UNIVERSITY OF CENTRAL FLORIDA

Robot Basketball

Final Project Documentation - Senior Design I, Group 9, Fall 2019

Brandon Gross • Electrical Engineering
Suvrat Jain • Computer Engineering
Cory Ricondo • Computer Engineering
Mathew Schneider • Computer Engineering

Table of Contents

Table of Figures

Table of Tables

1.0 Executive Summary

The project documentation is centered around the design of a Robot Basketball Arcade game. The arcade basketball arena sits on top of a tabletop and one to two players can pick up a controller and move around the robot basketball player to intake and launch a small basketball into a hoop. The arena garners attention from near and far with an exciting display of robot athleticism and engaging displays and sounds. The project provides an exciting platform to task the team with modern engineering challenges such as robotics, computer vision, game development, and embedded systems.

The major systems are designed with one primary goal in mind: player engagement. The final product is ultimately meant to be fun and entertaining such that people want to keep playing the game. In order to achieve this goal, the project's features and functions are fully described in requirement specifications and constraints, and relevant standards are researched and implemented where appropriate. The project is split into 3 major systems: Robot, Arena, and Game Systems. Each system contains many subsystems and components that interface with one another to implement functionality and features. The robot is responsible for picking up and launching a ball with a fast and capable mechanical system that feels fluid to the player. The arena handles high level logic and computer vision to maximize robot intelligence and autonomy. The game system provides a high-fidelity representation of the robot and arena to guarantee the player can fully engage with the system with minimal frustration. The game system also gives full control to the player to customize the robot's functionality to match the user's playstyle. The project includes both high and low-level software to hide complexities from the player to ensure maximum usability and accessibility.

The following report details the full design process including project description and narrative, engineering requirement specifications, realistic design constraints, system architecture, a detailed breakdown of system components and an administrative approach. Each system component contains a description, relevant research, design, and prototyping and testing sections. The component description translates the project's requirements and features to a narrative discussion detailing the various design aspects of that particular component. The research sections discuss in detail the possible technologies, high-level designs, or purchasable components that satisfy requirements for the component. The design section fully defines the ultimate design that the team utilizes to solve the requirements for the project. The prototyping and testing sections describe how the design is to be built and tested to ensure that the component actually solves the problem within required specifications. Each section is designed and described with the previous section's design decision in mind but attempts to be agnostic to it. Non-critical interfaces are defined in their relevant sections, but critical interfaces are designed and developed separately in another section to mitigate risk.

2.0 Project Description

2.1 Motivation

Entertainment is an essential part of life in the City of Orlando. Amusement parks, arcades, sports, movies and television retire us of our tiredness and fulfill our lives with optimism and sheer excitement. The Robot Basketball game project is chosen to create dynamic, interactive entertainment for everyone to enjoy.

This project is proposed in the spirit of RoboCup challenge; RoboCup is a standardized soccer-based robotic competition with a variety of leagues. In general, robots compete against one another utilizing complex algorithms developed by engineers. In the case of Robot Basketball, two human players can compete against one another by controlling the robot to move and shoot the basketball. However, due to perception and coordination problems that come from remotely operating robots, the players may need some assistance to maximize amusement. This introduces a complex engineering challenge that involves some level of machine intelligence to achieve high control fidelity.

The team proposes this project as a foundation for learning a wide variety of skills including Robotics, Computer Vision, Machine Learning, PCB Design, Bluetooth communication, Game and App development, and real-time control.

2.2 Goals and Objectives

The overall goal in this project is to create an arcade-style entertainment system that is both robust and intelligent. The product should be able to fit on typical foldable tables and should be playable by at least one, but preferably two people. The system should be designed modularly such that different subsystems can be designed, tested, and created independently without disassembling the entire system. The system should incorporate both high level software and low-level hardware interfacing. The robot should be low cost such that multiple robots can be created. The robot should be capable of collecting and launching the ball into a scale hoop with high accuracy and precision. The robot should be quick to traverse the court to increase mid-game activity. The system should assist the player by performing calculations to increase shot accuracy. The arena should display information to the player including game type, score, and debugging information. The final product should be engaging and attractive.

2.3 Design Process

The design process for this project follows the following pattern: Define the system features from market requirements, Define the subsystem components, Determine the requirements for each subsystem requirement specifications, define the tests to evaluate the subsystem requirement specifications, research components,

design subsystem, prototype subsystem, and test subsystem. This pattern is chosen because it follows the logical progression of system development such that a final product meets the actual market requirements defined by customer. Each test defined early in the process is directly traceable to an engineering requirement specification. The tests are defined before the design is complete in order to create an objective set of tasks to be completed such that the requirements are fully satisfied. This prevents changing tests in order to ensure the test passes. The pattern is shown graphically in Figure 1.

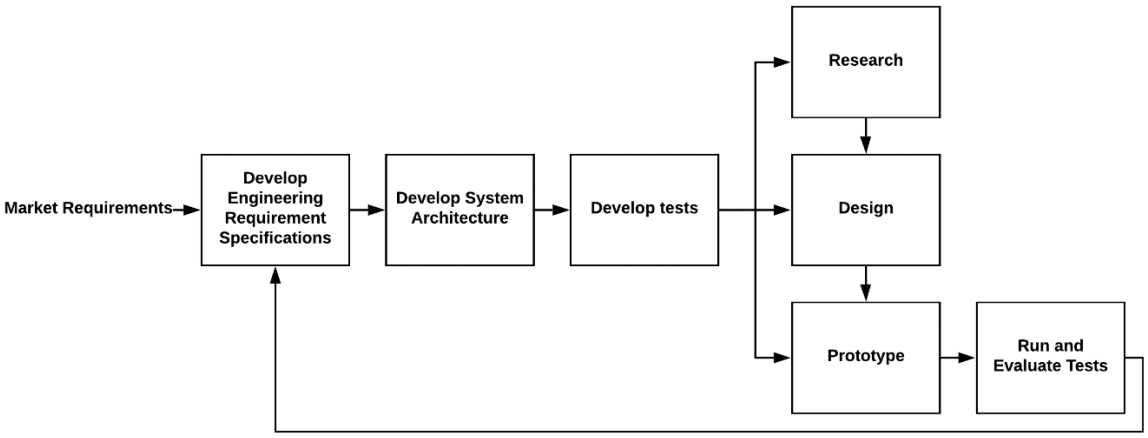


Figure 1 Project Design Process

2.4 Realistic Design Constraints

These constraints are those placed upon the project by environmental factors such as transportation, budget, or customer requirements. The constraints arise from the need to present the project in appropriate settings, and also to constrain the team adhere to deadlines and restrictions placed upon the project by the senior design committee. Additional constraints relate to arbitrary requirements imposed by the team to learn skills or increase understanding in some areas. Others are facility constraints for final presentations.

2.4.1 General

General constraints pertain to the constraints enforced by the university or by team members due to environmental factors or arbitrary distinctions. The identified constraints can be found in Table 1, Table 2, Table 3, and Table 4. The constraints pertain specifically to the Project, Arena, Robot, and game sections such that each high-level system is appropriately designed within boundaries for their respective sections. The project constraints are more general than the other sections due to the constraints imposed by the university for deadlines, transportation, and presentation of the project.

Table 1 Project constraints

Constraint	The project shall…
C.P.1	Be transportable in a standard-sized sedan
C.P.2	Be designed by August 2, 2019
C.P.3	Be built and tested by November 15, 2019
C.P.4	Utilize GitHub as a version control system

Table 2 Arena constraints

Constraint	The Arena shall…
C.A.1	Be powered by a standard US 120V 60Hz wall outlet
C.A.2	Be able to rest on two standard folding tables
C.A.3	Have only 1 cable that plugs into the wall

Table 3 Robot constraints

Constraint	The Robot(s) shall…
C.R.1	Utilize a custom PCB that fits within size constraints required by the project
C.R.2	Utilize a PCB that contains limited through-hole soldering
C.R.3	Be powered by a battery

Table 4 Game constraints

Constraint	The Game shall…
C.G.1	Utilize a market-available Game Engine

2.4.2 Economic Constraints

Economic constraints are constraints that pertain to the microeconomic and macroeconomic factors that affect design decisions. These factors can include things such as taxes, impacts to stock markets, and the general cost and value of a product. In the case of Robot basketball, the primary economic factors are those that limit the quality or quantity of the parts the project can afford. Further, if the project is to be utilized in an actual arcade, some analysis must be done to ensure marketplace viability. The identified constraints can be found in Table 5.

Table 5 Economic Constraints

Constraint	Economic Constraint
C.ECON.1	The project shall cost no more than $1000
C.ECON.2	The robot shall cost no more than $300
C.ECON.3	The arena shall cost no more than $450
C.ECON.4	The game system shall cost $0
C.ECON.5	The robot design and cost shall be scalable to multiple copies

2.4.3 Environmental Constraints

Environmental constraints pertain to the consideration of environmental impacts such as disposal, energy efficiency, or carbon footprint. For this project, the environmental considerations directly relate to the energy efficiency and battery technology. The identified constraints can be found in Table 6.

Table 6 Environmental Constraints

Constraint	Environmental Constraint
C.ENV.1	The project shall be energy efficient
C.ENV.2	The project shall utilize organic materials where feasible
C.ENV.3	The project shall utilize rechargeable batteries where appropriate

2.4.4 Social Constraints

Social constraints pertain to human factors such as psychology, social etiquette, privacy, education, and accessibility. Social constraints are the largest driving force in this project due to the nature of human interaction with the final product. The identified constraints can be found in Table 7.

Table 7 Social Constraints

Constraint	Social Constraint
C.S.1	The project shall be easy to utilize
C.S.2	The project shall display information to enhance understanding
C.S.3	The project and associated documentation shall ensure appropriate terms (Pronouns, avoid trigger words, etc.) are utilized

2.4.5 Political Constraints

Political constraints pertain to the government as an overseer and as a customer. There are no driving political constraints for this project outside of following governing laws and regulations.

2.4.6 Ethical Constraints

The ethical constraints pertain to the ethical design, construction, and operation of the product. The team adheres to the standard IEEE Code of ethics. [1]

2.4.7 Health and Safety Constraints

Health and safety constraints pertain to the safe operation of a product and ensuring no harm comes to a person by being associated with the product. There are several health and safety constraints for this project. The identified constraints can be found in Table 8.

Table 8 Health and Safety Constraints

Constraint	Health and Safety Constraint
C.HS.1	The project shall ensure all electrical components are properly secured and grounded. No bare wires are to be accessible without a locked enclosure
C.HS.2	The project shall ensure all flying objects are appropriately secured and cannot leave the Arena
C.HS.3	The project shall ensure no user can interact with the robot while it is under active power
C.HS.4	The project shall ensure ergonomically considerate devices are utilized when feasible

2.4.8 Manufacturability Constraints

Manufacturability constraints pertain to the construction of the physical device and development of any software required to operate the device. This includes utilizing widely available standard components such as screws, bolts, and designing custom devices that can be made with available tools and machinery. For this project, several mechanical devices are required, and effort is put in to ensure the product can be manufactured by University students with available resources. The identified constraints can be found in Table 9.

Table 9 Manufacturability Constraints

Constraint	Manufacturability Constraint
C.MANU.1	The project shall utilize ISO hardware where needed
C.MANU.2	The project shall be designed with the following available machinery in mind: Saw, Table Saw, Jigsaw, Drill, Laser-Cutter, 3D Printer, Heat Gun, Soldering Iron
C.MANU.3	The project shall utilize as few parts and custom components as possible

2.4.9 Sustainability Constraints

Sustainability constraints pertain to the maintenance and support of the project after development and release to reduce or eliminate the need for additional resources. Additionally, renewable resources are to be utilized to ensure the long-term sustainability of the planet. The identified constraints in Table 10 increase sustainability through the ease of repair, changes, and expandability of the product by the end of the term, and the use of organic materials where possible.

Table 10 Sustainability Constraints

Constraint	Sustainability Constraint
C.SUS.1	The project's mechanical design shall be maintainable
C.SUS.2	The project's mechanical design shall utilize locking hardware where feasible
C.SUS.3	The project shall include expandable hardware for future development

2.5 Engineering Requirement Specifications

The Engineering Requirement specifications found in the following tables are requirements developed by the project team such that the project is fully defined and constrained. The requirements are a guiding force behind the entire project, and each design decision made in the following sections are traceable back to these defined requirements.

2.5.1 Project Requirements

The project requirements in Table 11 define the major subsystem components and the large overarching requirements for the entire product. They act as a governing set of requirements that the project must achieve in order to be considered successful.

Table 11 Project requirements

Requirement	The project shall…
R.P.1	Contain three high-level subsystems capable of communication: Arena, Robot, and Game
R.P.2	Allow a human-player to control the robot-subsystem to drive and launch a ball
R.P.3	Take efforts to ensure safety of both human players and subsystems
R.P.4	Identify high-risk interfaces and fully define & design them

2.5.2 Robot Requirements

The robot requirements in Table 12, Table 13, Table 14, Table 15, Table 16, and Table 17 describe and define the functionality of the robot. The major subsystems under the robot are described in individual tables labeled appropriately.

Table 12 Robot requirements

Requirement	The Robot(s) shall…
R.R.1	Weigh no more than 8 lbs.
R.R.2	Contain a launching mechanism capable of launching a 1.5” diameter rubber ball
R.R.3	Contain an intake mechanism for acquiring a 1.5” diameter rubber ball from ground level
R.R.4	Be sturdy, robust, and resilient regardless of subsystem weight
R.R.5	Perform required functionality regardless of ball holding status
R.R.6	Be resilient to hitting the ball while driving
R.R.7	Be resilient to collisions

Table 13 Robot Base Requirements

Requirement	The Robot’s Base shall…
R.R.B.1	Be capable of holonomic locomotion
R.R.B.2	Traverse in one direction at minimum 0.3 m/s
R.R.B.3	Traverse the court without unintentional slipping
R.R.B.4	Be able to maintain a shot angle while driving

Table 14 Robot Launcher Requirements

Requirement	The Robot's launcher shall…
R.R.L.1	Contain no more than two motors
R.R.L.2	Maintain at least 75% shot accuracy from anywhere on the court
R.R.L.3	Be capable of launching a ball with different forces for a required distance

Table 15 Robot Intake Requirements

Requirement	The Robot's Intake shall…
R.R.I.1	Contain no more than one motor
R.R.I.2	Intake the ball while stationary and moving from a variety of angles

Table 16 Robot Electrical Requirements

Requirement	The Robot's Electrical system shall…
R.R.E.1	Utilize a battery that can safely operate at the loads required for the systems
R.R.E.2	Convert voltage from 12V DC to 9V DC, 7.2V DC and 5V DC with high efficiency
R.R.E.3	Support an embedded controller capable of processing controls for a minimum 6 motors
R.R.E.4	Be power efficient in operation to run more than 10 minutes
R.R.E.5	Utilize a microcontroller capable of communication protocols

Table 17 Robot Software Requirements

Requirement	The Robot's software system shall…
R.R.S.1	Communicate with the arena at a rate of at least 30Hz
R.R.S.2	Utilize sensor data to close feedback loops on relevant actuators
R.R.S.3	Utilize software that is fully unit tested
R.R.S.4	Utilize a robust deterministic state-machine

2.5.3 Arena Requirements

The arena requirements in Table 18, Table 19, Table 20, and Table 21 define the features and functionality of the Arena system and its respective subsystems. Each major subsystem's requirements can be found in the appropriate table. The arena requirements discuss specifications for a variety of components including the arena frame and hardware, court, hoop, and ball. These requirements are critical because they directly affect the performance of other systems such as the Game and Robot. Additional requirement tables include supporting Display requirements, and more importantly, Computer vision requirements that are fundamental to the successful operation of the project.

Table 18 Arena requirements

Requirement	The Arena shall…
R.A.1	Be no larger than 2 meters length, 2 meters width, and 1.5 meters height
R.A.2	Weigh no more than 75 lbs. total
R.A.3	Contain at least 1 rubber ball that is no smaller than 1.5" diameter
R.A.4	Contain at least 1 basketball hoop no smaller than 1.5" diameter
R.A.5	Have flat ground with scale basketball court markings
R.A.6	Be easy to put together and take apart (Less than 3 minutes each)
R.A.7	Contain a surface that is level
R.A.8	Be resilient to impacts such as falling or dropping
R.A.9	Contain walls such that the ball or robot does not go through
R.A.10	Contain accurate basketball court markings
R.A.11	Utilize a ball that weighs no more than 5 grams
R.A.12	Utilize a ball that is not severely impacted by aerodynamic conditions
R.A.13	Securely mount the hoop to the frame
R.A.14	Contain a hoop that can fit a ball no greater than 2.5"
R.A.15	Contain a display to show players and spectators game status
R.A.16	Contain LED lights for status indication and consistent lighting on the court
R.A.17	Employ software that is unit tested

Table 19 Arena Display and Sounds Requirements

Requirement	The Arena Display and Sounds shall…
R.A.DS.1	Contain a display that is widescreen with a refresh rate of at least 60 Hz and 720p resolution
R.A.DS.2	Contain a display that can be viewed outdoors from a distance of 10 feet
R.A.DS.3	Have speakers capable of being heard from 10ft away

Table 20 Arena Electrical Requirements

Requirement	The Arena Electrical System shall…
R.A.E.1	Utilize an AC-DC adapter capable of powering the required DC loads at a high efficiency
R.A.E.2	Contain a DC-DC adapter that converts from the voltage provided by the AC-DC adapter to the required DC voltages at a high efficiency
R.A.E.3	Communicate with the robot subsystem at a rate of at least 30Hz
R.A.E.4	Support a camera for top-down view of the court
R.A.E.5	Support an Embedded Controller capable of running a traditional Operating System
R.A.E.6	Convert voltage from 120V AC to 5V DC
R.A.E.7	Support at least two gamepads
R.A.E.8	Contain sensors to detect when a goal is made

Table 21 Arena Computer Vision Requirements

Requirement	The Arena Computer Vision System shall…
R.A.CV.1	Support vision-based position tracking of the ball and robots in the court with update rate of at least 30 Hz
R.A.CV.2	Have clear color vision from camera mounting height with objects moving
R.A.CV.3	Camera field of view covers the entire court area
R.A.CV.4	Camera is compatible with the arena controller
R.A.CV.5	Support vision-based detection of the ball and robots

2.5.4 Game Requirements

The game requirements in Table 22 define the features and functionality of the game system. Each requirement indicates an aspect of the subsystem that must be accomplished for the project to be considered successful. The Game system is fundamental to the player's interaction with the project. It is the front-facing system and thus should be engaging and entertaining. It also particularly conforms to constraints listed above. The game system has a few sub-components that interact directly with other systems and thus is critical to be well-defined and high performing. If the game system does not function well, it will affect almost everything else as it is functioning as the data hub for the controls sent to and from the robot.

Table 22 Game requirements

Requirement	The Game shall…
R.G.1	Create a 2D visual representation of the Arena and Robot Status
R.G.2	Have a menu to start, pause, and reset a timed match
R.G.3	Display current score and game time
R.G.4	Playback past 10 seconds of gameplay upon a goal
R.G.5	Play a 3D animation of the ball making it into the goal
R.G.6	Perform collision detection between the different objects
R.G.7	Employ software that is fully unit tested
R.G.8	Utilize collision detection to prevent dangerous actions

2.6 Standards

The standards found in Table 23 are relevant engineering standards that can simplify or increase the capabilities of the designs chosen. Utilizing standards results in inter-operability between various systems. It also streamlines decision designs in the event of an available standard that meets requirements. The table gives a quick summary of the standards investigated and the following sections describe the standards followed in detail.

Table 23 Relevant Standards

Standard	Name/Field
IEEE 1872-2015	Standard for Ontologies for Robotics and Automation
IEEE 1012-2016	Standard for System, Software, and Hardware Verification
IEEE/ISO/IEC 29418-2018	Systems and software engineering – Life cycle processes – Requirements engineering
IEEE 802.15.1	Bluetooth qualification
IEEE 1540	Software Risk Management
IEEE 42010	Architectural Description of Software -intensive systems
C Coding Style	C programming standards by Barr Group
IEEE 829-2008	Standard for Software and System Test Documentation
IPC-2221	PCB generic standard
Python style	Google Python style guide for clarify and bug reduction

2.6.1 Electrical Standards

The electrical standards listed below dictate the development, design, manufacturing, and assembly of components related to the electrical systems of the project. This includes PCB design, wire selection, cabling, signal generation, and others.

2.6.1a IPC-2221

The IPC standard IPC-2221 provides aa generic standard for Printed Circuit Board design. It provides general instructions and requirements for the design, mounting, and manufacturing of various PCB material types. The generic standard provides a wide array of other sub-standards to go into significant detail on their respective topics. [2]

2.6.2 Communication Standards

The communication standards define the operation of the system as the functionality pertains to wireless and networked communication. All communication systems must adhere to the standards below.

2.6.2a IEEE 802.15.1

The IEEE 802.15.1 standard applies to wireless arena networks for small, low power devices. The standard contains a wide variety of definitions, data-formats, message types, and structured formats. The standard is based on the Bluetooth standard developed by the Special Interest Group. All devices related to Bluetooth communication must conform to this standard. [3]

2.6.3 Software Standards

The software standards define the development, design, testing, and maintenance of the software systems of the project including style guides for different languages, testing procedures, and design documentation.

2.6.3a Barr Group's Embedded C coding standards

Barr Group's Standard for Embedded C coding follows several guiding principles to define a set of rules for developing software in the C programming language. This set of rules helps maintain strong consistency between different developers and keep a safe, bug-limited software. The standard explicitly calls out things that could avoid bug-related problems and gives examples and reasoning for each guideline. [4]

2.6.3b Google Python Style Guide

Google's python style guide defines the various coding standards to ensure interoperability between python code and developers. The style guide helps avoid a variety of common Python mistakes and reduces code complexity and increases readability. [5]

2.6.3c IEEE 829-2008

IEEE 829 pertains to the standard for Software and system test documentation. The standard defines the appropriate manner to test if processes meet requirements for the process and meets various standards defined by the design team. The standard defines various schemas for particular integrity levels that can be followed to appropriately test fidelity of the system. It also further defines various terms to guarantee the successful communication between relevant parties. [6]

2.6.3d IEEE 1540

IEEE 1540 pertains to Risk management in Software Life Cycle Processes. It helps define a process to determine potential problems, the consequences of problems, and how to address problems found. Key definitions are defined for risk management, and a very clear process diagram is described. [7]

2.6.4 Robotics and General Standards

The robotics standards define key-terms, operations, functionality, and interoperability between robotic systems. General standards pertain to general architecture design and testing outside of those explicitly created for software.

2.6.4a IEEE 1872-2015

IEEE 1872 is the standard for Ontologies for Robotics and Automation. The standard defines general concepts, relations and axioms for a seamless and unambiguous communication between robot engineers. The axioms contained within help define system inheritance in a standardized manner. [8]

2.6.3b ISO/IEC/IEEE 42010

ISO/IEC/IEEE 42010 is the international standard for Systems and software engineering for architectures. The standard defines how architecture descriptions of systems are organized and expressed. This includes viewpoints, frameworks, and language to adequately describe system architecture. [9]

2.6.3c IEEE 1012-2016

IEEE 1012 pertains to the Standard for System, Software, and Hardware Verification and Validation. Verification and Validation is a way to determine whether or not a product meets requirement specifications after the product has been designed and built. This standard is similar to IEEE 829 in that it defines integrity levels and processes related to the integrity level, except that this standard applies to all system levels including software and hardware. [10]

2.7 Project Research

There are several similar projects that are utilized as inspiration for the design, operation, and requirements for this project. The projects found below serve as a foundation upon which the team builds upon. Additionally, many of the mechanisms utilized in the robot's design directly pull from common mechanical applications from the various projects.

2.7.1 RoboCup

The RoboCup competition introduces a challenge for competitors to develop complex algorithms to enhance the capabilities of robots in sports. There are several academic papers published on the topics of computer vision, control, and robot architecture. A useful solution for tracking robots that was developed for RoboCup is the usage of an overhead camera utilizing computer vision to solve the localization and mapping problem. The camera provides a top-down two-dimensional view that is easier to process than a complex three-dimensional scene. RoboCup participants often utilize position and orientation data from camera views to generate paths to acquire a ball and score a goal. RoboCup serves as a platform for advanced robotics research that pushes the field further. [11]

2.7.2 VEX Robotics

The VEX Robotics platform provides a plethora of cost-effective robotics parts that can be utilized for this project. In addition, the Nothing but Net challenge from 2015 and Turning Point from 2018 involved several unique launching mechanisms and locomotion systems for a basketball-like challenge. The Vex robotics platform is a starting place for the mechanical aspects of the robot. The challenge provides a plethora of designs for launching a ball at different forces and ranges, and an inordinate amount of designs for locomotion in a competitive arena. The VEX Robotics platform provides a standard set of parts to compare quality and prices to from other vendors. Some of the parts utilized in the Robot are VEX parts due to their wide availability and versatility. [12]

2.7.3 Stanford's Battle of the Bots

Stanford's 2015 battle of the bots. This challenge very closely matches the scope and scale of our project. The students developed many unique robots that launch balls in a basketball competition at a very similar scale to the one initially considered for this project. This challenge provides a point of comparison for the scale, size, and capabilities for launching a small tennis ball in a basketball context. The robots in this challenge are approximately 1 cubic foot and shoot small dog tennis balls into large hoops that are amounted about a foot tall. [13]

2.8 House of Quality

The house of quality diagram shown in Figure 2 indicates the relationships and correlations between engineering requirements and market requirements. Additionally, the diagram indicates the relationship between different engineering requirements. In summary, some requirements that should be maximized causes an increase in a requirement that should be minimized. For example, increasing the shot accuracy of the project would result in an increased cost of the project. It is important to have a strong understanding of how focusing on one particular aspect of the project results in diminishing the quality of another aspect. Further, it is important to have clear optimization goals in mind to know what the appropriate amount of time is to spend on finalizing the product. While each of the requirements are critical to achieve success in the project, the house of quality also serves as a place to return to in order to identify which aspects of the project are more important to achieve than the others due to the tie back to the market requirements.

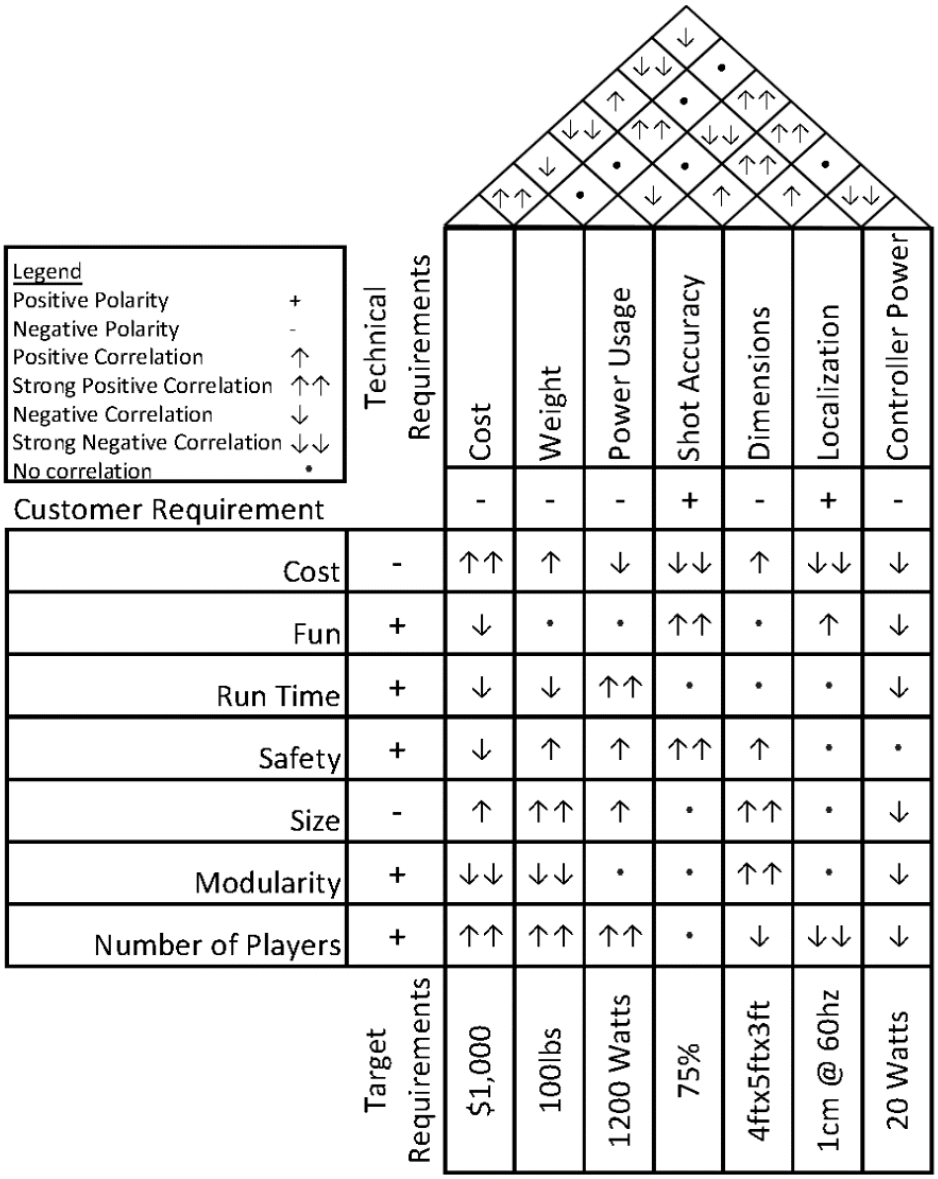


Figure 2 House of Quality

2.9 System Architecture

The system architecture defines the various systems included in the project, and their interactions between one another. The architecture is the highest-level guiding structure for all solutions to the engineering requirement specifications for both hardware and software systems.

2.9.1 System and Interface Identification

The Project is split into three primary systems: Arena, Robot, and Game. The Arena system encompasses all things related to the basketball court, basketball,

physical frame structure, and computer vision. The arena contains a control system for high-level planning and control for commands that are sent to the robot system. The Computer vision subsystem is to determine the position and orientation of the robot on the court. Additionally, it must track the position of the ball on the court. The Game System involves taking data in from the player and displaying information such as game and robot status, instant replays, and other high-level functionality. The robot system is the device for physically interacting with the basketball court and basketball. The robot receives commands from the arena control-system and executes them.

The subsystems identified to achieve the requirements are the mobile base, intake, launcher, and control subsystems. There are several critical interfaces identified for this project. These are looked at separately from their own subsystem such that the individual subsystems can be designed independently. However, this introduces risk that the systems are not compatible. Further, interesting behaviors can emerge when complex systems are put together. Thus, these integration systems are fully designed and tested in conjunction with the individual systems to ensure robustness and consistency. The system architecture is shown graphically in Figure 3.

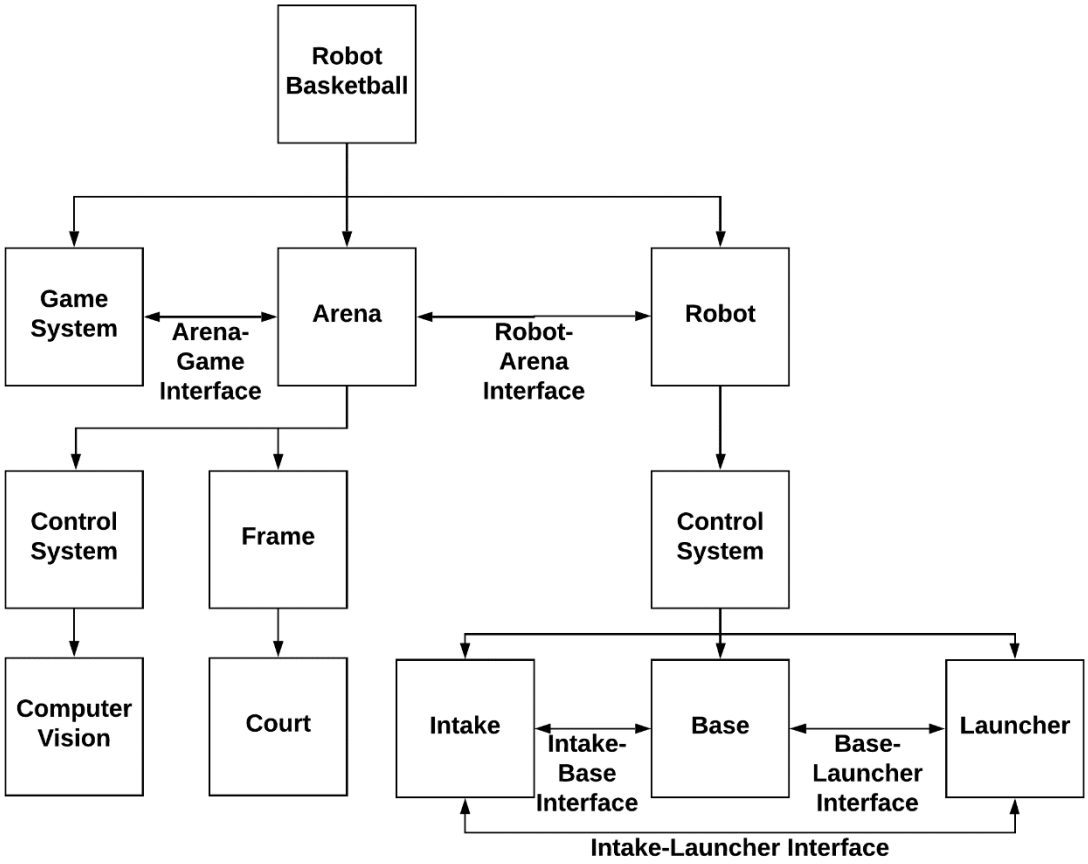


Figure 3 System Hierarchy and Interface Identification

2.9.2 Distributed Architecture

The project is designed and presented as a distributed system. A distributed system is an architecture that contains multiple independent systems that often rely on one another's components. In this case, the robots' responsibilities are separate from that of the arena both physically and computationally. This type of architecture is chosen due to the possibility of scaling the system to a larger number of robots without significantly increasing costs. Dozens of the robots could be built and the arena could be scaled up to a larger size, and the arena cost would remain the same as a single robot cost. The robot is treated as a *slave device* that does minimal processing. The higher-level control systems, computer vision, and hardware are handled by the *master device* (arena). This reduces cost and complexity for the robot by eliminating the need for a camera and a high-power processor. The arena can have increased complexity without significantly changing the system by only replacing a single arena device as the number of arenas or size of arenas increase. The distributed architecture diagram is shown in Figure 4.

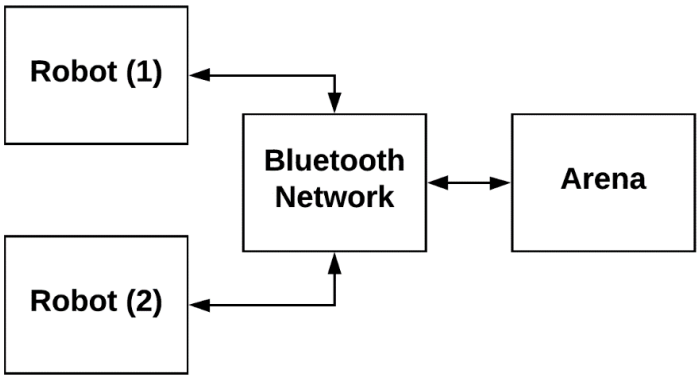


Figure 4 System Communication Diagram

2.9.3 Robot Control Architecture

A robot architecture defines how data should flow such that the robot can effectively interact with its environment. The architecture introduces constraints that drive the design and development of a robotic system. A deliberative robot architecture is chosen for this project because it provides a robust solution to systems that operate in a well-defined space. Due to the nature of the project, most variables related to the operating conditions of the robot such as the number of objects, color of objects, speeds and behavior of objects can be adjusted such that the robot performs adequately under the conditions provided. The general approach to this architecture is to take in data from peripheral devices such as encoders, computer vision, and a-priori knowledge to construct a virtual model that is then deliberated over to plan and act according to a set of pre-defined rules. The architecture is shown graphically in Figure 5.

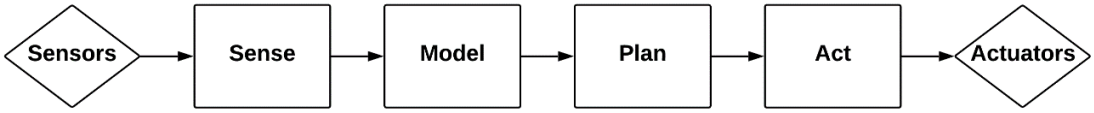


Figure 5 Deliberative Robot Architecture [14]

3.0 Robot

The robot subsystem is comprised of all the components required to pick up and launch a ball from different places on the court. The diagram in Figure 6 denotes the primary systems and their various connections to other systems. The final robot assembly rendering is shown in Figure 7. The robot acts as an I/O device that simply takes inputs from other systems and executes them based on a set of parameters on the robot. Additionally, it provides insight into its state by providing information to other systems.

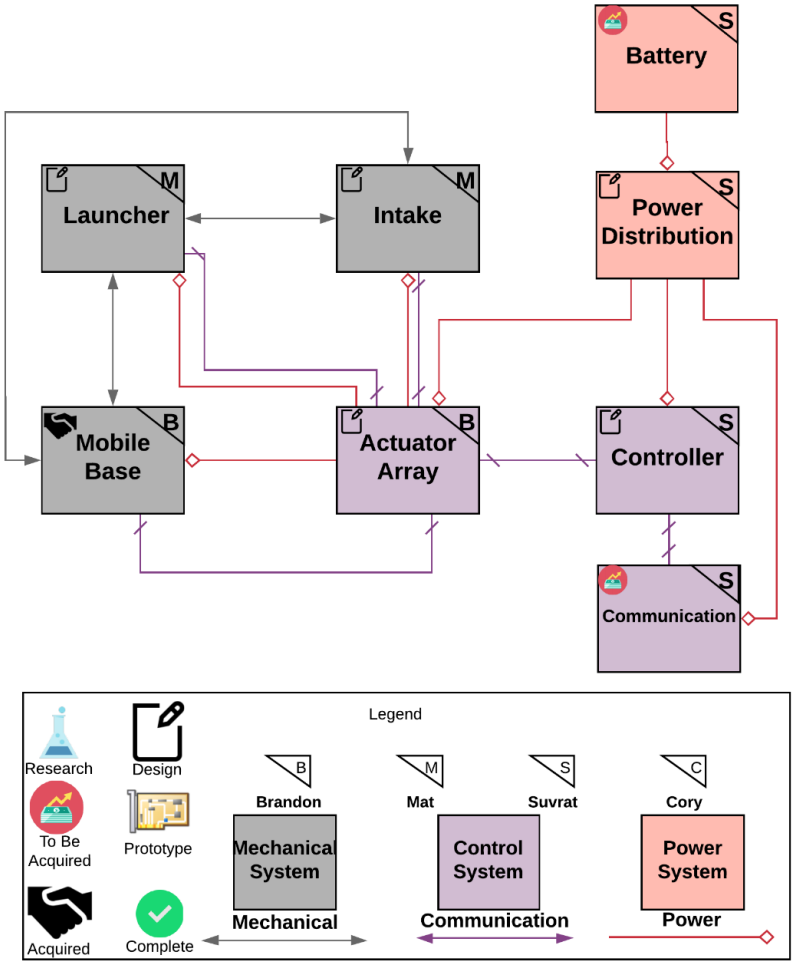


Figure 6 Robot Subsystem Power and Signal Diagram

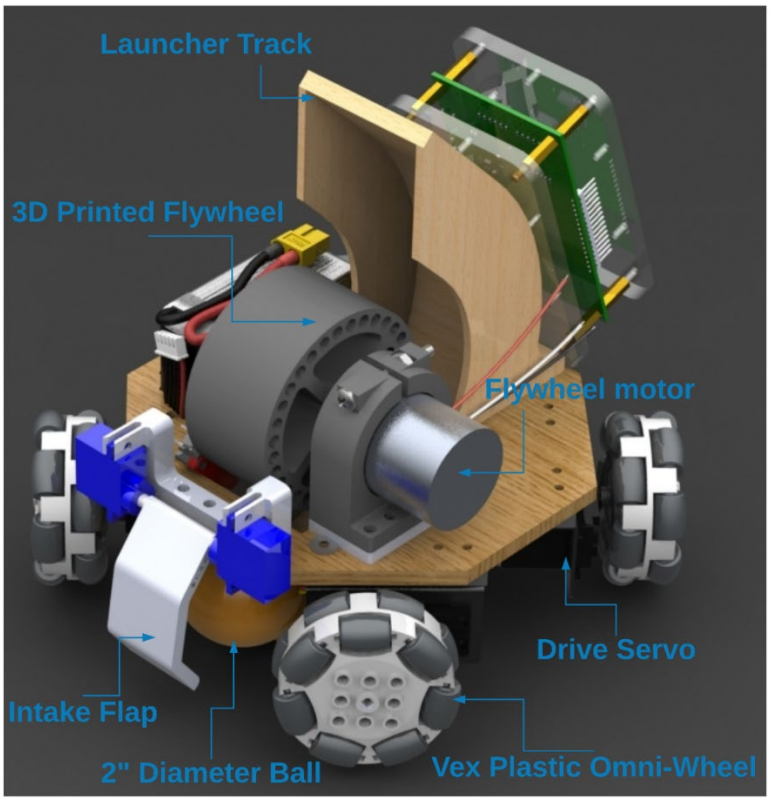


Figure 7 Final Robot render

3.1 Base

The mobile robot base is the locomotion piece of the system. It is the sole source of robot movement on the court. The mobile base is to be fast and agile in order to increase player engagement. If the robot is slow, the player will feel like they are not in control of the robot's actions, and they are not excited to play the game. If the robot is fast and agile, the player can perform complicated maneuvers and make exciting plays. The player experience is also significantly enhanced if the robot does not need to turn significantly to move around and shoot the ball. This way, the player can focus on moving the robot to specific positions and not worry about if the robot can rotate and shoot from that position.

