Robot League

Team: sdmay22-26

Client: Dr. Rover

Adviser: Christopher Pedersen

Team Members:

Cheyenne Smith - Team Organizer
Brogden Worcester - Client Interaction
Jordan Suby - Individual Component Design
Noah Brooks - Team Leader Hardware
Joseph Holtkmap - Team Leader Software
Dalton Holdredge - Document Submitter/Creator
David Quan - Progress Coordinator
Tejas Agarwal - Finance Officer

sdmay22-26@iastate.edu

https://sdmay22-26.sd.ece.iastate.edu/

Revised: December 05, 2021

o

Executive Summary

Development Standards & Practices Used

Software Development Practices

- Agile/Waterfall project management process
- Good code formatting and documentation

Engineering Standards

- Unit testing
- Integration testing
- Acceptance testing
- Hardware I/O testing
- Documentation of design decisions, including UI and architecture
- Documentation technologies and rationale behind them
- IEEE Standards

Summary of Requirements

- Robot cars are controlled over WiFi to play a game of soccer
- The bots can use machine learning to play the game
- Game play can be user vs. user, user vs. CPU, or CPU vs CPU
- Game can be built/collapsed with minimal difficulty with provided instructions
- Game can be stored in a standard vehicle trunk, and carried by one average person

Applicable Courses from Iowa State University Curriculum

- CPR E 288
- E E 185
- E E 201
- E E 230
- E E 425
- E E 466
- CPR E 329
- CPR E 430
- COM S 106
- COM S 309
- S E 319
- S E 329
- S E 339

New Skills/Knowledge acquired that was not taught in courses

- Embedded machine learning training/implementation
- Embedded system design
- Raspberry Pi development
- Application development with Flutter
- Video streaming technologies
- Better understanding of software development practices

Table of Contents

1 Team

1.1 Team Members

Joseph Holtkamp

Noah Brooks

Dalton Holdredge

David Quan

Brogden Worcester

Cheyenne Smith

Tejas Agarwal

Jordan Suby

1.2 Required Skill Sets for Our Project

Application development, machine learning, project management, raspberry pi development, python, robotics, circuit design, UI/UX, backend/networking development.

1.3 Skill Sets covered by the Team

Application development	Joseph Holtkamp
Machine learning development/implementation	Dalton Holdredge
Raspberry Pi development	Cheyenne Smith, Dalton Holdredge, Tejas Agarwal, Brogden Worcester
Python	Joseph Holtkamp, Brogden Worcester, Cheyenne Smith
Robotics	Dalton Holdredge, Noah Brooks
Circuit design	Dalton Holdredge, Noah Brooks
Project management	Joseph Holtkamp
UI/UX	David Quan, Jordan Suby, Joseph Holtkamp
Backend/networking development	David Quan, Jordan Suby, Cheyenne Smith

1.4 Project Management Style Adopted by the team

Our project management style is a hybrid of agile and waterfall.

1.5 Initial Project Management Roles

Team organizer: Cheyenne Smith
Client interaction - Brogden Worcester
Individual component design - Jordan Suby
Team leader hardware - Noah Brooks
Team leader software - Joseph Holtkamp
Document submitter/creator - Dalton Holdredge
Progress coordinator: David Quan
Finance officer: Tejas Agarwal

2 Introduction

2.1 Problem Statement

Our team will create two robots to compete in a soccer inspired competition. They will be controlled by an application that communicates with the Raspberry Pi microprocessor on the robots. The game will be able to support human vs human, CPU vs CPU, and human vs CPU competition where the robots compete autonomously via machine learning.

2.2 Requirements & Constraints

Functional Requirements:

- The bots can use machine learning to play the game
- Uses Websocket/handshake protocol over wifi
- Uses camera feed to provide information to the bots
- The bots must be robust enough in case they collide with each other
- The bots will be able to shoot a ball into a goal
- Bots are light enough to carry around
- The arena and robots can be collapsed into a size small enough to fit into a standard vehicle trunk

Resource Requirements:

- 2 geared DC motors per bot (for a total of 4)
- 3 microcontrollers: 2 Raspberry Pi 3b+ and 1 Raspberry Pi 4B
- Jumpers and wires for connecting the hardware
- Chassis for the bots, most likely an acrylic laser cut one
- Tools for machining the chassis
- Power supplies for each bot, and power supply for the arena camera (for a total of 3)
- 2 cameras (one camera each) for the robots to have first person view within the application, and 2 cameras above to utilize for object detection. Each camera will require at least 8MP resolution.

Application

- Cross-platform compatibility
- Low latency video streaming (MJPEG format)
- Socket server backend
- Database for user profiles and game statistics
- Server from ISU for hosting socket server and database

Movement/Aesthetics Requirements:

- Bots can travel at a minimum speed of 3 mph and maximum speed of 5 mph (**constraint)**
- 0 inch turning radius (**constraint)**
- Bots look aesthetically pleasing as determined by surveying volunteer users

Economic/Market Requirements:

- Cost: $600 budget for entire project (**constraint**)

Environmental Requirements:

- N/A

UI Requirements:

- Easy to understand and navigate
- Ability to set up profiles and track statistics
- Ability to access all streams from the bots and arena regardless of play status.

Other Requirements:

- Database for storing profile information of users

2.3 Engineering Standards

7007-2021 Ontological Standard for Ethically Driven Robotics and Automation Systems: Our bots will have both an autonomous feature and a user driven code. This code needs to follow the guidelines of being able to provide clear and precise communication between the bots' different systems and/or the user.

802.11 wifi standard: Our bots will be communicating over a wifi signal.

2.4 Intended Users and Uses

We benefit in learning different skills and new languages along the way for this project as something we can take to employers. Also, benefits others in the robotics fields to provide an example of what could be taught to future students. As well as, people who enjoy playing with RC cars or robots and want a fun little game to play.

3 Project Plan

3.1 Project Management/Tracking Procedures

Our project management style is a hybrid of agile and waterfall. We have adapted an agile-like task breakdown and story board to keep track of what stage each task is in and who is working on it. We do not yet have a daily standup although it may become necessary as we transition into the more implementation focused phase of our project. Due to the short timeframe of our project, the idea of success by incremental, continuous improvement will not cut it. To accommodate our limited time to implement the project, we have organized the implementation of our project in a waterfall model. The combination of these styles offers us a structure for success in our time frame, with weekly "sprints" and agile-like task decomposition.

For version control, we are using Git. Our repository is hosted on Gitlab as was provided to us for this course. For communication and collaboration, our team utilizes Discord. Our communication with Dr.Rover is carried out via Slack primarily. Lastly, Jira is where our tasks, stories, and epics are tracked.

3.2 Task Decomposition

Application Development:

Our team decomposes tasks as they come. When we add a new task to the storyboard, it is looked at what incremental steps are required for it to be completed. If those subtasks are important, they are documented within the story card. If they are significant enough in effort required, they become a card of their own and the tasks are linked as dependent. Jira provides great support for task breakdown and linking. We organize tasks into which Epic (or phase in our gantt chart) that they best fit in. If tasks are related in separate Epics, we use label tags to indicate their relationship.

Hardware:

Task Group 1: Machine Learning

- Develop the ML model for detecting the robots and the game ball
    - Using transfer learning to modify a pre-trained model with our dataset to fit our needs
        - collect dataset to train model
            - collect photos of the shape markers that will be on the robots, as well as the ball
            - classify the shapes in every single image
        - Split the dataset into train/test/validation sections
        - Upload dataset to Google Cloud Vision API
        - Train model using Google's server
        - Download the custom AutoML Vision Edge model in TensorFlow Lite format
        - Deploy the ML model on Raspberry Pi
- Create algorithm for teaching the bot to decide whether to be on offense or defense

    - Using the ML model, the coordinates of the robots and the ball will be used to determine, based on their relative positions with respect to the goals, whether the robot should try to hit the ball into the other goal or defend its own goal
- Create algorithm for steering the ball into the goal
- Create algorithm for teaching the bot to defend its goal

Task Group 2: Robot Prototype

- Decide what materials to use for constructing the chassis
    - Find out if we can find a 3D printer we can access
- Decide which model of Raspberry Pi to use
- Decide how much speed, torque, and current, and space is necessary for our application
    - Acquire motors, cameras, wheels, and motor drivers
- Decide how much durability the robots will require
- Construct robot chassis and assemble the bot
- Test the design for robustness and durability

Task Group 3: Frontend Application / UI

- Choose a framework to enable cross-platform development with a low barrier to entry for developers (Flutter)
- Team members focused on frontend development will learn the framework and tools necessary to be productive
- Design the UI/views
    - Home View describes bots in use and navigation
    - Control view video stream from bots and control interface
    - Game setup and training view
    - Player profile view
    - Statistics of profile view
    - Live game feed for non-participating viewers
- Implement UI
- Distribute application

Task Group 4: Backend Application / Server

- Create user profiles
- Implement TCP/Socket connections to UI
- Implement TCP/Socket connection to robots
- Implement RESTful API connection to database

- Stream video and sensor data between bot and application
- Research hosting services. Select the best one for our needs

Task Group 5: Database / Sensors

- Research and decide between SQL and NoSQL databases
- Design tables for user profiles and game results
- Create tables
- Write code for sensors on the bots, such as bump sensors
    - Decide what the penalties for bumping the other bot will be
- Add sensors to the robots

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

How we will determine progress on a given task:

We will split jiras into different categories: Large tasks, quantifiable tasks, and incomplete/complete tasks.

Large tasks will be split into smaller jiras and we can see progress as smaller jiras get completed.

Quantifiable tasks are tasks like improving the object detection machine learning model to attain 75% precision with IoU threshold of 0.5 for ball recognition. We will be able to see how much it has improved and by how much. For example, if we are able to attain 50% precision with IoU threshold of 0.5 for ball recognition, and the requirement was for 75% precision, then we would be 66.6% of the way complete with this task. These tasks will slowly improve as more work is completed and will be given a progress number (60% complete, 80% complete, etc.)

Incomplete/complete tasks are tasks that are binary - either they are complete or not. Such as, if we were making profiles for users the individual will talk about progress in developing those and we will mark them complete once it's all complete.

