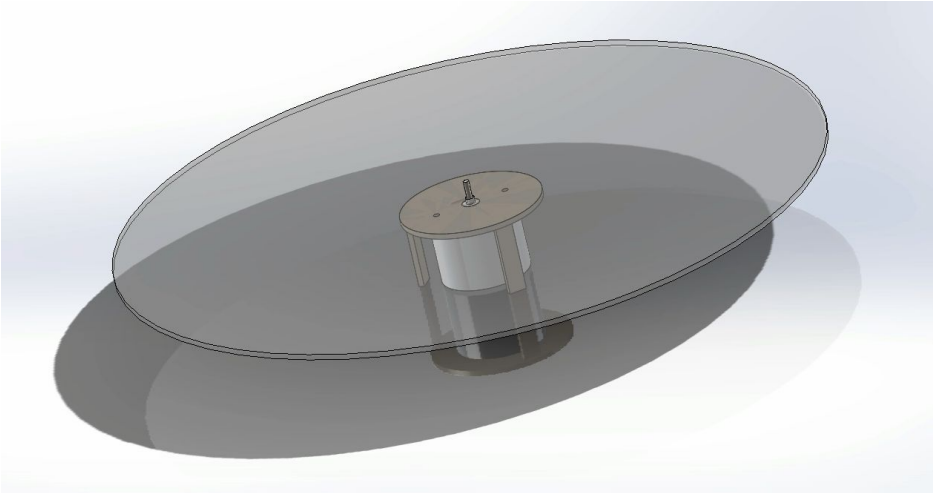
Quantitative Design Requirements

Requirement 1 The user should experience minimal noise under the same robot settings to maximize the skill aspect of the game (**Accuracy & Reliability**)	While aiming a target **1.5m - 3m** away ● **5 cm depth variance** when the ball is shot with the same power settings ● **2 cm horizontal variance** when the ball is shot without reaiming ● **5 cm error tolerance** on cup & ball detection
Requirement 2 The user should have seamless gameplay experience (**Responsiveness**)	Less than **5s of processing latency** for cup detection and ball trajectory detection *(network latency is not part of the requirement)*
Requirement 3 A new user can easily set up and learn to play the game (**Accessibility**)	Total weight of the robot **less than 5kg** Small form factor of the robot of **40cm * 40cm * 50cm** when stored Intuitive control and game state display