The base platform serves as the main structure for the other subsystems. The Intake and Launcher must seamlessly integrate with the base to ensure robustness and consistency. For example, there must be space for the launcher to extend and retract, and the intake must be able to mount and reach the ball on the ground without interrupting the intaking motions. In the likely event of collisions between robots or between the robot and the arena, the base must be sturdy and stable. The electronics on the robot also must remain safe throughout various operating conditions, and they should be secure and resilient to impacts. The drive system

should also be relatively low power to lengthen run-time, as most of the power in the robot is designated to the subsystem. Finally, the robots are generally the focal points of the entire project, thus they should appear both professional and exciting.

3.1.1 Research

3.1.1a 3-Wheel Holonomic

The 3-wheel design has omni-wheels mounted at 60-degree angles to one another. This allows for full holonomic motion with only three motors. There is power loss driving in cartesian directions because only two of the motors are contributing to the motion. An example of an available 3-wheel holonomic kit is shown in Figure 8.

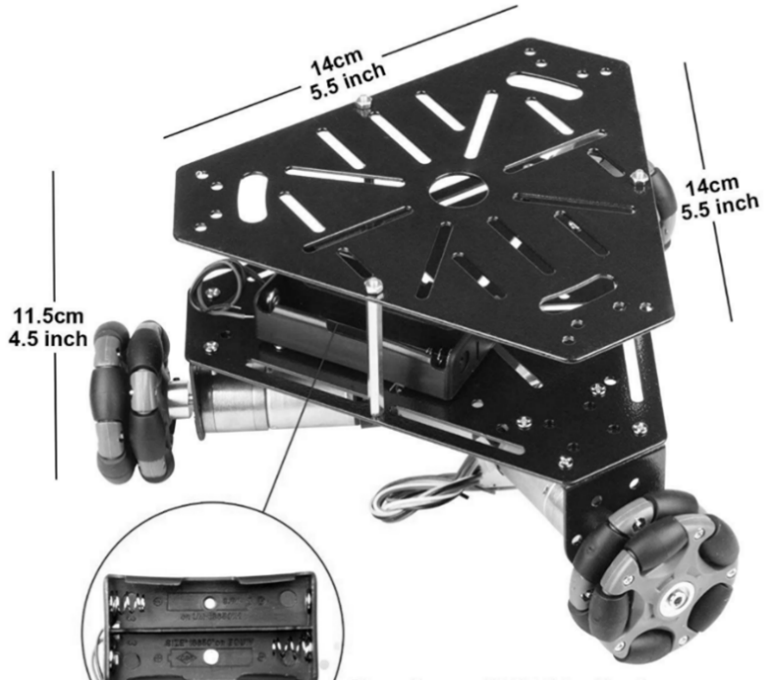


Figure 8 Example Omni-wheel base
Permission from Heneng shown in Figure 70

3.1.1b 4-Wheel Holonomic

The 4-Wheel Holonomic design is the same in principle as the design discussed in 3.1.1a 3-Wheel Holonomic. However, instead of three wheels at 60-degree angles, there are 4 wheels mounted at 45-degree angles. There is significantly more power in this design than in the three-wheel design because all four wheels are contributing to the motion at any given time. Additionally, the output speed is faster than the actual wheel rpm due to vector multiplication at the cost of torque. An example 4-Wheel holonomic kit is shown in Figure 9.

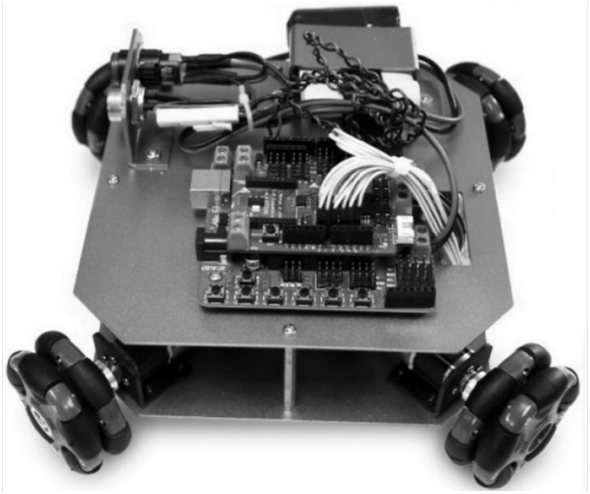


Figure 9 4-Wheel Omni Kit
Permission from RobotShop in Figure 72

3.1.1c Differential Drive

The differential drive design is a traditional approach to mobile robotic bases. This design generally involves 2 to 4 wheels mounted square to the base. Either two or four of the wheels contributed to the motion of the drive. This design is very robust and provides significant power, however it is not holonomic. This base requires turning of the entire robot to drive in directions that are not forward or backward. The wheels are not required to be Omni-directional, thus traditional wheels or treads could be utilized. In order to achieve the requirements, an additional mechanism for turning the launcher and/or intake would be required. This would ultimately achieve the same thing as the holonomic motion regarding launch angle, but it reduces the agility of the robot and ultimately the player engagement. An available differential robot kit is shown in
Figure 10.

Figure 10 Differential Drive Robot
Permission from RobotShop in Figure 72

3.1.1d Actuators

There are many available actuators with a variety of parameters that distinguish the different products. The actuators cost, RPM, voltage, current, power, control, and feedback types are the parameters that directly impact design decisions. A summary of the devices investigated in detail is shown in Table 24.

Table 24 Actuator Comparison

Actuator	Cost ($)	RPM	Voltage, Current (V), (A)	Power (W)	Control	Feedback
Heneng DC Motor	15	100	9, 1.2	10.8W	External MC	2 CPR Quadrature Encoder
Feedback 360 High Speed Continuous Rotation Servo	28	140	6, 1.2	7.2 W	PWM	2 CPR Hall Effect
High Speed Continuous Rotation Servo	17	180	7.4, 0.6	4.44W	PWM	None

3.1.1e Wheels

There are many wheels available to choose from, each with a variety of properties that affect design decisions. The wheel type, cost, size, and material are the main factors investigated for this project. A summary of the products investigated is shown in Table 25.

Table 25 Wheel Comparison

Wheel	Type	Cost	Size	Material
RobotShop Omni	Omni-Wheel	$15	60mm	Aluminum + Rubber
Lego Omni	Omni-Wheel	$7.60	58mm	Plastic
UniHobby Omni	Omni-Wheel	$15	38mm	Plastic
Micnaron Luggage Wheel	Standard	$10	60mm	Rubber

3.1.1f Frame Materials

The frame is a critical component in the base subsystem, and a huge selection of materials are available to achieve the requirements defined for the project. The parameters investigated are cost, modularity, strength, and manufacturability. The modularity property indicates how easy it is to adjust, modify, or change the design of the frame given designs of other subsystems. The strength is the sturdiness of the material. Manufacturability is how easy the material is to work with given the tools available. A summary of the investigated materials is shown in Table 26.

Table 26 Material Comparison

Material	Cost	Modularity	Strength	Manufacturability
Wood	Low	High	Medium	High
Aluminum	High	Low	High	Low
Plastic	Low	Medium	Low	Medium

3.1.2 Design

The researched designs are summarized in Table 27. Based on this information, the design chosen is the 4-wheel holonomic design. The design provides maximum usage of the power available in the motors and provides better mounting places for the launcher and intake systems. However, it is more expensive and there are no low-cost kits available.

The design chosen is a mash-up of custom parts fabricated by the team, and pre-existing components. The motor/encoder combination is to be a continuous rotation servo. The continuous rotation servo is like a DC-motor except that it has built-in open-loop position control and motor driver. This substantially reduces the complexity of the PCB required for the robot. Standard servo mounting plates are used to interface the servo with the frame. The best servo considering long-term goals is the Parallax High-Speed Continuous Rotation servo with feedback shown in Figure 11. This servo for $27 provides up to 160RPM with high torque and accurate position control. Although this is more expensive, it provides a way to close the control loop to improve base performance. The servos are mounted asymmetrically to allow for the wheel to be in the center of the hexagonal side, and to allow a channel underneath the robot to allow space for cuts and mounting of the launcher and intake. The final design is shown in Figure 13. The holonomic motion is described in Figure 14.

Table 27 Base Design Comparison

Design	Wheels	Motors	Speed	Cost	Agility	Strength	Modularity
3-Wheel Holonomic	3	3	Low	Med	Med	Low	Low
4-Wheel Holonomic	4	4	Med	High	High	Med	Med
Differential Drive	2-4	2-4	Med	Low	Low	High	High

The main frame piece for the design is a wooden plate cut on a laser cutter to quickly and accuracy cut out all the holes for the various hardware, and the cut-open sections that give space for the intake and launcher systems. It is also possible to utilize traditional tools such as a jigsaw and drill to build the design with enough tolerances.

The chosen Omni-wheels shown in Figure 12 are 60mm in diameter and are a mixture of aluminum and rubber. They are purchased from Robot-Shop for $15 each. This is the cheapest omni-wheel at this size. The size is chosen because it is just large enough to allow the ball to roll underneath given the wheel mounted directly center of the plate. Additional clearance is given by mounting it directly to the servo which is underneath the wooden frame. The drive servo directions indicated in Figure 14 show how the frame successfully achieves the holonomic requirements in each cartesian direction and both rotations.

Figure 11 Parallax Feedback 360 Degrees High Speed Servo Permission from Parallax Shown in Figure 71

Figure 12 60mm Omni wheel Permission from RobotShop in Figure 72

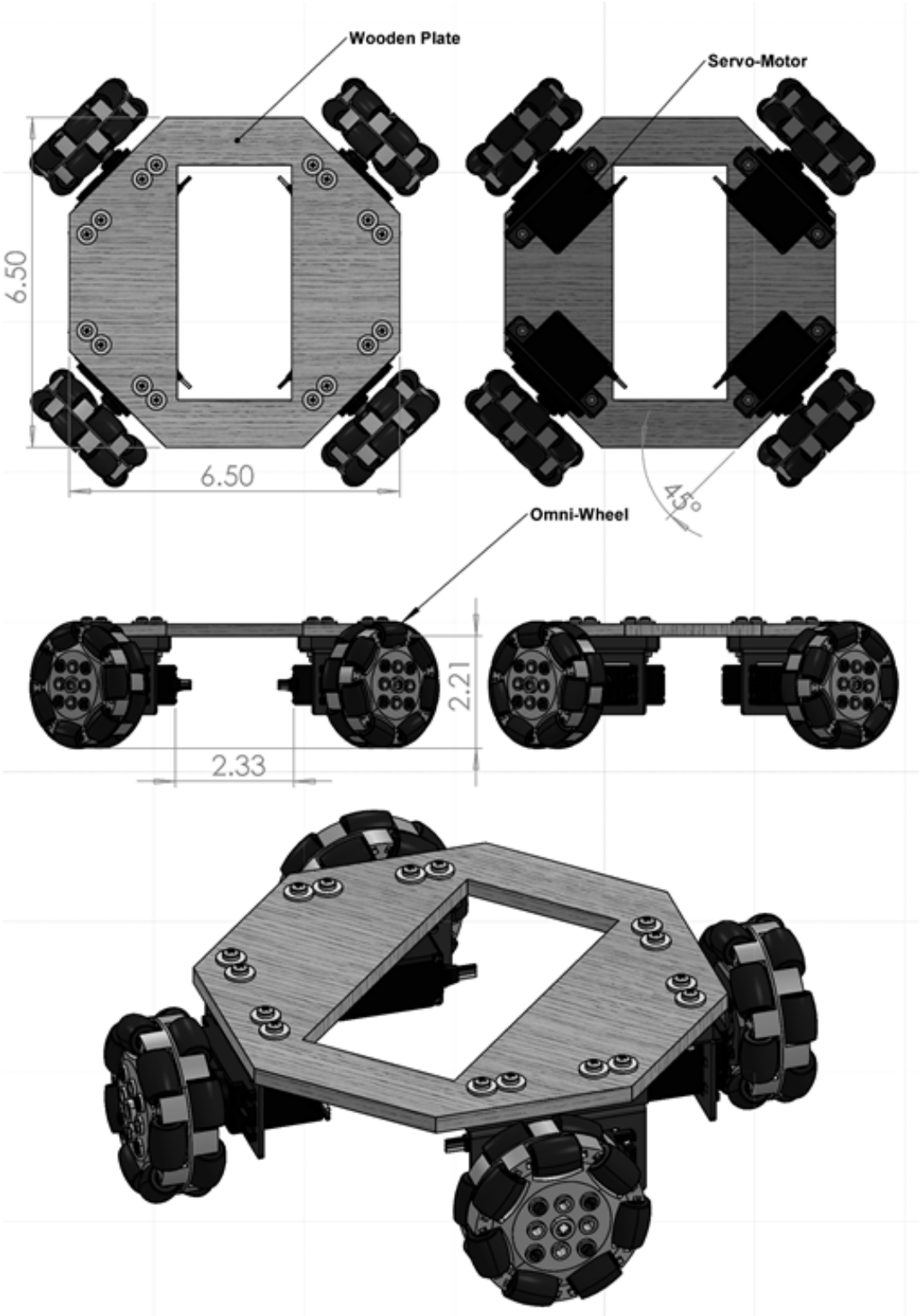


Figure 13 Robot Base Design

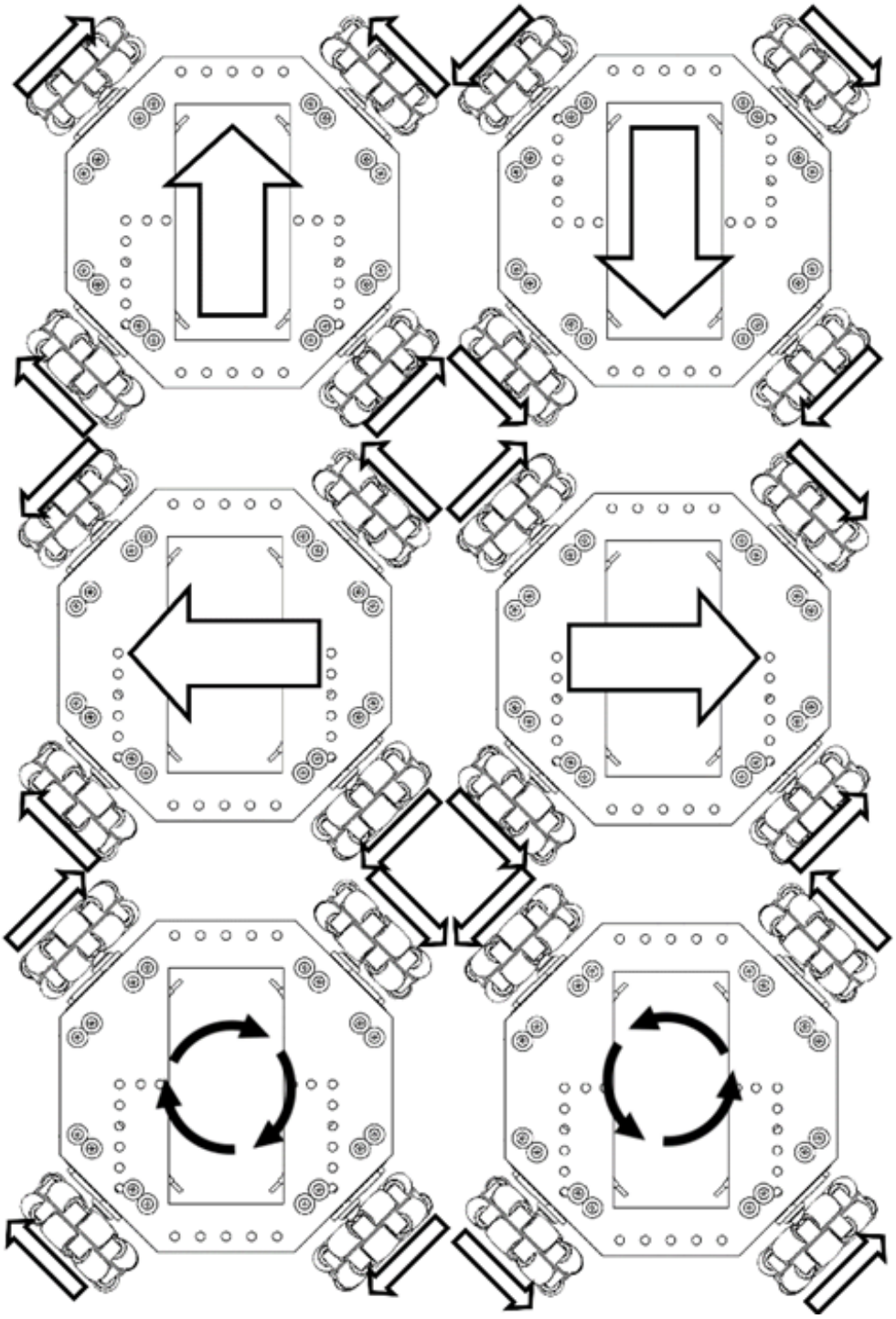


Figure 14 4-Wheel Holonomic Drive Configuration

3.1.3 Prototyping and Testing

The prototyping can be accomplished with a simple wooden plank of an appropriate dimension that is cut by a jigsaw or hacksaw and drilled appropriately.

Once tested, a more accurate, tolerance-sensitive version can be manufactured on a laser cutter. The electronics can be individually bench-tested utilizing a servo driver, power supply, and Arduino. The Servos and wheels can be purchased directly from their respective manufacturers.

The tests in Table 28 indicate the various tests required to evaluate the performance and capabilities of the Base design. Each test corresponds to a requirement or constraint. The equipment required to adequately complete the test is also determined such that the equipment can be acquired prior to manufacturing.

Table 28 Base Tests

Requirement	Test	Required Equipment
R.R.B.3	Determine if the base traverse the court without slipping	Court, rope
R.R.B.1	Determine if the base drives forward, backward, left, right, and rotates in both directions	Arduino, long USB cable, windows laptop
R.R.4	Determine if the base plate is sturdy enough to support the additional weights of the other subsystems	Weights
R.R.4	Determine if the base is heavy enough to support a moment about the expected launching axis	Weights
R.R.4	Determine if the base moves in all directions when additional load is added	Arduino, long USB cable, windows laptop,
R.R.5	Determine if the base has enough height for the ball to roll underneath on the side that the intake is mounted to	Ball
R.R.5 **R.R.I.2**	Determine if the base has low enough height to block the ball from rolling under on the sides that the intake is not mounted	Ball
R.R.4	Determine if the robot remains active after an impact	Rubber Mallet

3.2 Launcher

The launcher on the robot must be able to shoot the ball from anywhere on the court being played on. In order to accomplish this, the launching mechanism must be adjustable in some way, shape or form. This feat can be accomplished in a multitude of ways, however, to make it an achievable goal, the team narrowed the possible designs down to two ways: either lock the angle and have variable force or lock the force and adjust the angle. These paths require different solutions and

steps to be able to work properly, and the same type of mechanism may not work for both, or either of the ways chosen by the team and can influence other design choices. With a fixed angle, the force of the mechanism must be able to be easily and reliably changed. This makes the overall mechanism more complicated because more parts are required to make the launcher behave in the intended manner. A fixed force and variable angle bring up a different set of problems, such as the platform the launcher rests on will need to be more complicated instead of the launcher itself, and the equations become more complicated due to the changing height at each point of launch. Another point the team must keep in mind is that due to the steeper angle that would be required at some points on the field, the ceiling must be higher than it would be with a fixed angle. As previously mentioned, this would have an influence on the size and weight of the field, which has the potential to clash with our field requirements. The three main ways of implementing a launcher on the robot being explored are a flywheel, puncher, and catapult. These three methods were chosen because most of the ways to launch the ball reasonably will fit into one of these categories and the team can narrow it down more easily within the category before deciding which type overall to use.

3.2.1 Research

3.2.1a Flywheel

There are two main ways to implement a flywheel launching mechanism, using one or two wheels. Both offer their own specific problems that must be considered when doing calculations for the projectile coming out of the launcher. These situations are outlined in Table 29 below.

Table 29 Flywheel design problems

Flywheel problems	Outcomes
Wheel not up to full speed before shot	Shot comes out short
Ball enters wheel at different speed every shot	Shot is either short or long depending on speed and is hard to track and correct
Ball hits different part of wheel (isn't compressed as much or compressed more)	Length of shot is once again affected. Could also put a different spin on the ball
Wheels are not spinning at same speed (double flywheel specific)	Curve is put on the ball. This could also potentially change every time the ball is fired.

All these situations boil down to a flywheel just being too unpredictable at any given time. There are ways to remedy these problems, such as finding ways to finely control the speed of the ball entering the wheel, making sure the channel the ball follows into the launcher is a tight fit for the ball to disallow the ball to enter the wheel from a different angle each shot. The solutions to many of the problems

presented by a flywheel are mechanical in nature and are something that the team isn't built to implement well. Something that can be looked at positively about using a flywheel, however, is that it will allow the robot to put a more natural spin on the ball compared to the other options under consideration by the team. Since a huge part of basketball is getting spin on the ball to help make shots off the backboard, this is a rather good thing to be able to do. The flywheel design also would easily be able to fulfill our requirements of varying force, by adjusting the velocity the wheel spins at, and the ability to fix the angle that the ball is launched at easily. This could be done the other way around rather easily as well.

Comparing the two types of flywheels, one or two-wheel, both have their own advantages as well. A one-wheel flywheel will take up less space overall but won't be able to put out the same force as a two-wheel flywheel using the same motors. Also, due to having only a single motor, the one-wheel flywheel solution would require less power to operate as well as have an overall simpler design to implement. The two-wheel flywheel would allow for more finely tuned spin on the ball and more overall launching power. However, the extra motor would need extra consideration as it adds more weight to the robot in the form of extra parts needed to hold and support the extra motor and removes space needed to implement other systems on the robot. Depending on the parts chosen, this could put unnecessary strain on the base and could affect how the base is constructed. The two-wheel variant of the flywheel also has a greater chance of failing due to the extra wheel. This would require careful monitoring of more variables than the single wheel method as any sort of disharmony between the speed or angle of the two wheels essentially make the calculations done by the other systems of the project useless as the real-life motion of the ball wouldn't be able to match the projected numbers. Overall, the flywheel method's variability is both its biggest strength and weakness, in the form of being flexible enough to meet the team's launcher requirements whichever way ultimately is chosen while being unreliable in accuracy and precision needed for this task.

3.2.1b Puncher

A punching mechanism is a lot more straightforward than either a flywheel or catapult design. With a puncher there is a lot more control possible with it because the ball is always launched from the same spot and orientation every time. The first major downside of a puncher is that in order to make the force of it variable is to more hardware is required. If the team was going to make a fixed force mechanism for a shooter, the puncher would excel at that as it could be solved with a mechanism such as a skip gear. However, due to needing to meet the requirement of a variable force on the ball, an additional mechanism such as a linkage, actuator, or even another motor, would be required to release the puncher.

The puncher design currently being considered is a tension-based design powered by springs either extended or compressed with a sudden release. The spot that the puncher contacts the ball and the shape of the punch can be changed to

produce different effects on the ball. As the puncher and rail can be attached at basically any angle and won't need to move, the team can experiment easily and find the best angle to use before locking the angle in place to fulfil our requirement of having a fixed angle, variable force launcher. Due to the puncher traveling in a straight line and only acting a short impulse upon the ball, the calculations end up being projectile motion equations. The main pros and cons of the puncher are outlined below in Table 30. A huge con of the puncher design is the space required to implement it correctly and a sample design by the team is provided below in Figure 15. First, even though the slide component may look compact, it needs to be able to extend a certain amount outside of its at rest position, this size change can range from very little, like half an inch, to having to take up double the size of the initial position. Second, the extra component that would be needed in order to remove the gear from the slide to trigger the launch would have to include another motor or drive mechanism which also essentially doubles the space needed for the full system. However, if the puncher only needs to have a consistent force, the second part of the size requirement is removed, and it is only necessary to worry about the range of motion of the slide component.

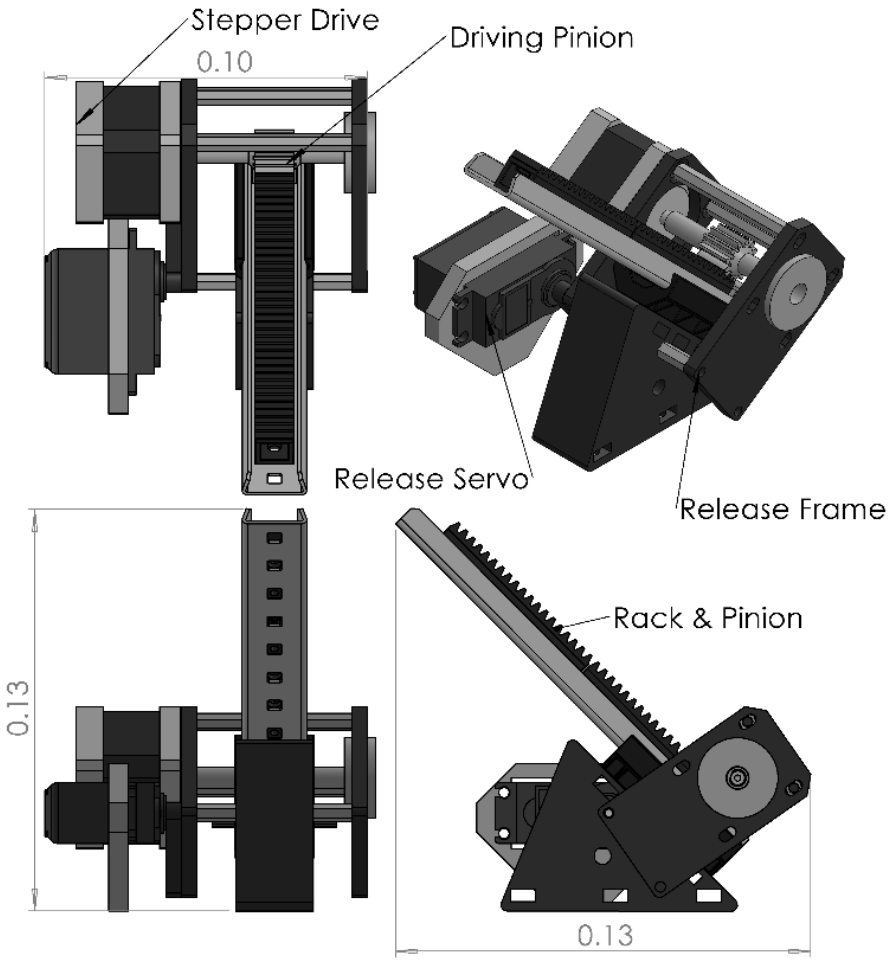


Figure 15 Launcher Design

Table 30 Pros and cons of a puncher

Pros	Cons
Consistent launch	Not easily converted to variable force
Consistent force	Spring/elastic mechanism can wear down
Angle easily changed	Large

3.2.1c Catapult

There are three main types of catapults, the ballista, the mangonel and the trebuchet. Since the construction of a trebuchet device would be unfeasible due to the complication of the design and the size constraint of our small robot, that idea was only very briefly explored. The ballista variant would be very similar in design to the puncher mechanism described above in section 3.2.2b, except for the fact that the ball would be pushed down the length of rail instead of a short, sharp contact to propel the ball. The ballista design shares a lot of the same advantages and disadvantages as the puncher except for being able to control the spin of the ball as it is launched. And in the implementation that would be used for this robot, the only difference between the ballista design and the puncher design being considered is a stopper that keeps the ball from falling into the channel left behind when the spring is drawn back.

The last type of catapult is the mangonel [15], which is what most people think of when they think of the word catapult. Using this design poses a lot of design problems. First, we would need to have a bigger and more complicated intake or put it in a place on the robot that doesn't make sense in order to load the arm of the catapult. Second, there would be little control over the angle unless the placement of the beam to act as a brake for the arm was very precise. Due to this, if the team was to try to make the robot have a variable launch angle, this design would immediately become unable to use as it would be difficult to get the correct placement dynamically on such a small-scale base. The team would also have to take special care to make sure that the arm was able to be fully drawn back, or at least drawn pack to a specific spot to be able to vary the force. The calculations for aiming the catapult and getting the correct drawback on the arm are more calculated than the relatively easier impulse and standard projectile motion formulas useable with something like the puncher.

3.2.2 Design

For all that the cons that a flywheel can potentially have, the team has decided that they are easier to mitigate than having to design an entire separate mechanism that would be required to get the correct variable force behavior that is needed for the robot. A layout of the design is provided below in Figure 16. The flywheel is

almost centered on the robot, slightly more towards the front side and sunken into the robot so that it is close to the ground allowing it to be able to function as the intake into the robot as well. The ball travels around the wheel until the correct angle for launch is reached and the ball is shot from the front of the robot.

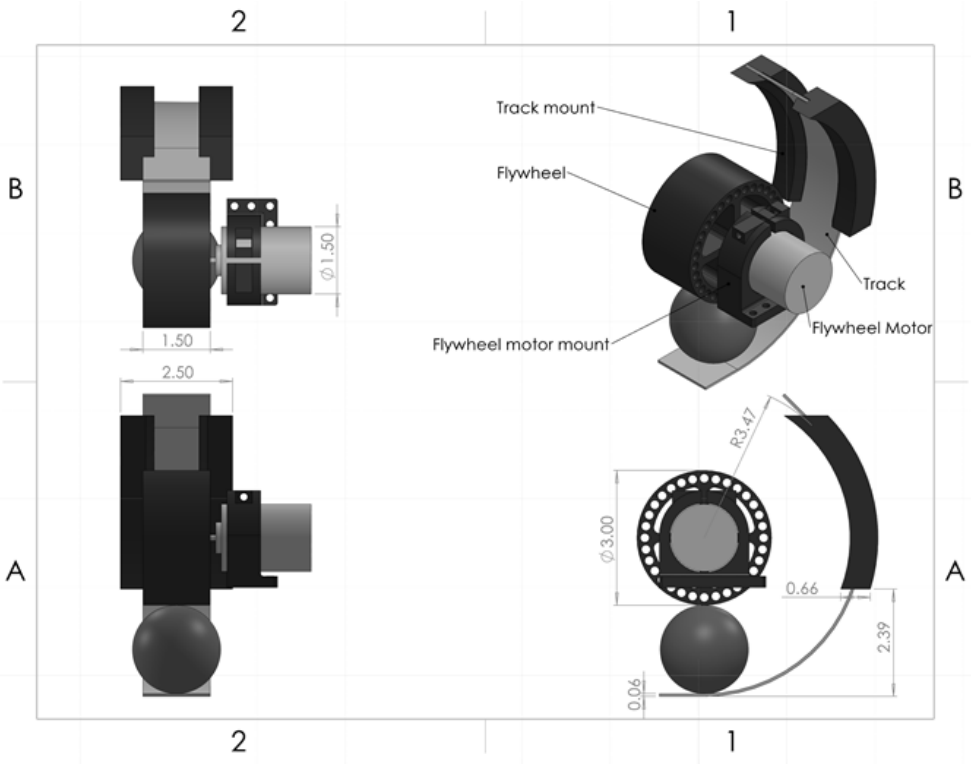


Figure 16 Launcher design drawing

3.2.3 Prototyping and Testing

The launching mechanism that is going to be used is a single flywheel device instead of a double flywheel. The team chose this after spending a large amount of time attempting to piece together a variable force spring mechanism which ended up being more complicated than what was initially thought. We also decided against the double flywheel design because we didn't want to have to worry about the calibration between the two wheels to prevent unwanted curvature. The overall tests for this subsystem of the robot are shown in Table 31. Testing for this mechanism is conducted in stages, starting with force and making sure it is consistent before trying with different angles. Although there is not varying angle capability included in the final design, it is important to test the angles in order to find the optimal one that uses less power and to make sure the path of the shot ball is contained within the arena that has been built for the robot. The angle is controlled by a piece of material attached at the end of the track that the wheel slingshots the ball around.

Table 31 Launcher Tests

Requirement	Test	Required Equipment
R.R.L.3	Test the launcher with different forces. Determine distance.	Tape measure, carbon paper
R.R.L.3	Test the launcher with different angles. Determine distance	Tape measure, carbon paper
R.R.L.3	Test the launcher for accuracy and precision at different shooting configurations	Tape measure Carbon paper
R.R.2	Test if launcher resets properly between shots	N/A
R.R.L.2	Check if the ball is hit consistently in the same area	Carbon Paper
R.R.E.4	Measure voltage and current draw across subsystem	Multimeter

3.3 Intake

The intake for the robot must be able to pick up a ball and transfer it to the launcher mechanism. There are both passive and active options to pick up a ball that the team has explored. Passive solutions require no power, or significantly less power than active solutions, however, there is a higher chance for them to not consistently pick up the ball. Options researched for our intake mechanism include a telescopic lift, a conveyor belt, or a wheel-based design. This mechanism would place the ball directly into the spot it is launched from. It's important that the ball is deposited into the launcher in the same spot each time because that has a direct impact on the accuracy and consistency of the launcher due to the puncher having to hit the same spot on the ball each time. The team has narrowed the decision down to a series of wheels, a conveyor belt, and a telescopic lift like what is seen on a forklift.

3.3.1 Research

3.3.1a Wheels

The first design being considered, as well as the first of the two active intake mechanisms is a wheel-based mechanism to pick up the ball and pass it up the intake. Wheels for the intake can be done in two ways, either on one or both sides of a channel, much like a single or double flywheel design except with a lot less power. Wheels are more useful for the intake than for the launcher because less precision is required. The design and calculations for the intake don't depend on something as small as making sure the ball comes in at the same speed every time. Since all that is required is to get the ball to the launcher, using wheels is necessary. A wheel-based intake mechanism would most likely require the most hardware out of all the designs being considered as it would require more than one

motor to implement. The wheel design the team is looking at is essentially a conveyor belt without the belt and the only major drawback besides the aspect of having to utilize more hardware is that if the wheels aren't placed in the right position the ball could get stuck between them or not move quick enough. Due to each wheel needing to be mounted individually, there is also more potential for a part to fail taking down the entire mechanism. The front of the wheels act as an active intake by spinning to physically pull in the balls, instead of just corralling the ball.

3.3.1b Conveyor Belt

The second active design being considered is a conveyor belt. There are only two versions of the conveyor belt that can be implemented for the robot. One with, and one without dividers in it. The only real distinction is that the one with tabs will have a more redundant mechanism for carrying the ball to the launcher. A conveyor belt can be implemented with a single motor potentially which makes it lightweight. The major failing point of using a conveyor design is that it must always be kept taut which requires a lot of attention and regular maintenance. If the conveyor belt isn't fully taut, the ball has the potential to just spin in place, which can be combated with plastic tabs that sweep the ball and act as a floor to prevent them from falling. Adding this to the conveyor belt doesn't come at the cost of too much hardware and extra weight typically. The two primary materials that the conveyor belt can be made from are either a smooth, continuous band or plastic links that look like tank tread. The tread design will allow the team to more easily. The conveyor belt is very similar to the wheel design in the fact that the front of the conveyor belt actively works to bring in the ball.

3.3.1c Telescopic Lift

The telescopic lift design is the only design being considered by the team that can be considered passive, as the end that contacts the ball would be like the fork on a forklift. The upside of this is that the fork part is simple to design and can be made from just about anything. It also has the perk of not being an active part that can break down and therefore must be replaced. The downside of telescopic lift is that the part that grabs the ball is passive. With a passive grabber there is a high likely hood of having to trap a ball in the corner to be able to pick the ball up. The inability to consistently pick up the ball is a huge detriment overall as it potentially leads to a huge loss of time in the game. The lift would be powered with either 1 or 2 motors attached to pulleys that would pull the different stages up. This introduces another problem in that the pulleys are another component that can potentially break if the cables that support the lift come off the pulleys or break. Out of these two problems, the rope breaking is the easiest to mitigate by simply making sure that the cable picked is the necessary strength. The lift could either go straight up or at an angle. To be able to drop the ball into the launcher at the top of the lift and to more securely hold the ball, a slight angle on the lift would be more beneficial than if it was perpendicular to the ground and base of the robot.

3.3.2 Design

The design of the intake will utilize two servos and a printed plastic piece that acts as a flipper to handle the ball. The servos have three modes or stages and are attached to the base of the robot with a 3D printed mount. The first stage is open, which allows the player to drive up to the ball without pushing it around. The second stage will have the flipper halfway down to trap the ball in front of the robot so it can still maneuver without losing the ball. It is important in this stage that the ball is not accidentally launched before the player is ready to fire the ball. To prevent this there is a small flap inside between the ball and the launcher channel that requires the launcher to be in its final stage, the fire stage. This will feed the ball into the flywheel and launch it around the track.

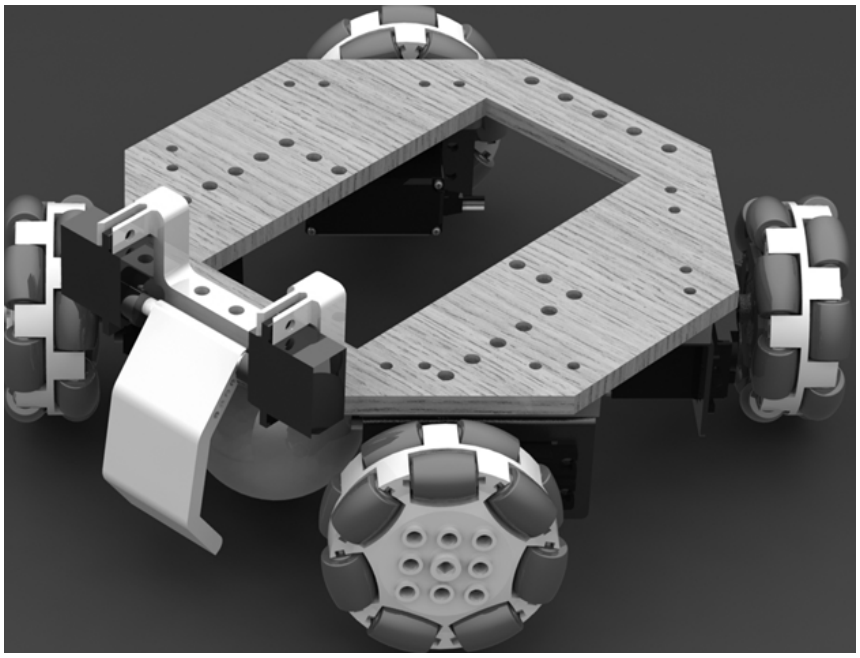


Figure 17 Intake Design

3.3.3 Prototyping and Testing

The intake is prototyped and tested the same way that the launcher is; first using premade parts and then getting them manufactured. As for actually carrying out the tests, until the intake is able to be mounted to the base of the robot, it will have to be hand moved to cover the tests that require the intake to be moving. The team will also be looking at the speed and consistency of the intake mechanism to determine what must be tweaked in order to make it better overall. Table 32 summarizes the tests for the intake subcomponent as well as what requirement they pertain to.