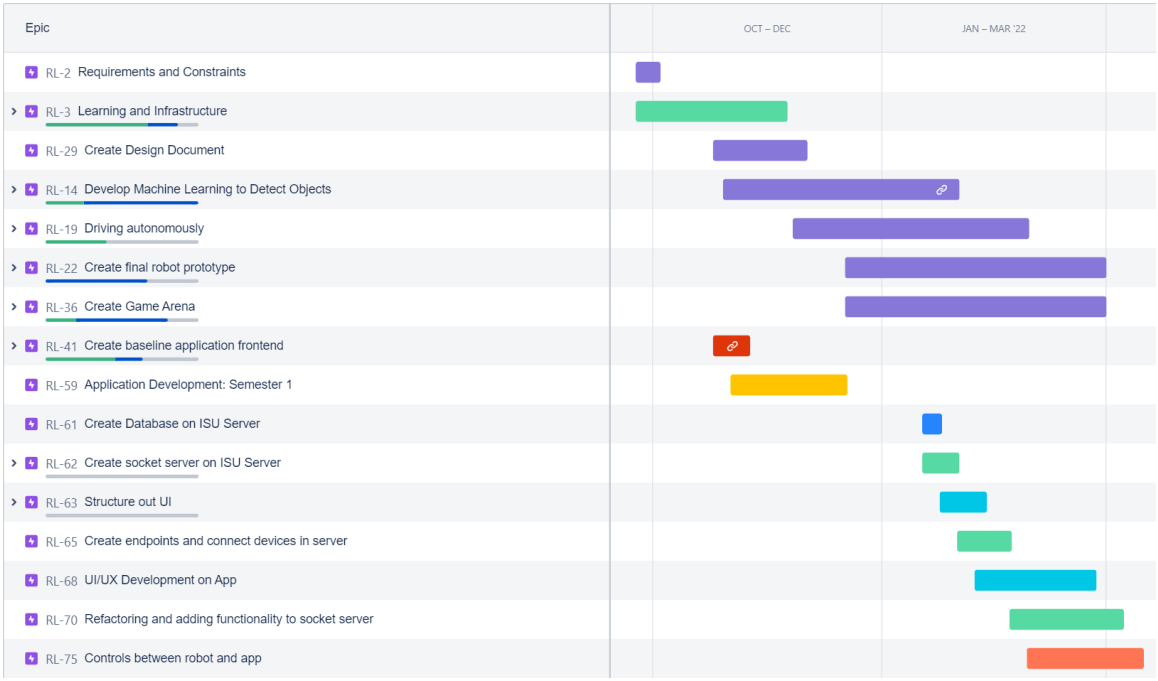
Key Milestones:

- Create a single prototype for our robot league robots
- Interface with our prototype over WiFi to be able to send and receive data
- Create embedded machine learning algorithm to recognize the ball at 75% precision with IoU threshold of 0.5, with a maximum of 250 milliseconds for each detection
- Create a second prototype for robot that can move more quickly, has all of the hardware components encased, is smaller, and has rechargeable batteries.
- Create the arena prototype
- Write python function for automatically recognizing when a team has made a goal based on the visual information from the camera, within 1 second with 95% accuracy
- Create a connection with the bots and retain connection 100% of the time
- Have the bots last 60 minutes of play time
- Write python function to allow the robots to autonomously play the game

Software Milestones:

- Create application interface
- Create socket server for communication between robots and application
- Display video from robot in app
- Stream controls from app to robot
- Create database structure to represent users and game statistics
- Video and controls must have low latency

3.4 Project Timeline/Schedule



3.4.1 Requirements and Constraints

The first task outlined in our Gantt chart is the Requirements and Constraints. This section is where we define the project requirements and constraints within which we must operate. This task was scheduled to be completed in early October, as these requirements determined the scope of the rest of the project.

3.4.2 Learning and Infrastructure

The second task group is Learning and Infrastructure, which is mainly the research into the various components for our design. We had to research and make decisions such as what materials to use for the robot chassis, find what resources were available for us on campus to construct the project, and research how to stream our video feed over a websocket connection. This was scheduled to be completed at the end of November.

3.4.3 Create Design Document

Creating the Design Document is the next item on our Gantt chart, and its completion is based upon the first cohesive draft of the design document being complete. We set the deadline for the rough draft in the beginning of December. We will be updating this document throughout the year into the Spring semester, so this is an ongoing process after the draft is completed.

3.4.4 Develop Machine Learning to Detect Objects

The next item on the Gantt chart is Developing Machine Learning to Detect Objects. In order to accomplish this, we had to train a machine learning model to recognize the robots and the game ball.

3.4.5 Driving Autonomously

The next item on the Gantt chart would be to utilize object detection to aid in an autonomous driving or mode so that a person could play against a robot. The machine learning object detection model provides the locations of the robots with respect to the game ball, which are inputs into the python function that allows the robot to drive autonomously to get the ball into the other team's goal.

3.4.6 Create Final Robot Prototype

The next item on the Gantt chart is the process of creating the final robot prototype. The finished product will be a functional car that is able to drive either via user control over a websocket connection from the application, or autonomously utilizing the custom machine learning object detection model.

3.4.7 Create Game Arena

The final task for physical deliverables is to create the game arena within which the game will be played. This involves picking dimensions that will work with the arena camera (narrow enough so the entire field is in the camera's field of view, yet wide enough to allow for adequate space for playing, etc.).

3.4.8 Create Baseline Application Frontend

This epic was centered around creating the skeleton of our application. It involved creating the flutter app, adding pages we will most likely use, and then routing between them. This was mostly a research and development based set of tasks designed to get us familiarized with the Flutter framework and using Dart. We also completed a basic stream of MJPEG content that was used as proof of concept that we could stream live video to our app.

3.4.9 Create Baseline Application Frontend

These tasks are for creation of the application that will interface with the robots and arena camera to facilitate the playing of the game. Tasks for this portion included items such as creating the empty application, configuring the endpoint for streaming video, displaying video feed within the application, and learning how to use Flutter.

3.4.10 Application Development: Semester 1

This epic is centered around creating a working application for semester 1 and how much progress we've made so far. Most of this was centered around having the bones of the project working so that we knew where screens went and what API calls to make.

3.4.11 Create Database on ISU Server

During this time, our team will set up an SQL database on our ISU server. It will consist of creating the database, creating a database schema to represent our data, and implementing the schema. After which, the connection details will be documented and handed off to the teammates implementing our backend/socket server.

3.4.12 Create socket server on ISU Server

This task is centered around creating a socket where data can be sent and received using a TCP connection protocol. This would involve testing our bots communication between the server and the web application.

3.4.13 Structure out UI

This task revolves using Figma to create screens to structure how our UI will look in the future. This includes learning Figma and creating example screens. We would also have to decide on how our UI would look together and how you would transition from one to another.

3.4.14 Create endpoints and connect devices in server

This task involves creating APi endpoints and connecting them to the server. This would mainly be after creating the database then seeing if info can be sent and received to the database. We would also create endpoints for other things like having the user see the viewpoint of the bot.

3.4.15 UI/UX Development on Application

This would involve actually creating the screens that we structured out in Figma. We would then like to replicate these screens for actual use.

3.4.16 Refactoring and adding functionality to socket server

This task involves actually getting the data from our socket that we established. With this, we can finally start talking to the robot and get data points from the bot. This can be information about where the ball is or video from the camera feed.

3.4.17 Controls between robot and application

This task involves sending information to the bot to control the movement. This would involve creating a screen and having the bot respond in an appropriate amount of time to the command from our web application.

3.5 Risks And Risk Management/Mitigation

This section outlines the risk of each sprint that the team came up with during the strategizing phase of the project. With respect to the requirements and constraints task. The risk that we assume is that we might have applied unrealistic constraints to the project that we are not yet aware of. To mitigate this risk, thorough research is put into each design element such as the bot's material composition, the user application framework, and the motor for the bot. Another prevalent risk that the team faces is the

development of the machine learning model (ML). With respect to the ML sprint, we face two distinct risks:

The live stream and the data captured by the arena camera can not be accessed in a way that allows for the use of that data for automation and furthering the ML model. We will be reducing the probability of this risk by training the model early. This will allow for the determination of the issues that the current solution might present. The second risk we face is the limits of the hardware. The field of view (FOV) of the arena cam could be too small for proper use. To mitigate this the cameras have been tested to determine a suitable one. Another risk we face with the camera choice is that the resolution might be too low to accurately detect the objects

The table below quantifies the risks associated with the project. For each risk, there is an estimated probability of that risk occurring, ranging from 0 (not possible) to 1 (guaranteed to occur). The "Weight of Risk Severity" is the relative impact to our success if this event occurs, ranging from 0 (negligible impact on our success) to 10 (catastrophic impact to our success). The "Probability*Weight" column is a representation of the combination of the probability and the risk severity. The higher the number for this column, the more concerned we should be that this risk might occur.

Task	Risk	Probability of Risk	Weight of Risk Severity	Probability *Weight	Mitigation
Requirements and Constraints	Game requirements are not easily feasible	0.25	4	1	Proper research and early prototyping
Learning and Infrastructure	Many of us will be working with tools we do not have prior experience with and they may be harder to pick up than we expect.	0.25	2	0.5	Ask for help from qualified sources.
Create design document	It does not get finished in time	0.1	10	1	We will start working on it early
Develop machine learning to detect objects	We cannot access the camera to live stream the data	0.3	5	1.5	Start working on camera in advance so we can work out any issues
	We pick a ML model that takes up too much time or data processing	0.25	3	0.75	Research existing ML models and decide on what will work best for us
	Cameras do not have high enough	0.2	4	0.8	Contrasting

	resolution to accurately detect the objects				distinguishing marks on top of the robots, as well as a high-contrast game ball
Drive autonomously	We don't have enough time to train the ML model	0.5	3	1.5	Use online resources such as Google Cloud Vision API or Roboflow to train the model
Create final robot prototype	We won't have enough time to create the prototype	0.5	5	2.5 0.5 0.6	Early prototyping now, and deadlines to keep us on track
	The motors don't have enough torque or speed	0.25	2		Early prototyping
	The chassis is not strong enough	0.2	3		Prototype with various materials
Create game arena	The arena is too large to be captured with the camera	0.1	6	0.6	Check camera field of vision beforehand
	Arena materials take up too much space when they are stored	0.25	1	0.25	We choose materials with this consideration in mind