Solution Approach - Aiming

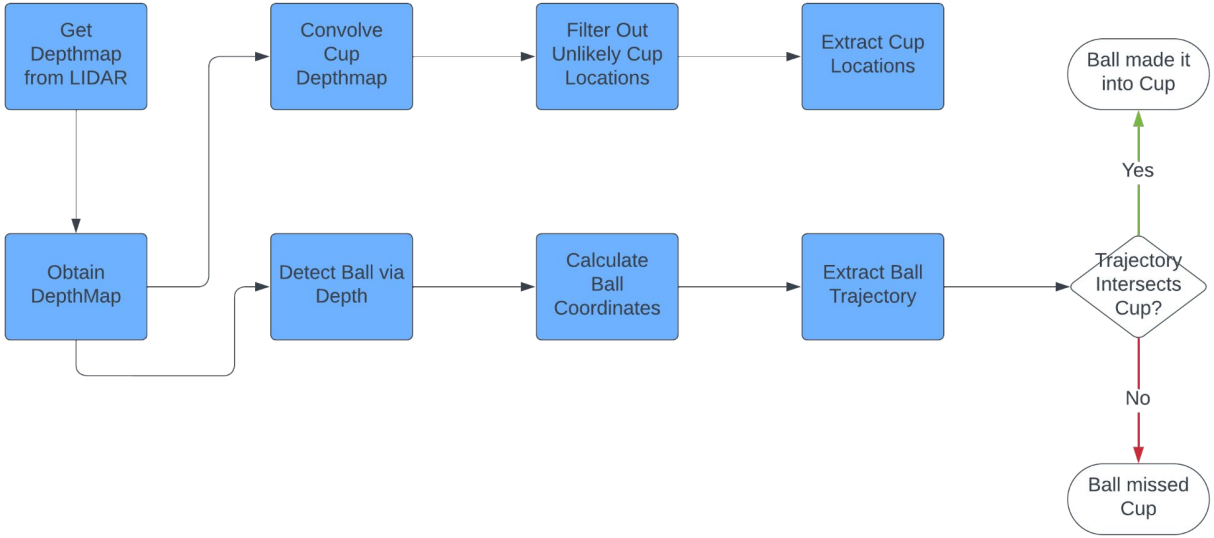


Vertical Aiming prototype

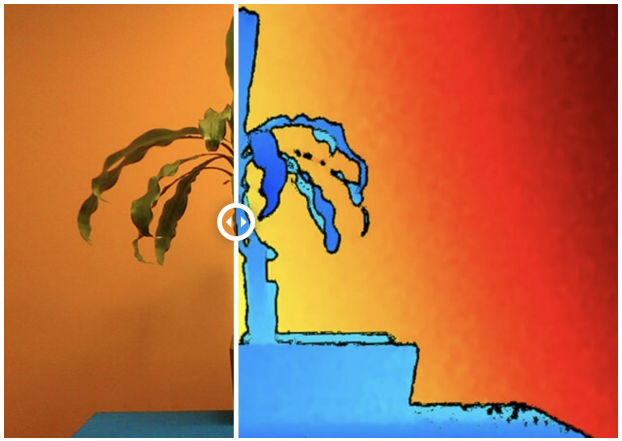


Radial Aiming prototype

Solution Approach - Vision

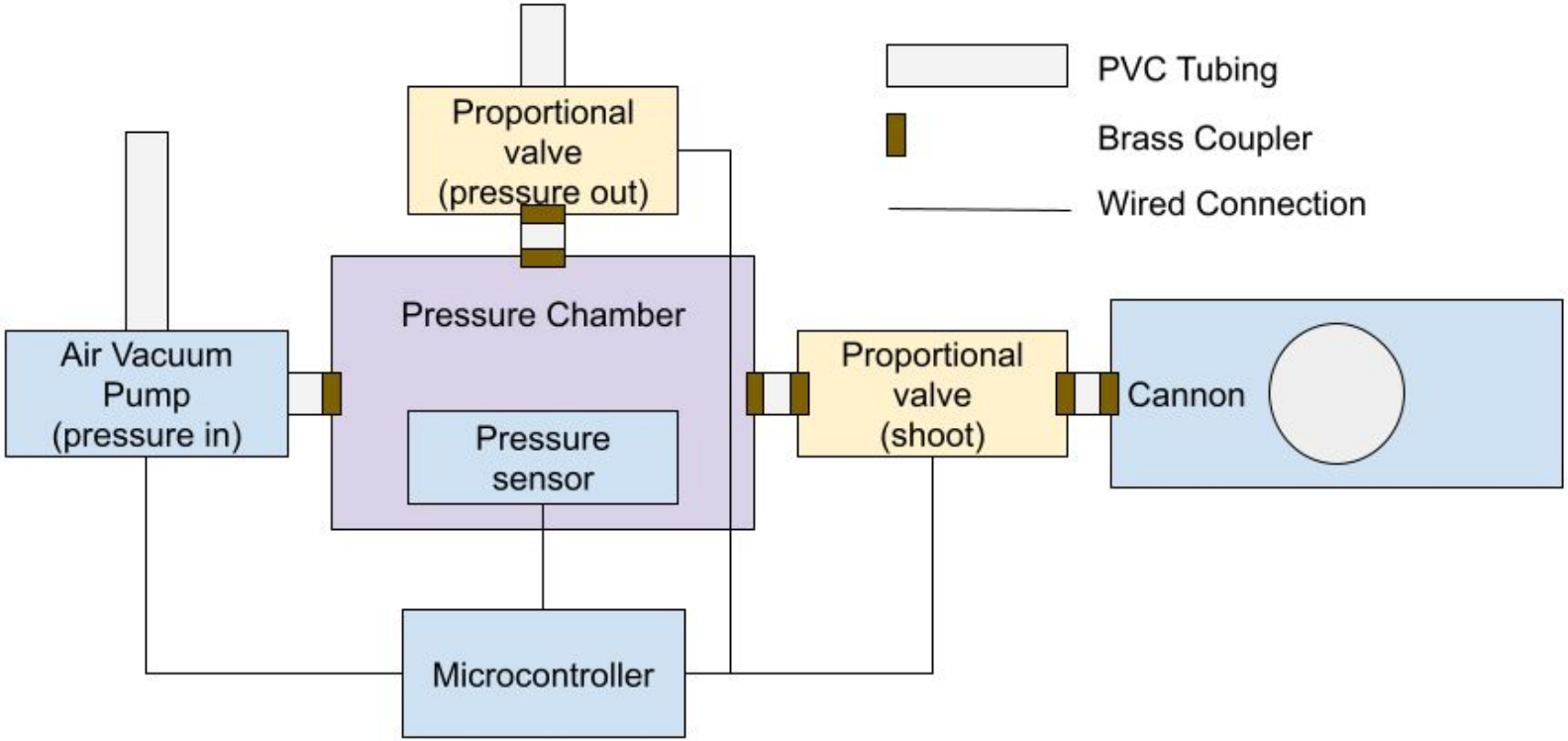


Cup & Ball detection algorithm flowchart