Table 32 Intake Tests

Requirement	Test	Required Equipment
R.R.I.2	Test if the intake can pick up a ball from different angles	Ball
R.R.I.2	Test the intake moving and pick up a stationary ball	Ball
R.R.I.2	Test the intake stationary and pick up a moving ball	Ball
R.R.I.2	Test the intake with both intake and ball moving	Ball
R.R.E.4	Measure voltage and current draw across subsystem	Multimeter

3.4 Actuator Control Array

The actuator array block exists primarily to interface the various actuator components of the Launcher, Base, and Intake systems to the electrical systems of the robot. This includes routing the signal parameters from the microcontroller to the motor controller, and routing power and ground to each device. The motor drivers for each of the drive motors exist within the servo itself, thus this component simply routes power and signal appropriately – There are no additional integrated circuits required. The intake and launcher systems are integrated into the same device, thus only a signal motor controller and servo controller port are required. The launcher motor is a DC brushless motor that requires an electronic speed controller to control. Thus, that device is investigated fully below. Additionally, components to simplify the control loop or servo control generation are also investigated.

3.4.1 Research

3.4.1a PWM Generators

PWM generators are evaluated to reduce the computational and output strain on the microprocessor. These devices can take in protocol base input to set/latch several PWM channels. These PWM channels automatically generate signals at the desired duty cycles.

One device under consideration is a PCA9685 which is a I^2C to PWM IC. It can drive up to 16 PWM channels at once with 12-bit resolution at a fixed frequency. This can be used in conjunction with the chosen motor controller to reduce load on the chosen microcontroller. This also simplifies the motor control process. Additionally, between this device and a voltage regulator, several servos can be controlled without significant overhead.

Another device being considered is a MAX31790 which is marketed as a 6-channel PWM fan controller. The duty cycles of the 6 channels are determined by the I2C. In addition to the outputs, the device also has inputs for tachometers to monitor the rpm of the fan. This device would work in this application based on the desired control of the servos. The drive servos are velocity controlled with positional feedback in the form of a PWM duty cycle. This device can take in the feedback and automatically adjust PWM duty cycle output to close the feedback loop on velocity. The other servos would operate in their typical positional mode without feedback.

3.4.1c Motor Controller

A speed controller is a device that generates pulses to accurately control the speed and output of a DC motor. This is required to control the primary launcher wheel motor at a higher RPM than the other actuators need to operate at. There is a wide variety of speed controllers available with many different qualities, features, and prices. A summary of the devices primarily investigated is shown in Table 33 Motor Controller research summary.

Table 33 Motor Controller research summary

Driver IC	L293D	L298	DRV8871
Supply Voltage (V)	**5 - 36**	**6.5 - 45**	**6.5 - 45**
Max Output Current (A)	**0.6**	**4**	**3.6**
Number of Channels	**2**	**2**	**1**
Footprint (mm^2)	**12.6 x 7.4**	**15.8 x 10.9**	**4.90 × 6.00**
Cost($)	**2.95**	**4.86**	**2.13**

3.4.2 Design

The elimination of motor control circuitry from the control array significantly reduces the complexity of the design. The servos integrate the motor control directly within their servo package which allows for just a Signal, Voltage, and

Ground line for each servo. In addition to these lines, the servos for the drive have positional feedback for verifiable velocity and position control. Figure 18 shows the control signal diagram indicating the signal flow from the microcontroller to the motors through the different connections. Notably, the microcontroller communicates with the PCA9685 PWM generator through I2C protocol. The PWM generator then outputs all of the required signals to drive the servos, and motor controller for the launcher motor. Figure 19 shows the power flow diagram between the battery and the actuator devices. Each of the power and signal lines can be broken out to a single array for ease of connection and expansion.

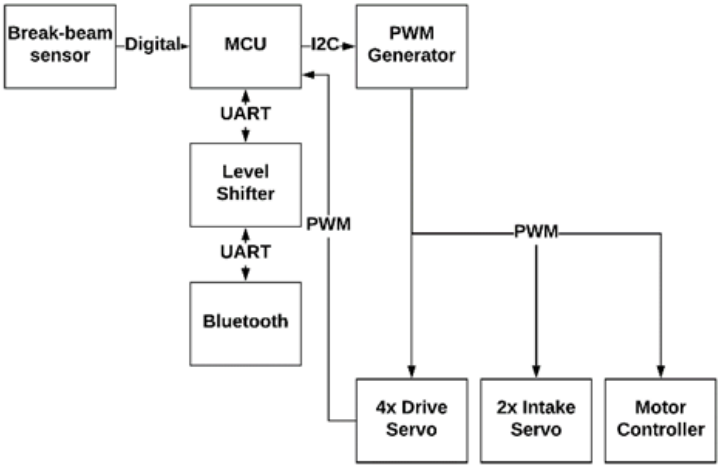


Figure 18 Control signal block diagram

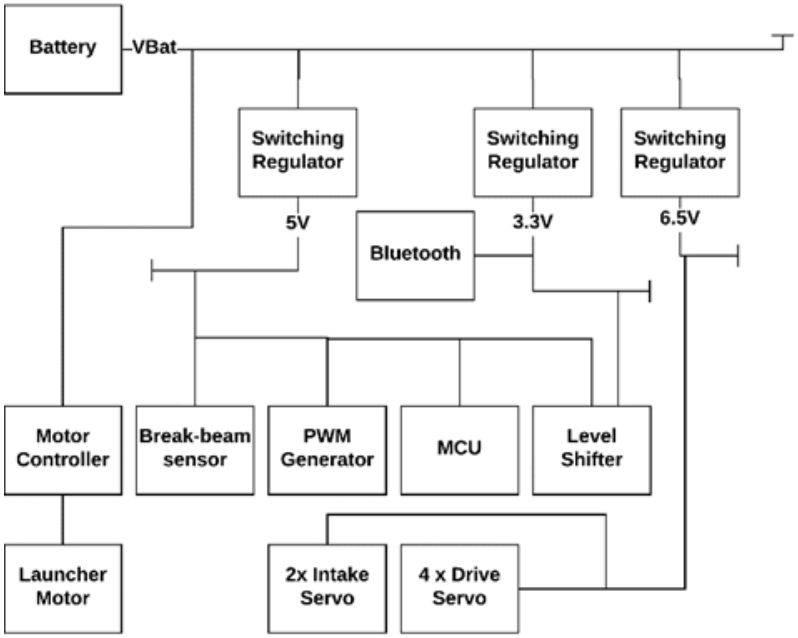


Figure 19 Power block diagram

3.4.3 Prototyping and Testing

The chosen devices can be evaluated utilizing evaluation boards available from Amazon. Each board can be purchased and tested individually to verify the design prior to the final PCB construction. To make sure the board and motor controllers are working correctly, a series of tests provided below in Table 34 Table of Motor Controller Tests must be conducted and passed.

Table 34 Table of Motor Controller Tests

Requirement	Test	Required Equipment
R.R.E.3	Drive each actuator utilizing the chosen motor controller and Arduino	Arduino, Power Supply, breadboard, actuators
R.R.E.3	Drive each actuator utilizing the chosen motor controller, PWM generator, and Arduino	Arduino, Power Supply, breadboard, actuators
R.R.E.3	Drive each actuator utilizing the chosen motor controller, PWM generator, I/O generator, and Arduino	Arduino, Power Supply, breadboard, actuators
R.R.E.3	Drive each actuator simultaneously using each evaluation device (Motor controller, PWM generator, I/O generator, and Arduino)	Arduino, Power Supply, breadboard, actuators
R.R.E.4	Determine the final load of each actuator at full speed simultaneously	Arduino, Power supply, breadboard, actuators, multimeter
R.R.E.1	Determine the stall torque of each actuator, and the current at which it stalls	Arduino, Power supply, breadboard, actuators, multimeter
R.R.E.1	Determine the actual range of the servo motor	Arduino, Power supply, breadboard, servo, protractor

3.5 Microcontroller

The robot requires an onboard processor to perform the necessary calculations for locomotion and making shots. However, it is still a slave device to the arena and therefore, a microcontroller and not a microprocessor is used. A microprocessor can carry calculations at nanosecond speeds whereas a microcontroller, well, in microseconds. To provide a rich user experience millisecond latency is enough and therefore, due to cost requirements and constraints a microcontroller is used to control the robot.

The controller is needed to control the dedicated tasks on the robot. These tasks require real time executions. The controller receives a packet from the Arena in a timely fashion and decodes them. The format of this packet is designed by the team. In excess to the overhead that comes with Bluetooth communication, the packet contains data that has substantial information for the robot to perform its activities. The update frequency of Bluetooth communication has to be 30Hz to meet the design requirement as it allows for a rich user experience. This high update rate will allow for error detection and correction most of which is inherently designed in Bluetooth's protocol allowing little to no lag on user end.

The packet received by the robot will have information on motors, velocities, configuration settings etc. Each motor is given an ID helping the microcontroller and the engineers in easily distinguishing them and applying varying velocities based on information contained in the packet. These motor values are converted to discrete values by the microcontroller and then fed to motor controller ICs using Pulse Width Modulation (PWM). The microcontroller also sends sensor data back to the Arena for feedback and makes the arena aware of the robot's location. The microcontroller also performs PD calculations for the motors to ensure accurate closed-loop control for the systems that require it. There is a myriad of options available in the market to use as Robot's "brain", however, due to the listed requirements and constraints only certain of them are feasible.

3.5.1 Research

Based on the market research there are many microcontrollers available to perform the job. The requirements however constrain the team from choosing just any microcontroller. As mentioned earlier, the microcontroller needs to control 6 motors and have the capability of getting encoder data for monitoring the velocities. When the arena sends the robot a Bluetooth packet, the onboard microcontroller parses the packet and breaks it into its respective components such as motor ID, velocity for that motor ID, Intake action commands, Launch action commands, no motion command and the like.

Due to the aforementioned tasks, the microcontroller is required to have Bluetooth compatibility for communication. There are several workarounds for this. Solution one is to get a controller with a built in Bluetooth module and have a Bluetooth stack available for programming it to send and receive data. However, microcontroller boards with built in Bluetooth tend to be expensive. Another option is to buy a simple microcontroller and have a separate Bluetooth module and use it via Universal Asynchronous Receiver Transmitter, also known as the UART. The UART is preferred method for exchanging data between the microcontroller and the Bluetooth mainly because the data format and transmission speeds are configurable.

Additionally, the microcontroller also needs to be able to send motor commands using Pulse Width Modulation and receive encoder commands via interrupts. There are boards available in the market which allow configuring every single pin as PWM and interrupt however, they tend to be expensive and constrain us in our spending limit. Therefore, the microcontroller needs to have a minimum of sending 6 PWM signal and have 8 interrupts for encoders. There are also multiple ways to work with this. First option is to buy a board with all features on board whereas another option is to buy modules and either find or create custom libraries to interface with them. Keeping such specifications in mind a list of required features was created and appropriate microcontroller technologies were studied. A summary of the findings can be seen in Table 35.

Table 35 Compare and Contrast of Different Microcontroller Technologies

Processor	ATmega328P	ATmega2560	MSP430G2553
Cost ($)	2.08	11.99	2.14
I2C	2	2	2
UART (Rx, Tx)	1	4	1
SPI	1	1	2
Interrupts	2	6	24
Digital IO	14	54	24
Analog IO	6	16	n/a
PWM	6	15	24
TTL Voltage(v)	5	5	5
Input Voltage (V)	3.3 – 5.5	3.3 – 5.5	3.3 – 5.5
CPU Speed (MHz)	16	16	25
EEPROM (KB)	1	4	n/a
SRAM (B)	2k	8k	512
Flash (KB)	32	256	16
USB	Regular	Regular	Regular

3.5.2 Design

Based on the research conducted, the microcontroller that seems most feasible for the robot is ATmega328P. ATmega328P is popularly used in Arduino development boards. There are multiple open source libraries and forums available on the internet that act as a valuable asset in the development of this subsystem. For the PCB design, a surface mount chip is used, and the pinouts is matched with the Arduino Uno board. Using an ISP connector, the Arduino bootloader is flashed on the ATMega328P microcontroller which gives this chip the same capabilities as an Arduino development board allowing development using various ICs such as the PWM Controller and the Bluetooth chip.

The ATmega328P controls the PCA9685 PWM Controller chip using I2C protocol. The microcontroller sends a 16-bit code at each loop which is generated by parsing the incoming Bluetooth packet from the Arena. The servos send a feedback signal to the microcontroller via PWM and GPIO pins. An encoder for the flywheel's brushless motor also sends data back as a PWM signal which interrupts ATmega328P's loop to increment or decrement speed. The Bluetooth chip uses UART protocol to communicate with the ATmega328P chip. The Transmission (Tx) line is pulled high to avoid noise that could be generated on an open trace. The flywheel that is used for intake and launching the ball spins with the help of a brushless motor. For simplicity of programming, the microcontroller is laid out such that the ATmega328P's ports and pins align with Arduino's digital and analog pins. Additionally, the ISP connector uses SPI protocol to upload the bootloader on to the ATmega328P microcontroller due to its reliability and high data transfer capabilities. An LED is attached to the microcontroller digital pin and upon the code startup, the pin is blinked to indicate the microcontroller's status. A detailed schematic is shown in the PCB section 3.9.2 Design.

3.5.3 Prototyping and Testing

The microcontroller is tested using multiple tools. The primary tool is the Arduino Development board as it allows ease of prototyping. Having print statements after a certain point in code execution allows the programmer to detect and eliminate unnecessary bugs. The UART, SPI, and I2C signals are tested using an oscilloscope. The Oscilloscope can be configured to decode the communication signals and display them as hexadecimal that can surely be used for debugging. The pin voltages and currents are checked using digital multimeter whereas the solder joints for the MCU on the PCB are checked using a magnifying glass. This is used to detect errors which would take the Serial monitor a delayed time to respond to in time critical situation. The requirements completed by these procedures is listed in Table 36. It also lists the equipment used to perform the tests successfully.

Table 36 Controller tests

Requirement	Test	Required Equipment
R.R.S.1	Determine that microcontroller uses Bluetooth Low Energy as a serial device	Oscilloscope, Serial Monitor
R.R.S.2	Determine if the encoder interrupts increment and/or decrement	Oscilloscope, Serial Monitor
R.R.E.5	Determine that the microcontroller implements communication protocols	Oscilloscope
R.R.S.2	Determine that the encoder channels properly interrupt the microcontroller	Oscilloscope

3.6 Communication

The communication subsystem allows the robot to receive commands from the arena. To accomplish this, the robot must have a communication system on board and receive data over a wireless link. The communication subsystem needs to have a data update frequency of 30Hz at the minimum. Failure to do so can cause latency in robot's motion. This latency hinders the robot from receiving data in a timely manner and constraints it from shooting successfully 75% of the time as per out requirements.

Another reason why the communication system needs to be wireless is that the robot is moving in the field. Having cables or wires can restrict the robot and introduce noise in the communication signals. Using a differential pair is a possible solution however, the robot could damage the cables by running over them. Therefore, wireless communication is a priority to prevent any potential damages to the entire game. However, with wireless communication comes with a possibility of potential packet loss and data corruption. This can inherently introduce the similar problem of latency due to which the communication system has to have error detection and correction schemes implemented. This achievable using TCP/UPD or Bluetooth. The received packet from the Arena is designed by the team. It has information regarding the motors to be operated (i.e. motor ID), the velocity for that motor, information regarding intake, launch, and other necessary configurations.

The robot is a slave device to the arena that will receive data over the radio to perform its actions. The implemented communications protocol will also allow the robot to send its sensor data back to the arena for monitoring and debugging purposes. This data is shown by the Arena on a screen to give users more information regarding their robot. These stats could include current motor velocities, battery status, communication link status etc.

3.6.1 Research

3.6.1a Bluetooth

Another technology which is under consideration is Bluetooth v4.2. Bluetooth is low power communication protocol which allows the entire system to be portable and cost effective. Connecting two Bluetooth devices together is a multi-step process. It requires an inquiry, paging, and connection. These steps are usually implemented in Bluetooth's firmware and API and are readily available on the internet. Additionally, there are various Bluetooth profiles and for two Bluetooth to exchange data the Bluetooth profile has to be the same. The Bluetooth also has to have compatibility with the microcontroller as creating a custom firmware could take more time than at hand for the project [16].

For the robot to receive data, a Serial Port Profile, or SPP, is used. Likewise, the arena shall also have a Bluetooth stack that supports similar profile to correctly send data. There are various Bluetooth modules available that are compatible with Arduino. Based on their characteristic analysis, the best Bluetooth module is chosen and tested with Arduino. Then, for the PCB the chip native to that board is used with its bootloader that will allow the team to program the chip in a similar fashion.

Different Bluetooth modules compatible with Arduino and ATmega328P can be seen in Table 37 [17]. The Bluetooth used is version 4.2 and is low energy allowing data transfer without wasting power. They use Serial Port Profile for communication and thus, the exchange of data happens via UART protocol. The data link layer for the modules uses standard Bluetooth profile and therefore, each packet sent can go up to 251 bytes where 14 bytes are overhead due to each layer in the Bluetooth stack.

The data rate is dependent upon the version of Bluetooth in use. The Bluetooth has data exchange rate of 3 Mbps and compromises distance for high throughput. It can go up to 10m unlike the previous versions that can reach 100m. This is ideal for the project as the Arena dimensions are far less than that [18]. The firmware used is proprietary firmware provided by Microchip and it supports connection via SPP over rfcomm.

Table 37: Bluetooth Module Comparison

Module Name	IC	Range (m)
HC-05	TI CC2451	9
BLE Link Bee	TI CC2540	60
BLE Mini	TI CC2540	50
BlueSMiRF Silver	RN-42	10

3.6.1b Wi-Fi Direct

Wi-Fi direct is a wireless communication and data transfer protocol that is used for browsing, file transfer, or any other communication. The difference between Wi-Fi

and Wi-Fi direct lie in the fact that Wi-Fi Direct opts for device to device communication. Some of the advantages of Wi-Fi Direct include the ability to connect to any device. For Wi-Fi direct only one of the devices has to have the technology to setup the connection. Majority of the communication uses TCP/IP and UDP to exchange data. It acts Bluetooth abilities to Wi-Fi at a higher bandwidth which makes it an attractive choice. One does not need to be connected to the internet to exchange data. However, there are limited number of technologies available that have libraries and firmware developed enough to work with Arduino, specifically ATmega328p [19].

The main device used to carry out Wi-Fi communication on Arduino is the ESP8266. This device can be setup as an access point that can connect to Arena's Wi-Fi however, this would make the Robot a master device unlike it's intended use as a slave device. A solution would be to find a module for Arena communication system that can work as access point that will allow multiple robots to connect via Wi-Fi [20]. The data rate provided by Wi-Fi direct is larger than Bluetooth, but it also uses more power. Both of them work at 3.3V but Wi-Fi modules can use currents up to 170mA of current whereas Bluetooth LE uses 50mA of current at maximum. These are important factors that will affect the choice of module used in the design of Robot communication which is discussed in section 3.6.2 Design.

3.6.2 Design

Based on the market research conducted on Wi-Fi Direct and Bluetooth technologies and weighing their advantages and disadvantages Bluetooth Low Energy is used as the primary mode of communication platform for exchanging data and commands between the Arena and the Robot. The version of Bluetooth LE that is used is v2.0 which provides communication range of up to 10 meters and uses 10dB power as it is a class 2 system. The profile of Bluetooth used is Serial Port Protocol, or SPP, and it connects to Arena's Intel Bluetooth module.

In a wired interface, RS-232 is used for UART communication, however, instead of RS-232 Bluetooth uses rfcomm protocol to exchange data serially. This can be advantageous as Arduino's bootloader, that resides on ATmega328P, has built in encapsulation in "SoftwareSerial" library that uses a typical UART to send data to the Bluetooth driver which translates it into rfcomm and send it wirelessly to its connected master. Therefore, using Arduino's firmware makes programming the Bluetooth module easy.

At maximum, the packet length cannot be more than 251 bytes where 14 bytes are used as overhead that contains information of the Bluetooth layer in the Bluetooth stack. The packet that is sent by the robot to the arena includes information on the status of the robot and battery information. The status of the robot includes moving, stationary, and shooting. The battery information data includes the LiPo cell voltage of individual cell and the battery pack. The structure of the packet can be

seen in Figure 20 [21]. Robot only sends 4 bytes of data to the Arena but receives about 16 bytes of data which is discussed in section 4.11.2 Design.

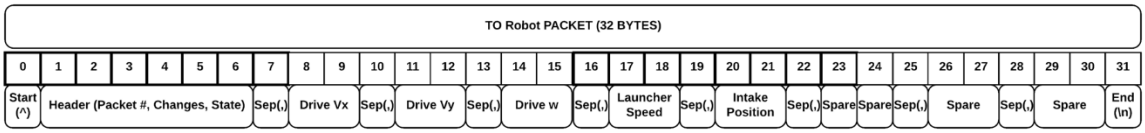


Figure 20: Bluetooth Packet sent by the robot

3.6.3 Prototyping and Testing

The Bluetooth connection is tested using Serial monitor. Initially, the robot sends a packet to the serial monitor as hexadecimal and the same packet is sent to the arena. The hexadecimal values are checked for their validity using serial monitor on the arena end as well. A time stamp is added to the test packet to determine the transmission time which allows the designer to determine the length of the packet to reduce latency in robot's operation if needed. The required tests for the prototype are listed in Table 38.

Table 38 Communication tests

Requirement	Test	Required Equipment
R.R.S.1	Packet is successfully generated by the master/slave	Serial Monitor
R.R.S.1	Packet is successfully received from the master	Serial Monitor, Oscilloscope
R.R.S.1	Packet is successful transmitted to the master	Serial Monitor, Oscilloscope
R.R.E.4	The system goes into sleep mode when no communication is occurring to save energy and system resources	Serial Monitor, Oscilloscope, Multimeter

3.7 Battery

The battery is the main source of power for the robot. The kind of battery to be used depends on the application and power requirements of the system. As mentioned earlier, the robot will have an onboard computer, up to 6 motors, sensors including motor drivers, analog to digital converters, DC to DC converters, and communication systems such as Bluetooth or Wi-fi Direct. Therefore, the battery needs to be strong enough to power it all.

For the motors, the robot has intake and launch mechanism that is implemented quite frequently. During this action, adding a load to the system will increase the

current draw from the power supply to the motors. Therefore, the battery needs to not only fit the voltage requirement but also the overall current requirement of the robot system. The battery supplies at least 12 to 9 volts and 8-10 amps to the system to overcompensate in cases of indeterministic power requirements. This is stepped down to a usable voltage for the microcontroller and its peripherals using a DC-DC converter and/or voltage divider with a buffer.

The battery technology is rechargeable mainly because it reduces the overall cost of the system. It allows reusability of the components and keeps the costs at minimum consequently meeting the project requirements and constraints. The battery has to have a safety rating that meets OSHA standards. The battery needs a voltage detection circuit to determine when it is going under its minimum voltage as for instance, LiPo batteries can catch fire when electrically over drained or mechanically damaged harming the user or the environment or both.

3.7.1 Research

3.7.1a Lithium Polymer

Lithium Polymer, or LiPo, batteries are quite popular due to their light weight and higher energy rate. A single cell can hold up to 4.2V when fully charged and they are sold as a pack of multiple cells such as 2S, 3S, 4S, 5S and even 6S or more. The S essentially signifies that they are arranged in series therefore, a pack of LiPo can provide voltage up to 12.6V in a 3S (3 * 4.2 V). This property makes them an attractive choice since different combinations can be used at an affordable market rate. They also have a low discharge rate which allows them to last longer. Therefore, depending on the power consumption by the robot, a LiPo can easily power the robot system for at least 30 minutes or more. A detailed calculation of this is done in section 3.7.2 Design based on which the desired battery is chosen. Another advantage of them is that unlike Lithium Cadmium batteries, LiPo's do not require to be fully discharged before being charged again. They can also be used in parallel to increase the current source to the system. LiPo batteries are also environment friendly unlike Cadmium, Lead or Mercury batteries which is also an important design decision for longevity of the system. [22]

LiPo batteries are rated with respect to their current and capacity rating. Therefore, a 2200mAh LiPo battery at 25C can provide 55 Amps of current at 11.1V for 1 hour, or 5 amps of current for 11 hours at the same voltage. This makes LiPo batteries an attractive choice as the launcher might use variable force to throw the ball which in turn would change the load on the motors. In addition to purchasing the battery, a proper battery charger and monitor is required as LiPo batteries come with inherit risk of fire and cannot be over or under charged due to their chemical composition. Additionally, they are quite expensive and their price increases with their capacity rating and number of cells. Therefore, an important design decision is to choose whether two 3S LiPos at 2250 mAh or one 3S LiPo

at 5500 mAh capacity as this causes a dilemma choosing between cost and weight and one has to be sacrificed for the other.

3.7.1b Nickle Cadmium

Nickle Cadmium is one of the oldest battery technologies that were revolutionary upon their arrival. They made low powered portable systems a reality however, lost their market share to Lithium batteries.

Some of the positive characteristics of NiCad include low internal resistance. This allows the energy to easily travel from battery to the system and therefore, is an important trait in choosing the battery technology. Modern digital systems require high current spikes from time to in operation unlike analog loads that work easily on steady current. Therefore, a lower internal resistance acts as an important factor in determining the battery to be used in building the robot system. NiCad batteries can be easily stored in charged or discharged state without harm unlike LiPo batteries that need to be at a certain voltage before being shelved for prolonged period of time. They are available in a large variety of sizes and capacities [23].

Some of the negative characteristics of NiCad batteries include their susceptibility to memory effect [23]. This effect causes the battery to remember its previous discharge state and hinders its next recharge cycle from reaching a full potential. This is usually prevented by either discharging the battery completely before recharging it or buying a charger with capabilities to carry out such operations. This can increase the cost of building the robot as such charges are expensive. Like LiPo batteries, NiCad are prone to damage by overcharging.

3.7.1c Lead Acid

Lead Acid batteries are an industry standard that are featured in robots, cards, industrial machinery, power supplied and much more. They are cheap and reliable which make them an attractive choice for a financial standpoint. However, one of their major limitations include their weight. They are typically used in situations where weight is not much of a problem or concern.

One of the major pros of Lead Acid batteries include its reliability. They have been in development for over a century and are scaled enough to be available at a cheaper price compared to LiPo or NiCad batteries. They are tolerant to abuse and overcharging and do not explode in strenuous environments unlike LiPo batteries. They have an indefinite shelf like which can a plus to the robot when kept dormant for prolonged periods and can deliver high currents required to run the flywheel for intake and launch and servos for locomotion.

However, their weight is a serious disadvantage. Due to their high reliability, they tend to come in bulkier packaging which will add on to robot’s overall weight and

put pressure on the electronics to function with ease. They also do not charge fast unlike LiPo and NiCad battery technologies which can deteriorate user experience exponentially. Finally, they overheat easily and can cause disruptions in sensor readings and wear the robot hardware [24].

3.7.2 Design

Based on the research conducted in section 3.7.1 Research a system power analysis was conducted to specify what battery met the desired requirements and specifications. The results can be seen in Table 39 that shows how much power each system will need to operate under worst case scenarios and the overall power robot will use to operate. Conclusively, LiPo battery seems like the optimal solution to driving the robot due to multiple reasons.

Table 39 Power Calculations of Robot's Subsystem and Components

Subsystem	Part Name/Number	Unit(s)	Voltage (V)	Current (A)	Power (W)
Bluetooth	CC2541	1	3.3	0.02	0.066
Microcontroller	ATmega328P	1	5	0.2	1
Encoder	TLE4946-2K	1	5	0.05	0.25
PWM Controller	PCA9685	1	5	0.04	0.2
Servos	Parallax #900-00360	5	6.8	1.2	40.8
Motor	DRV8871	1	12	2	24
Total Power					66.32

A LiPo battery can charge quickly and discharges at a longer rate. This allows the robot to run for a prolonged period of time. The specification of the battery that will run this robot need to be at least a 3S LiPo that can provide anywhere from 5000mAh to 6000mAh charge rate capacity. However, a cheaper solution would be to use two 3S LiPo batteries in parallel with 2250mAh capacity each, but it will increase the robot weight and occupy more space than a single LiPo battery. The specified battery can run the robot for approximately one hour on a full charge and 40 minutes on the minimum safest cell voltage (i.e. 3.7V each). Therefore, the battery should be able to easily support the robot and its activities for more than one hour. A test is conducted upon purchase to determine the actual time the battery can run the robot for.

3.7.3 Prototyping and Testing

The battery can be purchased and tested with the materials available in the Senior Design lab. No tests can be done prior to component purchase except making sure that the calculations in Table 39 in section 3.7.2 Design are correct. Table 40 lists

the requirements and constraints that need to be fulfilled by testing this component using the mentioned equipment.

Table 40 Battery tests

Requirement	Test	Required Equipment
C.R.3	Determine that the battery is not undercharged	Portable BMS unit, Multimeter
C.R.3	Determine that the battery is not over charged	Portable BMS unit, Multimeter
C.R.3	Determine that the battery provides the necessary voltage and current to the system	Multimeter, electronic Load
R.R.26	Determine expected runtime of the robot	electronic Load

3.8 DC-DC Converter

The battery provides a high voltage and high current supply to the entire system. This can be harmful for certain integrated circuits and sensor technologies. Most sensors work at a standard 5V transistor-transistor logic, or TTL voltage. However, it is not uncommon to come across technologies that run on 3.3V. The reason behind such vast changes in logic levels is inherent to manufacturers and power consumption requirements of the system. Lower voltage levels and current draw contribute to the longevity of systems. However, it could come at a cost of high performance, cost and rich user experience.

The DC-DC converter has to be 9V to 12V tolerant and therefore, a switching regulator is needed to maintain high power efficiency as the voltage is stepped down by this system. The power supplied to the robot with a LiPo battery is more than what a microcontroller can safely handle. The DC-DC converter takes in raw battery voltage and current and converts it into an acceptable power level for the system components. The microcontroller used for this project works on 5V Transistor-Transistor Logic and therefore, a 5V converter is required to power it on. The pins of the microcontroller can supply a maximum current of 20 mA and with a maximum of 20 pins a total of 400 mA can be drawn from Arduino pins at the same time which, however, is an overestimate as a phenomenon like this highly unlikely as per the design.

A 6V DC-DC converter is used to power the continual rotation servos that help the robot in its movement. The servos use 15mA of current when idle, about 150mA or current when rotating with no load, and 1.2A of current when stalled. Therefore, the converter needs to supply at least 4.8 amps of current in worst case scenario for all driving servos. This is an important requirement for the robot to move.

An additional 5V DC-DC converter is used for powering the microcontroller, encoder, and PWM Controller. These are low powered systems that use 200mA, 50mA, and 40mA of current, respectively. Therefore, a linear regulator that can support 1 A of current should suffice. The converter will take 9 to 12 V of battery input and will try to regulate the output voltage to a steady 5V with an error ±0.1V. An alternate solution would be using a voltage divider from a buffer that takes 6V regulated output as input. However, this causes issues such as failure in case the 6V switching regulator fails. Having separate voltage conversions will allow connecting a GPIO line from the microcontroller to the 6V DC-DC output that can interrupt the processor in case the line goes low. This can help in debugging the robot when it stops moving without the need of a multimeter. Another problem with the voltage divider with a buffer is that the output voltage is not regulated and therefore, it can be anywhere from 4.5V to 5.5V which is a huge change and loss of power.

A 3.3V DC-DC converter is used to power the Bluetooth Communication System. The Bluetooth communication system uses 50mA of current at maximum and hence it is low energy. Therefore, the converter takes 5V input from linear regulator and steps it down for the Bluetooth to use. To create these converters, an online tool named TI Webench was used which is discussed in section 3.8.1 Research.

3.8.1 Research

3.8.1a TI Webench

Texas Instrument has an online power tool that allows designers to specify input voltage range and the desired output voltage and current. Based on this information, the tool generates recommended schematics, their PCB layouts, a Bill of Material, or BOM, and CAD files for popular schematic and PCB design software such as Eagle. This tool is used to determine the details of DC-DC Converters that were mentioned in the description of Section 3.8 DC-DC Converter The tools also makes sure that the efficiency of the suggested schematics is close to 90% and has the ability to filter the results based on BOM count, BOM cost, ripple output, switching frequency etc. It is mainly used to design the DC-DC Converter for servo system. For the other two DC-DC converters Linear Regulators are used.

3.8.2 Design

Using TI's Webench tool a DC-DC Converter is designed that takes 9 to 12.6 V of LiPo battery input and outputs 6.8V for the servo rail and provides up to 6 amps of current. The efficiency of this circuit is 90% and the total cost of parts is less than $5. The schematic of the bypass capacitors and the switching regulator circuit are shown in Figure 21 and Figure 22, respectively. The design uses TI's TPS chip which has an enable pin. This enable pin is be controlled by the microcontroller and also provides a test point in debugging the circuit. The converter gives a maximum power of 40W in worst case scenario where all the servos are stalled

and are consuming significant current. Therefore, the LiPo battery can power this converter for hours.

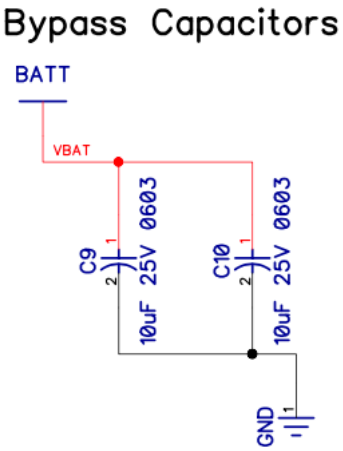


Figure 21: Bypass Capacitors for 6.5V - 5A DC-DC Converter

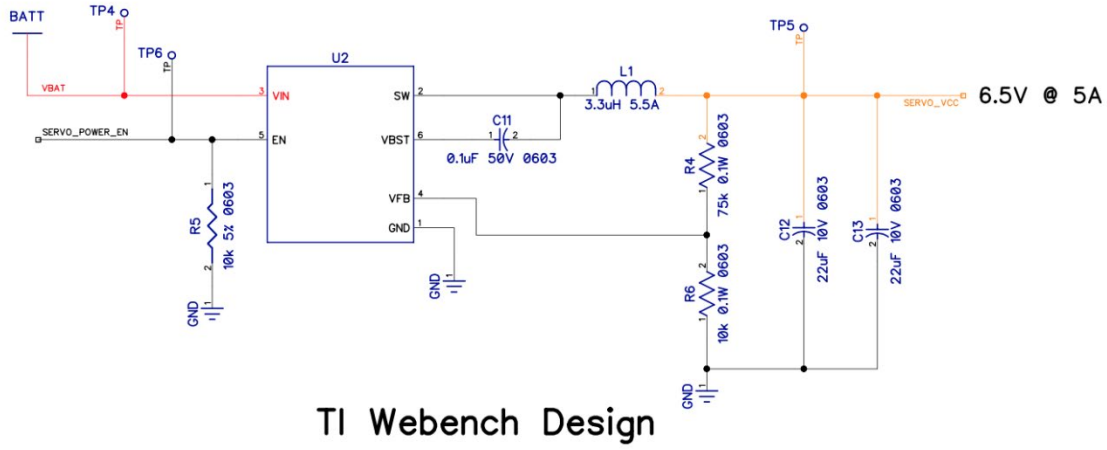


Figure 22: Schematic for 6.5V - 5A DC-DC Converter IC

The bill of materials, or BOM can be seen in

Table 41. The table displays the headings abbreviated as part, manufacturer, part number, quantity, price, footprint and description of the components, respectively for the schematics shown in Figure 21 and Figure 22. There is a total of 22 components that go into building the 12.6V to 6.8V DC-DC converter. The total price for buying the parts for this converter is $3.10 (excluding shipping) and easily fits the budget requirement. Most of the parts generated in the original BOM are obsolete. Therefore, with the help of DigiKey and Mouser Electronics, the BOM is modified as per part availability and solderable footprint size.

Table 41: BOM for 6.8V-6ADC-DC Converter

Part	Mfr	PN	Qt.	Price ($)	FP (mm²)	Des
C11	MuRata	GRM033R71C101KA01D	1	0.01	2.08	Cap: 0.1 µF
R5	Vishay-Dale	CRCW040220K0FKED	1	0.01	3	Resistance: 10 kΩ
R4	Vishay-Dale	CRCW0805102KFKEA	1	0.01	6.75	Resistance: 75 kΩ
R6	Yageo	RC0201FR-0710KL	1	0.01	2.08	Resistance: 10 kΩ
C12, C13	MuRata	GRM155R71C104KA88D	2	0.01	3	Cap: 22 µF
U2	Texas Instruments	TPS56637RPAR	1	1.5	16	
L1	Coilcraft	XAL7070-332MEB	1	1.19	87.4	L: 3.3 µH
C9,C10	TDK	C2012X5R1V156M125AC	1	0.23	6.75	Cap: 10 µF

For both 5V and 3.3V DC-DC converters Recom's switching regulator component is used. Recom is a packaged switching regulator that simply needs bypass capacitors to regulate the output voltage to 5V or 3.3 depending on the product chosen. It eliminates the use of adjustable resistor which are typically used in common linear voltage regulator and is 91% power efficient compared to Linear Regulator, which tend to be only 60-70% efficient and lose energy as heat. The selection guide in Recom's datasheet meets the project requirements which is summarized in Table 42. This component powers the Microcontroller, Encoder, Bluetooth, and PWM Controller circuits. For bypass capacitors two 10µF capacitors are used as per datasheet's recommendations. Both 5V and 3.3V DC-DC converter schematics can be seen in Figure 23.

Table 42: Recom Selection Guide Table from Recom's Datasheet [25]

Part Name	Input Voltage Range (V)	Output Voltage (V)	Output Current (A)	Efficiency (%)	Max Capacitive Load (µF)	Price ($)
R-78E3.3-1.0	7-28	3.3	1	87	220	3.26
R-78E5.0-1.0	8-28	5	1	91	220	3.26

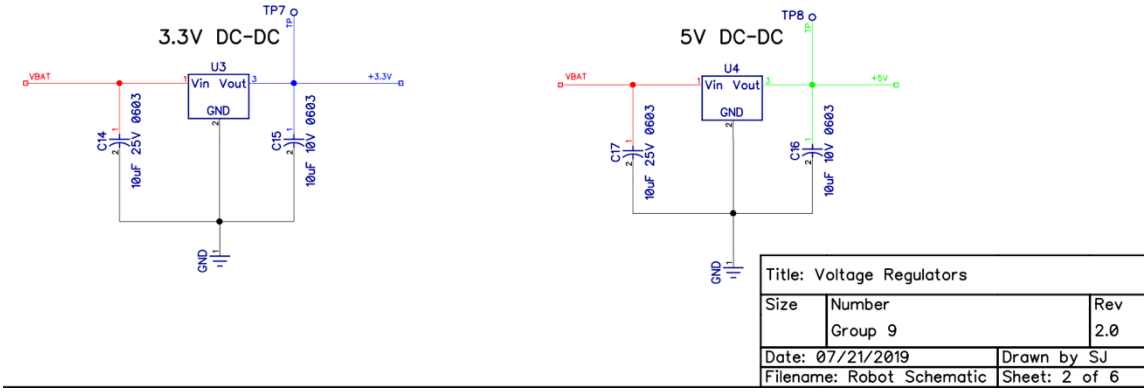


Figure 23: Schematic for 3.3V and 5V DC-DC Converter

3.8.3 Prototyping and Testing

The DC-DC converter is built for prototyping with the appropriate components on the breadboard in the Senior Design Lab. All of the appropriate equipment and resources are available to verify the design for current and voltage requirements. The power output is also to be tested via passive or active load. In some scenarios, such as when using a microcontroller in conjunction with the PWM controller and servos, actual components are necessary for testing instead of equivalent loads because different servo rotations and speeds pull different amount of current to generate power for motion. The tests along with their required equipment and the requirements that they satisfy are summarized in Table 43.

Table 43 DC-DC Converter tests

Requirement	Test	Required Equipment
R.R.E.2	Determine that the DC-DC converter outputs desired voltage and current	Multimeter, Electronic Load
R.R.E.4	Determine that the DC-DC converter is power efficient and does not lose energy as heat to the environment	Multimeter, Thermal Analysis

3.9 PCB

The PCB component is the implementation and integration of the various electronic systems defined previously. It must connect the various integrated circuit components to the microcontroller and have slots for the peripherals to plug into. The Printed Circuit Board is required to simplify the design of the robot. The electronics that make up the robot consists of 6 motors, a microcontroller, a DC-DC Converter, Motor Drivers, H-Bridge Circuit, encoders, and communication system such as Bluetooth. The PCB connects all these systems together and

gives the robot a sophisticated appeal. The PCB needs to be small enough to fit the robot and keep the costs as minimum.