3.6 Personnel Effort Requirements

Task	Person-Hours
1 Requirements and Constraints	3
2 Learning and Infrastructure	41
2.1 Set up a Jira Board	1
2.2 Set up repository	1
2.3 Find 3D printers and technology for 3D printing designs	2
2.4 Research and decide on tech stack for application	4
2.5 Document why we chose Raspberry Pi	2
2.6 Figure out and document specific funding amounts	3
2.7 Meet with hive to discover available resources	1
2.8 Find what resources are available in Student Innovation Center	2
2.9 Research Unity	4
2.10 Password Generator	2
2.11 Define and document game and rules	8
2.12 Decide if we are using embedded machine learning or using a server	3
2.13 Research how to stream video over a websocket connection. (TCP Protocol)	8
2.14 Create CI/CD Pipeline for repository	3
2.15 Learn how to broadcast mjpeg and/or .cgi from camera over http protocol	5
3 Create Design Document	15
4 Develop Machine Learning to detect objects	96
4.1 Be able to access camera data over WiFi	16
4.2 Decide which trained model will be used for the “skeleton” for the ML model	16
4.3 Attain 75% precision with IoU threshold of 0.5 for ball recognition	32
4.4 Attain 75% precision with IoU threshold of 0.5 for robot shape recognition	32
5 Driving autonomously	36
5.1 Create python function for allowing the bot to “decide” to be defense or offense	12
5.2 Create python function for the bots to drive on offense autonomously	12
5.3 Create python function for the bots to drive on defense autonomously	12
6 Create final robot prototype	40
6.1 Decide what materials to use for design chassis	16
6.2 Acquire motors for final design	20
6.3 Decide for Raspberry Pi 3B+ or Raspberry Pi Zero W as computer	4
7 Create Game Arena	104
7.1 Logistics for game camera (resolution, FOV, USB or camera cable)	16
7.2 Decide if it will be an open goal, goal post, or something else	8
7.3 Decide what materials will be used to construct the arena	16
7.4 Build arena camera holder	32
7.5 Build arena	32
7.6 Calculate estimated cost for arena design	1
8. Create baseline application frontend	31
8.1 Create basic form in Flutter	2
8.2 Create empty application	1
8.3 Learn how to create Mjpeg elements in Flutter	6
8.4 Learn how to switch views between in Flutter	4
8.5 Retrieve data from an API in Flutter	2
8.6 Implement routing in our app	2
8.7 Configure endpoint for streaming video	2

8.8 Display video feed in application	4
8.9 Learn how to construct layout in Flutter and position widgets	6
8.10 Create Use Case diagram for application	2
9. Application Development: Semester One	21
10 Create Database on ISU Server	2
11 Create Socket Server on ISU Server	6
11.1 Create TCP connection between video and app	6
11.2 Allow streaming through TCP	7
12 Structure out UI	22
12.1 Navigate between screens	4
12.2 Player screen	5
12.3 Video screen	6
12.4 Home	2
12.5 Statistics	5
13 Create endpoints and connect devices in Server	6
14 UI/UX Development on App	9
15 Refactoring and adding functionality to Socket Server	8
16 Controls between Robot and App	12
Total person hours:	385

3.7 Other Resource Requirements

Hardware Team:

Part No.	Description	Quantity	Rate	Cost
1	Canvas	3	$ 1.49	$ 4.47
2	PVC Pipes	13	$ 4.41	$ 57.33
3	T Connectors	14	$ 0.39	$ 5.46
4	Elbow with Side Outlet	8	$ 1.49	$ 11.92
5	Cross 4 way connectors	2	$ 1.69	$ 3.38
6	4 way T connector	1	$ 12.29	$ 12.29
7	Elbow Connectors	4	$ 0.39	$ 1.56
8	DC motors with wheels	4	$ 7.01	$ 28.04
9	Raspberry Pi 4 4GB	1	$ 55.00	$ 55.00
10	Raspberry Pi 3B+	2	$35.00	$ 70.00
11	Raspberry Pi Cameras	2	$ 24.99	$ 49.98
12	8 mp camera	1	$ 30.00	$ 30.00
13	Homemade battery pack	1	$ 33.00	$ 33.00
14	Front wheels	4	$ 1.00	$ 4.00
			Total Cost	$ 366.43

Application

- Low latency video streaming tool (MJPEG format)
- Socket server backend
- Database for user profiles and game statistics
- Server from ISU for hosting socket server and database

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	The product must provide entertainment to the general public and possibly an outlet for stress for participants and audiences.	Reducing stress impacts the wellbeing of individuals in a wealth of ways.
Global, cultural, and social	Competitions and games play a role in most communities and provide a realm in which unrelated groups can be introduced to each other.	Games can bring people together from anywhere, especially when available over the internet.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement. Our project should have little to no environmental effect. If any, slightly increasing the demand for rare earth metals used in computer hardware and electricity.	Increasing/decreasing energy usage from nonrenewable sources, increasing/decreasing usage/production of non-recyclable materials
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

4.1.2 User Needs

Children need activities that can tie their curiosities to productive fields of work so that they can explore technology as an industry for their future careers.

Stressed individuals need something as a distraction to divert their mind from the stress so that the next time they get back to work they have a fresh mind and high productivity.

4.1.3 Prior Work/Solutions

When exploring prior solutions, the technology used for driving modern drones with onboard cameras meets the solutions our apps are being designed for. From the camera on the drone, video feed is displayed on the controller or phone being used and controls are sent to the drone continuously.

RC cars exist in a similar capacity to what we need for the soccer-inspired game. They can be controlled in a similar capacity to our use case and can, in some cases, stream video feed.

Advantages (Drone Controller)

- Supports the functionalities we desire for direct interaction with our robots
- Is not limited by wifi connection

Shortcomings (Drone Controller)

- Most are not open source
- Does not support the streaming video over wifi
- User interface doesn't support 3rd party viewing
- Many don't support the UI customizations we would require

Advantages (RC Cars)

- Can drive and interface with RC controllers
- Are light and quick
- Can stream video in some case

Shortcomings (RC Cars)

- Cannot interface with wifi
- Do not have the interfaces necessary to apply machine learning
- Often does not have the mass to be as effective in a competition as we require
- Most often do not have the capacity to turn in a small enough radius

Below is a list of pros and cons of our target solution compared to all other related products/systems.

Pros

- Our product is a unique combination of the products available
- Our product is an interpretation of an existing game (rocket league) but in a real world context
- Our product could enable a unique learning environment in the fields of robotics
- Our product is sturdy enough to not break for repeated use
- Our product involved machine learning for a real life CPU opponent

- Our product will not require the user to be in the same room as the arena

Cons

- Rocket League, RC cars, and controllers with video capabilities currently exist
- Our product is more expensive than existing rc cars in the market

4.1.4 Technical Complexity

1. The design consists of multiple components/subsystems that each utilize distinct scientific, mathematical, or engineering principles

   Our product will be a combination of robotics and full stack application development that requires the following components:

   - Multi-platform application
     - Capacity to display low latency video feed
     - UI/UX design
     - Robot driving game control interface
   - Socket server
     - Coordinate players and robots
     - Handle account/statistic interactions
     - Hosted on and ISU server
     - Handle login validation
     - Multithreaded to optimize performance at bottlenecks
     - (Object detection data will likely be used to drive bots from here)
   - Database
     - SQL
     - Store account information
     - Store game statistics
     - Calculate rankings
   - Arena
     - Stream video of entire game
     - Camera with object detection
   - Robots
     - Stream video
     - Driveable with controls streamed over wifi
     - Agile enough to keep game speed competitive

2. The problem scope contains multiple challenging requirements that match or exceed current solutions or industry standards.

   Complex components

   - Machine Learning- Since our project has an option of a single user playing the game with the CPU, as well as the 2 robots playing autonomously using the CPU, we needed to train a machine learning model so that the CPU can maneuver in such a manner as to give a decent competition to the user. We are utilizing visual object detection using the arena

cameras, which will implement the ML model that we have created to detect the robots and the game ball.

- Autonomous Play- The ML model provides the location of the robots and game ball, which are used as inputs to the function that will ultimately provide the direction for the robots to drive autonomously.
- Power Supply- Our design consists of various components, all of which have a desired power dissipation which we couldn't supply from the design and the concepts we decided to implement. Adding more power could only be done by two methods, either we add the power supply which will increase the cost of the design or start from square one, redesign the robot and arena again and look for the components that run on less power which might be expensive or difficult to find.
- Design- Designing the robot and arena turned out to be a lot more difficult and time consuming than we originally thought. The robot had to be strong so that it doesn't take any damage in case of collision with the other robot or the structure of the arena. We wanted our project to be highly mobile due to its structure size, therefore the robot has to be small and lightweight so that it's easy to carry. The robot also has to be "nice and beautiful" all of which becomes difficult when we try to add more strength to it with a restrained budget. Similar complexity was with the arena as well. Since the arena was sized to be 8' x 4' x 5', its structure takes a lot of space and is difficult to carry, therefore we needed to come up with something that's strong enough to bear the collision force of robots and has good mobility.
- Socket server - Our server will need to receive connections from multiple different devices: users on the app, arena camera, and robots. The server will coordinate which robots get connected to their driver and ensure the correct driving commands go from the user to the robots. The server will also set up which IP's are streaming video for the app to connect to over HTTP protocol.

4.2 Design Exploration

4.2.1 Design Decisions

1. Multi-platform framework: **Flutter/Dart**
    - Single platform products have only a limited audience
    - Our team will more effectively be able to test our products by being able to reach every device our team members possess.
    - Targeting a single device would confine our team in ways that could limit product down the line.
    - Flutter apps can be compiled to a wealth of the common device platforms with a single code base.
    - Dart is an object oriented programming language which suits the strengths of our team
2. What computer(microcontroller) to use for the bots: **Raspberry Pi**
    - Ability to interface with wifi
    - Capacity to stream low latency video
    - Compatible and affordable video cameras available
    - Raspberry pi's have a variety of models available

- They are accessible and do not require the learning of a new programming language like arduino
- Capacity for embedded machine learning

3. Decide whether we are using embedded machine learning or doing the computation on a server.
    - The calculations need to be fast enough to be useful
    - Calculations on the server add latency
    - Calculations on the microcontroller may be limited for space or compute power
4. Database: **SQL (MySQL or SQL Server)**
    - Our data is going to be very structured which lends to SQL in strengths
    - NoSQL is good for highly connected data which our data will not be
    - Our data should be easily organized into a database schema which is best suited for SQL
    - Additionally it can be hosted on the server we have from ISU which is more suitable than cloud solutions for our project's duration.

4.2.2 Ideation

For what computer to use we used the lotus blossom technique.

Microcontroller options considered:

1. Arduino Uno
2. Arduino Nano 33 BLE Sense s
3. Raspberry Pi Zero WH
4. Raspberry Pi 3B+
5. Raspberry Pi 4

Frontend options:

1. React Native
2. Angular
3. **Flutter**
4. Native android app via Android Studio
5. Electron

4.2.3 Decision-Making and Trade-Off

We looked at the official Raspberry Pi and Arduino sites in order to see hardware specification and whether they aligned without goals. We didn't decide on Arduino because no one has used Arduinos which leaves the raspberry pis. We then discussed machine learning capabilities and decided that the raspberry pi 4 has enough ram for intensive computation that comes with ML.