LIDAR depth map example

Solution Approach - Pressure



Solution Approach - Front End

User Interaction Components

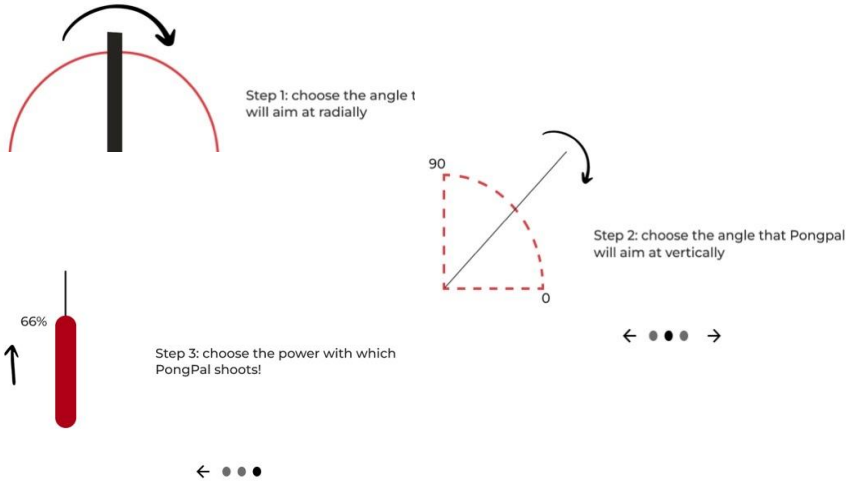
- Pressure Control
- Barrel Angle Adjustment
- Radial Angle Selection
- Fire Button