The Printer Circuit Board is a 2-layer copper board with silk screen and soldering pads on it. It consists of terminal blocks that intake power from the battery and then direct them to a voltage regulator by the means of traces. The width of the traces depends on the power it is carrying. The width of the trace carrying battery power is thicker than the trace carrying the power to the microcontroller. The PCB contains test points to check voltage and currents at certain spots. Additionally, adding test points helps in determining the signals using an oscilloscope which can help in debugging communication protocol problems that might arise while implementing I2C, SPI, or UART. The PCB also contains a reverse voltage protection circuit leaving the user with minimal adjustments and focus on playing the game out of the box.

3.9.1 Research

3.9.1a Autodesk Eagle CAD

Eagle CAD is a popular schematic capture and PCB design software that is used by hobbyists and beginners to make 2-layer PCB boards using the free version. The paid version allows users to make PCBs with up to 4 layered PCBs. For the robot system, a two-layer PCB will suffice which makes the free version of Eagle an attractive choice. Eagle has myriad of libraries available on the internet for parts and their footprints. Manufacturers such as Texas Instruments tend to give eagle or ".bxl" files with their part that can be used to generate an eagle footprint for PCB design and schematic capture. Eagle also allows users to create custom parts with their footprints and connects to LTSpice to perform circuit analysis of the electrical design. Eagle CAD, however, does not support 3D view of the generated PCB. It cannot export PCB files in ".stl" format which can be used for design in solid works. [26]

3.9.1b Diptrace

Diptrace is another schematic and PCB design software which is quite prevalent in the industry. The commercial version allows for multiple pin connections and sheets whereas the student version allows 300 net connections and is limited to designing two layered boards. The schematic capture allows the users to connect the pins visually, without wires, logically or using nets. It can annotate easily and can easily import or export from or to other Computer Aided Design software, which is a feature that can be valuable to detect maximum errors and is not available in Eagle. The PCB Layout software has the ability to generate from schematic, like Eagle CAD. However, unlike Eagle it has additional features which includes a verification tool that verifies the PCB nets and traces against the schematic file. Similar to Eagle, Diptrace also provides auto routing capabilities,

however, the designed needs to correct any 90° traces to avoid occurrence of EMF due to sharp electron turns.

Diptrace also has a pattern editor that allows designers to create custom parts of any shape and sizes. It has support for DFX files allowing imports from files of the parts generated by their manufacturers. This helps making difficult layouts easier. The key feature of Diptrace that beats Eagle is ability to 3D model the final PCB. This gives the designer most tactile feel virtually available and help them detect errors during hardware installations by exporting stl files to other CAD programs such as SolidWorks, etc. and looking at them in a three-dimensional axis. There are also a lot of tutorials available on the internet that can ease the design process for the team.

3.9.2 Design

After thorough analysis of different Schematic Capture and PCB design software tools, the team decided to use Diptrace for designing the Robot's PCB. Some of the reasons included the ability to create STL files from PCB designs which can be exported and analyzed in SolidWorks. Another reason is the prior experience in working with the software and hence, the familiarity with the tool. The schematics generated by using DipTrace can be seen in Figure 25 through Figure 27. These figures describe Figure 24 Robot Electrical Network Block Diagram in detail. A summary of the input and output connections is also shown in Table 44 I/O Schedule.

Table 44 I/O Schedule

	Type	Connected Devices	Signal Type
Drive Front Left	Servo	PCA9685	PWM
Drive Front Right	Servo	PCA9685	PWM
Drive Back Left	Servo	PCA9685	PWM
Drive Back Right	Servo	PCA9685	PWM
Launcher / Intake	Brushless DC	PCA9685	PWM
Launcher Release	Servo	PCA9685	PWM
Feedback Front Left	Hall	Microcontroller	1 Digital
Feedback Front Right	Hall	Microcontroller	1 Digital
Feedback Back Left	Hall	Microcontroller	1 Digital
Feedback Back Right	Hall	Microcontroller	1 Digital
Feedback Launcher	Encoder	Microcontroller	2 Digital
PWM Generator	PCA9685	Microcontroller	I2C
Bluetooth	RN-42	Microcontroller	UART

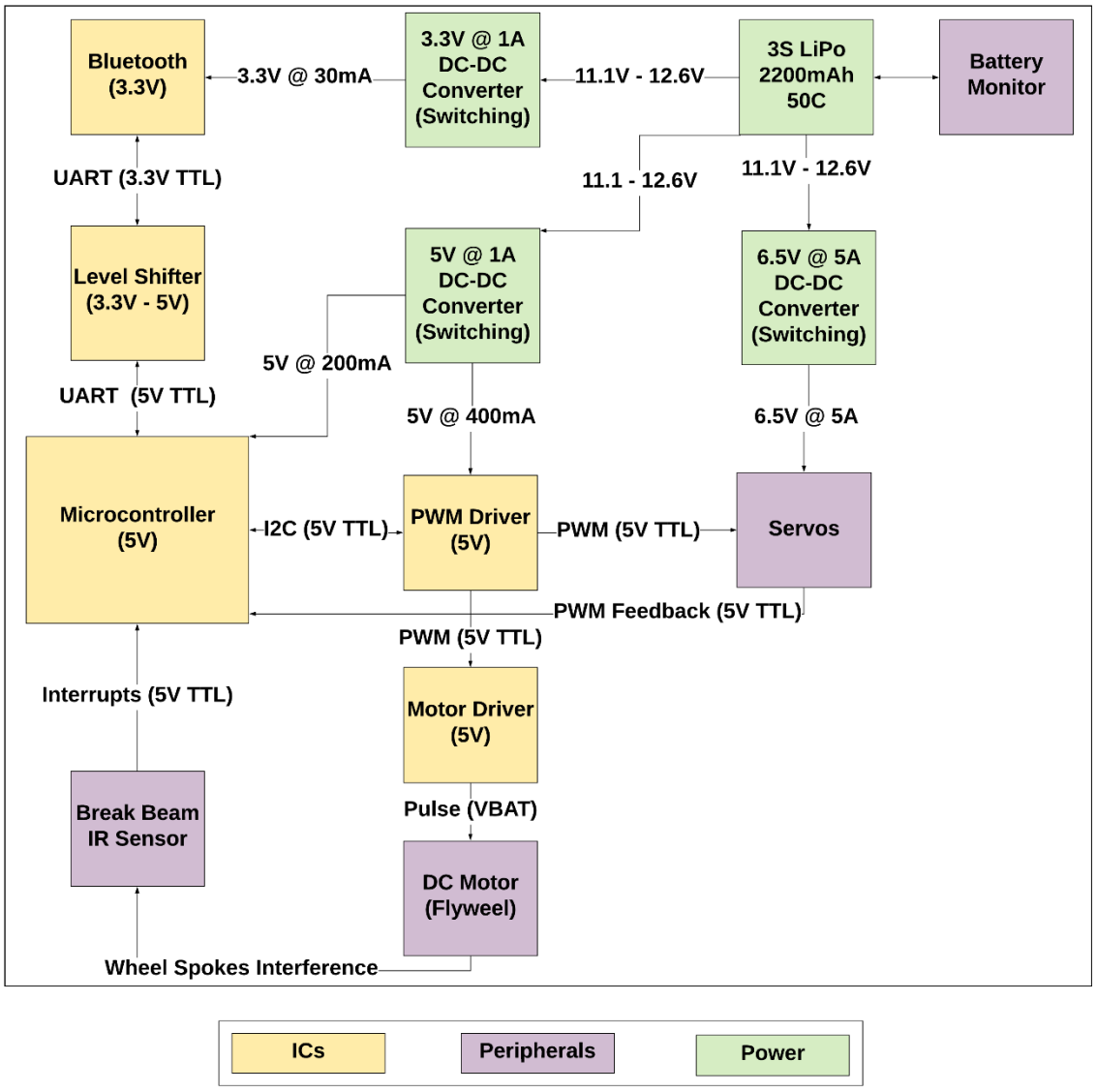


Figure 24 Robot Electrical Network Block Diagram

A schematic of the microcontroller subsystem is shown in Figure 25. The microcontroller used is an ATmega328P which is popular for its use in Arduino Uno development boards. The package is a QFP package unlike the traditional dip socket which is used in Arduino Uno boards. This saves spaces and gives more pins to the design. The firmware that is flashed on this chip, however, is the Arduino's bootloader. This is because it will allow the team to program the board just like an Arduino using their IDE and free libraries that abstract SPI, UART, I2C and other driver interfaces which otherwise would have required custom programming and increase the project timeline. The chip controls the output enable pin for the PWM controller allowing to turn on or off all 16 channels by flipping 1 bit. The PWM controller is communicated using the standard I2C protocol and therefore, two lines from the MCU, PWM_SDA and PWM_SCL, shown in Figure

25, connect to the PWM_SDA and PWM_SCL lines on PWM controller chip shown in Figure 27. The SDA and SCL lines have pull up resistors as the MCU pulls them down to begin communication, however, this is handled by Arduino's libraries and the designer does not need to trouble with them much. Additionally, UART lines are used to communicate with the Bluetooth chip shown in Figure 26 whereas SPI is used to program the microcontroller with ISP interface shown in Figure 25.

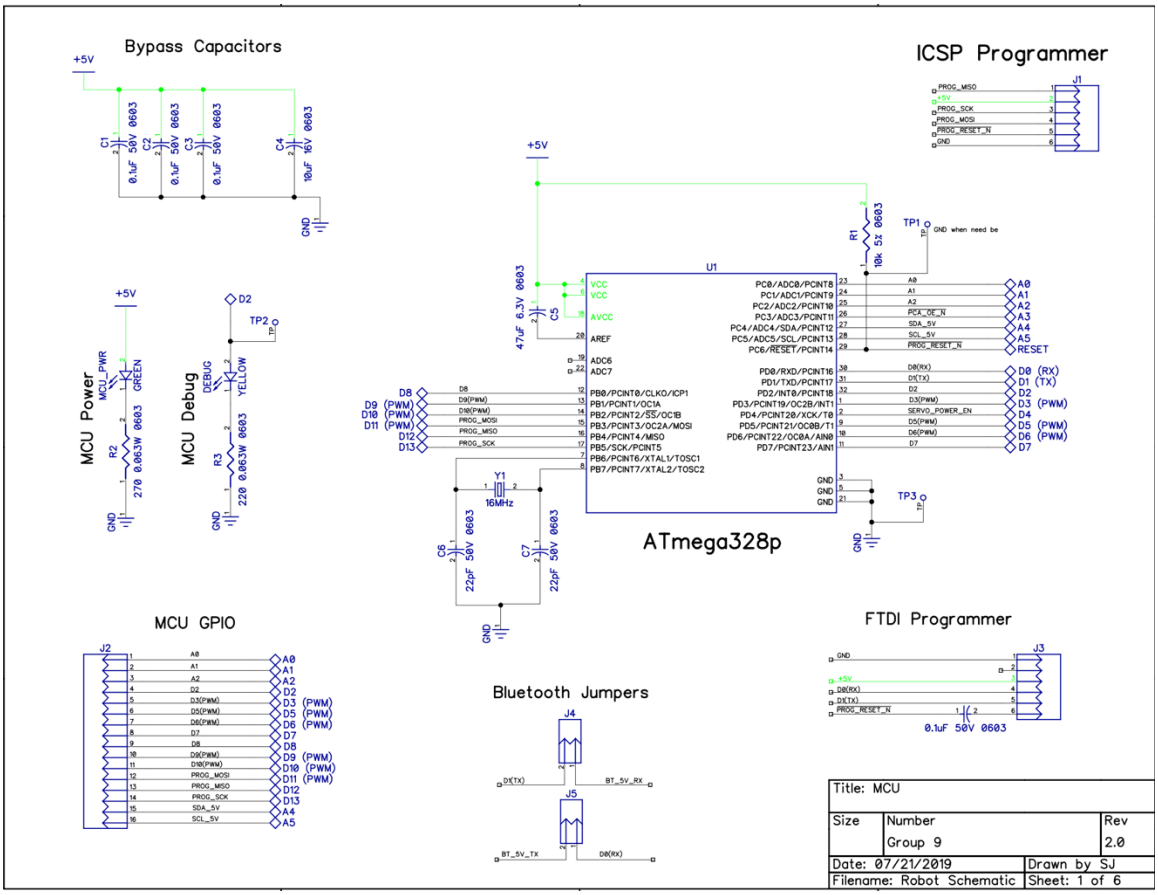


Figure 25: ATmega328P Microcontroller Schematic

Three bypass capacitors are used for the two VCCs, and one AVCC power lines shown in Figure 25. The values for these capacitors are a standard 0.1µF as their main purpose is to attenuate any noise entering the CPU and potentially damaging the IC with high voltage spikes. Also shown in the schematic is an In-Serial Programmer interface which is used to program and flash the ATmega328P MCU. Traditionally, Arduino boards come with a USB interface that in conjunction with an RS232 to TTL converter can help program the chip. However, that requires an additionally voltage regulator and the ability to add differential pair signals to the board increasing the overall complexity of pcb layout and designing the system. Therefore, to avoid such intricacies, a traditional ICSP program is used that uses SPI with high bandwidth to transmit signals and program the chip not only at a faster rate, but with ease.

The Bluetooth chip used for sending and receiving Arena packets is Microchip's RN-42. This is a popular IC that is used on multiple Sparkfun BlueSMiRF boards, which is compatible with Arduino and its clones. Therefore, the idea is to use the onboard firmware and interface it with the MCU to send and receive Bluetooth packets. The packet information is discussed in communication sections of this documents and can be seen in the design sections 3.6.2 Design and 4.11.2 Design for Robot and Arena packet structure, respectively. The schematic of the Bluetooth IC can be seen in Figure 26.

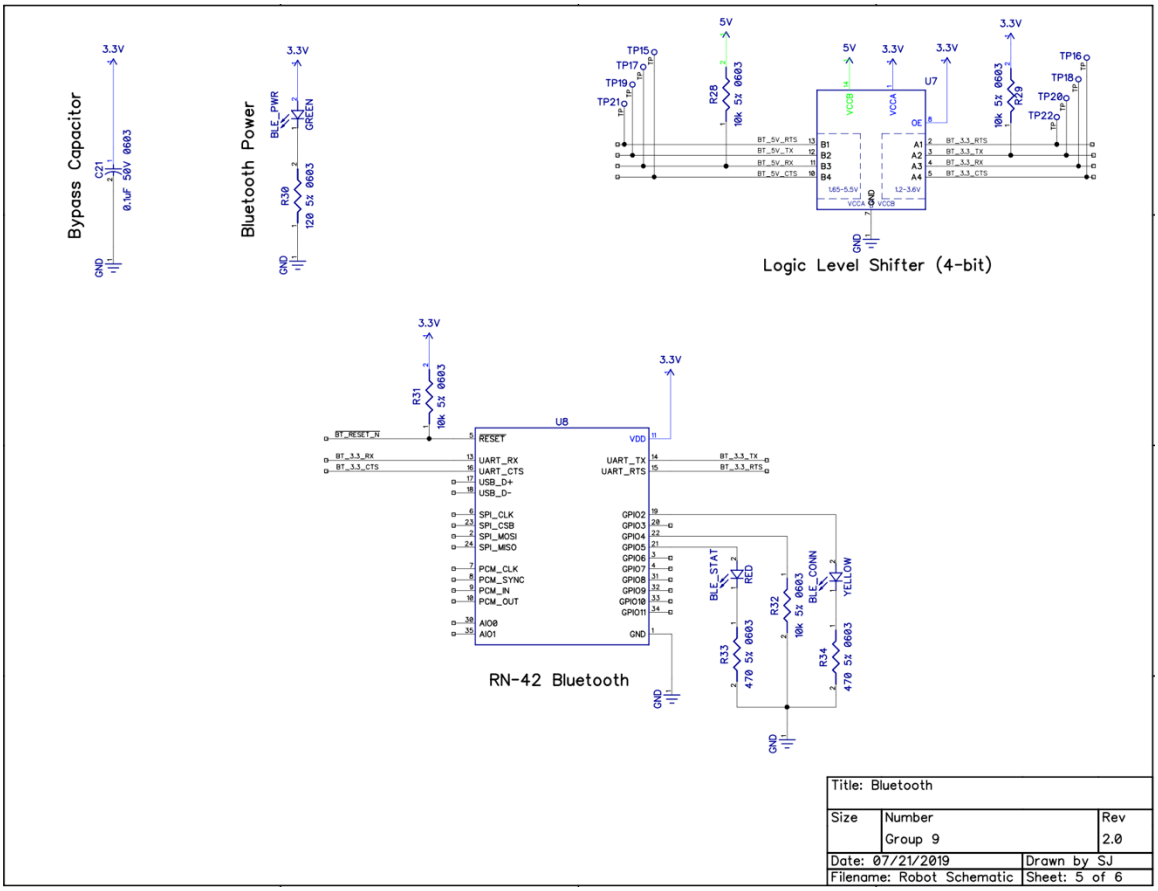


Figure 26 Bluetooth Schematic

The schematic for PCA9685 chip is shown in Figure 27. This chip controls the servos that drive the robot. The input signal is a 16 bit or 2 bytes of data that is received using I2C protocol. The I2C lines are driven high by pull up resistors to avoid any unpredicted behavior caused by floating pins. To initiate the I2C conversation the MCU pulls down the clock and starts transmitting the over the PWM_SDA line. The PWM controller takes input voltage at Vcc pin from the 5V Recom voltage regulator output (schematic shown in Figure 23). The servo rails are used to power the servos and send PWM signals to manage their rotation using GPIO pins shown in Figure 27. Feedback from is fed to the GPIO pins of the MCU

which adjusts the 16-bit code based on the received signal and sends it to PCA9685 via I2C.

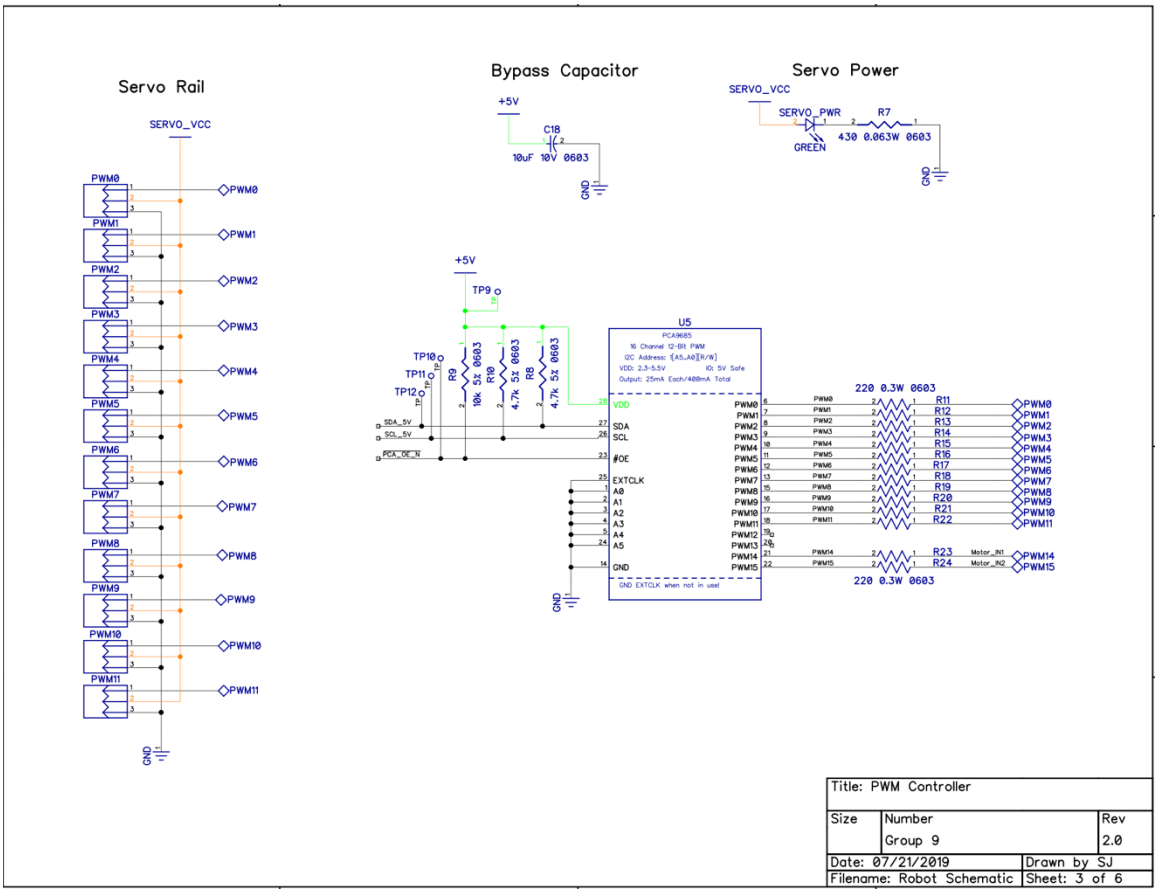


Figure 27 PWM Controller Schematic

At last but not the least, Figure 28 shows the schematic of the motor driver that drives the flywheel. The IC takes battery input to power the DC motor and is controlled by channels 15 and 16 of the PWM driver for velocity control. There are LEDs attached to the PWM lines. Based on what direction the motor is spun, the corresponding PWM signal toggles the LED. Therefore, the faster the DC motor spins to throw the ball, the brighter the LED gets and vice versa. Additionally, to add reverse rotation to the DC motor, second LED can be used to get a general idea of how the control logic is working. There is a current limiting resistor attached to ILIM pin of the motor driver chip. This resistor, as the name suggests, limits the current that the DC motor pulls. Care needs to be taken as more current provides more torque to the flywheel and therefore, after some experimentation and cross referencing the datasheet, a resistor value of 22k Ohms was chosen that limits the output current to 3A. The IC is designed such that any current over draw will automatically shut off the left half of the IC from the right half thereby preventing the PCB from getting damaged.

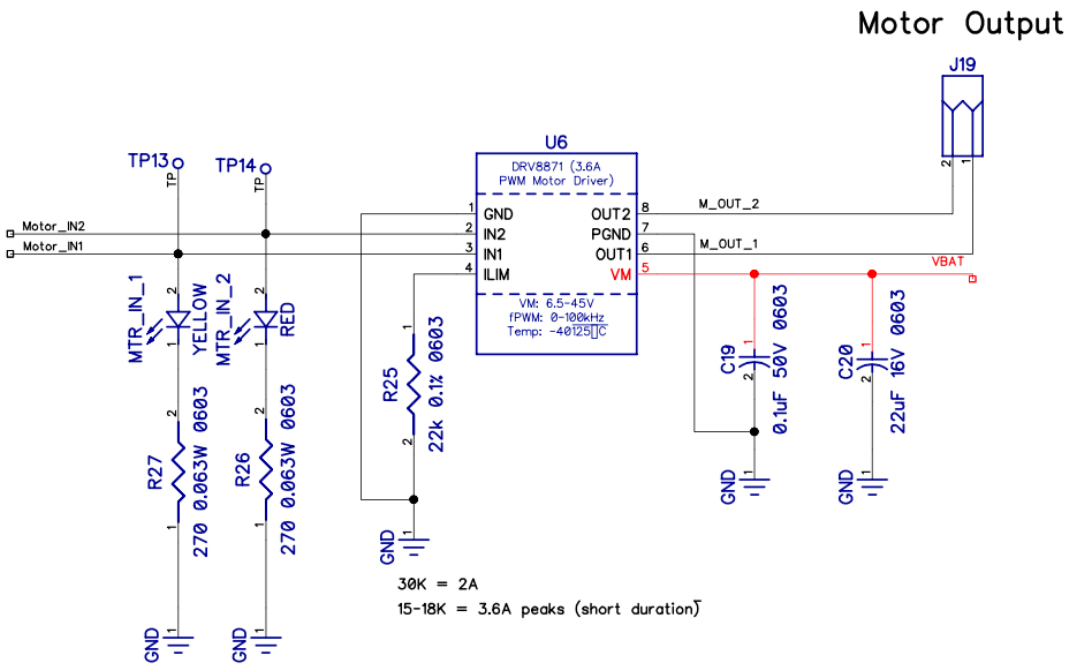


Figure 28: Motor Driver IC for the flywheel

3.9.3 Prototyping and Testing

The PCB is to be designed and purchased through a PCB manufacturer which sells 2 Layer PCB for $2 for 5 boards or cheaper if possible. Additionally, the stencils can be purchased for the board at just $6. The stencil is utilized in conjunction with the board and solder paste to quickly and accurately build the PCB. A heating chamber is required to evenly heat the board to prevent any damages. The tests and requirements are summarized in Table 45.

Table 45 Robot PCB Tests

Requirement	Test	Required Equipment
C.R.1	Determine that the PCB is an appropriate size to fit the robot	Ruler
R.R.4	Determine that the PCB does not have any shorts or opens	Multimeter

3.10 Software

The software component drives the hardware components in the PCB. This includes any control software such as various PID control, state machines, and Bluetooth communication. The software for the Robot must be at least Soft-Real

time to ensure that inputs and outputs are processed in a context that does not affect the fidelity of the system. For example, an input from the motor encoders should be processed and utilized in outputs for the relevant motor within a single deterministic loop. If the motors outputs are updated too long from the motor encoder input, the data is no longer valid and could be harmful to the system. Thus, a firm data flow structure must be followed for the entire software system. Additional limitations on the software are based on the microcontroller chosen in section 3.5 Microcontroller. Ideally, the written software follows a strict architecture to enhance readability and debuggability. Debugging is critical for the robot because there are several factors that could lead to failure: Mechanical, Electrical, and Software issues. Often each one of these have issues are observable only in another area. Thus, the chosen software and libraries must have thorough documentation, and be thoroughly tested prior to usage. Each function or block of code should be fully documented and contain a unit test that corresponds to the requirement that drives the function.

3.10.1 Research

3.10.1a Arduino IDE vs Atmel Studio

The Arduino IDE is a very popular software that includes a full development environment including a text editor, compiler, boot loader, and serial monitor. It has a very easy-to-use interface and a large amount of documentation due to the prevalence of Arduino as a hobby device. The IDE is strictly designed around particular Arduino boards, thus the support for the chip itself is somewhat limited and requires extra effort to work with.

The Atmel Studio software is a software provided by Microchip that supports development and debugging for AVR and SAM microcontrollers. The application contains a fully-fledged IDE that supports text editing, compiling, debugging, and deploying to AT chips. However, this does require (similarly to the Arduino IDE) an extra chip that acts as a programmer device. The IDE can also import Arduino sketches and libraries.

Another option is the Visual Studio Code extension for Arduino that extends the capabilities of the Arduino IDE. The Extension provides full IntelliSense and all of the advanced capabilities of the Visual Studio Code application. This provides all of the capabilities of the Arduino IDE with a much better text editor. [27] [28]

3.10.1b Libraries

The primary PID library available for Arduino environment (and thus the ATMega) is the Arduino PID library by Brett Beauregard. This library implements a traditional PID controller in a professional and well-documented way. Additionally, all of the PID code is factored into a self-contained class that can be instantiated as many times as necessary. Another library option is FastPID by Mike Matera. This library

is implemented with strictly fixed-point data types rather than utilizing any real data types. This reduces complexity and increases performance substantially on the ATMega. This is implemented by converting floating point coefficients into fixed point by a static conversion. This library does require a deterministic loop rate in order to function properly. The API is similar to the PID library in that instances of classes are instantiated, and functions are called to process updates. [29] [30]

I2C / PCA9685 Libraries

The best supported library for the PCA9685 is the Adafruit PWM servo library. This library implements the appropriate I2C commands and wraps them in a very simple to use and understand API. The library takes care of the I2C ID commands such as setting the clock, waking or resetting the device, and data transmission. The library is very well documented and thoroughly tested. Other libraries exist; however, they are fairly undocumented and lack support. A Servo library that supports the PCA9685 natively does exist and is discussed below. [31]

Servo Libraries

The default servo support in Arduino is handled by the Arduino Servo Library. This library allows users to create servo objects and set their speeds or positions in a few different ways. This library is limited to support up to 12 motors at once. This library is focused around the generation of PWM signals by the Arduino itself, and thus is unlikely to be useful for this project. Another available library is advanced Servo Easing library available from contributors. This library provides precise mathematical driven control over servos, and directly supports the PCA9685 device. The device also supports motor synchronization, and various easing functionality. However, it does not have explicit support for continuous rotation servos. The library may need to be adjusted manually to support this. [32] [33]

Bluetooth libraries

There are no widely available Bluetooth libraries for Arduino because the chip firmware generally abstracts data out enough to be a simple serial interface that native Arduino support handles. Thus, the Arduino Serial library is the primary library to use for the Bluetooth modules utilized in this project. An additional abstraction layer can be included to wrap the serial functionalities into a more usable interface.

3.10.2 Design

The design for this software is developed to ensure maximum robustness, scalability, modularity, and maintainability. The general structure follows a strict I/O paradigm to guarantee real-time reliability. The design does the following: updates inputs, processes the system’s data, and updates the outputs. This entire process must run each cycle with a deterministic scan time to ensure real-time operation.

The process is shown graphically in Figure 29. The software and appropriate unit tests are to be developed and handled in Visual Studio Code with the C/C++, and Arduino IDE extensions.

Figure 29 High level process flow

The system defines three finite state machines shown in Figure 30, Figure 31, and Figure 32. The Master state machine defines the states that the robot system can be in, and their valid transitions. The transitions are determined by the master arena software and faults when errors occur in any of the processes. The actuators and sensors on the system have their own state machines to reduce system complexity in fault-tolerance and error checking. In the actuator states, the actuator transitions from offline to tracking such that individual actuators can be deactivated without disabling the entire system. The tracking state indicates that the actuator is actively following commanded positions or velocities, and the fault state indicate that some error has occurred such as tracking errors or invalid state transitions. The sensor state machine indicates the validity of the sensor's information. The reset state indicates that the sensor's information is invalid in this cycle and it must reset accordingly. Active indicates that the sensor is actively tracking velocities / positions and the information appears valid. Fault indicates that the sensor has had an error or invalid transition.

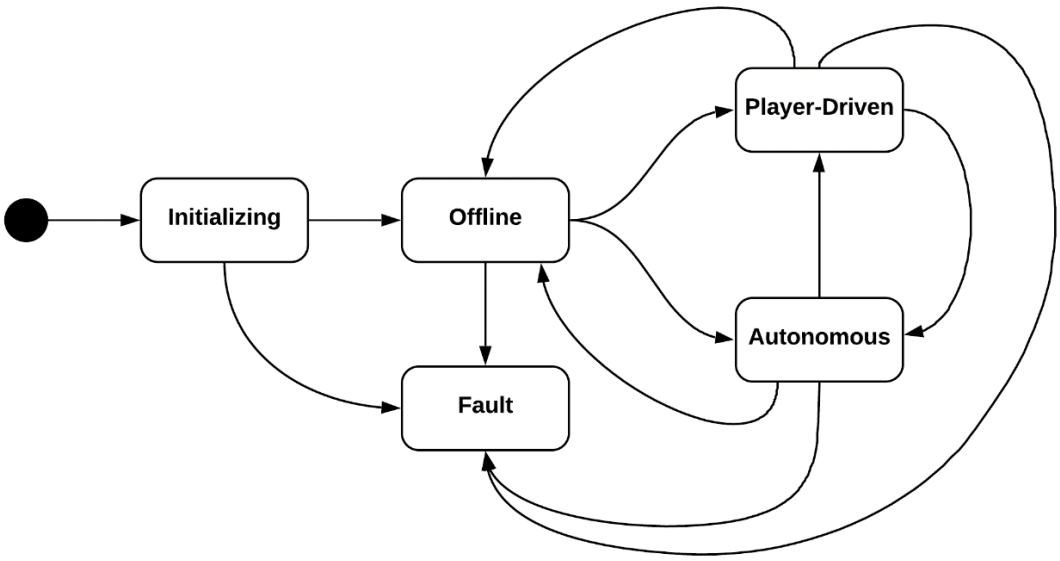


Figure 30 Master state machine

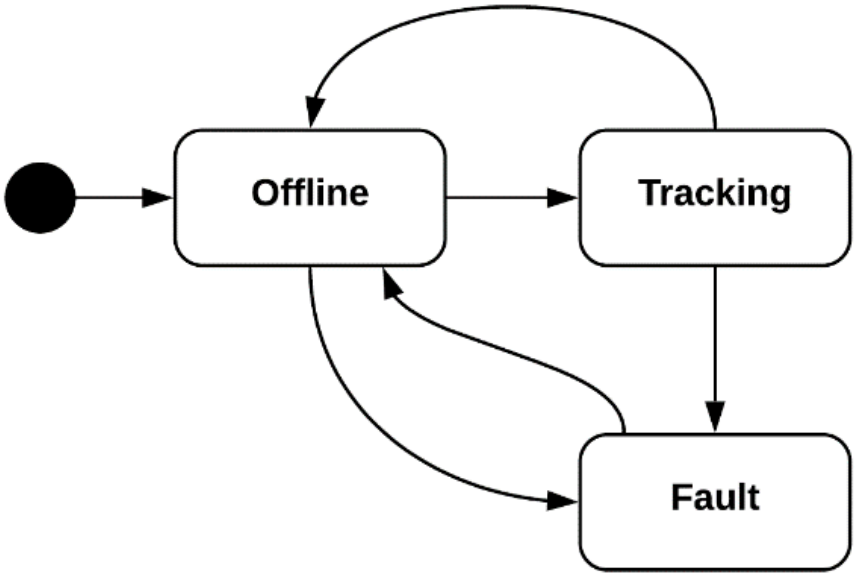


Figure 31 Actuator State Machine

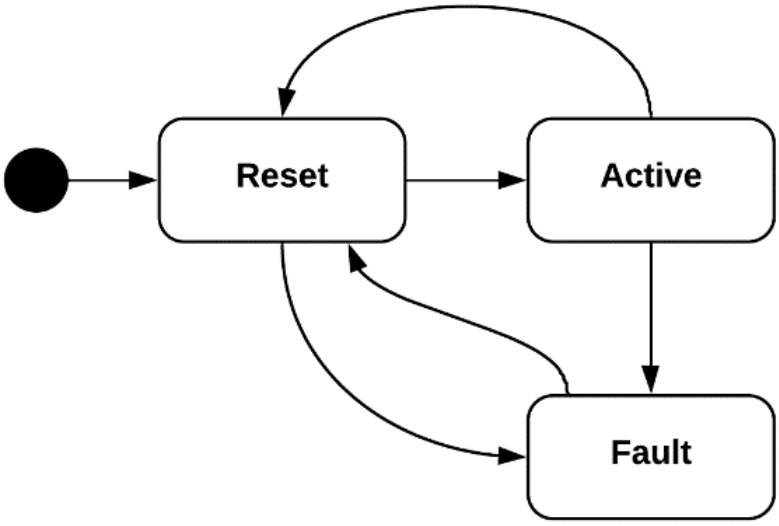


Figure 32 Sensor state machine

The software architecture and preliminary class diagram is shown in Figure 33. Four isolated layers are defined: Application layer, System layer, object layer, and library layer. The application layer contains the primary infinite loop for the robot's software that processes the inputs, data, and outputs in the appropriate scan time. It instantiates and calls methods from the classes in the system layer. The system layer contains classes defining robot-specific systems such as the intake, launcher, and drive systems. Each of these systems contain state-specific processing such as determining when an actuator should be active, how inputs from the master arena are handled, and general state machine I/O processing.

The object layer contains abstract code for actuators, sensors, and state processing. This layer must be instantiated and operated by an upper level layer. However, the majority of the data of the system is processed and allocated to this layer. Finally, the system and object layer leverage existing libraries when possible. These libraries include the PID, Servo, I2C(PCA9685) and Bluetooth libraries available for the electrical components. The specific libraries chosen are the FastPID library, the Servo easing library, and the standard Arduino serial library. The FastPID library provides a high-performance control loop that allows the Arduino to focus on other important tasks such as polling the encoders or processing other data. It reduces the reliance on floating point operations by working with a fixed update rate that the architecture already supports. The serial library provides a simple, robust interface for communication over Bluetooth.

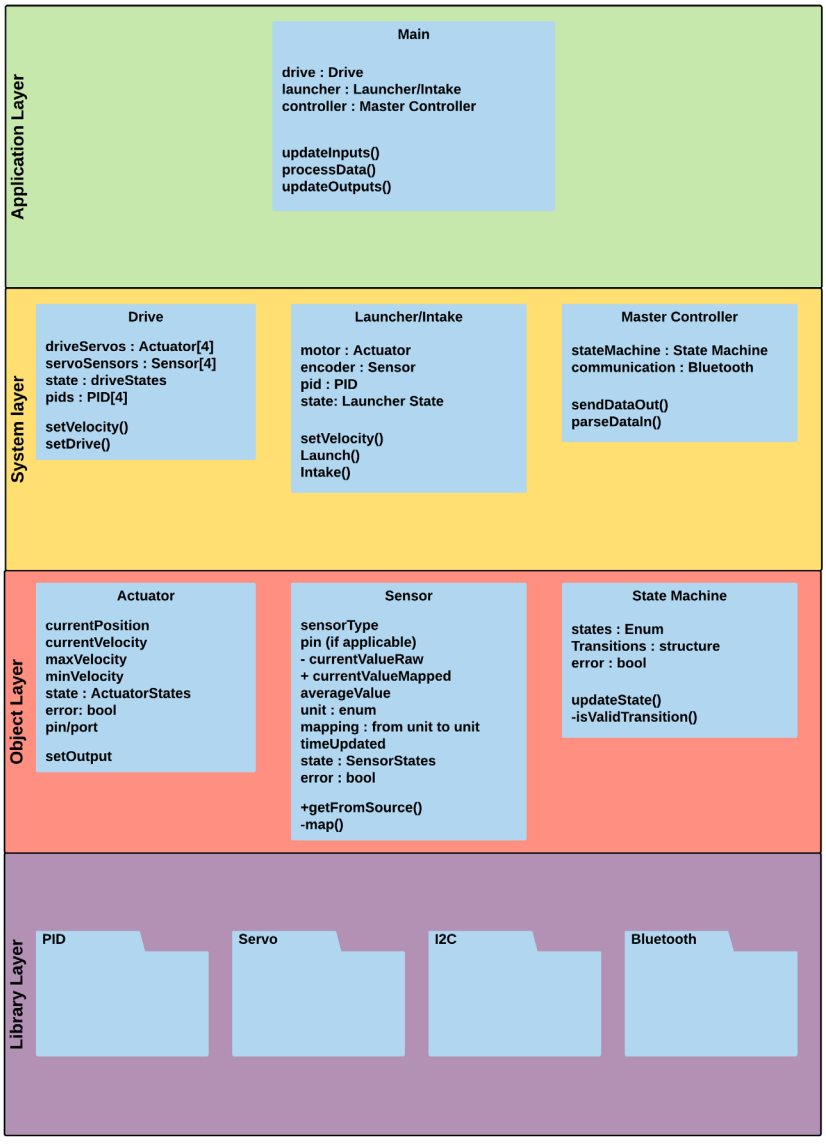


Figure 33 Robot software architecture design and class diagrams

3.10.3 Prototyping and Testing

In order to adequately prototype the software for the robot, an Arduino Uno and some evaluation boards are needed to mimic the functionality of the PCB if it is not finished yet. Otherwise, the software unit tests can be developed independently to the actual software such that each code block can be built and tested without the other blocks being complete. In order to verify that the system is working as it should be, every test listed in Table 46 will need to be conducted on the separate units of the system.

Table 46 Software System Tests

Requirement	Test	Required Equipment
R.R.S.3	Communication Unit Tests (Bluetooth, Wired)	Bluetooth Module, USB Cable, Arduino, Power Supply, breadboard
R.R.S.3	Actuator Unit Tests (Base, intake, launcher)	USB Cable, Arduino, Power Supply, actuators, breadboard
R.R.S.3	Sensor Unit Tests (Encoders, Switches, battery, etc.)	Sensors, USB Cable, Arduino, Power Supply, breadboard
R.R.S.3	Control Unit Tests (PID, etc.)	Sensors, Actuator, Arduino, Power Supply, breadboard
R.R.S.3	Safety system Unit Tests (Heartbeat, etc.)	Arduino, Power supply
R.R.S.3	Full software tests (Operations at max capacity, timing, etc.)	
R.R.S.3	State machine Unit Tests (including operating modes)	Arduino, Power supply

4.0 Arena

The arena subsystem is the subsystem physical frame that the robot can be placed on, along with all the components required for basketball gameplay. The way all the separate subsystems integrate together is laid out in Figure 34. It contains the computer vision component of the project for robot and ball tracking. The subsystem also contains all the player experience including lights, sounds, and display. The final mechanical model for the assembly is shown in Figure 35.