	React Native	Angular	Flutter	Android Native App	Electron
Supports android	X		X	X	X
Supports iOS	X		X		
Supports desktop	X		X		X

Supports web		X	X		
Supports devices we have access	X	X	X		X
Pure OOP			X	X	
Familiar to team members		X		X	
Results	**4**	**3**	**6**	**3**	**3**

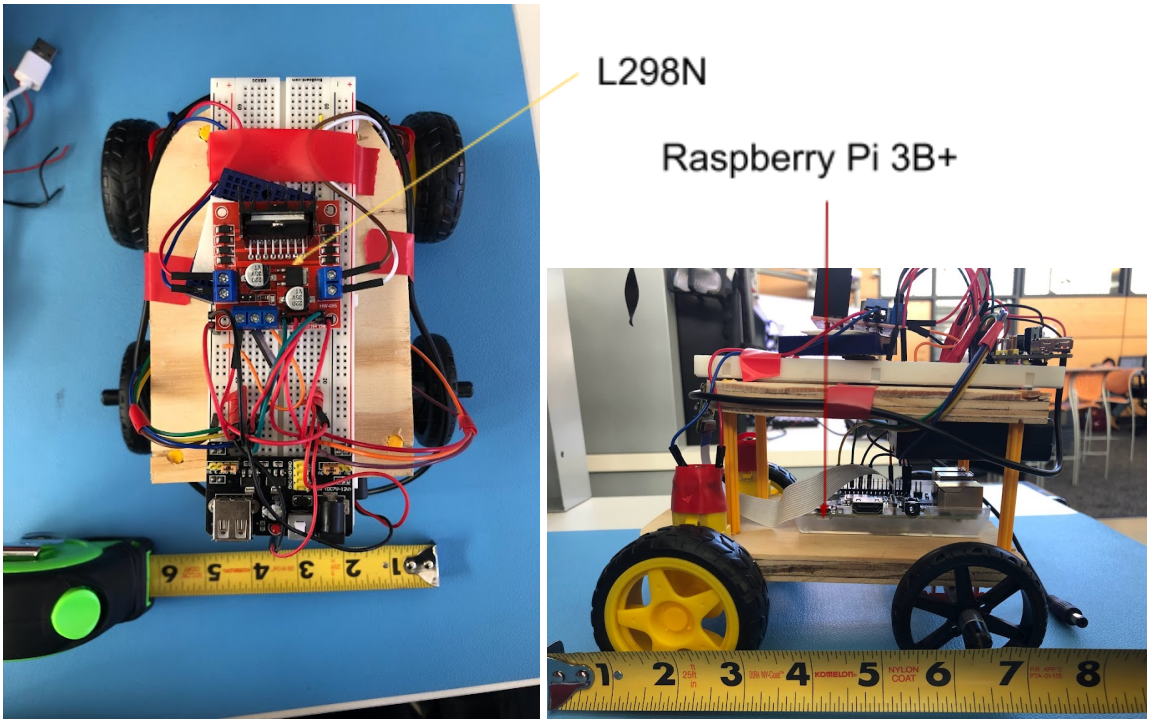
4.3 Proposed Design

On the application side of our project, we have implemented the skeleton of an application with basic routing between pages. We have also implemented the baseline for how we can display a video stream over HTTP that is MJPEG format. The application has been tested and works without customization on Windows Desktop and an android emulator.

On the Hardware side of the project we have implemented and tested the different power supplies to determine suitable for the needs of the project. Also we have tested the field of view of the designated arena camera to determine the course of action when it comes to the object detection model.

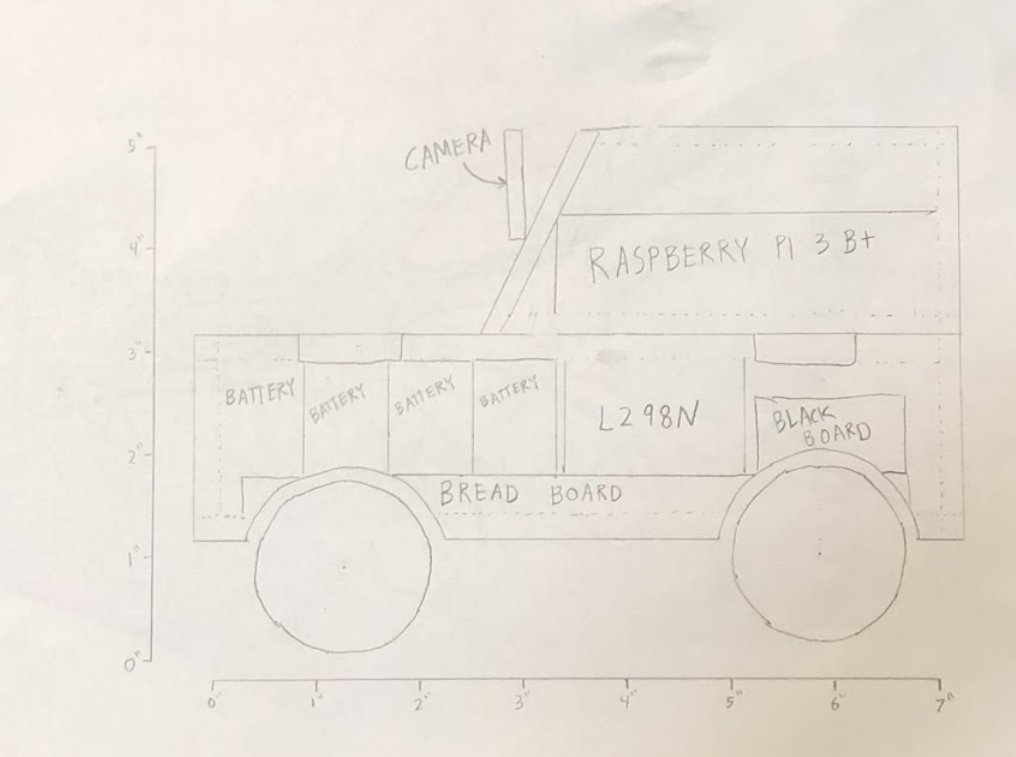
4.3.1 Design Visual and Description

Below is the first prototype P.H.I.L.L.I.P. It is our team's first prototype constructed of plywood, KNEX, and electronic components. The design is simple but inclusive, where we could work from to build better bots and improve features. The battery and motors on P.H.I.L.L.I.P. rated for 6 volts limited the speed. The next prototype will have 12 volt motors and an 18 volt battery. Another downside is the exposed wheels, which could cause unwanted behavior near other robots or objects.

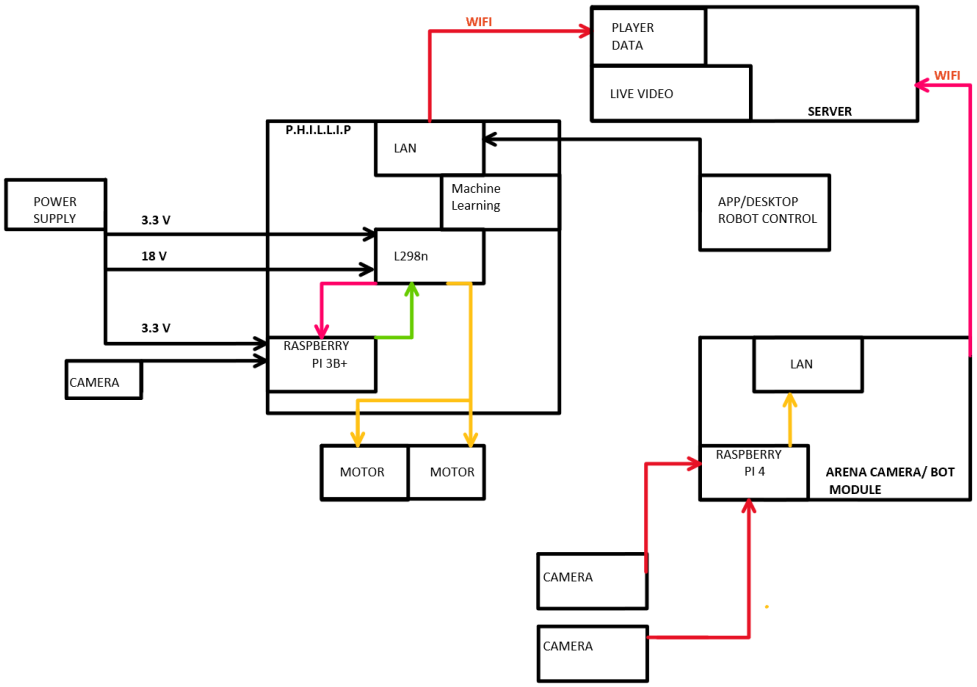


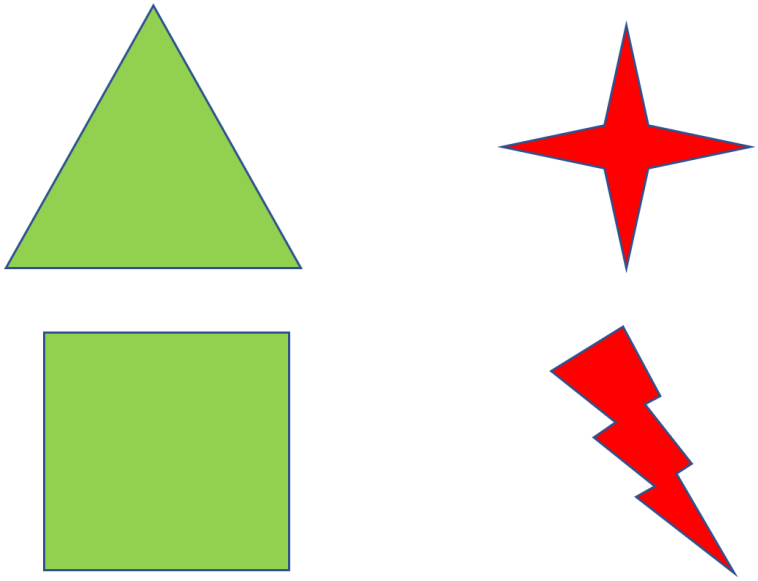
P.H.I.L.L.I.P.

The next prototype will be made of pressed material found in surplus in the scrap of the electrical engineering machine shop. By measuring the components on P.H.I.L.L.I.P and adding in the future batteries and motors, a design can be formed. This design will be more enclosed, and have a more aesthetic appearance. The upper half of the bot will be able to seperate allowing easy access to the breadboard and components. A large flat roof is needed for symbols used by machine learning.