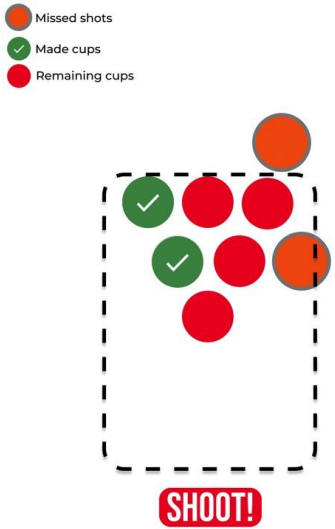
Real-time Feedback Mechanism

- Updates on the UI showing if the ball landed in a cup or where it missed

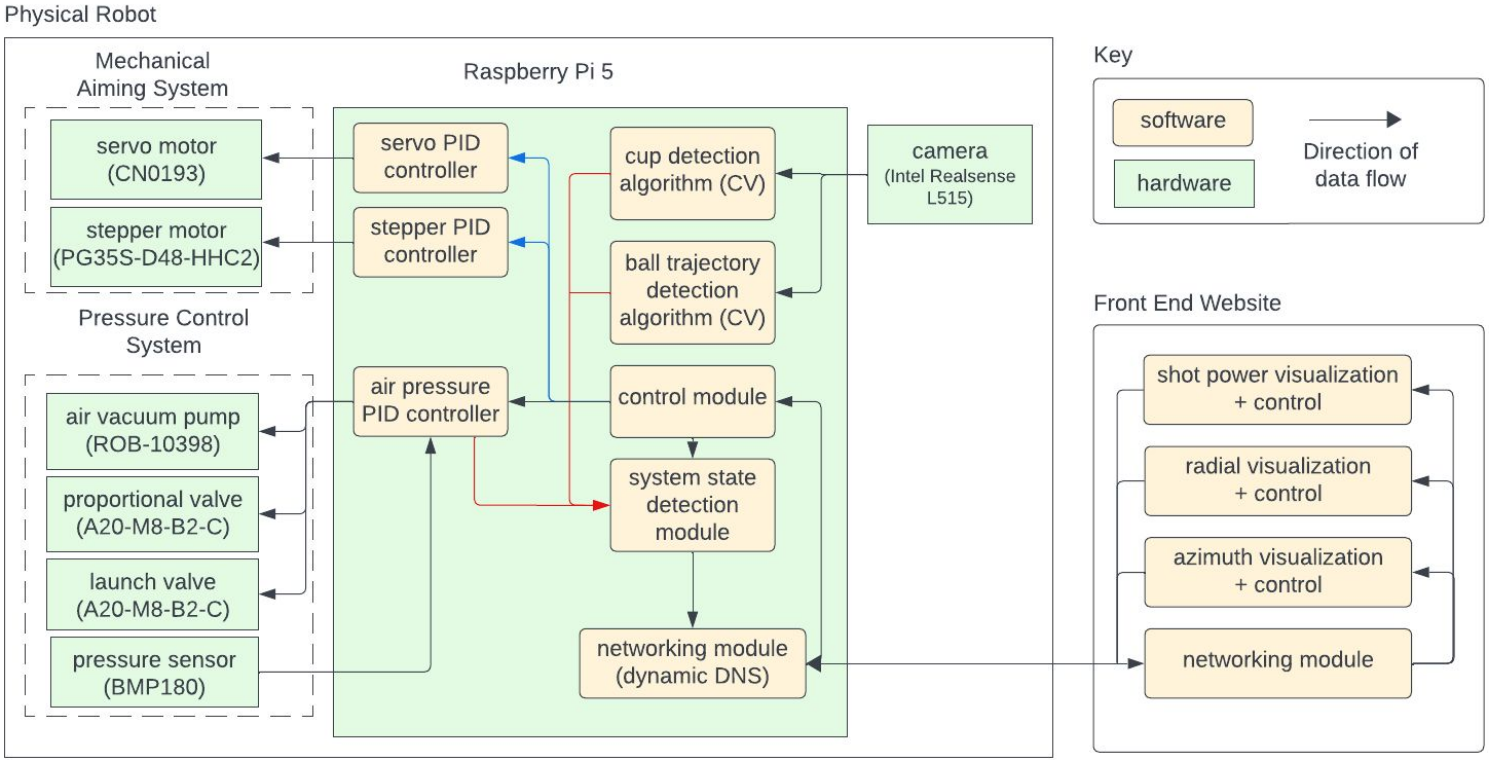
Responsive Design

- Compatibility across various devices and screen sizes

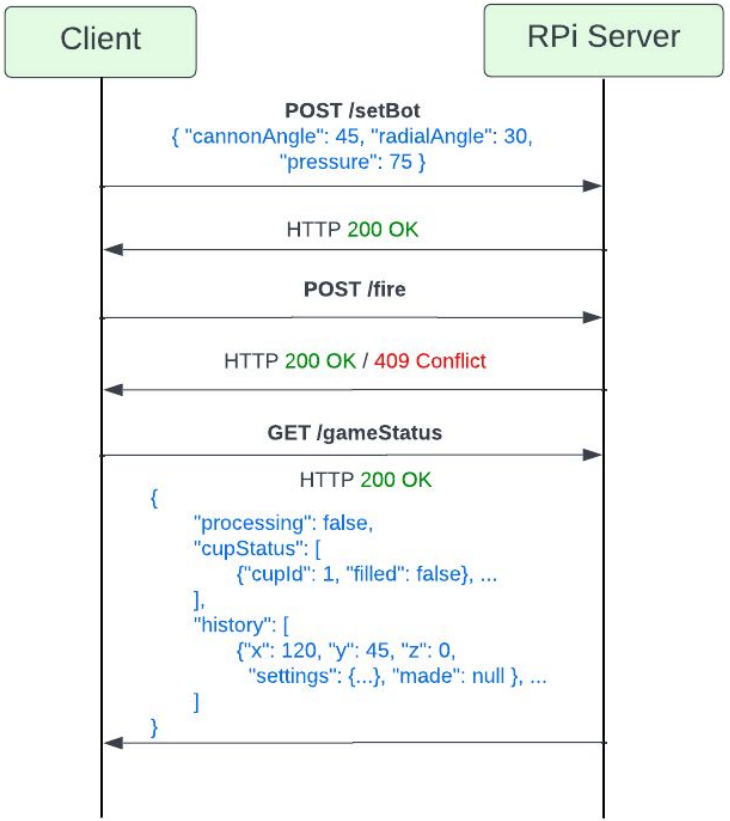




System Specification - Block Diagram (data flow)