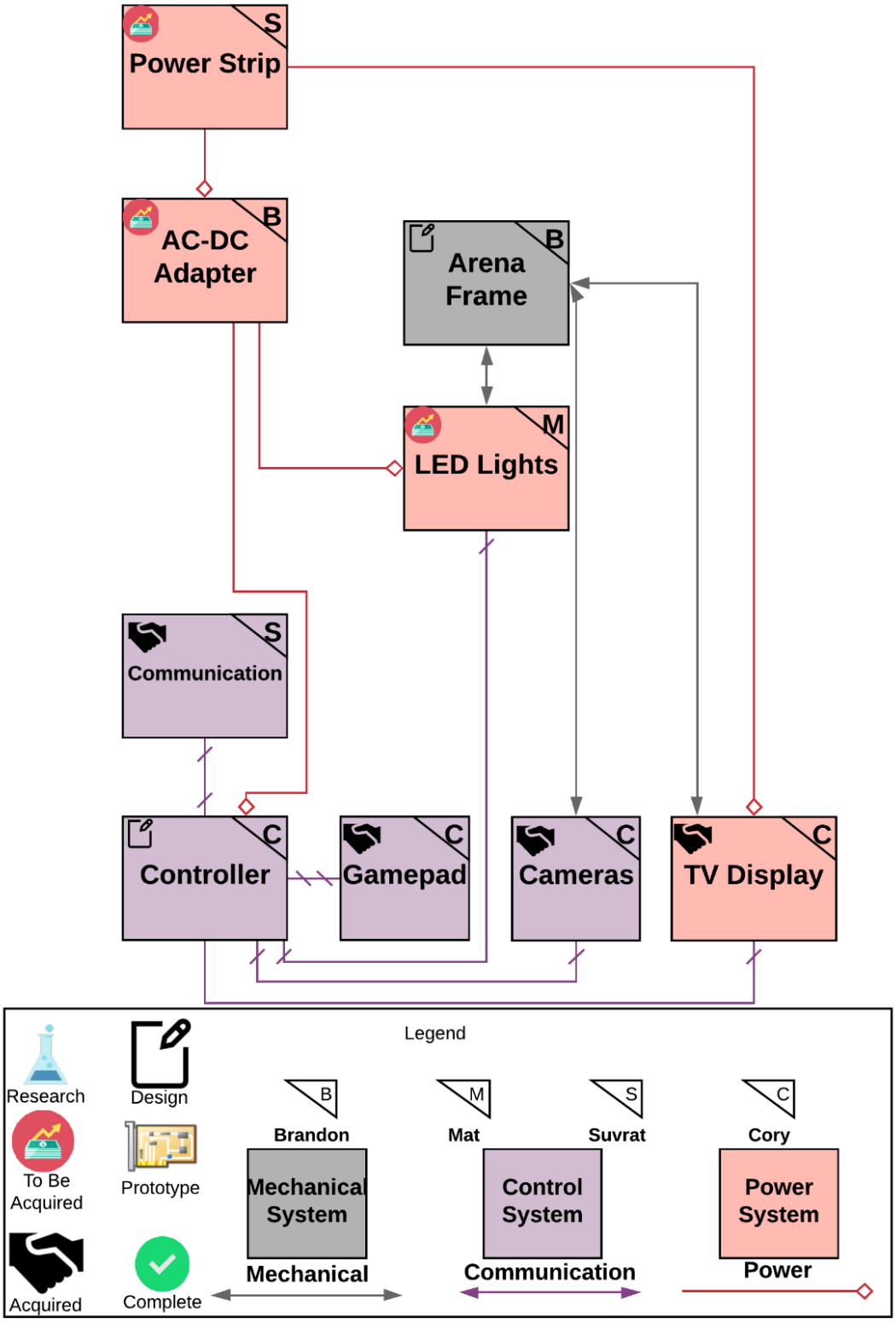


Figure 34 Arena Subsystem Power and Signal Diagram

Figure 35 Final Arena Rendering

4.1 Frame

The frame component is the structure to hold the robots, electronics, court, and other components physically. The frame can be quite large; thus, it is designed modularly such that it can be stored and transported in a small location. The court must be perfectly level to ensure the ball remains in place. The exterior of the frame is closed off to prevent the ball from flying outside of the arena. The material is transparent to make sure that spectators can see the entire field.

4.1.1 Research

4.1.1a PVC

PVC pipe is a light weight and low to medium cost material. The cost is dependent on how many connectors are used as they are the most expensive PVC part to buy. The number of connectors used depends on how much a PVC pipe can maintain level at varying lengths, if the PVC needs to be strengthened more to keep it level that increases cost. However, to alleviate the need for extra connectors we could use thicker pipe, again though the thicker the pipe the more cost increases. PVC pipe is a very portable material though since it does not need

any hardware or glue to hold it together. The downside to this portability is that when taken apart and put together repeatedly, it could go together at a different depth than it did before depending on the force applied by the person putting the pieces together. This varying depth can throw off calculations if the arena ends up being off level. A unique problem to PVC is that it is the only round material considered, a round material is not easy to mount other parts on and supplies little support to the flooring of the arena. Additionally, the round property of PVC means that when together it can rotate in place, this can again affect calculations if the camera mount is not placed in the exact same height and position every time. Lastly PVC pipe placed into connectors will always have a lip between the connection point. Because of this lip a piece of plywood may need to be added to the frame in order to ensure the flooring can sit above the lip to make it level.

4.1.1b Metal

A metal frame will cost the most out of all options. To create a metal frame would consist of at least four L angled brackets made of aluminum, steel, or another lightweight inexpensive metal. These L angled brackets would be used as the wall mounts of the frame and additionally for securing the particle board or similar material to the frame that is used to support the flooring. From our research, it was concluded that aluminum would have been the cheapest option for the L angled brackets with prices for all sides of the arena at over $100. Additional materials to construct the frame from metal would include the particle board or similar material to secure the flooring to the frame. In the research done the lowest cost material for this would be an OSB board which at the correct size for the arena would be roughly $8 at the lowest. The walls, hoops, and camera mounts would be connected to the L angled brackets using locking hinges so that the arena walls, hoops, and camera mount are capable of folding into the arena. These hinges were around $7 a piece and we would need at least two per wall, one per hoop, and one for the camera mount. The metal frame would supply the arena with the best compact design in that the arena would be able to fold into itself and carried. This material would also be near the highest weight of the researched materials. Even though the portability of this material is excellent, the minimum cost of around $180 is quite above the estimated budget for the arena. So, if this material is used either the budget of the arena would have to increase, the budget for the entire project would have to increase, or the budget would have to be lowered from another section of the project and applied here.

4.1.1c Wood

A wood frame would be one of the lowest cost options for the frame, but would require the most actual work to construct, in that there will need to be numerous cuts in the wood that require a tiny bit of skill in carpentry. Making a wooden frame would use four 2x4s as the walls and four 2x2s as the lengthwise flooring supports in the middle of the walls. Each cut in the four-foot sides (basket sides) is a notch. One notch on each side will hold the five-foot sides (lengthwise sides)

perpendicular but even height with the four-foot walls. The other four notches are evenly spaced and will hold the 2x2 strengthening planks an inch below the surface of the outer walls. The flooring will then sit on these strengthening planks with no glue or hardware to hold them down. For this reason, the wood used will need to be strong, rigid, as straight as possible, and with as little knots as possible to ensure the cuts and notches is even enough to hold the supporting braces level for the flooring. The wood will also need to be reasonably priced and accessible. For this reason, and all previous explained building choices, a kiln dried softwood, such as pine, spruce, or Douglas fir is preferable. This type of wood for a 2x4 would be about $5 for 10 feet of material and $2 for 8 feet of 2x2. The walls, camera mount, and basket mounts is attached to the frame with PVC pipe and a bolt that will go through the pipe and pipe holder to hold it in place. The hardware for this would be roughly $10. Using wood would come to a total cost of around $30 to construct the frame. The savings in the arena budget if using wood could then be used on better parts or parts that make other parts of the project easier. The downside of using wood is that it is difficult to get notches cut very level and it's of medium weight when needing to transport the eight planks together.

4.1.2 Design

The size of the arena is approximately 4 ft width by 5 ft length by 3 ft height which is not to an exact scale of a real court. After creating prototypes of the arena in SolidWorks using the previously mentioned materials and putting together a parts list including price for each, our team decided to use wood because its inexpensive, requires little hardware, and can be easily disassembled and reassembled.

The design of the frame uses four 2x4 kiln-dried heat-treated spruce-pine-fir wooden pieces as the base frame. Two pieces on each basket side serve as the main notched pieces that will hold the middle supporting braces. Each of these two pieces is cut to four feet in length with 1x1 inch notches cut two inches deep a half inch away from either side of the 2x4. These notches are used to slot the lengthwise 2x4s into place. The lengthwise 2x4s have the matching joint cut so that it fits tightly into the basket side notches and is level on the top and bottom of the joining pieces. With these four sides fit tightly together the inner perimeter of the frame measures 18 feet or 4x4x5x5 feet on each side. Additional 2x1 inch notches are cut into the basket-side pieces two inches deep. These notches are used to hold the supporting braces. The supporting braces are made of the same type of wood cut to a 2x2 inch plank five feet two inches long. Each side of the supporting braces has a joint cut to match the notch on the basket-side pieces. This cut places the supporting braces one inch below the top of the frame. With all walls and supports in place there is a level platform in which to place the flooring of the arena. The frame has lead screws attached to each corner area to keep it level on any surface. Figure 36 shows the arena dimensions and components.

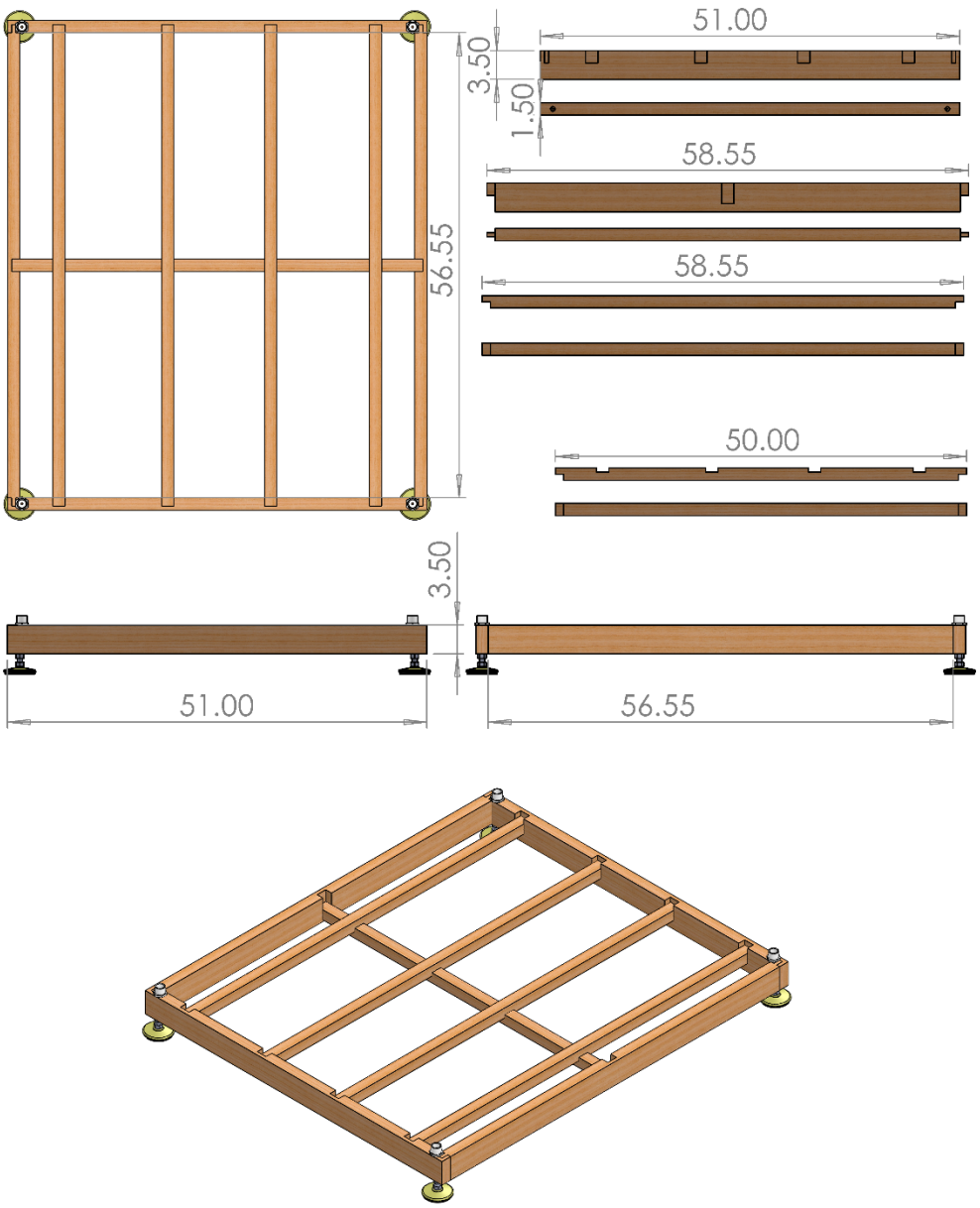


Figure 36 Arena frame drawings

4.1.3 Prototyping and Testing

The frame components were purchased from local hardware stores. The cuts required to correctly set up the frame were done with a jigsaw and table saw, both of which are available through team-member's families. Basic tests to confirm functionality of the arena frame are outlined in Table 47 below.

Table 47 Frame Tests

Requirement	Test	Required Equipment
R.A.6	Time how long it takes to put together	Stopwatch
R.A.6	Time how long it takes to take apart	Stopwatch
R.A.7	Determine how level the system is from different starting conditions	Level
R.A.8	Determine strength of the frame in event of falling	Weights

4.2 Walls

The walls surrounding the arena exist to prevent a rogue ball or robot from flying out of the arena and hitting someone or something it is not supposed to. It also exists to prevent people from placing arms or objects into the arena while the robots are running.

4.2.1 Research

4.2.1a Clear Acrylic Plastic

The first material to be researched was clear acrylic plastic. This material is very rigid and the strongest material to be considered. This rigidity would make acrylic the best material for a camera mount alleviating the need for extra mounting material should the camera instead need to be mounted to the frame. The solid acrylic panels would also be an excellent choice for dampening any wind that could occur from outside forces and affect the calculations for shooting the ball, which would be necessary should this project need to be demonstrated in an outdoor environment. A clear acrylic wall would also make the best choice for viewing the robot, making it very easy for any player to see and control their robot. Attaching the wall to the arena is also made easier by the acrylics rigidity as hinges would be all that is needed to hold the walls and would make them collapsible for portability. The downside of the acrylic material is that it is much heavier and would add a tremendous amount of weight to the arena decreasing portability. In addition, the cost of clear acrylic plastic is tremendously more than any other material considered. In fact, this material is the absolute best choice considering all aspects, however the cost is so prohibitive that our team is unable to purchase the necessary quantity needed for this project.

4.2.1b Clear Vinyl Plastic

A clear vinyl material is a medium cost solid, but not rigid material. For this project at least six gauge or thicker vinyl would be used. This thickness would allow for hardware to be installed without ripping the vinyl material when it is pulled tightly. The vinyl will need to be pulled as tight as possible in order to make the material

as clear as possible and help the ball bounce back into the court and keep the robot from falling out of the court. A vinyl material will also dampen wind almost as good as the rigid acrylic plastic if pulled tight enough. Due to the vinyl not being rigid the camera will have to be mounted on the frame, requiring more hardware. This is a lightweight material and will add almost no weight to the arena making it more portable. In addition, since the material is not rigid it can also be rolled up and carried separately. Attaching vinyl to the arena would entail the use of posts on each corner of the frame, again requiring additional materials.

4.2.1c Mesh

The third material considered is a mesh material, either plastic or nylon woven in a net like structure with one inch or less square holes. The mesh material is the absolute cheapest material considered, costing only several dollars for many square feet of mesh. Mesh is also not a rigid material and thus will not be suitable for attaching the camera boom and so additional hardware is required. Mesh is however extremely lightweight making it a very good material when considering portability. The ability of mesh to dampen wind is almost nonexistent, so there would require more work in the ball shooting algorithm to insure target goal probability. Mesh will also require additional materials to attach to the frame in the form of posts. These posts could be PVC or another sturdy low-cost material. PVC would be an easy solution as it is sturdy in short lengths and can easily be slotted into the arena by attaching PVC caps to each corner and placing the three-foot PVC pipe with mesh material attached into these caps.

4.2.2 Design

The final wall design is a combination of PVC pipe to hold each corner upright, a mesh material used for the physical wall itself, shower curtain rings to slide the mesh open for easy access, magnets to keep the mesh curtain closed, and some small ceiling hanging hooks to hold the base of the wall to the frame. Each wall is made of ¼ inch woven mesh material three feet tall and either four or five feet wide depending on which side it is attached to (Figure 38). A mockup of how the walls are attached to the PVC and surround the arena is provided below in Figure 37. The wall posts are made of 1-inch PVC pipe three feet long each. These PVC pipe posts fit into PVC couplings which fit into plugs that are mounted to each corner of the arena frame. Each PVC pipe has small holes drilled ¼ inch apart through both sides of the pipe down the full length of the pipe. The mesh has fishing line pulled through these holes and tied to keep the mesh tight to each PVC post. The frame has small ceiling hanging hooks attached along the base of the wall area. These hooks are used to hold the base of the mesh wall tightly in place. When completed there are four PVC corner posts with mesh connecting them.

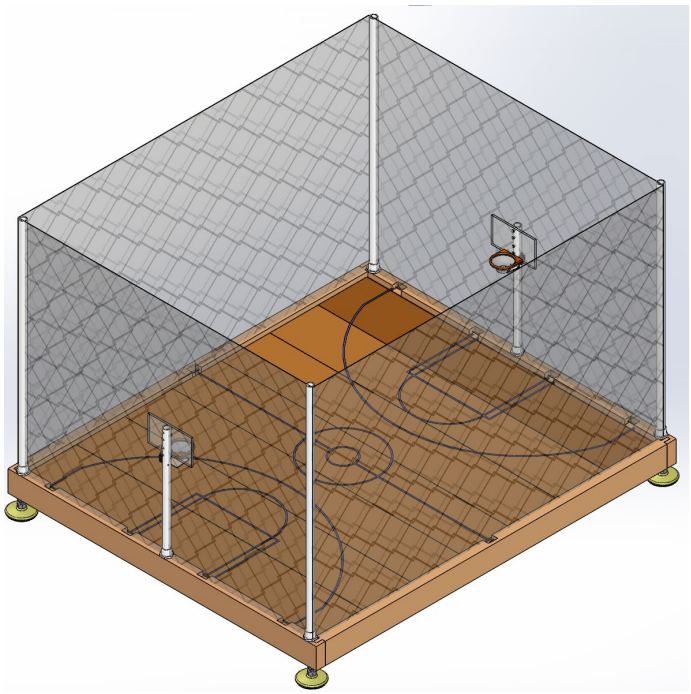


Figure 37 Mesh walls attached to PVC uprights

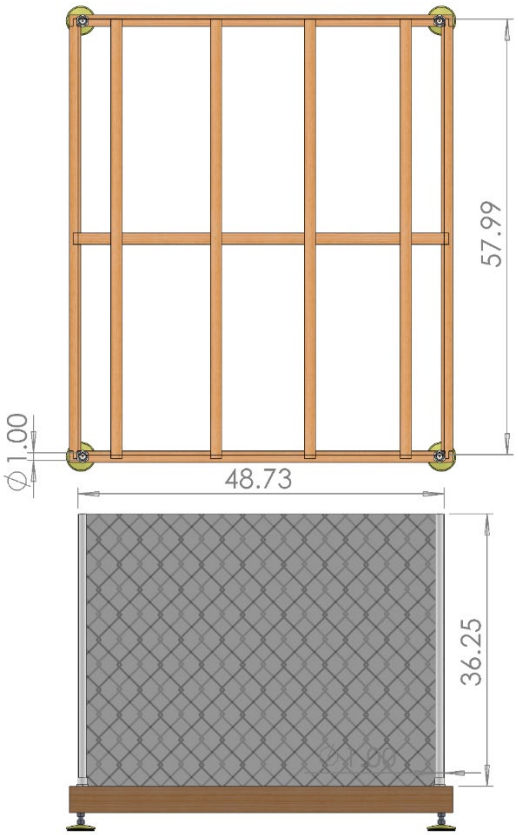


Figure 38 Mesh wall dimensions

4.2.3 Prototyping and Testing

The wall component is purchased from Amazon and the remaining components for PVC mounting are purchased from any local hardware store. The components are cut and manufactured at the UCF Innovation lab or at a team member's home. The tests for the walls are shown in Table 48. These tests verify whether the requirements are met to ensure the safe operation of the product.

Table 48 Wall testing

Requirement	Test	Required Equipment
R.A.9	Test if the ball goes through the mesh thrown at different starting speeds and locations	Ball
R.A.9	Test if the ball can roll underneath the wall	Ball
R.A.9	Test if the robot can push through the wall siding	Robot

4.3 Court

The court component involves the actual floor of the arena that the robots drive around on. The flooring is easily transportable and is able to attach to the frame described in section 4.1 Frame. The floor paneling lays flat on the frame and contains any required basketball court markings. This is because the ball will roll around without input force and end up in a hotspot on the court. Additionally, the basketball court markings are used as a way to ensure the court is placed together properly such that computer vision remains consistent between teardowns. The floor material has a coefficient of friction high enough that the robot can consistently traverse the court without fear of slipping in the driven directions. If the wheels cannot grip properly on the court, the robot will not move, or the holonomic motions is very inconsistent, leading to a poor player experience. The court is intended to mimic a half-size basketball court.

4.3.1 Research

4.3.1a Laminate

Laminate flooring is a low-cost portable option for the court of the arena. Laminate flooring comes in many different color variations as well, which is helpful in choosing a color that works well with the computer vision tracking program. Laminate flooring generally comes in lengths of around 48 inches and widths of around 8 inches per plank. Each laminate flooring piece connects in a puzzle piece locking manner, when locked together the flooring has little to no bumps, groves,

or creases. Additionally, as long as the frame holding the flooring up is level, the laminate flooring will also be level when locked together. Laminate flooring is also lightweight, about three pounds per flooring plank. For the entire arena to be covered, 4x5 feet of space, seven planks are needed. Eight to nine planks will come in one package of laminate flooring, so only one package would need to be purchased. Each package depending on color, brand, and thickness will range in price from $12 to $20 making this a very inexpensive choice for the court even if choosing the highest priced options.

4.3.1b Metal

An aluminum court would in theory be a great choice as metal is generally flat with no impurities in the surface that would cause bumps in the court and is lightweight. However, in practice it would depend on the thickness of the aluminum and how we transport, cut, and mount it. A thinner aluminum like 0.032 inches would be ideal for low weight as a 4x5 foot sheet would weight about 10 pounds. A thinner sheet though would be flimsy and need a solid frame below for support. If not, enough framing support is under a thin aluminum sheet it will start to develop waves in the metal, once the waves start to develop it is almost impossible to get the metal to be perfectly flat again. To fix this problem a thicker sheet of aluminum could be used, something like 3/16 of an inch. At this thickness the aluminum would not need much framing to support it and maintain its surface through transport. An aluminum sheet 4x5 feet at 3/16” thick will weight approximately 36 pounds, clearly a drastic increase in the weight of the arena. Regardless of the thickness of the aluminum sheet it will need to be cut in half either lengthwise or widthwise to make it portable. Because the sheet is cut in half it will have to be rejoined together when placed on the arena frame, this can be accomplished by either laying the two sheets down next to each other and hoping they don’t move or adding hinges to hold the two halves together. Whichever idea is chosen will create a small gap or possible difference in height between the two pieces which will have to be fixed somehow to make the court completely level again. Additionally, aluminum would be the most expensive material to create the court. From online quotes for a 4x8 foot sheet of 0.032-inch-thick aluminum sheets it would cost $109 and go up to $398 for an equal sized 3/16 of an inch-thick sheet. This would clearly break the budget for the entire arena assembly.

4.3.1c Particle Board

Particle board is an alternative to plywood. There are different types of particle board and the type chosen to discuss here is OSB, oriented strand board, as it is the lowest cost while maintaining uniform construction and rigidity. OSB comes in a variety of sizes and can be bought and cut such that the whole arena, 4x5 feet, could be covered by only one piece of OSB. As one solid piece OSB is very sturdy and if bought at the correct thickness would not need any framing underneath to keep it level. While it sounds great to only have one piece for the entire floor this project has portability as a restriction and therefore a 4x5 foot sheet of OSB would

not be portable. Thus, the sheet of OSB would need to be cut, at the least, in half to make either two 2x5 halves or two 2.5x4 halves. This half cut would be a detriment to the rigidity, levelness, and ease of the OSB sheet as joining the two halves together would almost certainly add a slight bump to the middle joint and add hardware to connect the two halves. Additionally, the longer and thinner width of the sheet the more likely the sheet is to start bowing thus increasing the need for frame supports or risking the flooring to be unlevel. OSB can easily be painted as well to any color and design that would work well with the computer vision program. However, OSB and all particle board, generally does not look very professional or sleek, even when painted. Lastly, for the amount of OSB that would be needed for this arena the cost would be around $8, which is clearly the lowest cost of any of the materials researched.

4.3.2 Design

The final design of the court uses the laminate flooring material because it was the best combination of lightweight, portability, cost, presentability, and would maintain a flat level surface after multiple instances of being taken apart and put back together. It requires seven laminate flooring planks to cover the 4x5 foot area. Each plank is locked into the previous plank by inserting the protruding locking plank side into the docking side of the previous plank at an angle and then pushing in and down to lock the two planks together. A diagram of locking two planks together is in Figure 39 Plank placement below.

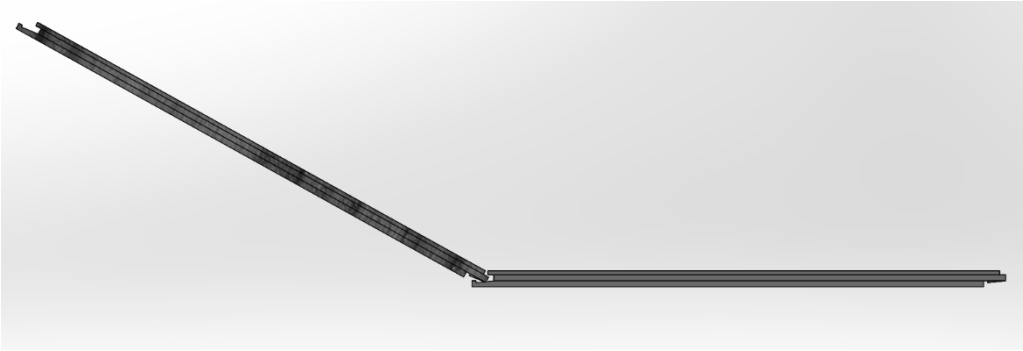


Figure 39 Plank placement

Since each plank is 8.03 x 47.94 inches, seven planks cover an area of 56.21 x 47.94 inches with no cutting of the planks involved. The court is inserted on the arena frame one plank at a time with the lengthwise side parallel with the basket side frame wall. Figure 40 and Figure 41 detail the specific dimensions of a single piece of the laminate flooring, and the full arena floor once assembled. As the dimensions for the frame are roughly 4'x5', the floor fits inside it.

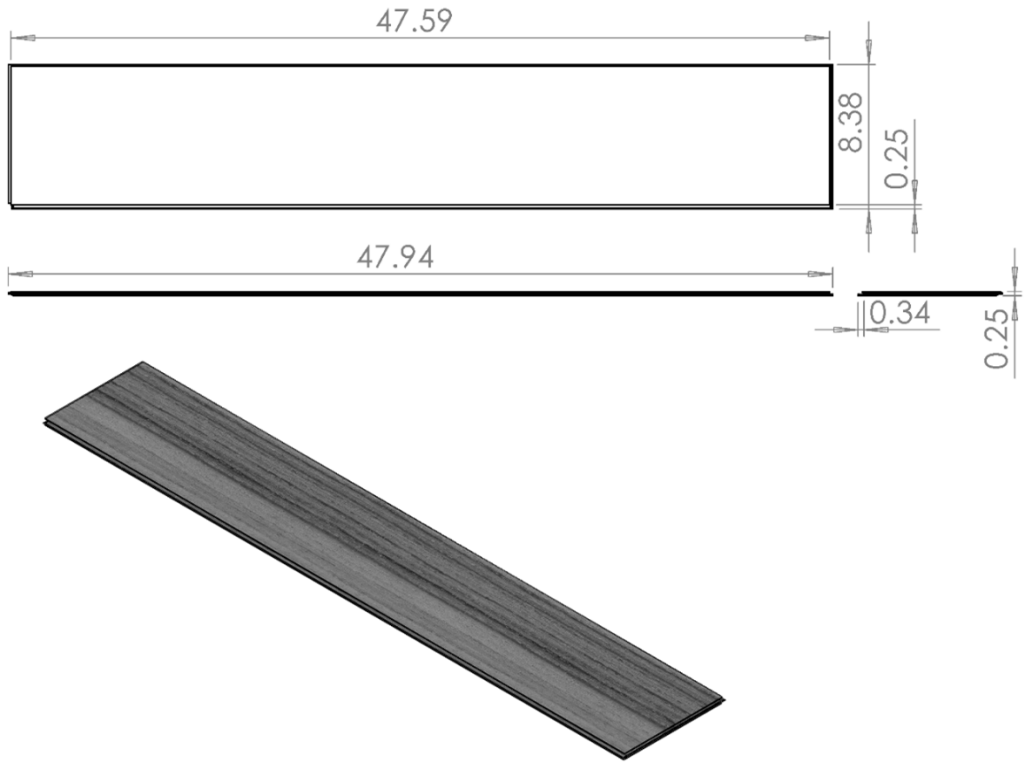


Figure 40 Plank drawing

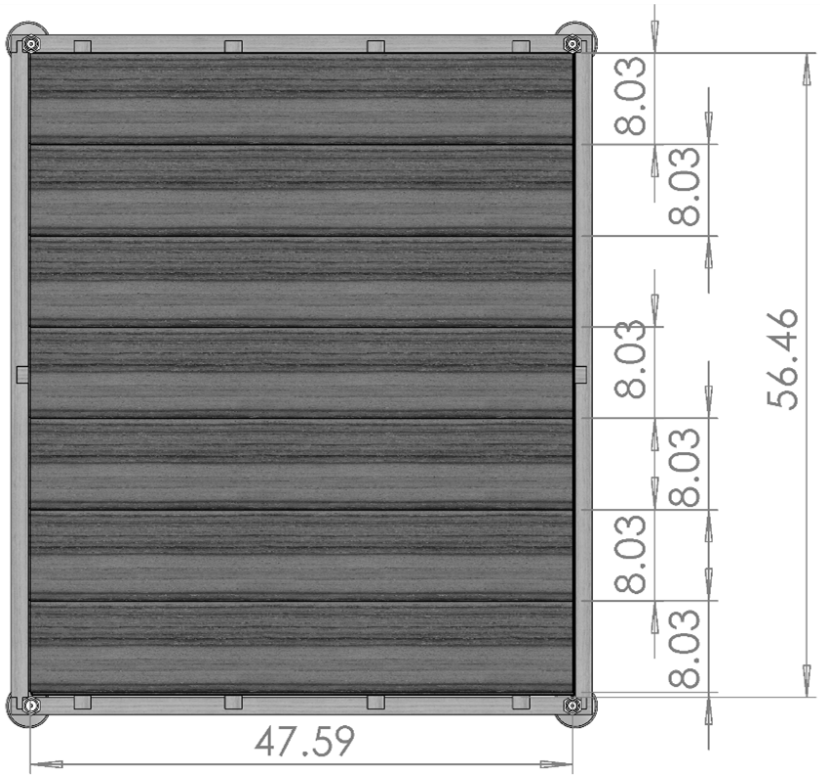


Figure 41 Court Floor panels

With all planks of the court together and aligned evenly the court markings are drawn. The basic court markings are general professional basketball court markings spaced and drawn to scale on this court. These basic court markings include the middle division line, drawn to separate the five-foot side length in two. The center circle, where the ball and both robots are located at the beginning of a game. A semi-circle free throw line for each basket. Finally, the free throw lane is drawn which is a rectangle and semi-circle that touches from the free throw line to the basket wall. All court markings are shown and labeled in Figure 42 below.

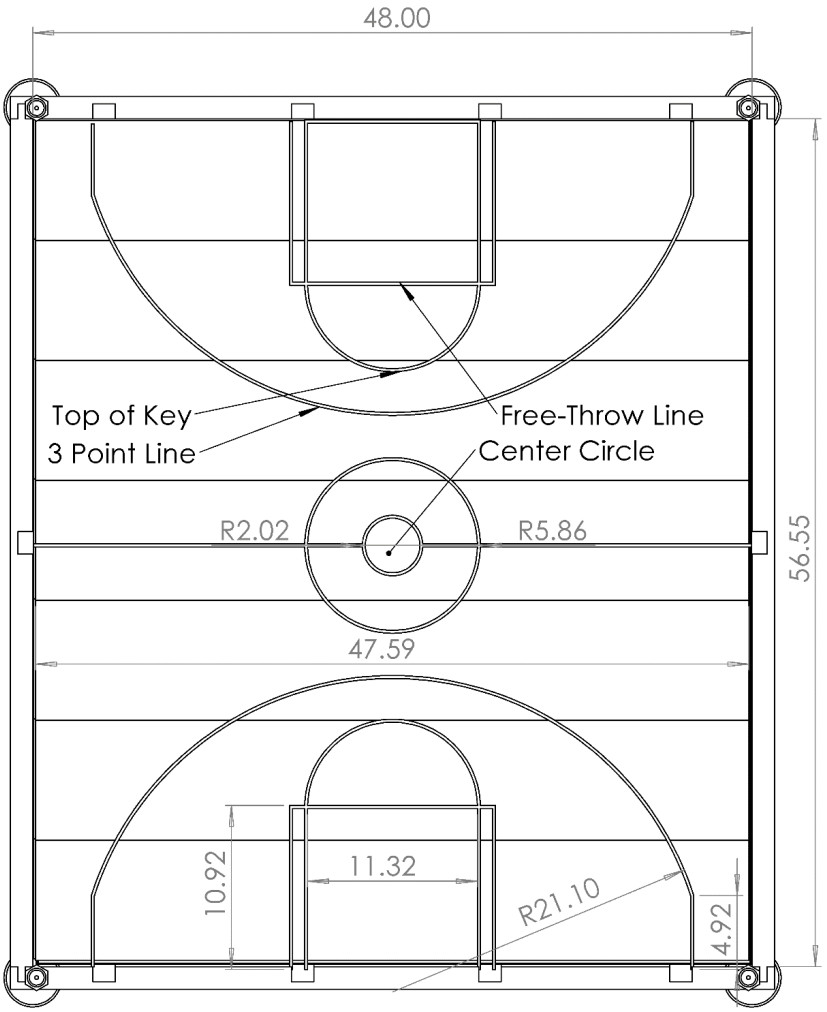


Figure 42 Court markings

4.3.3 Prototyping and Testing

The court is prototyped simply by purchasing a set of floor panels and marking out the court with a marker or tape. The arena frame must be completed to adequately install the panels. The tests are shown in Table 49.

Table 49 Court Testing

Requirement	Test	Required Equipment
R.A.7	Determine how level the flooring is in different conditions	Level
R.A.10	Determine if the court markings go back into the same place each time	Camera
R.R.B.3	Determine if the wheels slip on the floor	Wheel
R.A.7	Determine if the court lays flat inside of the frame	Level
R.A.9	Determine if the chosen wheel can roll over the frame wall	Wheel

4.4 Ball

The ball for this project represents a full-size basketball. However, it is much smaller scale and is throwable by a small-size robot. The ball is not heavily affected by aerodynamic forces to ensure repeatability. That is, the ball is not so light that a small gust of wind would affect its motion. Aerodynamic drag is expected and is utilized to gain lift based on the amount of spin on the ball. It also bounces on the court but not all over the arena from a single throw. This is to prevent the ball from being too difficult to pick up and to prevent the ball from landing on a portion of the robot that it gets stuck on. The ball is nearly spherical so that it has consistent rolling and launching. It is not deformable to the point that a force on the ball causes a permanent dent.

4.4.1 Research

4.4.1a Ping Pong Ball

Ping pong balls are 40mm in diameter and weigh about 2.7 grams. They are made from a thin plastic shell that is made of a material to meet a required bounce standard. The standard states that the ball should bounce “25 cm when dropped from 30.5cm.” This bounce is very significant and could lead to significant issues with the robot collecting the ball. However, the ball is very light and could be launched very easily.

4.4.1b Small Tennis Ball

A typical tennis ball is a bit too large for the scale of the robot. However, there are much smaller-scale tennis balls that exist for pets. This introduces a small difficulty as there are not standards related to the size and material. Thus, additional research must be conducted after the ball is picked and purchased because the material properties could differ from the documentation provided. The typical size for these tennis balls are about 1.5” in diameter, a perfect size for the scale of the

robot. Virtually any tennis ball of this size can be utilized, and there are hundreds of options that are offered in a variety of colors, themes, and prices.

4.4.2 Design

The small Tennis ball is chosen for this project. It is a small ball that has some weight and grip on it, and it has a fair amount of grippy material covering the rubber ball. The tennis ball chosen comes in a sports pack from PetSmart that contains one basketball themed ball. This provides a color with sharp contrast to the court so that it can be tracked more easily by the computer vision software. Additionally, it fits the theming of the game. The chosen ball can be seen in Figure 43.

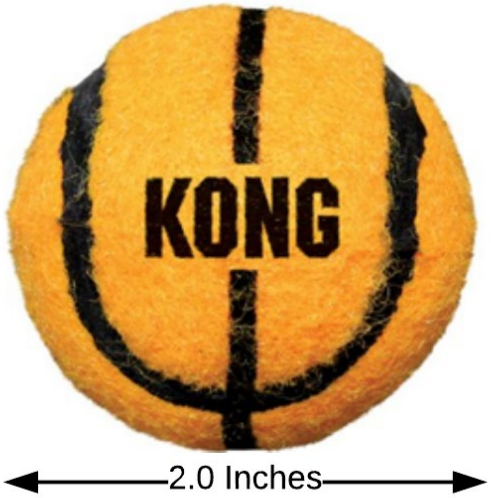


Figure 43 KONG basketball tennis ball chosen for this project
Permission from KONG in Figure 73

4.4.3 Prototyping and Testing

The ball is purchased from any local pet store. The tests for the ball are shown in Table 50. The ball is test thrown just by hand or a prototyped launcher to verify whether or not aerodynamic forces dramatically affect the balls flight path.

Table 50 Ball Tests

Requirement	Test	Required Equipment
R.A.3	Verify Ball size	Calipers
R.A.11	Verify ball weight	Scale
R.A.12	Test throw with different conditions	Tape measure

4.5 Hoop

There is one basketball hoop located on one short side of the arena. The hoop is mounted and is set to a diameter that is 2.5 times that of the ball for the launcher to remain accurate under all conditions. Each time a basket is made, the score for the game is updated, thus the hoop senses when a ball makes it all the way through. It is possible that the ball goes halfway in and pops out, so the sensor is designed such that it is resilient to false positives (I.E debouncing). The hoop structure maintains the appearance of a basketball hoop including a backboard, a rim, and a net. Each of the pieces remain sturdy when the ball inevitably misses and hits the structure. It is designed in such a way that improves accuracy. The hoop size can increase or decrease based on robot performance, and the backboard is angled in such a way that increases accuracy. The hoop can be broken into 4 sub-components: ring, post, backboard and sensor. The ring is the actual loop that the ball falls through, the post is the mounting interface for the backboard and ring, the backboard is the solid face that the ball can bounce in from, and the sensor determines when a goal is made.

