2 Project Plan

2.1 Project Management/Tracking Procedures

Which of agile, waterfall or waterfall+agile project management style are you adopting. Justify it with respect to the project goals.

We will be using a modified agile style of project management. This will allow us to work in iterations as we add features to both our application and our robots.

What will your group use to track progress throughout the course of this and the next semester. This could include Git, Github, Trello, Slack or any other tools helpful in project management.

We will use Git/GitLab for version control and coding collaboratively, our communication will take place on Discord, and our tasks will be organized on Jira.

2.2 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. This step might be useful even if you adopt agile methodology. If you are agile, you can also provide a linear progression of completed requirements aligned with your sprints for the entire project.

Tasks:

Machine learning:

- Create virtual environment to teach the bots, thus saving time
    - Research if this has been done before, or if we are starting from scratch for our application
- Write object detection ML algorithms for detecting the robots and the game ball
    - Decide if we are writing our own or using open source object detection algorithms
- Write algorithm for teaching the bot to get the ball into the goal
- Write algorithm for teaching the bot to defend its goal
- Write algorithm for teaching the bot to decide whether to be on offense or defense

Bot 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
- Acquire motors, cameras, wheels, and motor drivers
    - Decide how much speed, torque, and current, and space is necessary for our application
- Construct robot chassis and assemble the bot
    - Decide how much durability the robots will require

    - Test the design for robustness and durability

Application:

Frontend / 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
    - Control view: video stream from bots and control interface
    - Game setup view
    - Account creation view?
    - Player profile view?
    - Live game feed for non-participating viewers
- Implement UI.
- Distribute application.

Backend / 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
- Store the machine learning data sets (if not doing embedded ML)
- Research hosting services. Select the best one for our needs.

Database

- Research and decide between SQL and NoSQL databases.
- Design tables for user profiles, game results, and possibly machine learning training data.
- Create tables.

Sensors

- 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

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

What are some key milestones in your proposed project? It may be helpful to develop these milestones for each task and subtask from 2.2. How do you measure progress on a given task? These metrics, preferably quantifiable, should be developed for each task. The milestones should be stated in terms of these metrics: Machine learning algorithm XYZ will classify with 80% accuracy; the

pattern recognition logic on FPGA will recognize a pattern every 1 ms (at 1K patterns/sec throughput). ML accuracy target might go up to 90% from 80%.

In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

How we will determine progress on a given task:

We will split jiras into different categories: Large task, 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 machine learning algorithm x to have y accuracy. We will be able to see how much it has improved and by how much. 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. 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

-Interface with our prototype

-Create machine learning algorithm to recognize the ball at 60% accuracy

-Create a second prototype that improves on maneuverability by 20% (Speed, turn rate, etc)

-Create two goals and create a machine learning algorithm that can detect when a ball goes into the goal at 80% accuracy

-Create a connection with the bots and retain connection 100% of the time

-Have the bots last 10 minutes of play time

-Start creating a machine learning algorithm that would allow a bot to score 1 goal

2.4 Project Timeline/Schedule

• A realistic, well-planned schedule is an essential component of every well-planned project

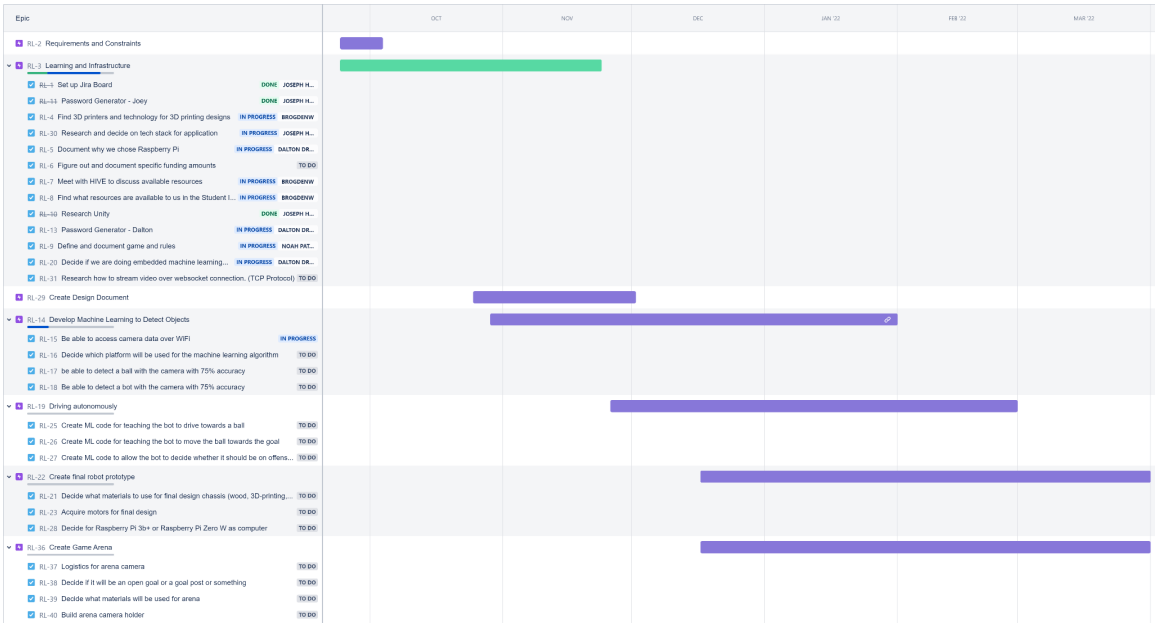
• Most scheduling errors occur as the result of either not properly identifying all of the necessary activities (tasks and/or subtasks) or not properly estimating the amount of effort required to correctly complete the activity