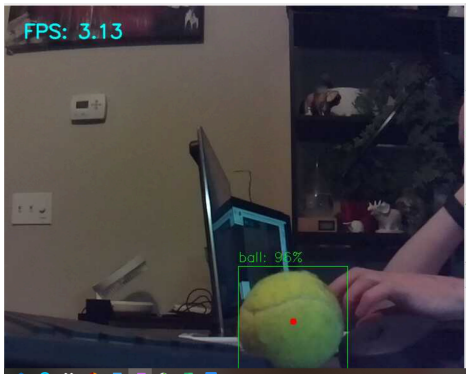
Next Prototype



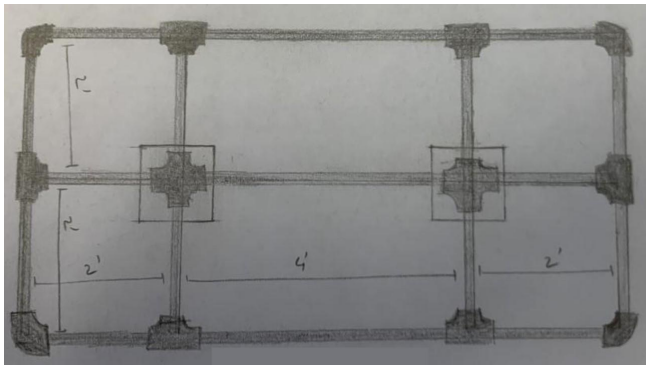


Instead of teaching the machine learning model to recognize the physical robots, we decided to instead teach the model to recognize the shapes shown above. Each robot will have two shapes on top of them, which allows for the object detection model to differentiate between the two robots. The reason that there are two shapes on top of each robot, instead of one, is because this allows for the model to determine which direction the robot is facing. In order to use the object detection data to provide input to maneuver the robots, we must have the ability to determine which way each robot is facing, in addition to their locations. Because the object detection model is only able to provide the location of each object, a single object would not be sufficient to determine which direction each robot is facing.

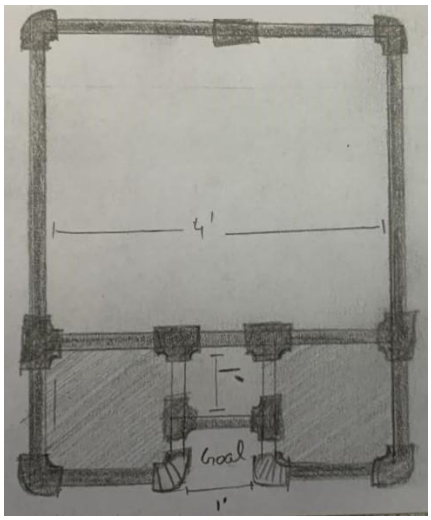
An example of our object detection model is shown below. In addition to training the model to recognize the shapes shown above, we used a tennis ball to train the model for the game ball. Tennis balls are inexpensive and easily replaced if they are lost, so that is why we chose them. For each object detected, the model determines the bounding box around them (shown in green below). The model also calculates the center of each object (shown by the red dot below), which is necessary in order to determine the location of each robot and the ball during the gameplay.

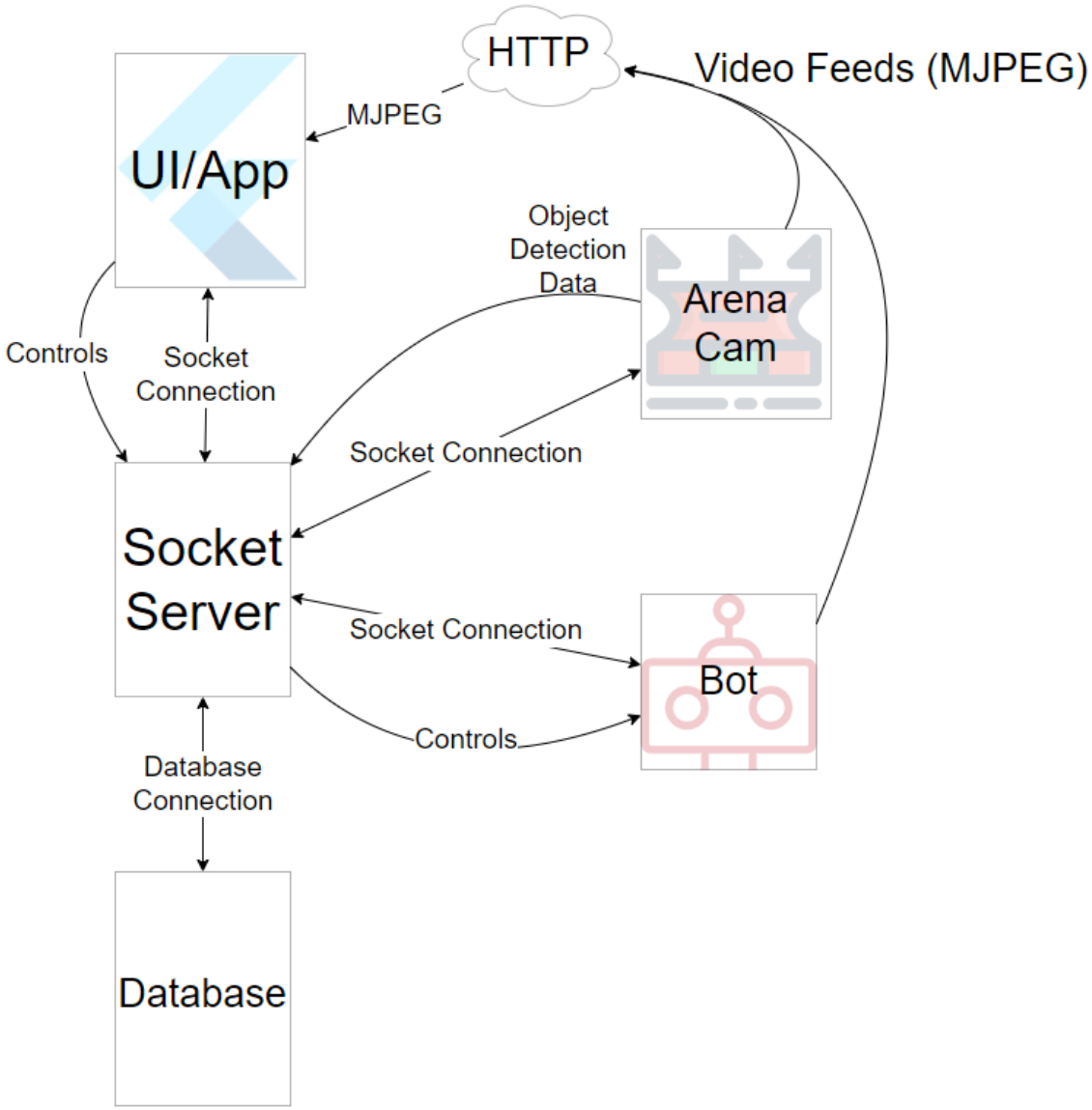


We designed our arena with dimensions 8’ x 4’ x 5’. Since the area to be covered by the camera for video analysis is too large, the camera must be placed very high which increases the amount of money needed to be spent on the arena. The extra height creates the problem that the camera will not be able to detect objects, so to mitigate this, we decided to use two cameras instead of one and merge the data collected from them.The image below shows us the top view of the arena with two cameras placed in the center of two halves of the arena.



The image below shows the view when standing from behind to the goal of the arena. The goal will have a structure similar to a draw bridge so that every time the ball goes in, it can be rolled out to reset the game. We also made sure that the size of the goal post is big enough for the ball to go in even if there is a little bounce but it's not too big so that the robot can enter and mislead the scoring system. with the lightly shaded area representing the walls which we created using military grade canvas so that it’s not too hard that can damage the robot and it’s not too soft that can be damaged by the robot.





High Level Architecture Diagram

Here is a high level drawing of the components involved in our project and how they interact. The socket server will orchestrate the various interactions between the bots, app, database, and arena camera. The cameras on the raspberry pi's will broadcast the video feed over HTTP in MJPEG format. The bots will send the host and port numbers to the server so that the user's applications can connect to the video feeds. The object detection data is also being planned to be sent to the server so that it can be overlaid over the video feed from the arena cam in a helpful fashion for users. Controls will be sent from the appropriate users to the correct bot for a given game.

4.3.2 Functionality

Describe how your design is intended to operate in its user and/or real-world context. This description can be supplemented by a visual, such as a timeline, storyboard, or sketch.

How well does the current design satisfy functional and non-functional requirements?

4.3.3 Areas of Concern and Development

The final design of the robot isn't finalized yet due to some errors and issues on the current design. Our current design has an issue that it's taking too much power that is very difficult to supply considering our resources and specifications. Another concern we have is the camera we want to use doesn't have a good frame rate which will affect the machine learning model and object detection. We are also concerned about connecting the bot with our web application. We haven't gotten the two to communicate and only have an idea on how this can be done. Proving that a TCP connection can transfer data efficiently without packet loss still hasn't been done.

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

We are still looking for potential solutions to these problems. As for now, we are looking for other cameras that have a better video quality and also fit our budget. We are also looking for some industry-grade battery that fits our budget but can supply more power to solve the issue of lack of power in the project. On the TCP connection we will begin implementing it in the upcoming semester. We have two students in CPRE489 who deal with sending TCP packets. We know the technology and understand it can be used for sending video feeds, but we haven't done it with our robot. We also have other solutions ready if this does not work such as using other forms of data transfer. We will be discussing these problems with the experts and our advisor to check if they have a better suggestion that would be useful in our case.

4.4 Technology Considerations

Different computer to place on the robot:

1. Arduino Mega - No one has used arduinos
2. Arduino Uno - No one has used arduinos
3. Arduino Nano 33 BLE Sense - No one has used arduinos
4. Raspberry Pi Zero WH - Works, but does not meet our computation/interfacing requirements
5. Raspberry Pi 3B+ - Works, but does not have the RAM requirements we may need for ML
6. Raspberry Pi 4

The Arduinos are microcontrollers and do not have the processing power for machine learning. The Raspberry Pi pico overheats and is also a microcontroller.

The Raspberry Pi 4 is expensive, but will work well.

The Raspberry Pi 3B+ has pins for I/O connections to easily test and should be powerful enough for machine learning. When we understand the processing power needed and power consumption, it may be

feasible to switch to either the cheaper zero or more expensive 4.

Different databases:

1. Firebase- Google's cloud no-sql database
2. SQL server- This server was provided and would store our data
3. no SQL server- This was not provided, but would allow us to grab data in a different way

Firebase would allow us to store lots of data in the cloud, this would mean that if our server were ever to be down we wouldn't ever lose our data. The only way the data would be lost is if Google's service went down. However, this platform costs money which would require more money from our budget to maintain.