4.5.1 Research

4.5.1a 3D Print

The hoop can be designed in SolidWorks and 3D printed in PLA or ABS plastic. This allows for easy integration between the hoop, hoop frame, and sensor technology by giving full control over the size, shape, and design of the hoop. 3D printers are readily available at UCF or by team-members with a variety of bed-sizes and printable materials. Thus, the actual cost of the print depends directly on the amount of print material required, and whether or not the design can be printed on a particular printer. However, the major disadvantage of the 3D print design is that the printer may not print the exact size or shape that is designed. It is common for prints to warp, bend, or shrink in the process of printing. Further, the strength of the design strictly depends on the material used and printing properties used such as infill density and infill pattern. Printing larger objects can also take up to days long which may affect the viability of the process.

4.5.2b Metal

A simple metal hoop can be utilized to fulfill the requirements for the hoop. Any metal material such as aluminum or steel can be utilized to form a ring. Additional hardware for mounting the ring, sensor, and post is necessary. Metal is sturdier than the 3D prints even at smaller sizes, so it is more resilient to impacts than the 3D prints regardless of diameter. The strength of the hoop design depends mostly on the interface between the ring and the post, as most of the force of an impact will go into a moment about the interface. It is most likely that the ring would bend downwards to the post upon impact.

4.5.1c Infrared Gate (Break Beam Sensor)

An infrared Gate utilizes Infrared light transmitter and receiver to determine when an obstacle is placed in the path between the transmitter and receiver. When the object blocks the light, the value of the receiver changes and that change is interpreted as a pulse by the microcontroller. The length of the pulse indicates how long the object has blocked the gate. Ultimately this shows whether or not the object actively passed through the hoop without bouncing out. These devices are relatively low cost and easy to set up. They are also contactless meaning they will not interfere with the object passing through the gate. Some key factors in determining the practicality of a particular gate is whether or not the beam can travel far enough to reach the receiver within the hoop, the width of the beam so that if the ball is not perfectly center the beam will still be broken, and the resilience to noise of the sensor. These sensors are dramatically affected by the amount of ambient light in a scene, thus outdoor use may affect performance.

4.5.1d Ultrasonic

An Ultrasonic sensor utilizes sound to determine the distance to objects. This sensor is like the IR gate in that it can detect when an obstacle passes in front of it by constantly determining the distance to a known plate on the opposite side of the hoop. Again, this can be interpreted as a pulse by the microprocessor and an appropriate response to the pulse can be executed. These sensors are more expensive and more difficult to work with than the Infrared Gate despite giving the same advantages. This sensor is not affected by ambient light.

4.5.1e Limit Switch

A limit switch can be utilized to detect if a ball has passed by opening/closing a digital circuit when interacted with. The major advantage of this is that the ball can be detected in a single direction from an angled switch. However, the device is contact-dependent thus it directly affects how the ball passes through the hoop. Similar to the previous devices, the digital output can be interpreted as a pulse and an appropriate response can be executed. This sensor is the most resilient to noise and environmental conditions.

4.5.2 Design

The final design is a combination of the metal and 3D print considerations. The metal ring is the sturdiest material and structure, but it suffers from poor interfacing with the post, backboard, and sensor. Thus, the metal hoop interfaces with a 3D printed bracket that integrates the sensor and backboard. The chosen sensor is the limit switch due to the ability to work in all environments, and it naturally prevents the problem of the ball bouncing from below the hoop being counted as a score. The limit switch is the Cylewet 6PC micro limit switch for snapping actions due to their small size, low expense, and ease of acquisition. It is also the cheapest

and easiest device to integrate into the rest of the project. The backboard is made out of polycarbonate to prevent warping or damage over continued use. All 3D printed parts are printed with a high infill density in ABS to maximize strength. A thin, lightweight nylon mesh is attached to the hoop rim to slow the balls trajectory through the plate, and to attempt to center the ball when passing through the plate to reduce false readings. Figure 44 shows the final hoop design with the various parts and dimensions called out appropriately.

Figure 44 Hoop & Mounting SOLIDWORKS design

4.5.3 Prototyping and Testing

The hoop can be rapidly prototyped with an available 3D printer either owned by team members or the University. The remaining parts can be purchased off the shelf at any local hardware store. The size of the hoop may change sizes in the future after testing, so it is important to have enough material to print the hoop again. Additionally, care must be taken to set the print settings such as resolution

and infill density to create the highest quality part. The tests for the hoop are shown in Table 51.

Table 51 Hoop Tests

Requirement	Test	Required Equipment
R.A.13	Test if the hoop is mounted securely and can take X force	Frame, Weight, Ball
R.A.14	Test if the ball can fall through the hoop	Ball
R.A.E.8	Test the sensors and verify accuracy	Ball
R.A.6	Test if the hoop is put into the same place each time the court is put together	Frame

4.6 Display and Sounds

The arena contains a visual display unit, like a TV, a monitor, or a tablet, that is used to relay information to the players. The display unit is capable of clearly showing the settings page for the game, like a dashboard on a video game console. This page is used to set up game modes, playback options, the score of the game, the current period out of four total periods, the remaining time for the current period, and to adjust the sounds for the game. The display unit also displays the live action 2D top down position on the court in a game engine, this is so the player can glance at important game information on the screen and not lose their place on the court and can continue driving. Showing the live location on the court is useful for spectators of the game that might not be able to see in the arena. It is also capable of displaying debugging and development information such as the live computer vision feed for any debugging that might need to happen during a game.

When searching for a display that will work for these tasks there are some features that will need to be considered. A high definition or super high definition display is ideal for spectators being able to see the information from a far distance very clearly. For this same reason a larger screen size is preferred. In addition, only widescreen monitors were evaluated so when the court, which is a rectangle, and robot location is displayed it can take up the whole screen space instead of making it smaller to fit on a square screen. The higher refresh rate on the display the better so that the game and settings will look smooth. Lower refresh rates might make the picture look choppy which can affect where the player thinks their robot is in respect to the court. Another consideration for choosing displays is how well it can display in daylight conditions. Should the game need to be played outdoors or near a window during daylight hours there may be too much ambient light to see the display. There is sounds enabled with the game and therefore speakers, either connected to the display or separate entities that will need to be able to supply loud enough sound for both players and spectators to hear. Sound is necessary

for this project as it will supply feedback to the user as well as add an emersion element to the game. An additional feature that is taken into consideration is the ability of the speakers sounds to be mixed with tactile feedback to enable a person who is blind to enjoy the game as well. In this case the speakers would have to produce adequately loud sounds in conjunction with the tactile feedback such as announcing location and orientation on the court. Finally, the price of the display and speakers should be reasonable for a self-funded college group of four to adequately purchase.

4.6.1 Research

4.6.1a Monitor

There are two categories of monitors examined and researched, those with speakers and those with no speakers (sold separately). Regardless of which category is chosen the total price for this section of the arena should be less than 70 USD. The screen of the monitor should be no less than 18”. The display is showing a live 2D position of the robot on the rectangular court. The monitor will need to display the camera feed for debugging. For these two reasons the display chosen should be widescreen to adequately scale the rectangular arena.
Any monitor with built in speakers must have an HDMI or DisplayPort connector to be considered, as these are the two best options for showing HD video and playing sounds through one plug. DisplayPort is prioritized higher than HDMI for its superior video quality capability. The lowest refresh rate on monitors today is adequate for this project and so will not be a consideration. The weight of the display is taken into consideration as the arena must be portable.

4.6.1b Speakers

The speakers for the arena need to be loud enough for the spectators to hear the game sounds and mountable or embedded into the display for portability. The cost of the speakers will also need to be low to meet the arena display and sound budget. For non-embedded speakers there are many choices available. Generally, all non-embedded speakers are loud enough for our needs and are relatively inexpensive, starting at roughly $10.

4.6.2 Design

The final design for the display and sounds is a combination TV with embedded speakers that is capable of mounting to the frame of the arena or stood up alongside the arena. The TV is the LED-LCD Sharp LC-32LB150U model which is 32” inches in size and weighs 13.9 pounds. The Sharp LC-32LB150U monitor can display in 1920x1080 resolution at 60 Hz. This meets our restriction for displaying the game dashboard, settings, and simulated 2D view of the arena. This model TV has a 10-Watt main channel 2 speaker system. Each speaker outputs 5 Watts RMS and uses a DTS digital output. The TV comes with 2 HDMI ports of which

only one is used. The actual dimensions of this TV are 28.8 inches width by 19.3 inches in height. The TV uses a standard 120V AC power supply consuming 65 Watts when operational or 1 Watt on standby. The TV can either be mounted on a post attached to the side of the arena or can be placed on the ground in the front of the arena depending on if the arena is on a table or not.

4.6.3 Prototyping and Testing

The Sharp model LC-32LB150U is tested for both display and sound requirements. Testing for the display size, display width, outside viewability, resolution, distance viewable, and refresh rate is conducted to test display specifications. Testing for the sound distance and quality is conducted to test sound requirements. A table of the requirements, test to be conducted for those requirements and the equipment needed to test the requirements is shown in Table 52 below.

Table 52 Display and Sound Test

Requirement	Test	Required Equipment
R.A.DS.1	Measure the screen from the bottom corner to the top diagonal of the opposite corner.	Tape measure
R.A.DS.1	Is the screen size 16:9?	Windows Laptop
R.A.DS.1	Use a program and run it on the display to determine the resolution	Windows Laptop
R.A.DS.1	Use a program to test the actual refresh rate of the display.	Windows Laptop
R.A.DS.2	View the display outside in a covered area with the correct settings. Stand in the player position and observe if the screen is clearly visible.	Windows Laptop
R.A.DS.2	Turn on the display to the proper settings. Walk backwards until the display can no longer clearly be seen. Measure this distance to the display.	Windows Laptop
R.A.DS.3	Play game sounds at max volume and continue moving backward until the sound can longer clearly be heard. Measure this distance to the arena.	Windows Laptop

4.7 Camera

The camera for this project is used for computer vision to track the robots, ball, and goal. It will need to be very accurate to ascertain the exact position of the robot

in comparison to the goal so that the robot can make the goal within accuracy requirements. The camera is placed above the arena a certain distance so that it may see the entire arena, robot, and goal without moving. The camera needs to supply clear video and bright colors along with fast speed so that we may update locations in real time, as accurately as possible. It will need to determine the robot's orientation in the arena so that we may use this to turn the robot toward the goal when the gamepad's shoot button is pressed. In addition, the camera will need to be fast enough to track the ball going through the goal so that we may register a point and trigger the replay on the display. The camera will connect to the controller directly, so it must have compatible connections and firmware to be able to achieve this.

4.7.1 Research

4.7.1a Pixy2

The Pixy2 is a small camera that comes with computer vision and tracking built in making it an excellent choice if it can perform the necessary tasks adequately. The Pixy2 uses an Aptina MT9M114 image sensor capable of displaying video at 1296x976 resolution at 60 FPS, which in theory should be perfectly fine for our application. The camera has a 60-degree horizontal and 40-degree vertical field of view. With this field of view the camera would have to be mounted six feet above the arena to have a full view of the entire court. The arena is only three feet high, so a six-foot mounting height is a detriment in terms of aesthetics. Additionally, at six feet high the camera might not be able to distinguish and track the objects it needs to. The Pixy2 uses a color-based object detection algorithm that should be capable of following a ball or a shape that we design for the robots. It also has built in 20 lumen lights to keep the vision area cleanly lite at all times. The Pixy2 uses an NXP LPC4330 204MHz dual core processor with 264Kb of RAM and 2Mb of flash memory. It will consume roughly 140 mA of power with either a 5V USB input or an unregulated 6V-10V input. The Pixy2 outputs data through either a UART serial, SPI, I2C, USB, digital, or analog connection. This variation in output data connections is useful because depending on the controller we use, there may not be enough of a certain port on said controller for all items to plug into if everything uses USB or UART.

4.7.1b Logitech C920

The Logitech C920 is a wide view full HD webcam. This camera was chosen to research because one of our team members owned it, there are other possible better options to research, but to stay in budget for the arena we will attempt to use parts already owned. For documentation purposes the Logitech C920 is sold for $60. This camera can produce a full high definition resolution of 1080p at 30 fps or 720p at 60 fps. This resolution is quite adequate for computer vision and is the best of the three researched cameras. Additionally, the C920 has an autofocus feature which is good for tracking quick moving objects. The camera is also wide

view having a 78-degree horizontal and 43.3-degree vertical field of view. This means the camera can be mounted at a minimum of roughly three feet above the court in order to view the entire horizontal part (5 ft) of the court without moving and roughly five feet above the court to view the entire vertical part (4 ft) of the court. Since the vertical height needed to see the court is much higher than our goal max height two cameras might need to be used. This will need additional testing and research to conclude, but most likely two Logitech C920 webcams is necessary to see the entire court at three feet.

4.7.1c Logitech C270

The Logitech C270 is a standard HD webcam. Like the previous Logitech camera this camera was also chosen for research it because it is already owned by a team member. For documentation purposes the Logitech C270 is sold for $40. This camera can produce a high definition resolution of 720p at 30 fps. This frame rate might not be good enough to follow fast moving objects like the ball flying, but further testing is needed to discern this. In conjunction with lower ability to track quick moving objects this camera only has a fixed focus which makes the previously mentioned quick moving objects harder to track. The Logitech C270 has a field of view of 60 degrees, meaning that in order to see the whole court it will have to be mounted 1.4 feet above the court.

4.7.2 Design

For the camera design we first chose to use the Pixy2 as it simplified the computer vision object detection and tracking. However, after testing the Pixy2 it was discovered that it would not work for this project as it was incapable of detecting unmoving objects from six feet above the ground (the height needed to view the entire court) at a reasonable rate. The cameras actual video input quality was also very low, requiring many lights to make the court bright enough for even slight object detection.

The C920 camera is the camera used in the final design and was chosen for its widescreen view. It’s also the best quality resolution of the three cameras researched. The two downsides of the C920 are the use of a USB connection for power and data transfer which will take up one of the few USB slots available on the arena controller and the fact that we will have to now write the computer vision software for the camera. In order to see the court with one camera, it would have to be placed somewhere around 5 to 6 feet above the arena. A mockup of this configuration is given in Figure 45 below. In order to work around this to cut down on vertical height in the arena, the team also explored a two-camera configuration that is detailed in Section 6.4 later in the paper.

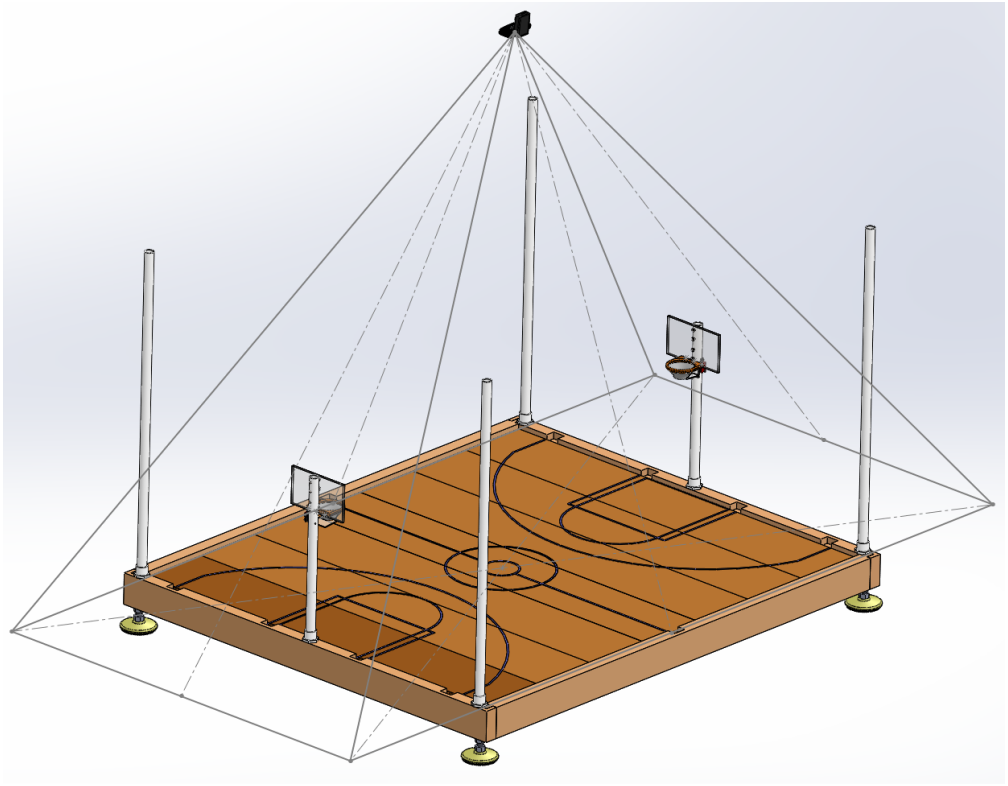


Figure 45 Camera field of view indicating area of Arena the camera can see

4.7.3 Prototyping and Testing

The Logitech C920 camera being used must be tested thoroughly before using it in the project as any malfunctions or different parameters will affect the computer vision portion of this project. Tests of the camera's clear viewable height, color, field of view, and compatibility are performed. If any test fails to meet the minimum requirement, then this camera will not work for this project and a new one must be picked and tested. Table 53 below shows the test to be performed for each requirement and the required equipment to perform the test. These tests can run in any order.

Table 53 Camera tests

Requirement	Test	Required Equipment
R.A.CV.3	Determine if the camera can view the entire field and objects	Webcam, Varying sized objects
R.A.CV.5	Determine color accuracy of objects within the field	Webcam, Varying colored objects
R.A.CV.4	Read the camera documentation and the controller documentation to ensure the parts is compatible	Camera and Arena controller documentation

4.8 Gamepad

The gamepad is the first thing a player will touch when playing this game, so it's important to choose a gamepad that will feel familiar. When choosing the gamepad our team felt that an often overlooked, but important feature is tactile feedback. Tactile feedback aids in the feeling of control over the robot and adds another level of response to the player so they feel like their driving has an impact on the game.

Our communication between the arena processor and robot is accomplished through Bluetooth. We anticipate that this communication will need to be very fast to make driving the robot feel good and reactive. Because of this we are keeping the amount of information sent over Bluetooth to the lowest amount possible and using a wired gamepad will remove information needed to be communicated over Bluetooth. The gamepad could also communicate using WIFI direct, but we have opted for wired because adding WIFI direct will add an additional module that will need to be purchased, which could break the budget requirement. Choosing a wired gamepad will also add a layer of reliability. If we use a Bluetooth gamepad and we are having problems with driving, is that a problem with our Bluetooth or our code for driving? We are eliminating the possibility of errors occurring from wireless communication.

Additionally, the gamepad should be easy to write code for and have thorough documentation. This will make working with the gamepad quick and easy and allow us to focus our efforts into other parts of the project. For these reasons we decided to pick between two popular gamepads; the Xbox One wired gamepad and the PlayStation 4 wired gamepad.

4.8.1 Research

4.8.1a Xbox One

The Xbox One wired gamepad is one of the most widely used gamepads for computer based and robot-based applications. Therefore, the Xbox One gamepad has a lot of documentation, especially for robotic applications like ours. The Xbox One gamepad was developed with comfortability in mind when holding the gamepad for long periods of time. Therefore, the gamepad fits comfortably in the hand while also allowing the user to be able to hit any button and any button combo with ease. This gamepad features ten digital buttons, a syncing button, two analog triggers, two analog sticks, and a digital D-pad. The two triggers feature independent rumble motors (Impulse triggers) that can be programmed to vibrate directionally. This rumble is useful for giving the user an in-depth experience, such as rumbling harder and harder while spinning the flywheel up to launch the ball when not in autonomous mode. The right side of the gamepad contains four of the ten digital buttons; the green 'A', red 'B', blue 'X', and yellow 'Y' buttons. These buttons are useful for main actions like 'Choose' or 'Go Back'. The left and right side also contains one of two analog sticks each, these also contain a digital button

activated when the analog stick is pressed in. Analog sticks are very important for driving and directional aspects of controlling the robot. In the center of the gamepad is two more digital buttons and the syncing button, generally used for pausing, menu, and turning the gamepad on and off. The left side of the gamepad also contains a digital D-pad generally used for choosing options quickly. Located on the shoulders of the gamepad are the two more digital buttons generally referred to as “bumpers”. Finally, the back shoulders of the gamepad each have one of two analog triggers. These triggers have the rumble feature and therefore can be used for processes that require feedback to make the game feel more natural. All together the Xbox One gamepad contains sixteen possible buttons, many more than this project should need to make it feel good to the player.

4.8.1b DualShock 4

The DualShock 4 is the gamepad used for the PlayStation 4. The DualShock 4 is not typically used in many robotics operations. The DualShock model line of gamepads has kept its design similar for many years, which could be seen as an advantage to players who have used this gamepad since the first generation, which was released well before the first-generation Xbox gamepad. The DualShock 4 is smaller gamepad compared to the Xbox One gamepad. It also contains two vibration motors, one inside the left handle and one inside the right handle. The right handle motor is smaller and less powerful than the motor on the left, this allows the vibration to vary based on what feedback the developer wants the player to feel. The DualShock 4 also incorporates a clickable two-point capacitive touch pad on the front along with motion detection through a three-axis gyroscope and accelerometer. The buttons on the DualShock 4 include two analog sticks, two analog triggers, two pressure sensitive buttons, ten digital buttons, and four directional buttons. Located on the right face of the gamepad are four of the ten digital buttons: green ‘triangle’, orange ‘circle’, blue ‘X’, and pink ‘square’. These are the main action buttons, such as ‘select’ and ‘back’. Also located on the right face is the right analog stick in addition to the fifth digital button activated by pressing the analog stick. Similarly, on the left face of the gamepad is the left analog stick and sixth digital button, again activated by pressing the analog stick inward. These analog sticks are generally used for movement, such as driving. On the left face of the gamepad is also located the four directional buttons: ‘up’, ‘down’, ‘left’, and ‘right’. These buttons are also generally used for movement tasks. On either side of the capacitive touchpad (located in the middle face) are the ‘options’ and ‘share’ buttons, which are two more of the ten digital buttons. On each side of the gamepad, located on the shoulder, lies the two pressure sensitive buttons, also referred to as “bumpers”. Lastly below each bumper on the shoulder of the gamepad are the two analog triggers, again which are usually used for performing action tasks like accelerating a car. The DualShock 4 gamepad is sold starting at $30.

4.8.2 Design

Between the two gamepads we believe the Xbox One wired gamepad has the most documentation and support as well as ease of programming, thus it is used for the final design. We are opting to use a wired controller for two reasons: lower the amount of information needed to be transmitted to the Bluetooth module and to add a layer of reliability.

The Xbox One gamepad also has the individual rumble motors on each trigger button, which will add more immersion to the game. In full autonomous shooting mode, the right bumper is used to launch the ball. When pulled, regardless of how hard, the flywheel will start spinning up which will enable the rumble feature, which will increase in intensity as the wheel spins faster and continue rumbling until the ball is launched. If autonomy is turned off the player will control the speed of the flywheel, this is done by pressing the right bumper button multiple times, the rumble in the trigger is just like in autonomous mode, but the ball will only be launched when the player hits the ‘A’ button. The ‘X’ button is used for intake and ‘A’ for launching the ball.

The left analog stick is used for main robot movement. When leaning the analog stick forward or backward the robot will move forward or backward. When leaning the analog stick left or right the robot will strafe left or right. All combinations of movement are supported as well: forward and strafe left or right, backward and strafe left or right. The right analog stick is used to rotate the robot. Moving the stick to the right rotates the robot clockwise and moving the stick to the left will rotate the robot counterclockwise.

The green ‘A’ button is used as the ‘select’ button and launching button. The red ‘B’ button is used as the ‘back’ button. The ‘menu’ button is used for pausing the video game portion of the game to view video settings, exit the game, or restart the current game mode, in addition this menu is used if the player would want to invert their movement controls. This mapping is reset back to default every time the main dashboard is accessed. The ‘view’ button is used for pausing the game. The ‘Xbox’ button turns the gamepad on and off. The left bumper is used to spin down the flywheel. The blue ‘X’ is used to grab the ball, and the yellow ‘Y’ is to open the ball grabber. The D-pad and triggers do nothing and will not have mapping.

This control mapping should feel comfortable and natural to the player, whether they play with launching autonomy or manual launching mode and regardless of the players left hand or right hand preferability. A visual aid for the controller mapping is given in Figure 46 while a quick summary of the individual controls is given in Table 54.

Table 54 Player input functions and gamepad mapping

Function	Type	# Of Axes
Forward/Backward + Strafing	Left Joystick	2
Rotation	Right Joystick	1
Launch Ball	A	1
Flywheel Speed Control	Left Bumper/Right Bumper	1
Intake	X	1
Select	A Button	1
Back	B Button	1
Player/Game Settings	Menu Button	1
Pause	View Button	1
Gamepad Power On/Off	Xbox Button	1
Open Ball Graber	Y	1

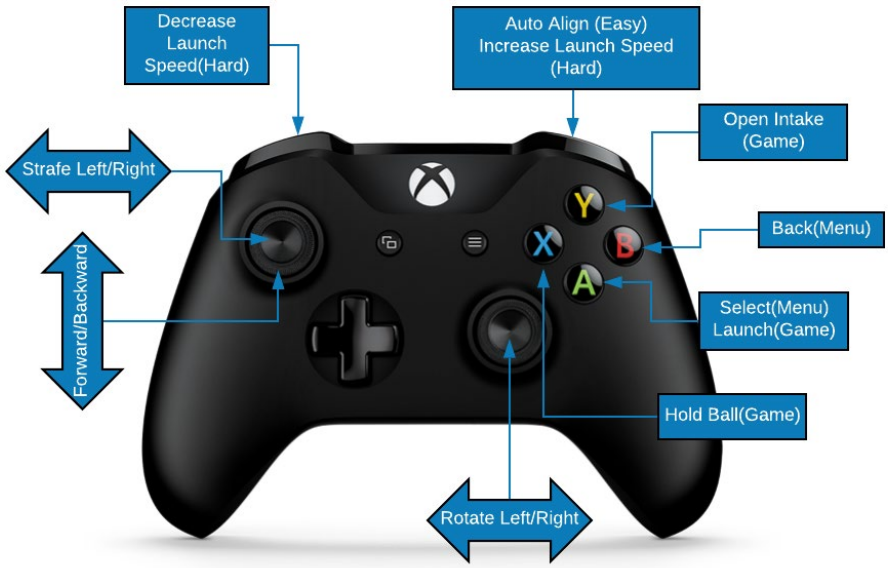


Figure 46 Gamepad control layout
Used with permission from Microsoft

4.8.3 Prototyping and Testing

To test the gamepad the following items are needed: the Xbox One wired gamepad, the robot, Bluetooth, arena controller, and video game portion need to be working and turned on. The first tests should be ran using launching autonomy, then the same tests shall be repeated for manual launching mode. Note that the right bumper button should only work in manual mode. Using the gamepad button mapping table and diagram press each button one at a time and observe that the correct function occurs. Ensure that when the triggers are pressed the rumble

function works properly and is synced with the flywheel spin up. Next, change the driving controls to inverted and observe that the robot is still moving in the correct directions. Lastly, after inverting the driving controls and certifying that they work correctly, return to the main dashboard and back into a game. The driving controls should have reset to default, test this by driving the robot and observing that the drive controls are now back to default settings. A summary of these tests is given in Table 55 below. The team will also have to test whatever configurable or custom controls that are decided upon being used to make sure they are easy to use and change on the fly.

Table 55 Gamepad tests

Requirement	Test	Required Equipment
C.S.1	Set a function to rumble the gamepad on command. Perform the command and observe the rumbling feeling of the gamepad.	Gamepad, Gamepad software, Arena controller
C.S.1	Ensure each button performs it's given task by pressing each button one at a time and observing the buttons output.	Gamepad, Gamepad software, Arena controller

4.9 LED Lights

The arena system uses computer vision to detect distance between the robot and the hoop. Based on these distances, the arena converts them into motor velocities for the robot to adjust and shoot the ball. The update rate of the computer vision system is 60Hz and therefore, the arena can perform calculations quickly. However, none of this is possible without proper illumination. This is where the LED lights play an important role.

Light Emitting Diode, or LEDs, are a common occurrence in present day. Therefore, due to such high-volume availability they are affordable. Using appropriate current limiting resistors and a microcontroller, one can turn them on or off in a timely fashion. Having a strip of them around the arena will not only illuminate the arena for computer vision, but also make it aesthetically entertaining. There are different colors of LEDs and they can be combined to form different colors by simply mixing their RGB values. Consequently, when the player makes a shot, a sensor will trigger a sequence of LED blinks and create an animation for user entertainment. Each action has an LED sequence preprogrammed into the arena. Making the shot causes the arena to turn green, a Bluetooth connection turns the arena blue whereas when pairing the arena blinks blue light. A lost connection or fault causes the arena to turn red.

LEDs tend to draw a lot of current to shine brighter and therefore, based on the kind used, their current and voltage requirements are used to calculate current

limiting resistors. They surely can add an entertainment value to the project and make it more professional. There are multiple LED technologies available which are discussed in section 4.9.1 of this document.

4.9.1 Research

4.9.1a Adafruit NeoPixel

NeoPixels are LED strips with individually addressable RGB LEDs. They must be programmed and run through a microcontroller such as Raspberry Pi. They have a refresh rate of around 400 Hz. The limiting factors to the number of NeoPixel strips able to be chained together are RAM and power to run, and time to process data from the microcontroller hosting them. There is no set limit on the number of NeoPixels that can be chained together, however they will eventually need more resources than the computer is able to provide for them to be able to function properly. Additionally, a Task Scheduler library is used to program different animations and Boolean flags are triggered based on ASCII input from Jetson Nano to the Arduino board. [34]

4.9.1b Traditional LEDs

Traditional LEDs are usually just a strip of LEDs controlled by one controller. This is the biggest downside of traditional LEDs because they typically must all be the same color at the same time which limits the number of custom configurations. Traditional LEDs are also usually cheaper than other “smart” LEDs such as the NeoPixel discussed in Section 4.9.1a. There are also more options for traditional LEDs than if the team was to use a smarter LED strip such as the Adafruit NeoPixel.

4.9.2 Design

The team is using NeoPixels as they allow for the customization needed for the arena. The ability to individually control the LEDs on the strip is invaluable to display different status signals such as a malfunction or “goal made” to the players. This feature can also be used for testing integration between all the objects involved with the arena. The design will use 4 NeoPixel Strips around the walls of the arena that is connected to our microcontroller brain and receive signals from the arena about what to display. This will also help to keep a consistent lighting amount on the arena for computer vision purposes.

4.9.3 Prototyping and Testing

The tests for the LEDs consist mainly of just checking if they integrate well with the controller and arena and is performed according to Table 56. The LEDs will also need to be tested to make sure that they are sufficiently bright and responsive to

the controller. If they do not respond in a timely manner without latency, they will have to be replaced.

Table 56 LED Lights tests

Requirement	Tests	Required Equipment
R.A.16	Determine the voltage used by the LEDs to power on	Multimeter, Power Supply, Jetson Nano
R.A.16	Determine the current draw by the LEDs and contain their brightness using Ohm's Law	Multimeter, Calculator
R.A.16	Determine the animation sequence via timers	Oscilloscope, Serial Monitor
R.A.16	Determine logical value of each LED at a certain instance for debugging	Logic Analyzer, Multimeter, Oscilloscope

4.10 Controller

The controller for this project strongly depends on the computational power that is required by the various components. This controller performs calculations for computer vision, Bluetooth communication from arena to robot, calculations for robot location, calculations for force to launch the ball, and be able to show video on the display using the game engine. In addition, it will also control any LED lights that are installed in the arena. The controller will need to be capable of running an operating system to allow the use of certain software, like the computer vision and game engine software. For this reason, the controller will need to be powerful, but also compatible with the other parts chosen for the project. Lastly, the controller will need to have the proper slots for additional hardware that will have to be interfaced with.

4.10.1 Research

4.10.1a Raspberry Pi 3 Model B+

At the time of writing this paper the Raspberry Pi 4 was released. The Raspberry Pi 4 is not being considered due to it being sold out. Instead, the older generation Raspberry Pi 3 B+ is researched. The Raspberry Pi line of controllers are generally used for the purpose of robotics and artificial intelligence applications. The Raspberry Pi 3 Model B+ uses a 1.4 GHz Broadcom BCM2837B0 Cortex A53 64-bit Arm8 processor, it has 1 GB of SDRAM. This processor gives the Raspberry Pi 3 Model B+ the ability to perform 21.4 billion floating point operations per second, or GFLOPs. With only 1 GB of RAM though this controller will have a hard time running an operating system, something integral to this project, as it is needed to run the computer vision. The Raspberry Pi has wireless LAN, Bluetooth 4.2, and

Bluetooth low energy capabilities. The Bluetooth 4.2 is what is used to communicate with the robot and is a very important feature. It also has a HDMI port and a display serial interface port for video implementation, important for connecting the display and sound. The Raspberry Pi 3 Model B+ also requires a micro SD card for loading an operating system and storing data, adding an additional expense to this controller. Additional ports on this controller include: Extended 40-pin GPIO header, CSI camera, four USB 2.0, and a 4-pole stereo output and composite video. The controller is powered by 5V/2.5A DC power, a standard amount for a controller of this type. The Raspberry Pi 3 Model B+ sells for $35. The Raspberry Pi 3 B+ was originally suggested because of its built-in Bluetooth capability, low price, and processing power, however, due to the inclusion of the game engine video display, we believe more RAM and an onboard GPU is necessary for smooth video and processing.

4.10.1b Jetson Nano

The Jetson Nano is a new addition to the Jetson family at the time of writing this paper. Typically, Jetson controllers are used for artificial intelligence and computer learning applications and so are a natural consideration for this project since highly accurate computer vision is necessary. Since the Jetson line of products is built for artificial intelligence and robotics it makes sense that the Jetson Nano is optimized for machine learning, this generally means a stronger processor, more RAM, and a stronger GPU. The Jetson Nano uses a 64-bit Quad core Cortex A58 CPU with 4 GB of RAM. This processor will give the Jetson Nano the ability to perform 472 billion floating point operations per second, or GFLOPs. Additionally, the 4 GB of RAM will make the Jetson Nano good at running an operating system and performing many computations, such is needed for accurate computer vision. The Jetson Nano is very capable of video processing, because of its onboard GPU, the 128 core Nvidia Maxwell. It can encode, decode, and display 4k videos. This GPU can run up to eight 1080p video feeds at 30fps or eighteen 720p video feeds at 30fps at the same time and is still able to run any video processing algorithms and processes needed. This feature is very important since this project will require at least two cameras running in parallel in either 1080p at 30fps or 720p at 60fps.

The Jetson Nano also has an M.2 Key E slot for mounting a wireless or Bluetooth card, which would be used to communicate with the robot. However, the price of a Bluetooth card must be considered as well since Bluetooth is not built in and requires an additional purchase. Also, the Jetson Nano, being a newer controller on the market, does not support all Bluetooth cards and might not have the drivers required for the card purchased. This means the drivers would have to be written by the team, adding additional responsibilities to an already large list. The Jetson Nano runs the Linux4Tegra operating system. This operating system is Jetsons adaptation of Ubuntu 18.04. This controller also requires a micro SD card to store the operating system and all other information. For video ports the Jetson Nano contains one HDMI and one display port capable of 4k quality. Additionally, the Jetson Nano has four USB 3.0 ports, a great feature since the cameras will most

likely be connected via USB. The Jetson Nano runs on a 5V-2A micro USB connection or a 5V-4A DC Barrell jack adapter if more power is needed to keep the controller powered smoothly. Additional connections include: 40 pin GPIO, I2C, I2S, SPI, and UART. The Jetson Nano will cost around $99 at the time of writing this paper.

4.10.2 Design

The final design for the controller came down to multiple aspects. The controller needed to be capable of running two cameras in parallel and operating on the images as quickly and accurately as possible. This will generally be comparable in GPU strength, processor power, and amount of RAM. The controller has onboard, or be capable of connecting, a Bluetooth controller since Bluetooth is used to communicate with the robot. Additionally, the controller has enough USB ports to connect the necessary hardware, two USB webcams and up to two gamepads. Since the USB devices are cameras and gamepads the USB ports have the best possible data transfer speeds, which between the 2.0 and 3.0 standard the 3.0 standard is preferred. Also necessary are ports for the display and sound. A HDMI or Display port is preferable as they are capable of sending both display and sound data through only one connection. Lastly, the price is, of course, an important thing to consider as well. It is easiest to show a table comparison of the Raspberry Pi 3 Model B+ and the Jetson Nano to show the important features of each and decide with the mentioned features in mind. Refer to Table 57 below for this comparison.

Table 57 Controller Comparison

Feature	Raspberry Pi 3 Model B+	Jetson Nano
CPU	1.4 GHz 64-bit Quad-Core ARM Cortex A53	1.4 GHz 64-bit Quad-Core ARM Cortex A57 MPCore
GPU	Broadcom VideoCore IV	128-Core Nvidia Maxwell
RAM	1 GB LPDDR2 SDRAM	4GB LPDDR4
Operation Performance	21.4 GFLOPs	472 GFLOPs
Wireless	Dual-band 802.11ac wireless LAN, Bluetooth 4.2 onboard	M.2 Key E Slot (None onboard)
USB Ports	4x USB 2.0	4x USB 3.0
Video Ports	HDMI, DSI	HDMI, Display Port
Price($)	35	99

It is evident after examining Table 57 that the Jetson Nano is superior to the Raspberry Pi 3 Model B+, but also costs significantly more. For the final design of this project a Jetson Nano is used. The Jetson Nano was chosen almost entirely because the Raspberry Pi 3 Model B+ was not good at running an operating system, had such little RAM, and not a great GPU. This project is heavily reliant on reading information a camera, detecting, tracking, and calculating locations and distances from objects and therefore will require a controller that can support this

feature. Of which the Jetson Nano would be capable of because of its USB 3.0 slots, larger RAM size, better GPU, and vastly superior operational performance. While the Raspberry Pi 3 Model B+ does have onboard Bluetooth already included, it would save a lot of time and money to not implement it ourselves, but it has been concluded that the Raspberry Pi 3 Model B+ wouldn't be capable of supporting this project.

4.10.3 Prototyping and Testing

The test processes for the controller are listed below in Table 58. They mainly consist of determining that the controller is sending and receiving data on time and not experiencing latency. The controller also needs to be able to fit inside the power requirements set aside by the team.

Table 58 Controller tests

Requirement	Test	Required Equipment
R.A.E.3	Determine if the controller can communicate over Bluetooth at the correct rate	Part documentation
R.A.E.5	Determine if there is latency in video playback	Controller Monitoring Software
R.A.E.6	Check that the correct number of slots and slot type is included in the controller.	Controller documentation
R.A.E.7	Determine output power is sufficient	Controller documentation

4.11 Communication

The communication subsystem allows the Arena to send commands to the robot. To accomplish this, the Arena must have a communication system on board and send data over a wireless link. The communication subsystem needs to have a data update frequency of 30Hz at the minimum. Failure to do so can cause latency in sending commands to the robot and thereby an overall latency in the system's response. This latency hinders the robot from shooting successfully 75% of the time as per the requirements.

The Arena is a master to all the robots in it. It takes in commands from the controller and the computer vision system and combines them into a packet in a systematic way. This packet is sent to the robot(s) via a Wireless link. The packet is designed by the team and passes multiple checks to ensure accurate transmission of data. The robot is a slave device to the arena that will receive data over the radio to perform its actions. The implemented communications protocol will also allow the robot to send its sensor data back to the arena for monitoring and debugging purposes. This data is shown by the Arena on a screen to give users more

information regarding their robot. These stats could include current motor velocities, battery status, communication link status etc.

The communication system is low energy because it will allow the onboard computer to use its resources for high powered activities such as driving the LEDs, powering the controller, powering the webcam(s), displaying contents on a TV screen using HDMI and running the game engine.

4.11.1 Research

4.11.1a Bluetooth

A detailed study on the use of Bluetooth as a communication system is conducted in section 3.6.1a Bluetooth. For Arena Bluetooth to work well with the Robot, it has to be the same profile and version as the Robot's Bluetooth. Therefore, a research for Bluetooth v4.2 modules were conducted for the Arena. The onboard computer for the Arena is a Jetson Nano which has a special M.2 Key E interface to attach a Wi-Fi and/or Bluetooth adapter. There are only a handful of modules compatible with Nano and therefore, choosing one was not a hard decision. The modules available for Jetson Nano provide a range of Bluetooth profile options unlike Robot's Bluetooth that come with dedicated profiles in their firmware

An alternative option is to use a USB adapter for Bluetooth however, due to the usage of one or more web cameras, and one or more D-pad controllers using a USB for Bluetooth appears to be a waste of resources. Therefore, due to efficiency using the M.2 connector for the Wi-Fi card is highly likely. It also provides a higher quality and reliability of communication signal as per suggested on NVIDIA's forums [35].