• A detailed schedule is needed as a part of the plan:

– Start with a Gantt chart showing the tasks (that you developed in 2.2) and associated subtasks versus the proposed project calendar. The Gantt chart shall be referenced and summarized in the text.

– Annotate the Gantt chart with when each project deliverable will be delivered

• Project schedule/Gantt chart can be adapted to Agile or Waterfall development model. For agile, a sprint schedule with specific technical milestones/requirements/targets will work.



2.5 Risks And Risk Management/Mitigation

Consider for each task what risks exist (certain performance targets may not be met; certain tools may not work as expected) and assign an educated guess of probability for that risk. For any risk factor with a probability exceeding 0.5, develop a risk mitigation plan. Can you eliminate that task and add another task or set of tasks that might cost more? Can you buy something off-the-shelf from the market to achieve that functionality? Can you try an alternative tool, technology, algorithm, or board?

Task	**Risk**	**Probability of risk**	**Mitigation**
Requirements and constraints	We make requirements for the game that are not feasible to implement, such as choosing the wrong microprocessor or too many functionality requirements	0.25	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	Ask for help from qualified sources.
Create design document	It does not get finished in time	0.1	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	We will start working on accessing the camera data in advance, so we can work out any issues we have
	We pick a model that takes up too much time or data processing	0.25	We research existing models of object recognition algorithms, and decide on one that will work with our application
	The cameras do not have high enough resolution to accurately detect the objects	0.2	Putting distinguishing marks on top of the robots such as different color symbols with high contrast to the background, and having a ball with a contrasting color to the ground
Drive autonomously	We won't have enough time to physically train the ML models	0.5	We will work on creating a virtual environment to train the models instead, so it will save time

Create final robot prototype	We won't have enough time to create the physical prototype of the robots	0.5	We are starting early prototyping now, and we have deadlines for deliverables for our other tasks to keep us on track to complete everything on time
	The motors don't have enough torque or high enough speed	0.25	Early prototyping to find motors that will be suitable
	The robot chassis is not strong enough	0.2	We will prototype with various materials to find the best one for our application
Create game arena	The arena is too large to be easily captured with the arena camera	0.1	We check the camera's field of vision beforehand to see what its capabilities are
	The arena materials take up too much space when they are stored	0.25	We choose materials

Agile projects can associate risks and risk mitigation with each sprint.

2.6 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in the total number of person-hours required to perform the task.

Task	Man-Hours
1 Requirements and Constraints	**3**
2 Learning and Infrastructure	**43**
2.1 Set up a Jira Board	2
2.2 Password Generator	2
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 to us in the 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 doing embedded machine learning or using a server	3
2.13 Research how to stream video over websocket connection. (TCP Protocol)	8
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 platform will be used for the machine learning algorithm	16

4.3 Be able to detect a ball with the camera with 75% accuracy.	32
4.4 Be able to detect a bot with the camera with 75% accuracy	32
5 Driving autonomously	**384**
5.1 Create ML code for teaching the bot to drive towards a ball.	128
5.2 Create ML code for teaching the bot to move the ball towards the goal.	128
5.3 Create ML code to allow the bot to decide it should be on offense or defense	128
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	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
Total person hours	**685**

2.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) required to complete the project.

- At least 2 geared DC motors per bot (for a total of 4, but possibly 8)
- 3 microcontrollers, most likely raspberry pi 3b+ or raspberry pi zero w
- Jumpers and wires for connecting the hardware
- Chassis for the bots, most likely a basswood or acrylic laser cut one. We may get a 3D-printed chassis if we have access to a printer.
- Tools for machining the chassis
- Power supplies for each bot, and power supply for the arena camera (one for the motors and one for the Raspberry Pi on each bot, for a total of 5)
- 2 cameras for the robots to have first person view within the GUI, and one camera above to utilize for object detection. Each camera will require at least 8MP resolution.

Application

- Frontend framework with cross-platform compatibility.
- Backend: server to orchestrate communication between the bots, users, and the database.
- Database: to possibly store user profiles and maybe store ML related data.
- TCP connection between robots, application, and server.
- Hosting service to run server on