Firebase also is easily implemented via Flutter and would be easily implemented by our devs.

SQL servers were provided which would provide zero maintaining cost. Also, all devs have used SQL queries and a relational database would be reasonable for our database.

SQL servers are more complex and the queries can get complex based on your structure. Configuring a SQL server would be difficult with a different IDE and a different language from what the devs are used to

No SQL server is simpler to query as they look like JSON objects. Also, our project does not require much relational database needs.

No SQL servers were not provided and all the devs would need to learn how to query. There are also a plentiful amount of No SQL databases to choose from.

4.5 Design Analysis

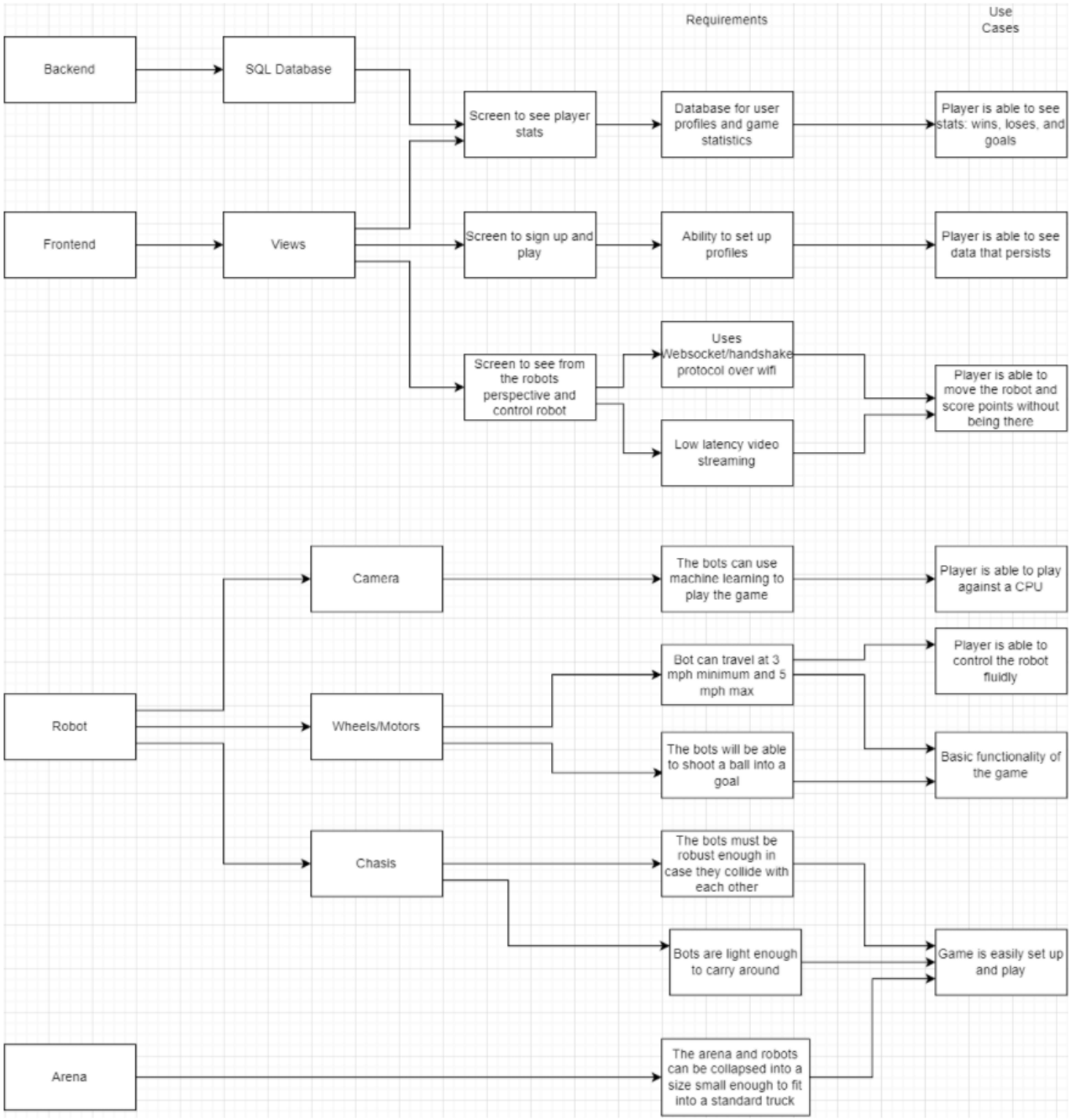
– Did your proposed design from 3.3 work? Why or why not?

Yes, our design from the hardware and software side are flushed out. The only thing that needs to be worked on are the details and problems that will arise while implementing the project.

– What are your observations, thoughts, and ideas to modify or iterate over the design?

Some parts of the document are more flushed out than others. For example, we have details on how we want to implement each part of our software side, but we haven't figured out how to connect every piece yet. We also have a plan to communicate with the bot, but we haven't implemented this yet. These will be more flushed out as we start implementing it and as problems arise with our solution. On the hardware side we have a plan of how we want the arena to look like and where everything should go, but physically it isn't there yet. The details on where things should go will be modified as we start creating the arena.

4.6 Design Plan



5 Testing

Our testing plan is split between hardware and software. Software side we plan to use unit testing that's connected to our CI/CD pipeline to ensure that our software continues to meet our requirements. Some unique challenges are having multi platform support. Thus, it's hard to ensure that all our platforms have fluid UI design.

Hardware plans to test system requirements using a trial approach. This means that we will have number requirements that will be met by having multiple trials and average out the trials. This will allow for us to take out any outliers and allow us to show graphically that our system meets requirements.

5.1 Unit Testing

Our main units to be tested are the widgets that make up our Flutter application and the helper functions that run on our server. Flutter provides its own tools for unit testing, and we can also use Docker to automatically execute our unit tests, as well CI/CD in git. We will use unit tests on the functions we create to alter or interpret our data within a closed environment. When it comes to testing our hardware components, we will feed the motors hardcoded movement commands to see how the bot reacts to a given input value. For the bump sensors we will use print() methods to display on the console when bumps are being detected to check if it matches to when the bumps are actually happening.

5.2 Interface Testing

Our backend will be tested using unit testing. We will do this by testing whether our api calls return the appropriate values from our database. This can be done by simply doing an api call and comparing those values to the assumed values that it should return. For example, we tested the goals made by a user by comparing the api call and how many goals the user has in the database.

Hardware has tested the speed of the robot. Hardware did this by testing how fast the motors registered over a 100m distance. We then did multiple trials, took our outliers, and averaged the numbers. This gave us appropriate numbers that we were happy with.

5.3 Integration Testing

Critical parts to integrate is the connection between the camera feed and our website. This is critical as the user should be able to see what the robot is seeing and move accordingly. We will test this by seeing if our network packets are being transmitted correctly and whether there are any packet drops. We can use tools like ping to see if the Raspberry Pi has a stable connection for this.

Another critical part would be the camera able to identify important objects in the arena: robots, ball, and the goal. This is critical as our game requires these be identifiable. We will test certain parts of our machine learning to see if it can identify the objects and at what confidence. We will also test if it can continue to do this while the objects are moving at their max speeds.

5.4 System Testing

Since our robots will be operating under predefined circumstances and rules, testing the system as a whole can be done simply by setting up the game in the scenario that we want to test and observing if everything proceeds in an acceptable manner. For example just by giving the robot set of input commands that equates to something as simple as "proceed forward at maximum speed for 2 seconds" we can test that the connection between the robot and the controller is working, the robot can properly decode the "move forward" instruction, the motors respond to that instruction by actually moving forward, the maximum speed is an appropriate speed(as defined under project requirements) for our gameplay, and the robot stops when expected to, all at once. This can be done either with our physical prototype robot or on an online simulator depending on what's more convenient for what is currently being tested. When doing these tests it is important that we cover every possible scenario that can occur throughout playing the game and that we are thorough enough to ensure that the desired outcome is reliably achieved and not just coincidentally working on the few times it was tested.

5.5 Regression Testing

During this section, we will be describing how we plan to handle new features to our project that will be building on old ones. Our main tool for handling this kind of regression testing will be through the use of Git .

Git allows each individual programmer to create a new branch whenever they begin a new task. That branch can then pull existing code from the master branch and create a new feature or fix a bug in existing code. By having a new branch created, the master branch is unaffected by the new code while it's being worked on the new branch. Everything will continue to be available in its previous state since no new code will be pushed to the master until its completion. If that new branch accidentally breaks any of the features with their changes. Git will allow us to set up automated tests such as CI/CD pipeline that runs when merging the new branch back into master. This will catch any conflicts of code that broke any features that were already in place.

In the event that a game-breaking bug goes undetected to the master branch, Git allows for the entire branch to be reverted to any previous commits. This will delete any new work created from the last commit but will allow for the branch to return to the previous state.

Finally, in the event two users make different changes to the same code, Git will not allow for a full merge to that branch until the conflicts are resolved. This means it will catch scenarios where the code being worked on by one user can be tested and can pass on their branch and another user's code can also work

perfectly when tested on their branch. However, upon pushing that code to the master branch the code will fail in conjunction because of a dependency that no longer exists in the format expected once the other user's work is considered on the same branch.

By using Git, if a new feature or patch would break an existing feature, the option to choose the better version between the existing code and the new code. This allows the users to prioritize the versions where the core gameplay mechanics are intact at the expense of extra features regardless of which version is newer.

5.6 Acceptance Testing

This section will describe how functional and nonfunctional requirements are being met. As well as how we plan to involve our client/advisor in our acceptance testing. With our end product being designed to be played as a game, it would be best for many of our functional and nonfunctional requirements to be measured by third party input. In order to test our ease of usability and UI/UX requirements, we would be required to see how well end users learn the game controls without assistance. Since our team will not be around every time to teach how to play the game.

This can be measured either in a guided way by timing and observing how long it takes an end user to learn the game mechanics, and/or by asking the user about their experience with the controls and interfaces via survey. This data can be collected via survey. Additionally, nonfunctional requirements such as the aesthetic appeal of the bots and application can be inquired about in the survey to see how our product stands up to meeting aesthetic appeal to end users.

Our client can be involved in the acceptance testing by playing the game, driving the bots, interacting with our application, and providing feedback through each testing phase to help us iron out details before displaying it to all end users. Our acceptance testing should take place over several demos with our client and/or alpha test users so that we can improve our end product with each iteration of feedback.

5.7 Results

During this section, we will be explaining how during our testing process we will measure our compliance with the requirements. We can ensure compliance with the requirements by testing our qualitative requirements to meet those expected projections that our client/advisor and team members have come up with during our design phase.