System Specification - API Design



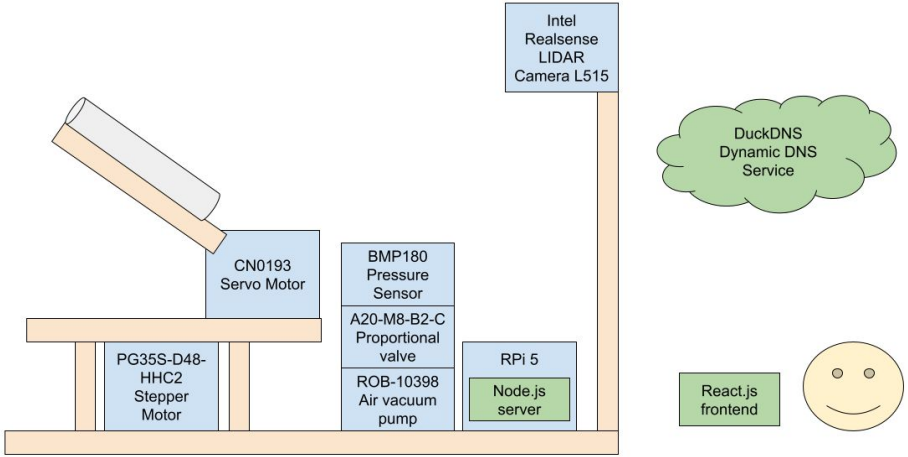
Implementation Plan

Buying / Off the shelf

- PG35S-D48-HHC2 Stepper Motor
- CN0193 Servo Motor
- BMP180 Pressure Sensor
- A20-M8-B2-C Proportional Valves
- ROB-10398 Air Vacuum Pump
- RPi 5
- Intel Realsense LIDAR Camera L515
- Node.js, React.js
- DuckDNS