One of the most trusted and used module that provides both Bluetooth and Wi-Fi capabilities to Jetson Nano is Intel 8265NGW [36]. The Intel 8256 supports 802.11 ac Wi-Fi dual band that can deliver speeds up to 867 Mbps and also has hardware support for Bluetooth v4.2. Since, Nano runs on Linux operating system, there are myriad of kernel modules that can be used to use a specific Bluetooth profile for Nano. There are two recommended antennas for this radio module, one is a 6dBi RP-SMA Dual Band antenna and the other is a Molex antenna with 3.3dBi of gain. In terms of cost, the Molex antenna is 78% cheaper than the RP-SMA Dual Band antenna and also weight less. Additionally, the arena is only four feet by five feet in dimension and so, using a 3.3dBi will cause no harm to the communication strength. However, if multiple robots were to be supported the 6dBi antenna is ideal.

To program Bluetooth for development the BlueZ stack is the most reliable stack. The hcitools allow sending data to a connected device whereas hcidump is used to receive Bluetooth packets. The BlueZ stack is an open source codebase which is available on git and can be used for development purposes. Additionally, the

packages can be installed using the “apt-get” utility in Linux. This stack has support for various Bluetooth profiles however, to be able to communicate with Arduino, Serial Port Profile has to be used. The time taken to send a certain amount of raw data bytes are specified in section 3.6.1a Bluetooth.

4.11.1b Wi-Fi-Direct

A detailed explanation of advantages and disadvantages of using Wi-Fi direct are presented in section 3.6.1b Wi-Fi Direct. To implement Wi-Fi Direct in Linux the same adapter that is used for Bluetooth, Intel 8265NGW, is used. Wi-Fi Direct is simply a software manipulation on the existing Wi-Fi hardware, therefore, for Linux Wi-Fi direct support wpa_supplicant package is used [37]. This package uses EAPOL (IEEE 802.1x) standard and acts as a middleware between the hardware and application layer. To use this efficiently, the functions provided in the library’s API should be sufficient.

The wpa_supplicant acts a control interface to the Wi-Fi hardware. The external program can use C or C++ to interface with it to suit their needs. The wpa_supplicant contains hostapd which includes the IEEE 802.11 access point management, EAP server, and RADIUS authentication server functionality. It can be built with various configurations which are specified in the documentation. The software uses subscribe-event software implementation where the client subscribes to host Wi-Fi and the host acts as the access point to deliver and accept information from the client.

For the project implementation, the Arena acts as an access point to which the Robot computer connects to. The robot’s Atmel uses ESP8266 chip to establish Wi-Fi connection with the adapter. After setting up the Wi-Fi the layer of TCP/IP can also be used in conjunction with wpa_supplicant to provide code modularity to ease the development process. However, due to high energy consumption it is unlikely that a Wi-Fi Direct approach is used for Arena-Robot Communication.

4.11.2 Design

The Arena acts as the master device to the robot(s) which send Robot the commands to perform certain activities. The arena uses computer vision to accept a stream of raw data bytes which are stored in a data structure and used by the CV script to decipher and get distance between the robot and the Arena. This distance is then used as an index to a look up table or hash table which outputs a motor velocity. This motor velocity is used to create a Bluetooth packet and sent to the Arena using rfcomm Serial Port Protocol Profile built into Linux’s kernel and used by the BlueZ stack to send and receive signals.

The data packet consists of 14 bytes of overhead along with the size of the data. Bluetooth v4.2 allows sending data up to 251 Bytes which is specified in Sparkfun’s tutorial on basics of Bluetooth [16]. The data packet as mentioned in 4.11.1a

Bluetooth 14 bytes of packet overhead. The data packet consists of motor velocities for driving purposes, the flywheel velocity command for shooting the ball, command to open or close the gate to let the ball in or let it out of the robot, and a state byte. There are multiple ways to frame the data packet. The indices can be used as an inherent indication of the motor and the value at that index can be used as the velocity of the motor, whereas another option is to use an array with first index specifying the motor whereas the second index specifying its velocity. There are pros and cons to both implementations. The first implementation allows using less data bytes to send more information whereas the second implementation secures the packet if the data bytes were to arrive out of order. The time taken by the first implementation is calculated to be ~6 ms and for the second implementation to be ~7 ms. As it can be seen the first implantation uses less time and less bytes for communication, it is used in the design. Since both the arena and the robot are designed in-house, loss of data packets is less of an issue as the communication field is quite small. The packet design can be seen in Figure 47.

LSB MSB

Preamble	Access Address	Header	Payload (Data) [Up to 251 Bytes]	MIC	CRC
1 Byte	4 Bytes	2 Bytes	14 - 18 Bytes	4 Bytes	3 Bytes

Figure 47: Bluetooth packet sent by the Arena

4.11.3 Prototyping and Testing

The Bluetooth communication is tested using RSSI readings and serial monitor. The hcitool library consist of an RSSI command that outputs the signal strength of the established Bluetooth connection. A signal strength of -30dBm or more is considered according metageek documentation and is used as a reference to determine Arena-Robot communication signal strength [38].

To check the validity of data, serial monitor on Arduino IDE, and Command Line on Ubuntu is used. The data bytes transmitted are ascii characters and because ASCII is a global standard, it makes deciphering the code easier. A good software test is to run a loop that transmits packets continuously and the receiving end sends an acknowledgement in return to validate that the data has been received. A counter should be incremented every time a packet is received and sent to track the number of packets exchanged and test if any packets were lost during transmission. These tests and requirements are summarized in Table 59.

Table 59 Communication tests

Requirement	Test	Required Equipment
R.A.E.3	Determine that the communication system successfully form a connection with the slave devices	Bluetooth module, Bluetooth App
R.A.E.3	Determine that the commination system successfully reads the packet generated by the Arena from a buffer	Serial Monitor, Bluetooth Module
R.A.E.3	Determine that the communication system successfully transmits the packet	Oscilloscope, Bluetooth Module, Bluetooth App
R.A.E.3	Determine that the communication system successfully receives a packet	Oscilloscope, Bluetooth Module, Bluetooth App
R.A.E.1	Determine that the communication system saves system resources by going to sleep when no communication is required	Multimeter, Serial Monitor, Bluetooth Module

4.12 Electrical System

The Electrical System subsection defines how the Arena components are wired and work together. Supplying power directly to a microcontroller from the power outlet can be dangerous. Due to this proper AC to DC conversion is required meeting the operation requirements of the Microcontroller and Arena peripherals. The AC-DC converter needs to be energy efficient and provide at least 30 to 40 watts of power for the entire electrical network of the Arena. One can design such a converter with enough time and resources. However, they also act as constraints for our purpose due to which an AC-DC converter is purchased instead. Overall, time vs cost analysis was done by the team to arrive at this conclusion.

Typically, microcontrollers run on 3.3 – 5 V Transistor-Transistor Logic (TTL) and so, the AC-DC converter needs to be able to step the power down to that voltage. Additionally, the NeoPixel LEDs, which are used to light up the arena, work on 5V input. The microcontroller can consume up to 4-6 amps of current as it is going to support the D-Pad controller, output video to the TV, and update the LED colors. The LEDs consume up to 2 amps of current when all of them are turned to full brightness. The TV monitor will run on AC output voltage. This allows the team to eliminate the need for a Printed Circuit board for the Arena thereby reducing the cost. The electrical wiring is hidden in a box which is attached to the arena. This electrical panel will give the arena a professional look and keep the electronics safe from potential damages caused by human interaction and carelessness. A power surge is required to power the monitor, Jetson Nano, and the LEDs. These technologies are further discussed in section 4.14.1 of this document.

4.12.1 Research

4.12.1a UPS / Surge Protector

The Power surge is attached to the arena to power the microcontroller, the NeoPixel LEDs and the display screen. For simplicity, the team decided to have one cable from the arena go into the power outlet. This also serves the arena requirement of being portable. To do so, a surge protector seems like an ideal option. One cable from the surge protector will go to the wall whereas all the components will connect to the surge protector. The surge protector needs at least 2 outlets: one for the display and one AC-DC converters to power the LEDs and the microcontroller. However, depending on the current consumption and equipment protection, the LEDs and the microcontroller might use two different power adapters. There are multiple surge protectors available in the market with varying features and a comparison between them can be seen in Table 60.

As seen in Table 60, different surge protectors provide different benefits. The arena requirement states the surge should be able to support at least three plugs for the microcontroller, the LED and the display screen. However, it also depends on the type of connectors used to power the equipment. The microcontroller can be powered from a DC barrel jack connector, using GPIO pins, or using a standard micro USB cable. However, each of them provides different amounts of current to the system based on which the DC barrel jack is chosen as it provides adequate amount of current required to run Jetson Nano along with its peripherals. The NeoPixel LEDs can be powered with either standard USB type A or an AC-DC adapter. The specifics of these connectors and adapters is discussed in more detail in sections 4.12.1b and 4.12.1c but this gives an idea on how many outlets and/or USB ports does the power surge need to support at the minimum.

The surge provided by Belkin has 12 outlets and can provide a maximum of 15 amps of current. The energy rating is 3940 joules and it costs approximately $25. The dimensions of this surge protector are 15.6 x 6.10 x 2.10 inches and it only weighs 2.1 lbs. However, it is quite big in size and it will not attach well to the arena making it a bad choice from an aesthetic standpoint. An important requirement for the arena is that it needs to be portable. Due to this the surge protector needs to be thin enough to able to be glued to the frame allow portability and ease of use. The surge protector provided by Amazon Basics consist of 6 outlet and can output 15 amps of current at maximum. The dimensions of this product are 11.9 x 2.2 x 1.75 inches and will definitely attach to the arena effectively. However, the energy rating is only 200 joules and therefore, it cannot protect against high voltage surges. Lastly, the surge protector by TonBux is a sure upgrade from Amazon Basic and Belkin but comes are a higher cost. It has built in Wi-Fi that connects to an app allowing toggling over the air. It has 4 outlets and 4 USB ports and can supply at most 16 amps of current. The energy rating is much higher than Amazon’s Surge but less than Belkin and it has a market price of approximately

$34. The dimensions of this supply fit our needs however, it is the most expensive option out of all the three surge protectors.

Table 60 Comparing Surge Protectors

Brand	Belkin [39]	AmazonBasics [40]	TonBux [41]
Number of Outlets	12	6	4
USB Ports	-	-	4
Length (inch)	15.6	11.9	12.2
Width (inch)	6.10	2.20	2.44
Height (inch)	2.10	1.75	1.26
Maximum Output Current (A)	15	15	16
Weight (lbs.)	2.1	1.1	1.6
Energy Rating (J)	3940	200	1700
Cost ($)	24.99	11.49	33.99

4.12.1b AC-DC Adapters and Peripheral Connections

This component is required to convert power from a standard US 120V 60 Hz outlet to DC power. This component must be highly efficient; thus, it is purchased. A single outlet is expected to support the entire Arena subsystem including TV display, Controller, and other loads. A household power strip is attached to the frame to split AC power from the outlet to the TV Display and AC-DC adapter systems. The total power between the two must be calculated to ensure a single outlet is not tripped, however the TV display is separate from the AC-DC adapter power requirements.

The AC to DC adapter will power the Jetson Nano Controller which sends data to TV display using HDMI and communicates with D-Pad controller using USB. The TV Display needs 3 amps of current and the D-Pad controller uses less than 0.5 amps of current. Therefore, Jetson Nano needs at least 4 amps of current to power all its peripherals easily. The DC barrel jack can support 4 amps of current at 5 V which is more than enough required to run the system effectively. The AC-DC used for Nano is specified in the datasheet provided by NVIDIA however, that component is obsolete. Due to this, the technical parameters of the said component were studied, and an equivalent AC-DC converter [42] was chosen. This converter has a 5.5 x 2.1 mm barrel jack connector that is compatible with Jetson Nano and successfully converts 100-240V to 5 V and can output a maximum of 5 amps of current.

The NeoPixel LEDs also require a 5V adapter however, the current requirements are at most 2 amps in case when all the LEDs is lit up. This scenario is highly unlikely mainly because the LEDs are used for animation purposes and therefore, they will never use their maximum current. This can be used as an advantage and

help save cost. The LEDs can be powered by Jetson Nano's 5V pin which at maximum load can output 1.5 amps and at minimum load outputs maximum current of the power supply. Therefore, a 5V 4A barrel jack connector could be used to run the TV, D-Pad, Camera and NeoPixels. On the other hand, if Nano cannot supply enough current to the NeoPixels then there are two possibilities. One, they can run at low current and is less bright and second, an additional 5V 25W AC-DC converter adapter can be used to power the LEDs and the data cable can be connected to Nano's GPIO to address and program the LEDs.

The peripherals are interconnected using different connectors and exchange data using drivers that are built into Linux's Kernel. The TV Display uses HDMI to receive and display data. The D-Pad sends controller commands via USB whereas GPIO pins and PWM is used to address and program the NeoPixels. The camera data is exchanged using USB as well. The Bluetooth modules is connected using a special M.2 Key E connector which is built into Jetson Nano and does not require any purchase. This information is also summarized in Table 61 in an organized fashion. The GPIO pins use approximately 0 Watts of power because they provide high impedance signal to their respective sensor/device.

Table 61 Arena I/O Schedule

	Type	Connected Devices	Connection Type	Power Requirements
Bluetooth	Intel Module	Microcontroller	M.2 Key E	~ 10 W
LED's	NeoPixel	Microcontroller	GPIO	~ 0W
Display & Sound	TV	Microcontroller	HDMI	< 5W
Gamepad	Xbox controller	Microcontroller	USB	~ 2.5W
Gamepad 2	Xbox Controller	Microcontroller	USB	~ 2.5 W
Switch	Digital	Microcontroller	GPIO	~ 0 W

4.12.2 Design

Based on the research conducted in section 4.12.1 an overview of the Arena Electrical Network can be seen in Figure 48. The block diagram shows that the Arena will mainly be powered by a single outlet. The outlet will power a 4 port, 4 USB power surge to which the display monitor/TV and a 5V 15A AC-DC converter adapter is plugged in. The AC-DC converter has a barrel jack connector that powers the Jetson Nano, the brain of the arena. The barrel jack also powers the LED strips that go around the arena and perform animation for entertainment purposes.

The Jetson Nano will consume 4 amps of current while the LEDs will consume up to 2 amps. The camera, D-Pad, and HDMI plug into Jetson Nano using their appropriate peripheral connector cables and they all run on 5V input. Two Arduinos are used to control the LEDs and Goal Sensor, respectively. Serial Communication is used to interface with the Arduinos and ASCII characters are sent and received that trigger the Arduinos to perform their appropriate tasks. For addressable NeoPixel LEDs the GPIO pin has to provide pulse width modulation signal as that is a requirement of its drivers.

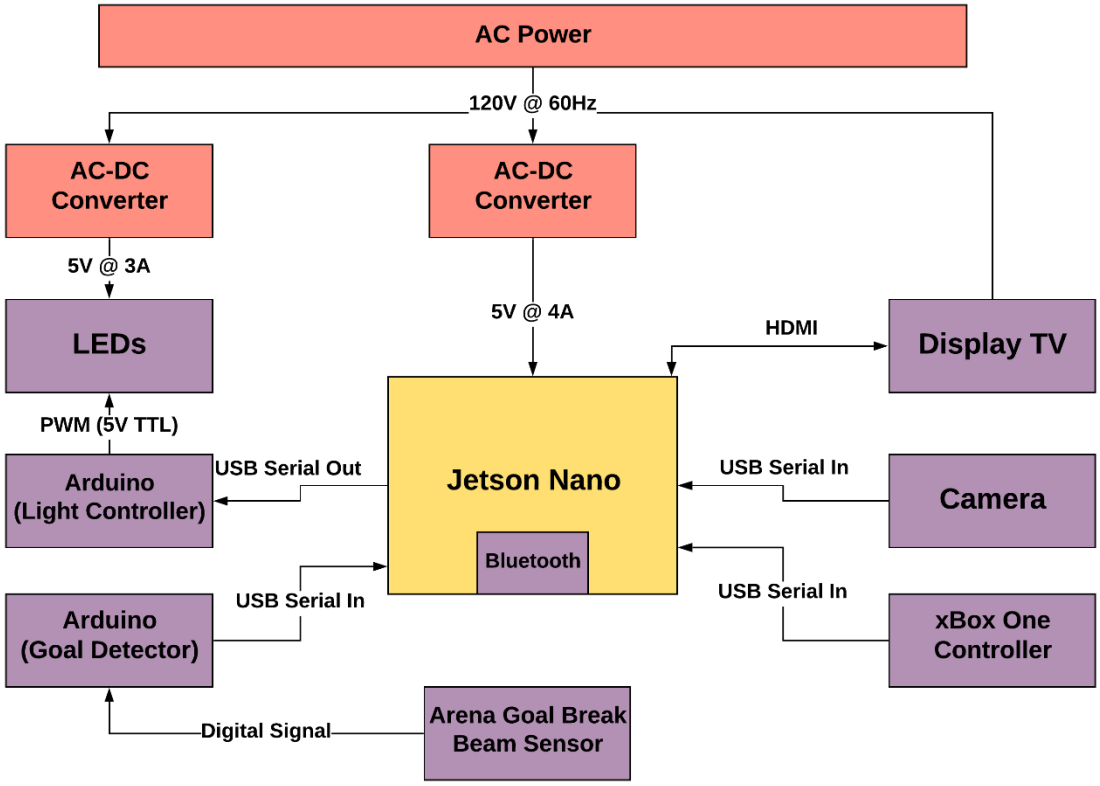


Figure 48 Arena Electrical Network Block Diagram

4.12.3 Prototyping and Testing

The electrical system is tested using multimeter and oscilloscope. The LEDs work on Pulse Width Modulation signal and therefore, an oscilloscope is used to decode the signal generated by the controller. A multimeter will allow to check the voltages and currents at input, output of the controller and all the peripherals allowing the team to make sure no excessive current or voltage spikes occur with the potential of damaging the system and harming the user. These tests will also verify the constraints specified in 2.4 Realistic Design Constraints section of this document. Table 62 shows how these constraints are tested.

Table 62 Arena Electrical System Tests

Requirement	Test	Required Equipment
C.A.3	Determine that the power surge supports all the adapters	Multimeter
C.A.3	Determine that the power surge plugs into the wall	Eye Test
R.P.3 **R.P.4**	Determine that the controller, peripherals, and the sensors are within acceptable voltage and current ranges	Multimeter, Oscilloscope, Datasheets
R.A.E.6	Test if the AC-DC adapter regulates voltage at the correct value	Voltmeter, electronic load
R.A.E.6	Test if the AC-DC adapter can support the required loads value	Voltmeter, electronic load
R.A.E.6	Test the AC-DC adapter efficiency	Voltmeter, electronic load

4.13 Computer Vision

The computer vision portion is one of the most vital aspects of this project. It will detect and track the location of all moving objects and defining court features. It is capable of distinguishing between different robots and supply an accurate location to be used in other portions of the project, most notably, shooting the ball. This information will then be used for multiple parts of the project. The most important part is for calculating the force or angle needed to shoot the ball into the hoop. The game engine will also use the information from the computer vision to show the robots on the court in an overhead 2D visual representation that is a part of the Game system. For these reasons, the computer vision will need to be good enough to distinguish between robots and track them if one should be obscured by another. It will also simultaneously need to detect and track the ball of the court. All these calculations must be done rapidly such that the robot's control loop can be updated with adequate accurate information. If the information is outdated or inaccurate, nearly all systems in the project suffer.

4.13.1 Research

Instead of discussing different software to implement computer vision only OpenCV is discussed because it really offers such a plethora of libraries that encompass every portion of computer vision and machine learning that would possibly be needed for this project. In place of different software will instead be an evaluation of the different methods that exist for performing the tasks. Such as, types of learning and tracking algorithms for OpenCV.

4.13.1a OpenCV

OpenCV is a library of programming functions and libraries used for almost every aspect of computer vision. The library is open source and cross platform, it can run on Windows, Linux, Android, and Mac. OpenCV was built using C++, but supports C++, Python, Java, and MATLAB languages. OpenCV caters to real time computer vision tasks incorporating almost all methods available to detect, track, and learn objects. These methods include the neural net, deep learning, and various tracking algorithms that is discussed further down in this section. Additionally, OpenCV looks to optimize performance by featuring the CUDA parallel computing and application programming interface that was created by NVIDIA. CUDA is a platform that tries to use the GPU to perform general purpose processing tasks. This is important in computer vision as the GPU is generally better at computing parallel elements.

So, the introduction of CUDA to OpenCV should allow the Jetson Nano controller the ability to use its onboard GPU to assist the CPU in large calculation tasks. OpenCV, being the largest and most well-known library for computer vision, will also have more than adequate documentation and it should be easy to find answers to any questions that may occur in production. One of the key concepts that is incorporated in OpenCV is the Neural Net. The Neural Net is a method of computer learning that attempts to perform tasks without telling the program task specific directives. Instead, the Neural Net looks at many examples and will come to conclusions about the examples. It does this by examining connections, known as edges, between artificial neurons. The edges and connections then use a weight system to determine aspects of an object and perform transformations. The other key concept supported by OpenCV is deep learning. Deep learning is a computer learning method based on neural networks. Its main feature is the use of layers to extract additionally higher-level features from input at each successive layer. OpenCV supports the deep learning frameworks TensorFlow, Torch, Py Torch, and Caffe. Caffe is the most important of the four for this project. The Caffe framework is built for image processing and supports the neural network architectures CNN, RCNN, and LSTM. Additionally, Caffe supports GPU and CPU based acceleration libraries.

4.13.1b Tracking vs Detection/ Online vs Offline

Identifying, following, and calculating information about objects falls into two categories, tracking and detection. Both are necessary to perform the task adequately, but when and how to use either is the main difference between algorithms in this field. Detection general refers to the identification of a type of object in an image. Detection can’t discern different objects of the same type only that the objects of that type are present in the image and where their located. Detection algorithms always start from scratch, meaning they will not consider any previous frames, only the current frame. This makes detection slower as it will need

to find the object in every image and therefore will need to search the entire image for that object instead of just searching a portion of the image. To alleviate the burden of having to detect objects generally only every nth frame or when the object is known to be lost will a detection be needed.

Tracking refers to the ability to distinguish objects and follow or predict their movement from frame to frame. Tracking is faster than detection because tracking keeps useful information about the object, such as its location in previous frames. This alleviates the need of a costly detection in every frame by predicting a search area for the object in subsequent frames, reducing the resources required to scan an entire image. Tracking, unlike detection, can distinguish objects of the same type meaning that if multiple objects of the same type appear in the image then each one can be marked as a different object of the same type and can be followed individually. In general, the tracking algorithm is running every n-1 frame. Tracking and detection must work together though because tracking can suffer from object obscurity or fast-moving objects causing the trackers bounding box to move progressively more off center of the object. This problem is fixed by detecting an object every so often to make sure the tracker is still following the object or if the object has left the field of view of the camera entirely. Different tracking methods employ different variations and mixtures of tracking and detection.

Another feature to be considered when researching tracking algorithms is the training method. This refers to the teaching of an object to a program by familiarizing it with the object through multiple viewings of the same object in different scenarios. The two types of training are online and offline. Online training is the harder of the two training methods because the algorithms are trained at runtime and have no information about the object prior to running. It is trained using positive and negative examples of the object. This method can potentially take many cycles to correctly and repeatedly detect and follow an object. Offline training is the more upfront time-consuming training method. Offline training is when an algorithm is shown thousands of examples of the object to be tracked in different scenarios in addition to showing thousands of examples when the object is not in the image. This method trains the algorithm to recognize the object as soon as the program starts running but has a high upfront cost of time to train the algorithm.

4.13.1c Tracking Algorithms

GOTURN - GOTURN stands for generic object tracking using regression networks and is the oldest deep learning-based object tracking method. It is an offline based tracker that will require thousands of examples of previous-current frame pairs. It works by looking at the previous frame cropping it and drawing a bounding box around the object to be tracked. The current image is then cropped using the bounding box of the previous frame and a new bounding box is drawn around the object in the current frame. Because of the extensive offline training GOTURN is one of the faster tracking methods.

BOOSTING - Boosting is the oldest tracker using an online version of AdaBoost for its algorithm. This tracker is online based and requires that the user or a detection algorithm draw the bounding box in the first frame shown. This frame is now the positive identification for the object and anything outside the bounding box is treated as the background. Subsequent frames will look for this object near the area it was located in previous frames and give a score to the frame based on how much it believes the two frames objects are a match. The new location of the object is the frame with the highest score and is counted as an additional positive identifier. This tracking method does not work well as newer methods employ the same strategies but without the drawbacks of not knowing when tracking has failed and fixing the problem of drifting bounding boxes.

MIL - The multiple instance learning tracker is like boosting and is also an online based method. Unlike boosting though the MIL tracker does not only look in the location of the object in the previous frame but around that area too to create more positives. Instead of scoring these frames and taking the highest score the MIL tracker will place frames into positive and negative bags. These bags don't have to contain entirely positives or negatives. In the positive bag there exists at least one frame where the object is centered. This method yields good performance, corrects the drifting problem seen in boosting, and can still reasonably track the object when it is obscured. However, like boosting, tracking failure is not reported reliably and it suffers from a lack of recovering when the object is fully obscured.

KCF - The kernelized correlation filter is another online tracker that utilizes features of both the boosting and MIL trackers. The KCF, like the MIL will take multiple samples of the surrounding area of the object. Unlike the MIL tracker, the KCF tracker will leverage the overlapping regions that occur from taking multiple bounding boxes from positions close together around an object and uses mathematical properties to calculate the predicted location of the object. This tracker has better accuracy and is faster than both the boosting and MIL methods. Additionally, the KCF tracker also reports tracking failures better than boosting and MIL. However, the KCF tracker is unable to continue after the object is fully obscured.

TLD - The tracking, learning, detection method looks at long term tracking, separating it into three individual jobs. The tracker will follow the object from frame to frame. The detector will investigate the frames and update the tracker. The learning portion calculates how much error there is between appearances of the object and remembers the error, so it doesn't occur in subsequent frames. This tracker uses the online learning method. This tracker performs the best when an object is obscured for long periods of time but suffers from repeated false positives.

MEDIANFLOW - The MEDIANFLOW tracker is a bit different in its method. Instead of comparing the previous frame to the current frame this tracker will compare a previous frame to the current frame and the current frame to the previous frame in time. It will then calculate the difference in the trajectories of the object and supply

an accurate prediction of the objects path. This is the best tracking method for failure detection. This tracker will work for small predictable movement that is unobscured but will fail if the motion of the object is too high.

MOSSE - The minimum output sum of squared error is a correlation filter-based tracker, not a deep learning based like the previously discussed trackers. It works by producing stable correlation filters only using a single frame. The benefits of this tracker are its ability to adapt to changes in lighting, scale, pose, and any non-rigid deformations that may occur during tracking. Additionally, this tracker is very good at following objects that are obscured as it will pause itself and start tracking again when the object is visible again. It is one of the easiest trackers to implement and can operate a higher fps than any of our cameras can supply, while also being one of the fastest methods of tracking. While the MOSSE tracker sounds like all positives it has lower performance compared to other deep learning algorithms.

CSRT - Lastly the discriminative correlation filter with channel and spatial reliability is based on spatial reliability maps. It relies on adjusting the area selected for tracking, which is able to make the tracking area larger and increase localization of the object area. This tracker is good for tracking nonrectangular objects. Additionally, it works at a much lower fps than the MOSSE tracker. In return though the CSRT tracker does supply high accuracy.

4.13.2 Design

Python will also be used for all of the OpenCV code. Additionally, all code is run on the Jetson Nano Linux based environment. The final design for computer vision will not implement tracking or machine learning, but instead more traditional computer vision methods. Firstly, a VideoCapture() device is started and set to take in frames at 30fps. The frames are resized to fit the arena perfectly. This first step can be seen in
Figure *49* Collecting the original frame below.

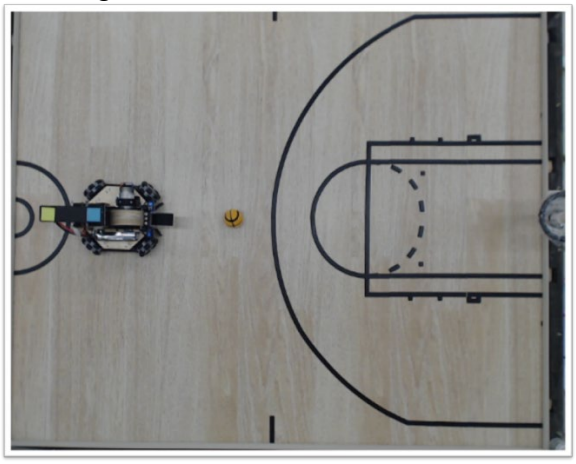


Figure 49 Collecting the original frame

A Gaussian Blur is applied to smooth the frames. The next step is to create a background subtraction object that will store the images that make up the model used for background subtraction in further steps. This model is given the first 30 frames which consist of the court with no robot or ball on it. It was determined that 30 frames are sufficient to create a suitable model, but the more you collect the better the model will be. A foreground mask is obtained from background subtraction by placing objects on the court after the model has been created. This can be seen in Figure *50* Foreground Mask below.

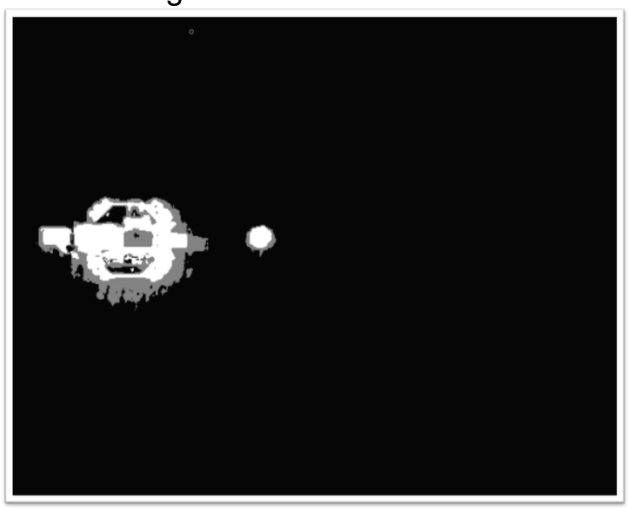


Figure 50 Foreground Mask

A bitwise ‘AND’ is conducted on the foreground mask and original frame to get a color version of the background subtraction. An image of the result of this process can be seen below in Figure *51* Background Mask 'AND' Frame.

Figure 51 Background Mask 'AND' Frame

A lower and upper bound is then used in conjunction with an inRange() function to find each color using the HSV range. An example of a square and the ball being

found using color detection is seen below in Figure 52 Color Detection on a square and Figure 53 Color Detection on a ball.

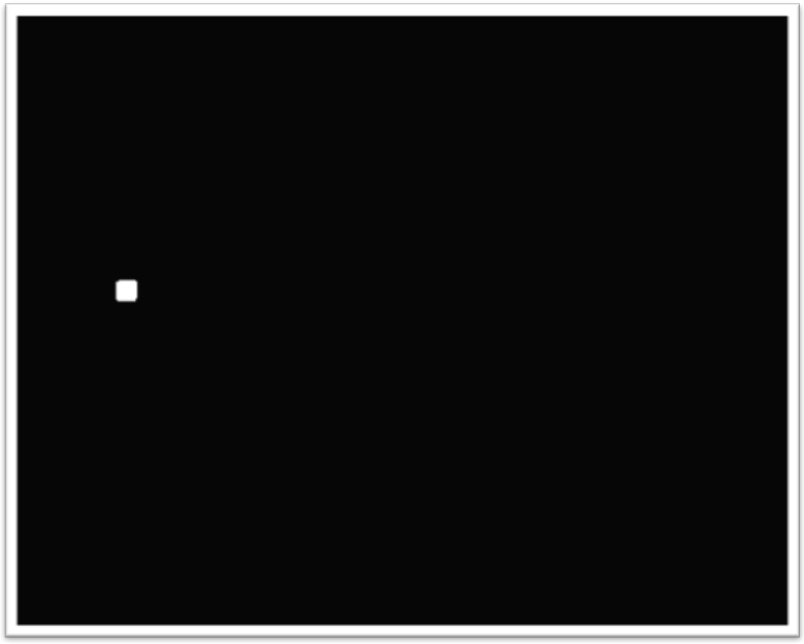


Figure 52 Color Detection on a square

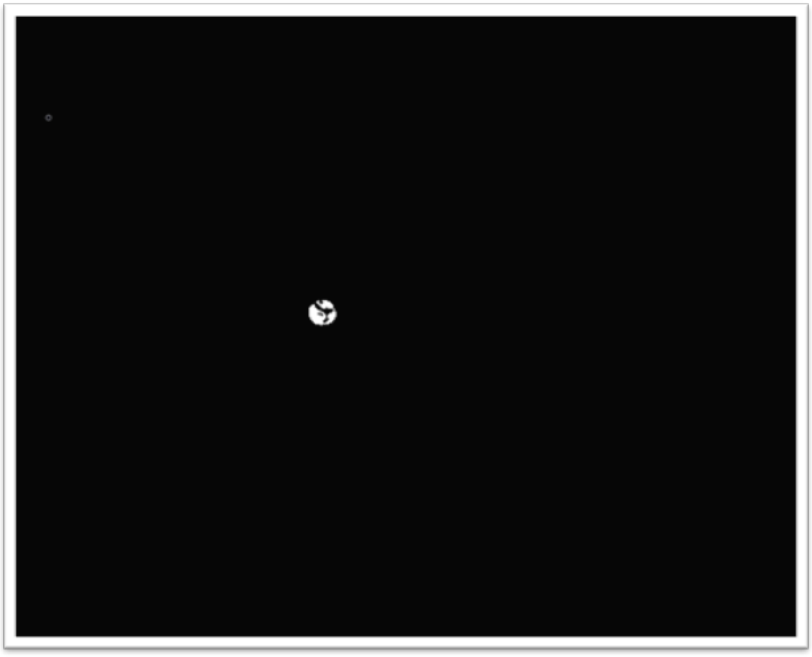


Figure 53 Color Detection on a ball

Once the colors are identified the center points of the shapes need to be found. To find the shapes contouring will be used. Contouring gives a list of all contours found like squares, triangles, and squares within squares. To find the best midpoint a

shape of best fit needs to find from this contour list. This is accomplished by filtering the list by perimeter and number of sides to give a square of best fit. A moments function is then used to find the center point of the square. The result of contouring and midpoint detection can be seen in Figure 54 Contour Detection and Midpoint below.

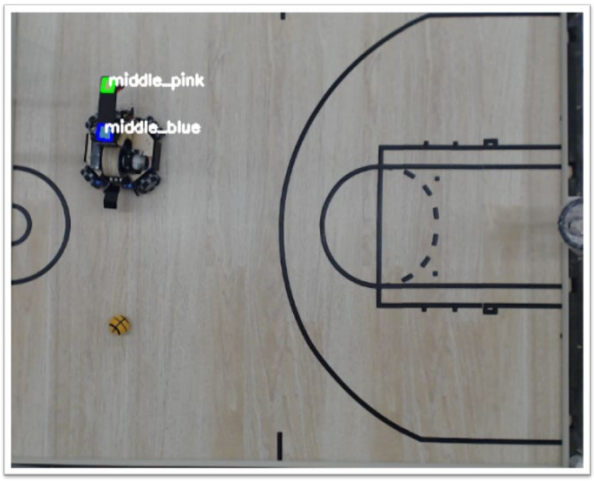


Figure 54 Contour Detection and Midpoint

The ball is slightly different because the ball is broken up into multiple shapes due to the black lines that are a part of the design of the ball. A method similar to contouring is used called Canny Edge Detection which will outline all the shapes that make up the ball. A Hough Transform is then used to create a circle of best fit around the shapes found by the Canny Edge detector. The midpoint of this circle is then calculated. The result of this detection can be seen in Figure *55* below.

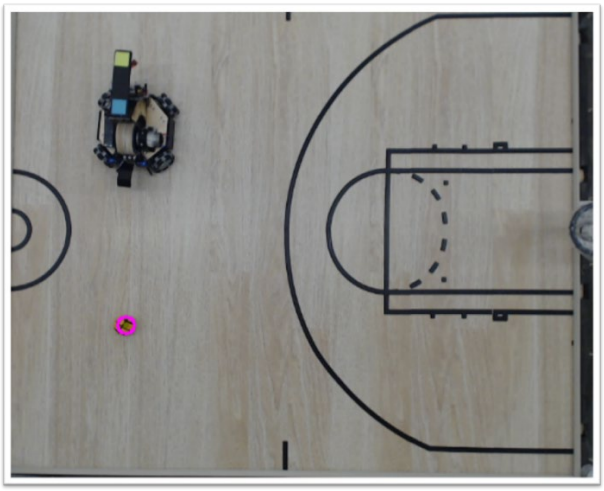


Figure 55 Canny Edge Detection & Hough Transform

The final objective is calculating a unit vector using the two square midpoints and centering this unit vector at the middle of the robot. This unit vector and location will be sent to the game for use in determining orientation and location in the game.

Finally, the location of the ball is returned to the game for further manipulation. The final image with all detections is shown below in Figure *56*.

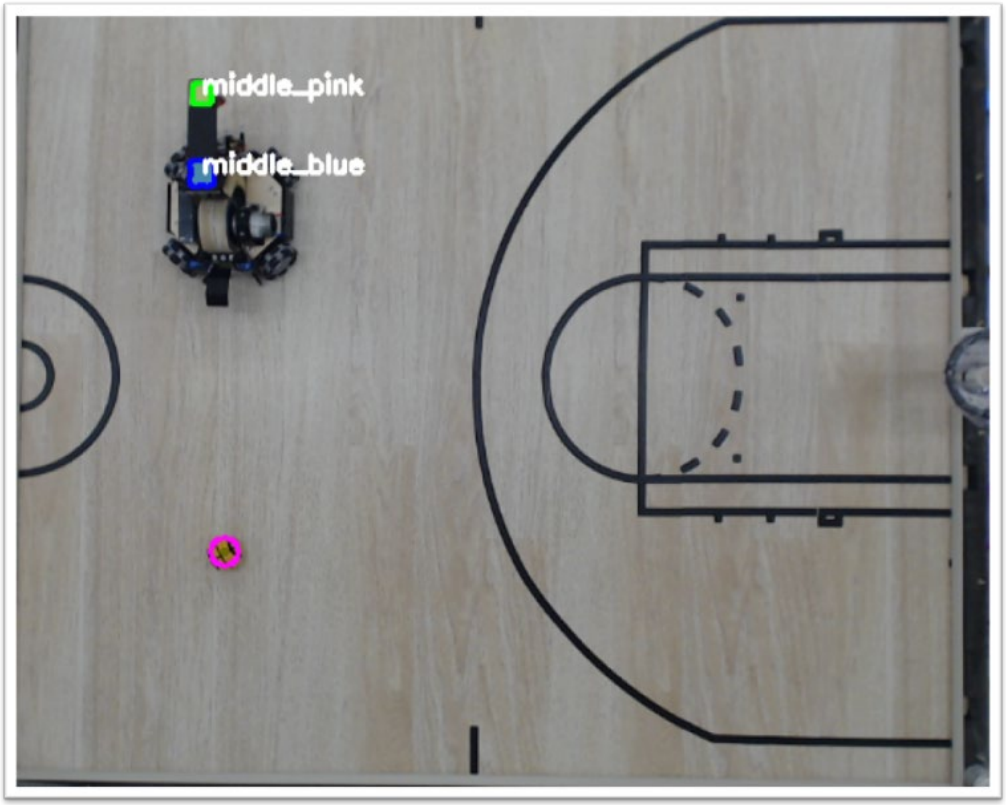


Figure 56 Final output of all detections

4.13.3 Prototyping and Testing

Prototyping the computer vision simply requires the camera mounted above the arena court. The robot does not necessarily need to be complete for prototyping, but a similar sized object is required. Testing of the above described steps will require a moving robot, but basic tests can be conducted with a simulated robot made of black board and two-colored squares before that time. If the above method works with the basic arena setup then the next step is to test it with the arena LEDs and arena display turned on. All tests should be conducted again with this scenario. If this above method should fail to meet the needs and requirements, then another method will need to be selected from the list above in section 4.13.1c and all tests ran again. In Table 63 below is a clear description of each test to be conducted with the adjoining requirement and required equipment to conduct the test. The computer vision tests are more robust than other sections due to the critical risk nature of the component. The position tracking must be highly accurate for the remainder of the project to function properly.