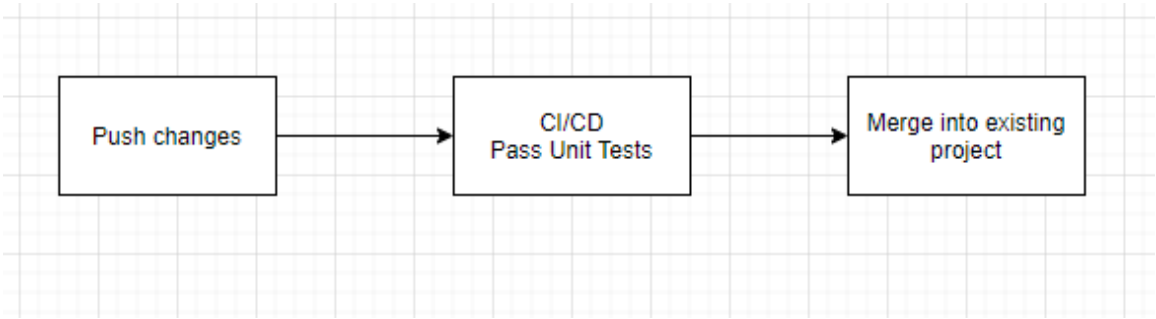


Figure 5.7.1 Testing Process on Code Level

Figure 5.7.1 shows how our code testing process will be completed by using Git as described in section 5.5. We will be pushing code from separate branches onto the master branch. Before it can be applied to the master branch, it must pass the CI/CD pipeline tests that show that there are no errors or conflicts between commits before being merged into the existing code on the master branch. In order to test for UI appeal and ease of use, we will be using alpha users to describe their experience while learning the game and how well the UI was laid out.

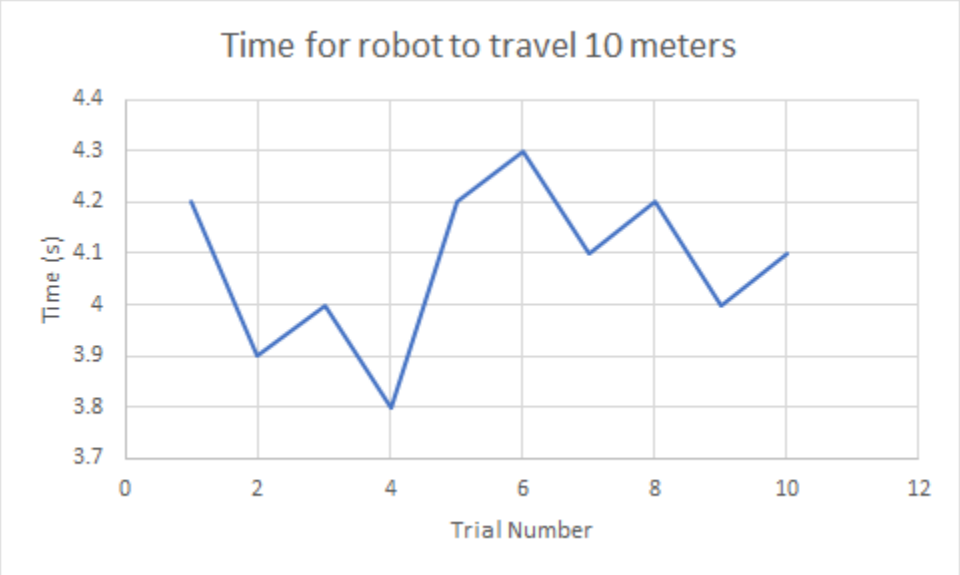


Figure 5.7.2 Testing Hardware

Figure 5.7.2 shows the hardware testing process. The bots will be tested via how long it takes for them to reach 10m. This will then be compared to the other values to see if the average values are within our requirements. The recording of this data can be done via an excel spreadsheet to display a trendline and see if any test falls too far below the accepted values on multiple runs.

6 Implementation

App-Dev Team

- Database
    - Create a database (SQL Server or MySQL) on our ISU server
    - Create a database schema for profiles and game statistics
    - Document connection details for the database
- Server
    - Ensure our ISU server has proper software installed for running our server software
    - Implement socket server
    - Configure connection to database
    - Create endpoints for socket connections with user app and raspberry pi's
    - Document connection details for the socket server
- Flutter App
    - Implement user authentication
    - Implement socket connection to server
    - Create page for profile viewing and editing
    - Create page for statistics and rankings viewing and querying
    - Create game setup page
    - Create game spectator viewing page
    - Create game play page with control interface for user
    - Customize app for different platforms as needed

Hardware Team

- Bots
    - Double check prototype plan with P.H.I.L.L.I.P components sizes
    - Order components needed (motors,pi,L298N,Camera,Batteries) for second prototype
    - Build composite shell in machine shop using scrap material
    - Add electronic components
    - Glue sections together
    - Modify and repeat above steps for another bot.
    - (Optional) Add additional features (headlights,speaker,turn signals)
- Arena
    - Double check plan dimensions with bot plans and check order quantity
    - Order / buy parts needed for the arena
    - Build arena frame
    - Add canvas walls
    - Add arena cameras and power supplies
    - Adjust arena cameras' positions
    - (Optional) Add additional features (Lights, drawbridge, score display, jumbotron)

Dependencies

- Video and controls need both teams to collaborate

-Create an existing prototype

-Test our current speeds of the prototype

-Create an arena that suits the need of the game

7 Professionalism

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

IEEE Code of Ethics:

https://www.ieee.org/about/corporate/governance/p7-8.html

Area of Responsibility	IEEE Code of Ethics
Work Competence	To maintain and improve technical competence.
Financial Responsibility	Honest estimates, reject bribery, give proper credit to the contribution of others
Communication Honesty	Protect others' privacy, and disclose conflicts to affected parties.
Health, Safety, Well-Being	To hold paramount the safety, health, and welfare of the public. Treat all fairly
Property Ownership	To accept criticism, correct errors, and give credit to whom it is due.
Sustainability	To strive to comply with sustainable development practices. Reveal factors that might endanger the environment without delay.
Social Responsibility	Improve the understanding of society about emerging technologies

7.2 Project Specific Professional Responsibility Areas

	Application to Project	Team Performance
Work Competence	The EE students are mainly focused on the hardware side of the project while the CPRE/SE students are mainly focused on the software side.	High: The separate teams make individual progress as a team and meet up weekly to further the development of the project.
Financial Responsibility	Deliberations are made as a team to determine the budget for each aspect of the project.	High: Every component is researched to find the best deal for the desired performance.
Communication Honesty	Weekly reports are submitted to the advisor, while an open form of communication is present for real time collaboration between the project team as a whole and the individual teams.	Medium: Weekly formal reports are submitted to dr. Rover and the team members utilize slack and discord for informal communication. Some weeks we have issues getting everybody to report their progress in a timely manner.
Health, Safety, Well-Being	Our project has lots of physical components so we must consider potential safety hazards whenever we are making design decisions.	High: We've already decided on a limit for the robot's maximum speed. We just need to be mindful of potential dangers posed by the materials we select to construct our project out of, especially if they were to become damaged.
Property Ownership	We have to respect each team member's ideas for our brainstorming sessions, as well as acknowledge the creators of the game our project is based upon (Rocket League).	Medium: We share our ideas with each other, and we respect each person's input as part of the decision making process. We are basing our game off of the video game Rocket League without contacting them, which may be disrespectful towards their creators.
Sustainability	Minimize the amount that could end up as waste.	High: Most of the components can be repurposed and scavenged from the project easily. We are trying to get old electronics and materials to use for our project, so we don't have to use new materials.
Social Responsibility	We are making a game that will bring joy to the users, and show Dr. Rover that embedded machine learning can be a potential course for students at ISU.	Medium: We have not been focusing on our social responsibility, and we have instead been mainly focusing on the technical aspects of the game.

7.3 Most Applicable Professional Responsibility Area

Being that our project is primarily designed for recreation and not financed by a company, financial responsibility is key to the success of our project. Financial responsibility is also an area where we have exercised a high degree of proficiency.

Currently, our project has no external financial support outside of the provided budget of $ 500-$ 600 provided by this being a senior design project. While Dr. Rover has offered to support us within reason if that budget is not sufficient, we still are planning only within the already allotted financial constraint. Every decision we have made as a team has been made considering the financial implications of it. When considering which areas our project can grow in with the most potential, the software space is favored as open source software is available for most solutions and otherwise, we can create what we need while hardware has a material cost in most situations. Many costly materials we have needed for the prototyping phases of our project have been borrowed from the ETG rather than purchasing what may not be the best fit. This allows us to experiment and determine what materials (raspberry pi's, servos, etc.) will best suit our project within our budget constraints.

8 Closing Material

8.1 Discussion

We have been working this semester to design both the hardware and software portions of our Robot League game. Before we could begin with the design, we had to decide upon the rules and come up with the entire structure of the game. After we reached a decision on the rules and composition, we had to design the robots, arena, and the application.

While we still have multiple components of our project to complete, we are comfortable with the progress that we have made up to this point. All of our requirements have been addressed, whether that be by satisfying them, coming up with solutions to requirements that are not met, or modifying requirements that we decided were not sensible.

8.2 Conclusion

In order to design the robots, we started early on in the semester with a prototype to determine if the materials we already had would be sufficient. We found that the battery pack we had, which had a voltage of 7.5V and capacity of 2 Amp-hours, was not sufficient to power the motors. The motors required a higher voltage, and they also operated at a much higher RPM than we need for our robots to travel at the speed requirements we chose.

For our arena design, we have not really made any significant changes than what we first decided for the layout. It will be approximately 4 feet wide, 8 feet long, and 5 feet tall. We originally planned to have the arena taller, but decided that using 2 separate overhead cameras would allow for them to be closer to the ground.

For the machine learning portions of the project, we had to make some changes to what we originally planned. While we originally decided to implement machine learning for both object detection and autonomous driving, we had to remove the machine learning requirement for autonomous driving. We realized that we would not have enough time to create such a complex machine learning model, and did not want to risk not completing the project after allocating most of our time to one aspect. We also originally planned to train the machine learning models ourselves using transfer learning and adapting a pre-built model such as the ssd_mobilenet_cpu_coco model, but we were unable to accomplish this due to processing constraints for the computer devices we were using.

For the software side, we set about this semester to learn how to use the flutter IDE. None of us had used this software before but during our research it checked off many of our boxes. We spent much of our time making sure we understood the software while continuing to design basic outlines of screens and refine our criteria needed for different target users.We wanted to make our software available for multiple platforms and create something that’s aesthetically pleasing. While Flutter provided us with a useful platform to build an application that could handle these requirements, the fact that we all had to learn a brand new technology was our biggest hurdle this semester and consumed the vast majority of our time, leaving little room to work on much else as our design decisions are closely tied to our ability to understand Flutter’s capabilities. This was mostly unavoidable, but the time spent learning Flutter this semester should put us in a better position to succeed when dealing with it in the future implementation phase.