Our own

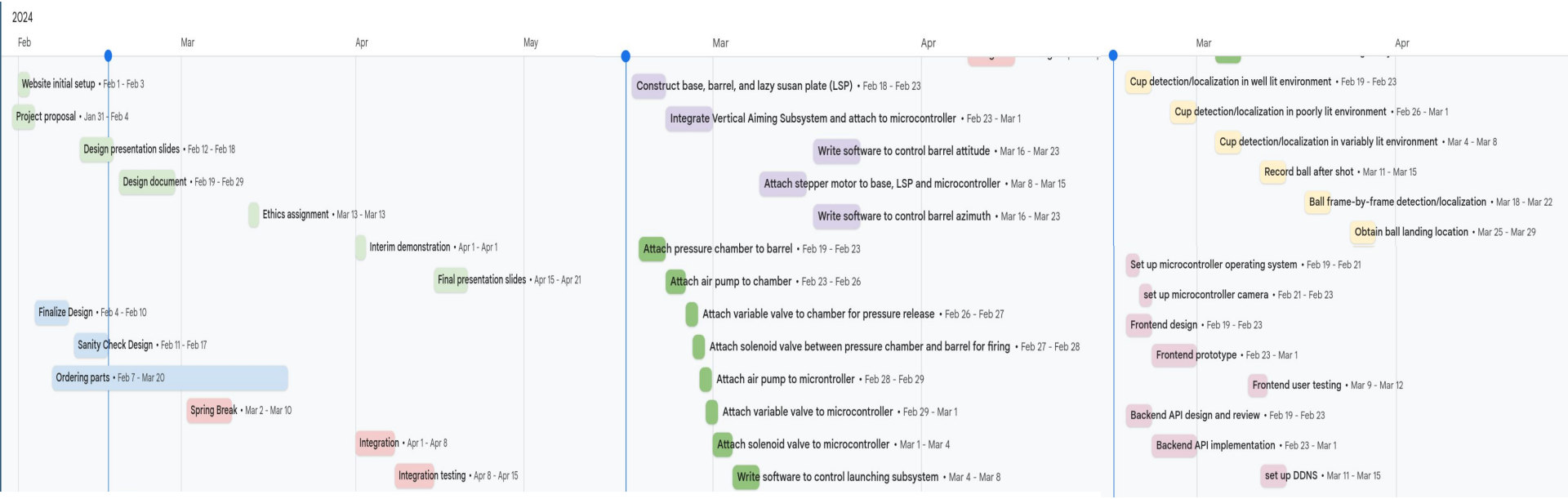
- Stepper PID controller, Servo PID controller
- Pressure PID controller
- Frontend, Backend implementation
- CV algorithm for ball & cup detection



Test, Verification, and Validation

Requirements	Testing	Verification
Accuracy Minimal noise under the same robot settings	Shoot 10 shots under the same setting, and record where the ball landed on a sand target (1.5m away, 3m away)	Maximal bounding box that contains all 10 shots < **5cm** * **2cm**
Reliability Consistent CV performance under various lighting conditions	Place robot and cups in a known position, and measure the cup detection's deviation from ground truth (outdoor lighting, indoor lighting, and indoor flashing lights)	Average deviation from ground truth < **5cm**
Responsiveness Real-time feedback loop for the user	Perform 10 shots, measure the processing latency of ball trajectory detection	Average latency < **5s**
Accessibility Intuitive UI for users to control the robots with minimal guidance	Perform user testing by having a randomly selected individual to control the robot using the UI	**90%** of the users can play the whole game without guidance

Project Management



- Deliverable
- Design General
- General
- Aiming Systems (Seung Yun)
- Pressure (Alex)
- Frontend/Interface (Mike)
- Detection (Alex)