8.3 References

[1] “Automl Vision Object Detection Documentation | google cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/object-detection/docs., [Accessed: 06-Dec-2021].

[2] B. Peabody, “Server-side I/O performance: Node vs. PHP vs. Java vs. go,” *Toptal Engineering Blog*, 11-May-2017. [Online]. Available: https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go, [Accessed: 06-Dec-2021].

[3] “Flutter documentation,” *Flutter*. [Online]. Available: https://docs.flutter.dev/, [Accessed: 06-Dec-2021].

[4] “Flutter_mjpeg: Flutter Package,” *Dart packages*, 11-Jun-2021. [Online]. Available: https://pub.dev/packages/flutter_mjpeg, [Accessed: 06-Dec-2021].

[5] N20 Electric Mini Micro DC geared motors 3v 6v 12v low speed 15 600rpm in DC motor with mounting bracket wheel tire for DIY Toys: DC motor: - aliexpress,” *aliexpress.com*. [Online]. Available: https://www.aliexpress.com/item/4000125484788.html?spm=a2g0o.detail.1000023.9.532c2e09BGMML6, [Accessed: 06-Dec-2021].

[6] “Object detection : Tensorflow Lite,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/lite/examples/object_detection/overview, [Accessed: 06-Dec-2021].

[7] “Raspberry pi documentation,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/documentation/, [Accessed: 06-Dec-2021].

[8] V. Q. T. (vuquangtrong@gmail.com), “Camera live streaming using Mjpeg format,” *Code Inside Out*. [Online]. Available: https://www.codeinsideout.com/blog/pi/stream-picamera-mjpeg/, [Accessed: 06-Dec-2021].

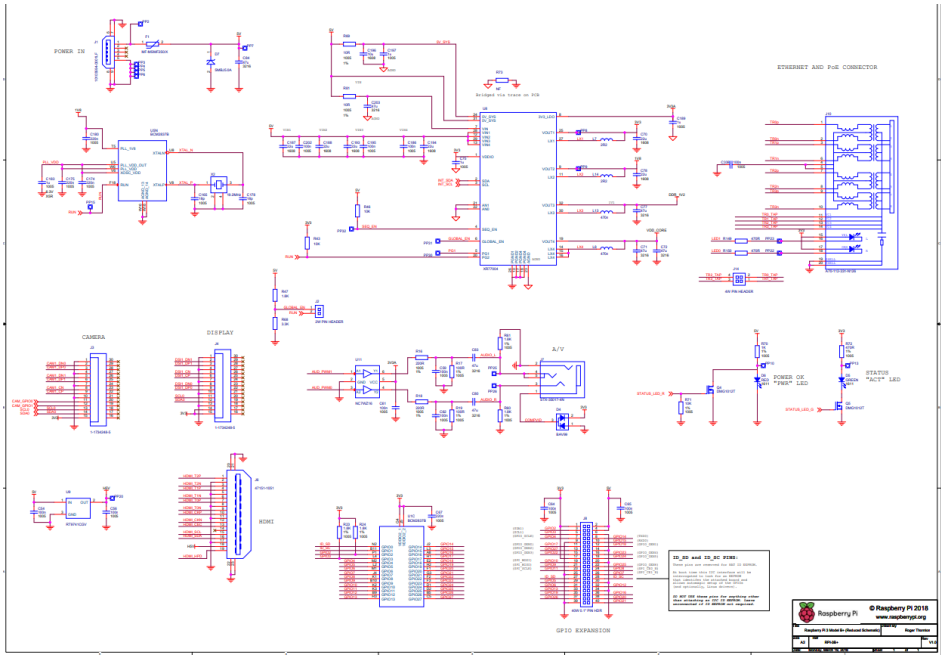
8.4 Appendices

Table 1 is a data sheet of potential motors to use for the next prototype. These brushed DC motors operate at 12 volts and include an attached wheel. Based on the RPM, the 600 RPM unloaded is the most likely candidate.

Reduction ratio (коэффициент уменьшения)	Rated voltage (Номинальное напряжение)	No load speed	Load (Рабочая нагрузка)			Stall	
			Speed	Current	Rated torque (крутящий момент)	Torque (крутящий момент)	Current
(1 :00)	(V)	r/min (±10%)	r/min (±10%)	(MA)	(kg.cm)	(kg.cm)	(MA)
30	12V	1000	800	300	0.3	2.4	300
50	12V	600	480	300	0.4	3.2	300
100	12V	300	240	300	0.5	4	300
150	12V	200	160	300	1	7	300
200	12V	148	118	300	1.2	9	300
250	12V	120	96	300	2	16	300
298	12V	100	80	300	2	16	300

(Table 1 - Motor comparison)

Table 2 is a circuit schematic for the raspberry pi 3B+. This schematic can be used as a guide especially when connecting other components to the raspberry pi.



(Table 2 - Raspberry PI 3B+ schematic)

8.4.1 Team Contract

Team Members:

1) Brogden Worcester
2) Dalton Holdredge
3) Cheyenne Smith
4) David Quan
5) Tejas Agarwal
6) Joseph Holtkamp
7) Noah Brooks
8) Jordan Suby

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
   Friday 10am, Wednesday morning 10am

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
   Discord

3. Decision-making policy (e.g., consensus, majority vote):
   Majority vote

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
   Scribe will type things during the meeting, or we will summarize at the end somehow

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
   We will probably have multiple meetings per week, so try to make at least one meeting a week since we should be able to work that out. Communicate if you cannot make it to the meeting ahead of time

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
   Everybody will do at least 12.5% of the total work for the project

3. Expected level of communication with other team members:
   When we break up into groups, one person from each sub group will update the entire group what was covered

4. Expected level of commitment to team decisions and tasks:
   Everybody puts in their input for decisions and tasks, and participates in the meetings

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
   Team organizer: Cheyenne Smith
   Client interaction - Brogden Worcester
   Individual component design - Jordan Suby
   Team leader hardware - Noah Brooks
   Team leader software - Joseph Holtkamp
   Document submitter/creator - Dalton Holdredge

Progress coordinator: David Quan
Finance officer: Tejas Agarwal

2. Strategies for supporting and guiding the work of all team members:
    - Weekly updates from team members
    - Reach out to team for help if stuck for over an hour
    - Reach out to advisor for help if team is stuck for over a week

3. Strategies for recognizing the contributions of all team members:
   We will have a running record of the tasks and accomplishments that everybody has done

Project Management Style

We will be implementing a Scrum style of project management.

Collaboration and Inclusion

Name	Major	Skills / useful experience / interests	Weaknesses
Dalton Holdredge	EE (Semiconductor Materials)	Arduino experience, willing to learn whatever I need to in order to complete the project, I know some C and Java, construction/building, circuit design, and embedded system design	Not the best at coding
Noah Brooks	E E	Circuit and Embedded System design, also willing to work for as long as it takes to finish a project.	Not the best at coding
David Quan	CPRE	Strong worker and knows a lot of languages. Can create good looking code and well documented code. Able to understand system design.	Poor at hardware and using boards. Can learn and willing to learn.
Joseph Holtkamp	SE	I work currently as a Full Stack Web developer for the Iowa DNR. This gives me experience with UI/UX, software architecture, and general application development. I am familiar with many programming languages and love learning new ones.	I have poor time management skills, I am easily distracted, and I have a lack of experience with circuits/hardware.
Brogden Worcester	EE + CPRE	I have worked with arduino, C, Java, and some python. I have done some android studio but are more interested in website development and Raspberry Pi.	Struggles to complete tasks on time when my schedule is full.
Tejas Agarwal	EE	I have experience with arduino, circuit design, and mechanical aspects to build a robot. I also have a little experience of C and Java.	I am not great at coding and have poor time management skills. I get distracted with other things quite easily

Cheyenne Smith	CPRE	Coding languages known: C, c++, c#, unity, Android studio, java, linux, python. Can find examples of something similar to code I'm working on quickly. I Like learning new things. Keep track of deadlines. Tries to have everyone involved, likes to hear others opinions and ideas. Have little experience using pcb software. Programmed applications for frontend work. Have converted code from one language to another. Easy going nature. Likes to joke but serious when need be. Steps up when needed. Embedded systems work. Worked with a pi before. Can read datasheets. App design.	Struggles with attention span when it comes to meetings and/or lectures. Horrible when it comes to making circuits design. Struggles with conflict ie arguments that are non productive.
Jordan Suby	CPRE	My biggest strength is I can code for hours without getting bored. Most of my experience is in C or Java, but I am perfectly happy to learn new things if the project calls for it. I do have some past experience with CAD, but it was long ago so it is very rusty.	I often write messy code at first while I'm working out the kinks and then try to go back and clean it up later, which can cause issues if I run out of time to do so before a deadline. I also tend to struggle with providing extensive documentation because I get locked in on getting my code to work.

1. Strategies for encouraging and support contributions and ideas from all team members:
   Consider all ideas, give thumbs up in the chat to make people feel good about themselves

2. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)
   If anyone has a conflict with another member, they can speak to them individually.

Goal-Setting, Planning, and Execution

- Team goals for this semester:
  - Figure out what the project is going to be
  - Define interfaces between the hardware and software
  - Define the rules for the game

- Everybody contributes as much as they can
- Fluid communication between team members
- Attend meetings regularly when you can

- Strategies for planning and assigning individual and team work:
  Everybody plays to their strengths or volunteers for what they are interested in. If you do not volunteer for tasks, you will be voluntold what to do.

- Strategies for keeping on task:
  The team leader in the meeting can make sure to keep the meetings on task, and talk with team members
  Weekly meetings and updates, milestones, larger tasks will have more members.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
   Deduction of grade points, talk to see what is the problem to see if they need help.

2. What will your team do if the infractions continue?
   Talk to Dr. Rover or instructors

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _David Quan_____ DATE _____9/10/2021_____
2) _Tejas Agarwal_____ DATE _____9/10/2021_____
3) _Dalton Holdredge_____ DATE _____9/10/2021_____
4) _Cheyenne Smith_____ DATE _____9/10/2021_____
5)_ *Brogden Worcester*_____ DATE _____9/10/2021_____
6)_ Joseph Holtkamp_____ DATE _____9/10/2021_____
7)_ Jordan Suby_____ DATE _____9/10/2021_____
8)_ Noah Brooks_____ DATE _____9/10/2021_____