Table 63 Computer Vision Tests

Requirement	Test	Required Equipment
R.A.CV.5	Place one robot on a blank surface with the webcam directly overhead at the correct height and attempt to identify the robot.	Robot 1, Webcam, Arena Controller
R.A.CV.5	Place two robots on a blank surface with the webcam directly overhead at the correct height and attempt to identify both robots simultaneously.	Robot 1, Robot 2, Webcam, Arena controller
R.A.CV.1	Place one robot on a blank surface with the webcam directly overhead at the correct height and attempt to track the robot while in motion from one point to another.	Robot 1, Webcam, Arena controller
R.A.CV.1	Place two robots on a blank surface with the webcam directly overhead at the correct height and attempt to track each robot from one point to another simultaneously.	Robot 1, Robot 2, Webcam, Arena Controller
R.A.CV.1	Place two robots on a blank surface with the webcam directly overhead at the correct height. Attempt to obscure one robot from the camera. Observe that after the robot is unobscured that it is still being correctly tracked.	Robot 1, Robot 2, Webcam, Arena Controller
R.A.CV.1	Place the ball on a blank surface with the webcam directly overhead at the correct height. Move the ball and observe if the ball is being tracked.	Ball, Webcam, Arena controller
R.A.CV.1	Make two markers at a known location at a known distance apart. Observe the output of the computer vision matches the known locations.	Robot 1, Webcam, Arena Controller
R.A.CV.1	Perform all previous tests, but now using the arena laminate flooring as the surface.	Robot 1, Robot 2, Ball, Webcam, Arena controller

4.14 Peripheral Software

The peripheral software involves all the software related to devices and hardware for the arena. This includes driving the LED lights, communication through Bluetooth, hoop sensing, and robot control. This software also needs to directly communicate with the Game system; however, this interface is discussed more

thoroughly in section 6.5 Arena – Game. This code runs on the microcontroller selected for the arena; thus, it is not as critical to design memory efficient code. A strict structure is not required to achieve high performance, thus object-oriented design is more appropriate than functional design. However, the architecture of the code is dictated by the libraries available to achieve the functionality the software requires. The code in this section needs to quickly process the data from the other systems and generate outputs to maintain some semblance of real-time control. Data from the gamepad input in the Game System sent to the peripheral system and then finally sent and processed by the robot can introduce a huge amount of latency, particularly in the arena-robot interface.

4.14.1 Research

4.14.1a Bluetooth

The most popular python Bluetooth library is pybluez. Pybluez is an open source library that extends the operating system's Bluetooth resources into a python application. Although the library is not under official development, there are active contributors implementing the software onto new devices. The library has a wide range of documentation. Another option is to utilize Python to execute and communicate with a C++ process that utilizes another Bluetooth library such as BlueZ, Qt Bluetooth, or libblepp. Each of these C++ libraries implement well documented code for interacting with the available Bluetooth resources. The node would communicate with Python through something like shared memory or a socket. This option is likely more difficult to implement. [43]

4.14.1b GPIO Library

The Jetson Nano GPIO is primarily implemented in Python. The python interface interacts with system files that are connected to the GPIO registers. There are many examples and sample applications, and the library is fairly well documented. The library supports interrupts, which is extremely useful for the limit-switches and frees up resources from polling. The library has identical API to the Raspberry Pi GPIO library, thus the applications written utilizing this library can be ported between boards without issue (In the event of hardware issues or additional testing). There are additional ways to interact with GPIO on the Jetson without this library, including writing directly to the system files related to the GPIO. This is significantly more difficult to implement and maintain. [44] [45]

4.14.2 Design

The peripheral software architecture for the Arena closely follows the robot architecture paradigm despite not having the same level of I/O. The only I/O for peripheral devices in the arena is two limit switches for hoop goal counting, and an output LED signal for the various Arena lighting. However, this software does have to pass data between the game engine and the robot due to the Bluetooth

communication. The message passing structure between the two other software implementations is shown in Figure 57. The software architecture and class diagram are shown in Figure 58. This figure indicates the three layers of the peripheral software: Application, implementation, and library layers. Application is the main program that performs the instantiation and method calling of the classes implemented in the implementation layer. The implementation layer abstracts the I/O, State machine, and communication interfaces. Each class contains relevant data and methods to perform all of the appropriate actions for that object. The library layer contains libraries external to the team that generally implements low-level implementations such as Bluetooth communication and GPIO control. The selected Bluetooth library is the pybluez library due to its simple implementation in python. The standard Jetson Nano GPIO library is utilized.

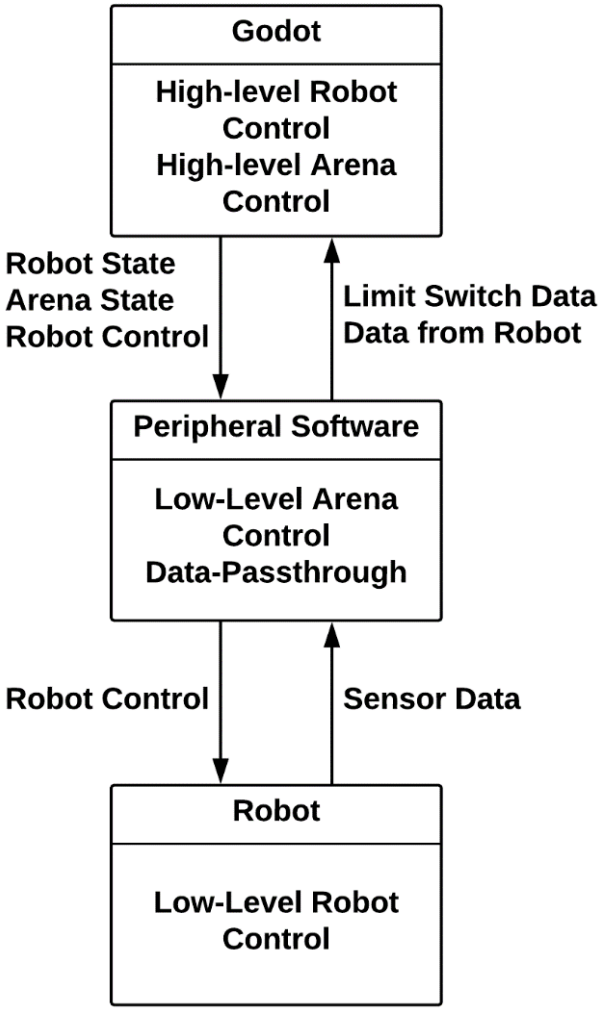


Figure 57 Peripheral Software Communication Diagram

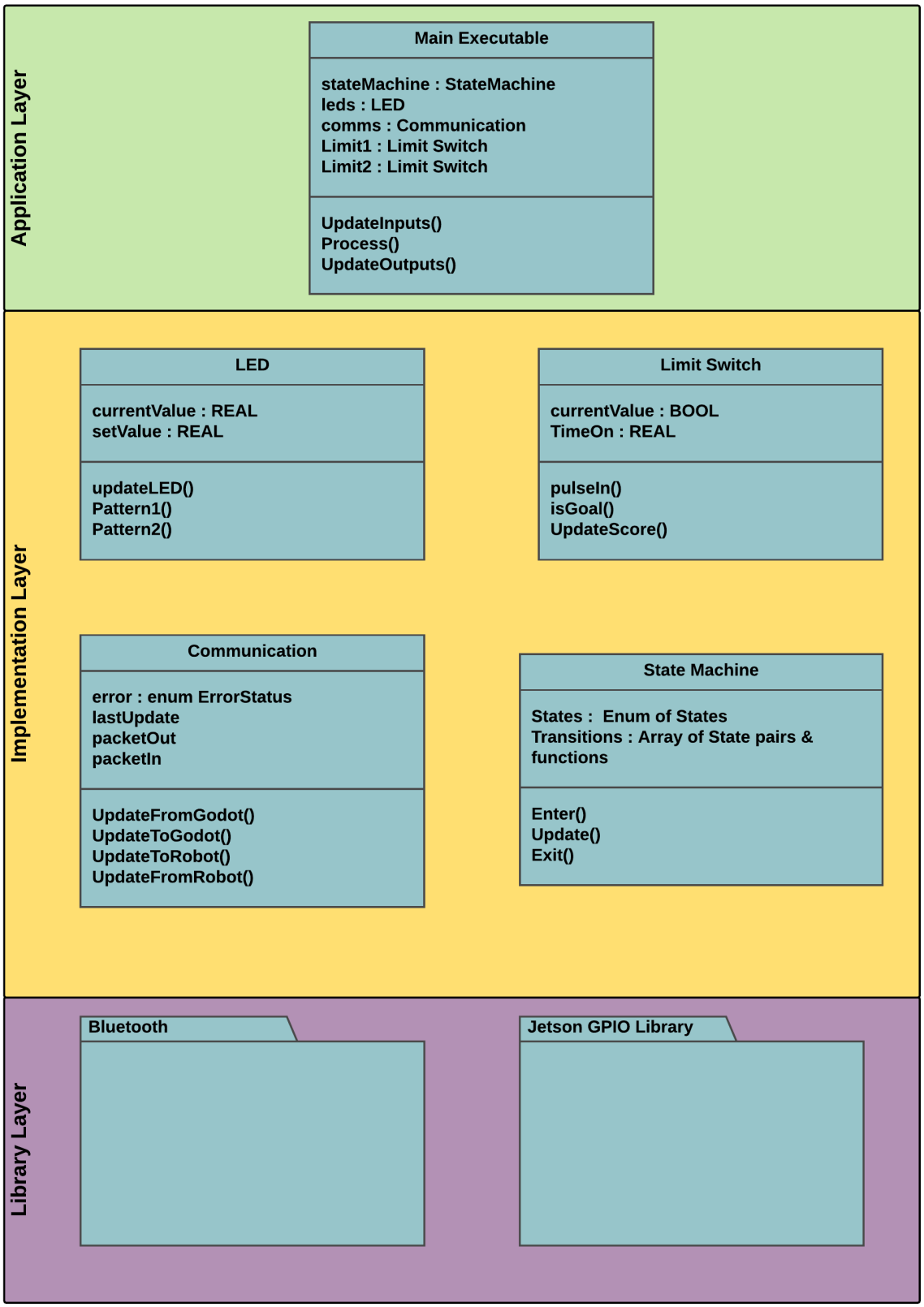


Figure 58 Peripheral software Architecture and class diagram

4.14.3 Prototyping and Testing

The prototyping for peripheral software requires the acquisition of the Jetson Nano primarily. Once that device is acquired, all the software and libraries can be installed and developed. The communication and GPIO tests require additional components such as the communication hardware and limit switches. A full-communication prototype requires the Robot and Game software to be complete. However, each section of code can be independently tested with appropriate unit tests as described in Table 64.

Table 64 Peripheral Software tests

Requirement	Test	Required Equipment
R.A.17	Communication Unit Tests	Bluetooth Module, power supply, Nano
R.A.17	LED Unit Tests	Nano, Power supply, LED's
R.A.17	Sensor Unit Tests	Nano, Power Supply, Sensors
R.A.17	Full software tests	Nano, Power Supply, terminal monitor

5.0 Game System

The Game system harnesses the power of a game engine to deploy commonly used features that are available in a virtual environment. The project requires players to be able to adjust settings, start and stop timers, display scores and other feedback, and assist the user by showing a 2D virtual representation of an environment. The game system is essentially the primary software arm of the Arena system, but it can be developed and act independently from the Arena system. This system is the most feature-scalable system in the project. A large number of extra software functionality can be added to the project through the game system. These things include different robot settings based on a chosen player. This feature could adjust speeds, accuracy, or force limits to vary the player experience. Additional control logic such as autonomy or machine learning could be introduced into the game system to change the player experience quite dramatically. Figure 59 contains a block diagram to illustrate the separation of components for the game system and how they interact with each other. The three main components of the game system are the collision detection, video playback and robot control. The game system is also responsible for displaying data to the player that is currently playing the game and any spectators watching.

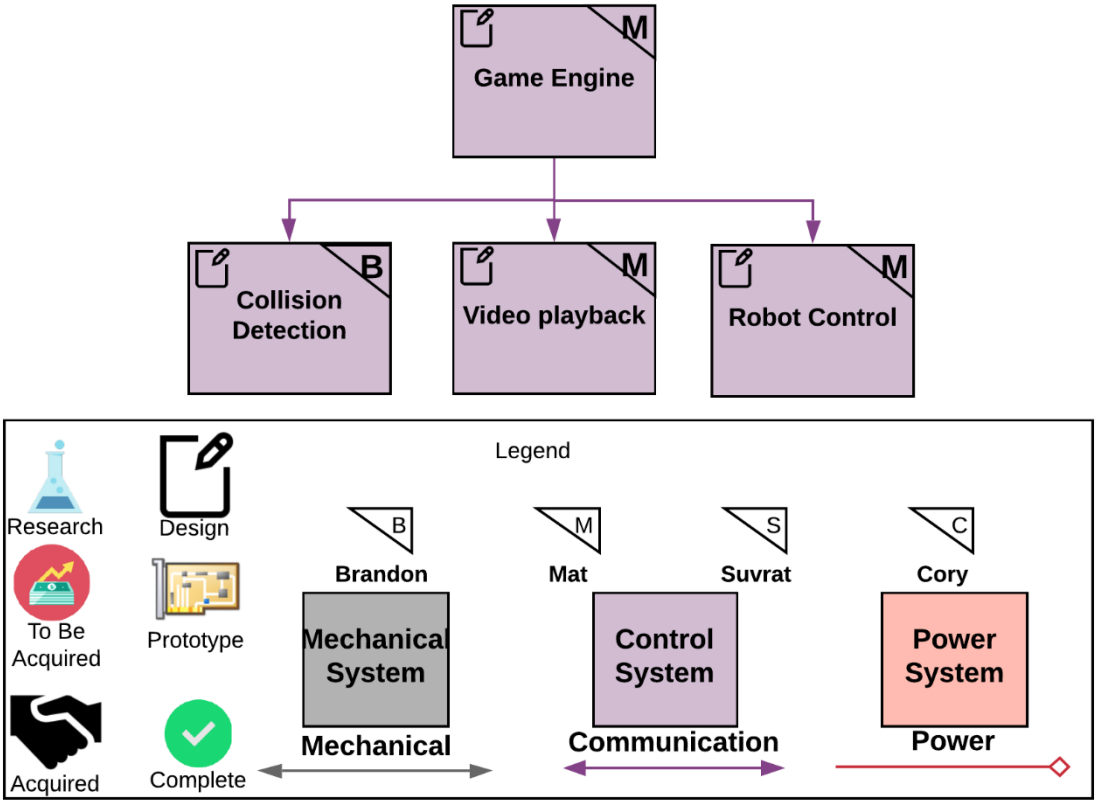


Figure 59 Game system block diagram

5.1 Game Engine

The game engine is responsible for a few tasks overall and will act as a hub for data to flow in and out of. It must be able to handle data visualization such as showing a mockup of where the physical components such as the robot and ball are on the field or playing an animation when a shot is made or missed. Using data sent from the field and robot, the game engine will also handle collision detection and send feedback to the gamepad being used by the player. The game engine will also need to be capable of both 2D and 3D animation to accomplish its tasks.

5.1.1 Research

5.1.1a Unity

Unity is useable for both 3D and 2D games and simulation. Most of the group has used Unity before so there is some experience there. A major plus is that the base edition of this engine is free to use. Due to that Unity is under consideration for being used for both the 2D visualization and tracking as well as the animation after a made shot. Unity looks to be a primarily 3D based engine and there seems to be more material for tutorial in 3D rather than the 2D side. A downside of unity is that the UI can get rather cluttered and unusable. It also has a rather steep learning

curve if the developer is just learning how to use Unity with C#. On top of that, there are solutions, but no simple answer to doing inter process communication between Unity and other things, such as a C++ program or Python script.

5.1.1b Godot

Godot seems to be slightly opposite of Unity in that it looks more 2D friendly than 3D. It comes with a lot of tools to help a first time Godot developer get started creating what they need to. Like Unity, it has its own suite of animation tools for the developer to use instead of using a separate software such as Cinema4D or Maya. As well as supporting C++ and C#, Godot also has its own language, GDScript which is a lightweight Python-like language. Godot in general is a more lightweight program and requires less resources to run. This may prove to be useful as the resources to run the game system through whatever controller is chosen for use may be limited. Godot, like Unity, has an asset hierarchy that dictates how and what items are allowed to interact with.

5.1.1c PyGame

PyGame is a cross platform set of python modules. It has been around for a while and supports 2D game making on python 2 and python3. Because it is cross platform it will be able to be used on both Windows and Linux, and will also be able to be used on the aarch64 architecture of the Jetson Nano being used to power this project, unlike the Godot engine that the group was leaning towards using. There is no GUI for this like with Godot/Unity and would be implemented entirely in code for this project.

5.1.2 Design

The GUI for the game aspect of the project is created using the PyGame engine. It will consist of 4 main screens that are laid out in Figure 60. The main menu is the screen that is shown first when the game is first initialized. It will consist of 3 options in the form of buttons for the users to pick, Play, Controls, and Exit.

The first option is “Play” which will bring the player to another option screen. The Play screen’s options consist of easy and hard mode. Easy mode will lead into a game where the robot will handle the different aspects of shooting for the player, such as power of the shot and making sure the robot is lined up with the basket. Hard mode will disable these assistive options and allow the player full control over the robot. The second option of the main menu is to bring up the controller menu. This screen will contain an image of the controller labeled with its mappings (Figure 46 in Section 4.8.2) as well as two checkboxes. One checkbox is to invert the x-axis of the controllers and the other is to invert the y-axis of the controllers. This allows the user to reconfigure the controller to match where they are standing around the arena to give them the easiest and least confusing control of the robot.

The last option is to simply exit the game. Figure 61 contains a flowchart on how the screens are accessed.

After the player selects their preferred mode of play, easy or hard, the game will start, and the screen will display a representation of the actual game arena. This screen contains all the elements on the physical field, the robot, the ball, and the hoop. All three is tracked by our computer vision program, and the positions will refresh in real time to display an up to date position of the game components. This will also allow the hoop to be placed wherever the owner wants to configure it, and have it still accurately shown in the simulation aspect of the game. The other two components are a match timer that displays the time remaining in the game, and a counter to display the score the player has accrued by making baskets. Baskets will also be scored by the distance the shot occurred from in order to provide more of an incentive to make shots from farther away.

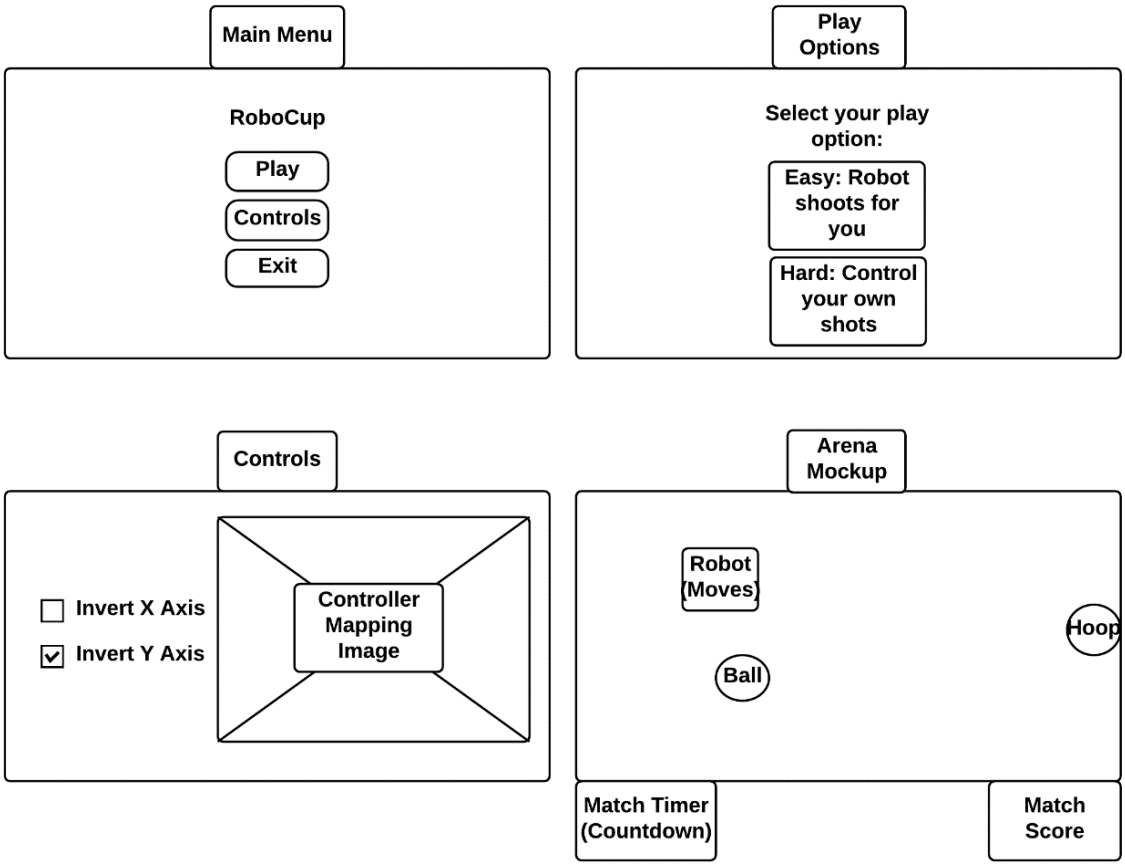


Figure 60 Wireframe screen layouts Screens of the game system

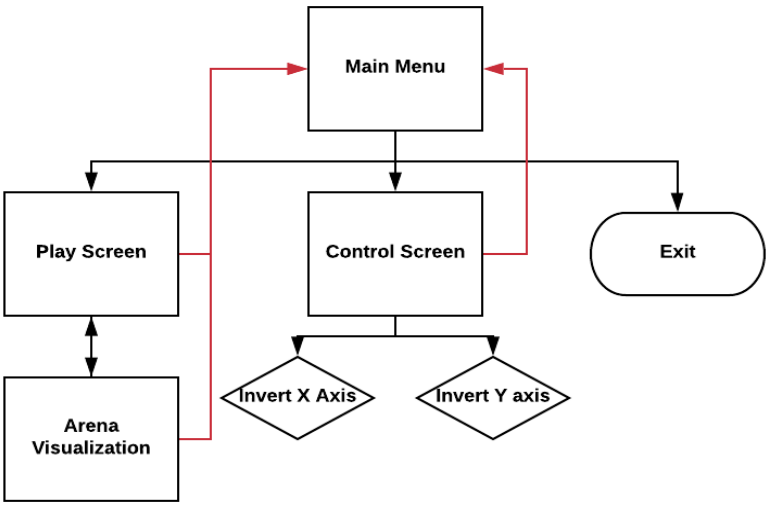


Figure 61 Flowchart for screen navigation

5.1.3 Prototyping and Testing

Prototyping is done using PyGame. The first objective for prototyping the game system is to create a base scene that takes the shape of a rectangle to act as the arena. On this scene there is a few shapes that will act as the ball, rim, and robot. After the scene is set up, the first 2 tests from Table 65 below is able to be tested. The tests are conducted in the order of the requirements fulfilled as each requirement is a concrete subcategory of the game system. Verifying that the game submodules work effectively together will require the rest of the subsystems work first, therefore it doesn't explicitly fall under a requirement listed in section 2.5

Table 65 Game engine tests

Requirement	Test	Required Equipment
R.G.1	Moving shapes programmatically	Laptop
R.G.1	Moving shapes via gamepad input	Laptop
R.G.2	Check data is displayed properly	Laptop
R.G.3	Check date and time are accurate	Laptop
R.P.4	Verify game submodules work together in the intended fashion	Laptop

After the tests are validated, the parts are combined into one scene that contains the visualization for the court, as well as the data display parts and rechecked to make sure the components still function properly. Once that is verified it is made into the official design.

5.2 Collision Detection

The game engine system is responsible for protecting the robot in events of poor user input. For example, if the player constantly runs into the wall, the robot would either drive over the wall and flip, or it would burn out the motors and cause electrical or structural damage. Another instance that requires collision detection is when two robots run into each other. Again, these events could cause electrical or structural damage and prevent consistent playing. In both instances, the collision detection should be aware when a robot is entering a zone that could be dangerous and protect the robot. The protections could be reducing motor power, slowing the robot, or preventing input entirely. Another useful feature of collision detection is automatic intaking when the ball is near the front of the robot. This is a player-assist feature that can have adjustable settings.

5.2.1 Research

5.2.1a Game-Engine Collision Detection

Collision detection is possible with both Godot and Unity and is done in a very similar way in both engines, with the game objects being designated as collision objects, and then monitoring the different objects in order to check if they are overlapping, and sending a signal when two objects are found to be overlapping. In PyGame, it can be done using either collision objects or a more primitive way in manually checking locations. Using an object is more streamlined and allows for more fine-tuned control.

5.2.1b Collision Response

There are a couple paths to take in terms of interacting with the user to let them know a collision has occurred or is imminent. The first is by sending a rumble pulse through the controller. This pulse can be practically any length, it just has to get the users attention and they will likely pay attention to their surroundings a little bit better. The system can also take control from the user to prevent them from moving the robot in the direction that the collision is occurring in. This will help keep the components on the robot from suffering wear and tear. However, this isn't the preferred method as the team wants to avoid taking unnecessary control of the robot to keep the players immersion intact. The last way that this can be done is to flash a symbol on the GUI/data visualization screen warning the player of a collision. However, this would require that the player actually be looking at the screen instead of the court, which isn't recommended.

5.2.2 Design

The collision detection is set up in such a way that as the robot moves closer to the designated wall area of the arena, the controller isgin to vibrate, and the

intensity of the vibration will increase the closer that the robot's position to the wall is. This is accomplished by layering bands of detection objects in a procedural manner leading up to the perimeter. Each band is assigned a value for vibration that is triggered upon the robot's sprite in the visualization entering its area. It is important that there is a reliable scale between the visualization and the actual arena. If this is not the case, the controller may vibrate for no reason, or not vibrate when it should be doing so. It is also possible to attempt to send a command to the robot to not allow it to move in a specific direction, preventing the continued attempt to move into a wall, which potentially can damage motors and components.

5.2.3 Prototyping and Testing

The collision detection system can be prototyped with Godot or PyGame and a simulated robot-input and output. This allows the team to verify that the safety functionality of the system is in place prior to testing with actual hardware using the unit tests below in Table 66. Another important aspect of the testing not mentioned in the table is that based off the feedback, the collision detection can be further optimized to change when it needs to trigger so it isn't always sending notifications to the system.

Table 66 Collision Detection Tests

Requirement	Test	Required Equipment
R.G.8 R.P.4	Verify Safety System	Robot, Display, Frame
R.G.6	Verify Collision Avoidance algorithms	Robot, Display, Frame
R.G.6	Verify accuracy of simulation versus physical	Robot, Camera, Court. Tape Measure

5.3 Video Playback

It is very common to have a replay of events that happened prior to a score in any sport. When a player scores a goal, it would be exciting and useful for spectators to see the motions of the robot and ball in the time leading up to the robot shooting the ball. This requires a storage buffer containing the positional data of the robots and ball, and timing for ball entering the hoop. At the time of scoring, a short playback of the positional data (in 2D) and then a pre-rendered 3D animation of the ball being launched and going into a hoop play. This is very similar to what bowling centers do for different types of pins being knocked down. The pre-rendered 3D animation reduces complexity of the simulation while still providing the feeling of experiencing the goal again. This gives a small reward to the player

when they score hopefully giving them a sense of accomplishment and achievement.

5.3.1 Research

There are a couple of paths that the video playback could potentially follow. The first one is like what someone would see at a bowling alley, where there is a little animation for making a strike or a spare, that rewards the player, but doesn't play back any real information. Another potential path is one that relays positional and input data leading up to the goal to allow the player or audience to "relive" the shot. The timeframe on this can be either lengthened or shortened to change the focus of the video.

15.3.2 Design

Videos is played back when a made basket is detected from the arena, and possibly when a shot is attempted and not made. The video is picked from a bank of premade videos depending on the situation and is rotated so the player doesn't see the same one every time. These videos will follow the bowling alley celebration of a goal and not the full playback. One of the options for playback is that the screen will replay the last few seconds of the 2D field visualization before it plays the animation for the celebration.

5.3.3 Prototyping and Testing

Prototyping of the video playback does not require the actual video that is played to be done in order to be completed. The team can substitute any video to use for testing purposes and just swap it with the correct rendered animation once it is complete. The testing will follow an order outlined below consisting of unit tests that slowly scale up until we get the full project. The procedural steps needed for full functionality are outlined in Table 67.

Table 67 Video Playback tests

Requirement	Test	Required Equipment
R.G.5	Manually trigger any video	Laptop
R.G.5	Make sure our video is rendered properly	Laptop
R.G.5	Trigger scene on basket score/non-score	Nano

5.4 Master Robot Control

The master robot control software exists within the game system and is the ultimate high-level controller for the robot. The game system acquires user input from the gamepad and then converts that data into appropriate robot commands. The robot simply acts as an I/O device that the game system is controlling. Robot kinematics, master states, computer vision, collision system, and other inputs are utilized to convert user input into servo commands that are sent through the Arena system to the robot and interpreted there.

5.4.1 Research

5.4.1a Inter-process Communication

Inter-process communication (IPC) is the act of transmitting data between two processes either on the same processor or between processors. There are two common ways to achieve this: Shared memory, and messaging. Shared memory is when two processes have access to the same memory hardware and access them at different times. Messaging is when two processes communicate through a channel such as a socket, pipe, or file access. This topic and design are discussed in depth in an integration section: 6.5 Arena – Game.

5.4.1b Kinematics

Kinematics is related to bodies in motion. In this case, kinematics is referring to forward and inverse kinematics of the different mechanical systems of the robot. Kinematics are particularly concerned with converting between different domains, or spaces. These domains are related to physical information about the system. For example, converting a robot arm's end-effector pose in 3D space to the series of joint values required for the robot to achieve that location.

5.4.2 Design

The Robot Master control accumulates all of the data relevant to robot function and determines the high-level functionality of the robot. The system dictates the state machine listed in Figure 30 by determining the appropriate transitions based on the input from the various components including Collision Detection, Kinematics, Computer Vision, Game State, Gamepad data, and Peripheral software data. This data flow is shown in Figure 63. The kinematics component itself is also completed in this section due to its reliance on Peripheral software data, and computer vision data. The high-level motion control is derived from the directions shown in Figure 14. The software is implemented in the chosen Godot software and does not require library support outside of the native Godot

functionality. The robot control inputs are dictated by the gamepad mapping shown in Figure 46.

There are three kinematic spaces we are concerned with: Task Space, Robot Space, and Actuator Space. The task space is the domain that computer vision, collision detection, and players perceive the environment. It is the actual full 3D representation of the arena, robot, and other physical components. Robot space is the robot's understanding of the environment, and how to interact with it. That is, the robot can move in various directions, spin up a wheel, and other actions that refer to velocities or positions relative to the robot transformation frame. The actuator space relates to the effort and electrical feedback required to drive the servos and motors. Each space requires a transformation between each-other as shown in Figure 62. The transformation between task and robot space converts between robot pose and robot velocities to servo speeds. This is critical to drive the robot in the correct manner. The servo velocities are then transformed to PWM signals or duty-cycle percentages. The forward motions take current servo speeds and determine the robot's overall velocity.

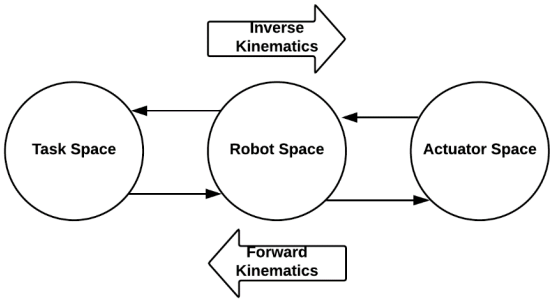


Figure 62 Kinematic Transforms

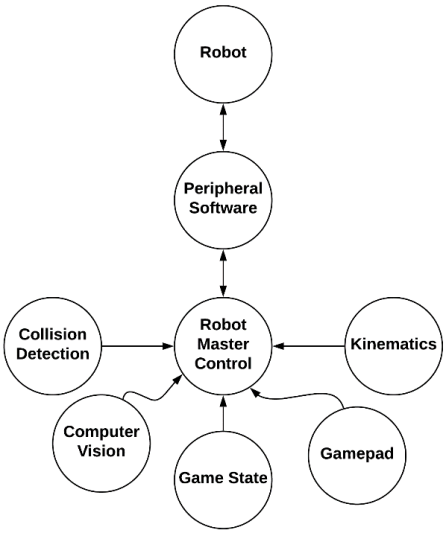


Figure 63 Robot master control data flow

5.4.3 Prototyping and Testing

The prototyping of this component only relies on the team's choice of game engine. Once the general game framework is implemented, the robot master control can be implemented. Final testing requires the Arena and Robot to be complete. The tests for this component are shown in Table 68.

Table 68 Master Robot Control software tests

Requirement	Test	Required Equipment
R.P.4	Kinematics Unit Tests	N/A
R.P.4	Data Process Unit Tests	N/A
R.P.4	Master Control Tests	N/A
R.P.4	Robot Control Tests	Complete Arena & Robot

6.0 Subsystem Integration

The system integration section identifies high-risk interfaces that must be carefully designed and tested to avoid problems that occur when multiple systems are designed in parallel. Three robot interfaces are identified, and two major system interfaces are identified. The robot interfaces are high-risk because they are the most likely point of failure in the project, and the entire project depends on the robot's capabilities to be completed appropriately. Although these components exist within the same subsystem, their critical risk status elevates the importance of integration. The system interfaces are not within the same subsystem; thus, their integration is not discussed within their respective system discussions. As such, the interfacing between the major systems is developed in this section.

6.1 Base – Intake

The Base-Intake integration is identified as the mounting interface between the base subsystem, and intake subsystem. This interface ensures the compatibility between the intake and the wheel locations, and the existence of mounting locations for the intake to be attached to the base. The ball must be able to be picked up from the ground and in various orientations around the court. Corners are particularly difficult for the intake to reach in, so the intake-base integration must ensure that the intake can reach the ball from each orientation at each position in the court. The overall goal of the base-intake integration part is to lower the skill and time needed to consistently grab a ball off the ground. If we are able to accomplish this, it will make the game more entertaining and less stressful to play.

6.1.1 Design

The intake requires the ball to go to a particular location without getting stuck. Thus, the design involved for this component is a shovel/gate type apparatus that directs the ball into the correct location. This design is provided in Figure 64 and functions as a way to keep the ball from getting stuck underneath the robot's base. This is vital to make sure that the sensitive components inside the base are protected from getting hit and potentially disconnected. The intake is the same as the launcher thus the mounting and cuts are the same as in section 6.2 Base – Launcher. This design does not solve the problem of picking up the ball from corners as the intake is located within the frame. However, the angled parts on the front of the robot that act as a funnel will also be able to displace the ball. In the event that this becomes a larger or more common problem, additional designs / components are introduced.

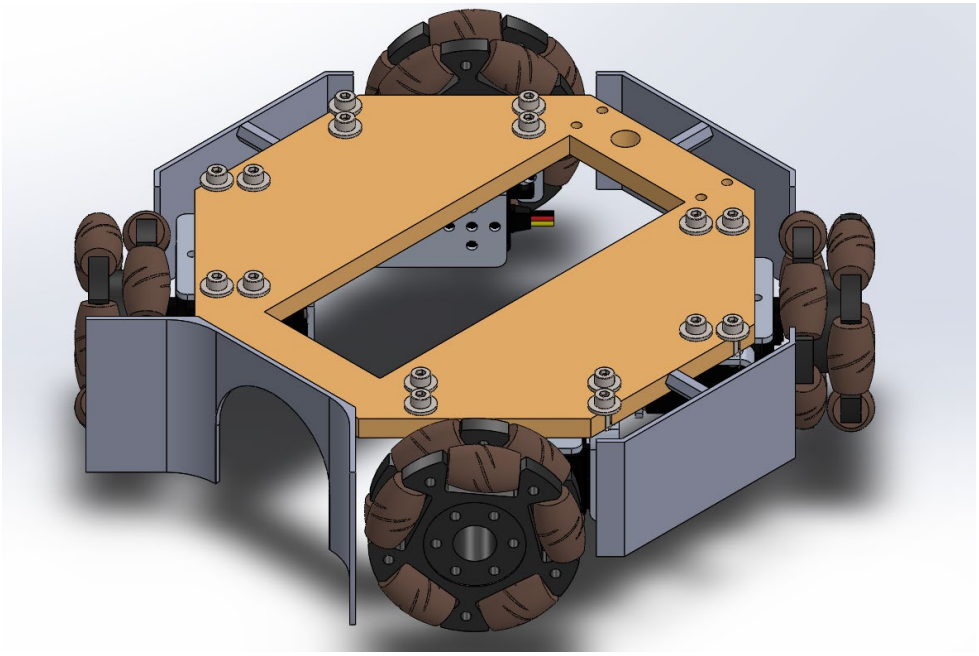


Figure 64 Ball-Prevention plates & cut-outs

6.1.2 Prototype and Testing

The base-Intake integration can be prototyped with the two sub-system components and some additional hardware. It includes additional plates that have bends in them which requires a heat gun to heat up the material and bend it at a particular angle. The tests for this integration determine the validity of the design and verifies that all requirements are met. The tests are shown in Table 69.

Table 69 Base-Intake integration tests

Requirement	Test	Required Equipment
R.P.4	Does the intake mount securely to the Base?	Base, Intake, hardware tools
R.P.4	Does the intake reach the ground to pick up the ball?	Base, Intake

6.2 Base – Launcher

The base-launcher integration is identified as the mounting interface between the Base subsystem and Intake subsystem. This interface ensures the compatibility between the base and the launcher, including the existence of mounting locations for the launcher to be attached to the base, and clearance for the launcher mechanisms to fully actuate

6.2.1 Design

The Base-Launcher integration design consists of a cutout for the wheel and track mechanism, and mounting holes for the various subsystem components required to operating the systems. This includes mounting holes for the lever servo, and the wheel motor bracket. The wheel size and cutout are variable such that the best sized-wheel can be printed or adjusted after additional testing. However, the cutout must be small enough that the frame remains strong despite the hole in the center. Dimensions for the cut outs to mount the base are provided in Figure 65.

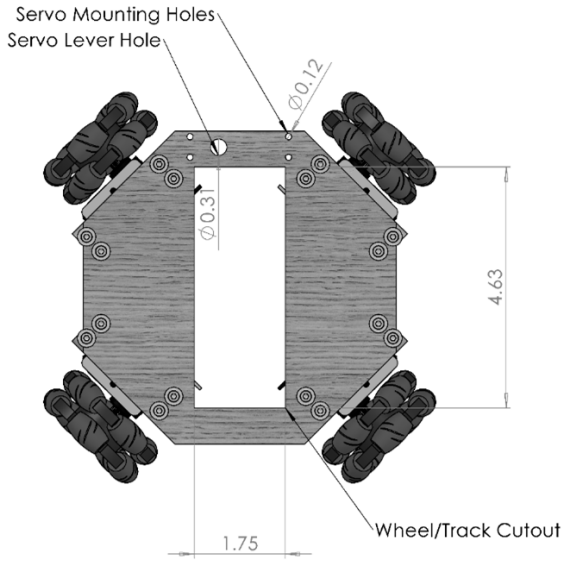


Figure 65 Base-Launcher Integration

6.2.2 Prototype and Testing

The base-launcher integration is very similar to the base-intake integration because of the combination of the two systems. However, testing is critical because the motion and accuracy of the base directly affects the consistency of the launching. The tests are shown in Table 70.

Table 70 Base-Launcher Integration tests

Requirement	Test	Required Equipment
R.P.4	Does the Launcher mount securely to the Base?	Base, Launcher, Hardware tools
R.P.4	Can the launcher slide fully extend and retract?	Base, Launcher
R.P.4	Can the launcher release fully engage or disengage the gear?	Base, Launcher

6.3 Intake – Launcher

The Intake-Launcher integration is identified as the design interface between the intake and launcher such that the intake places the ball in the correct location each time for the launcher to hit consistently. Further, the intake must not interfere with the launching mechanism.

6.3.1 Design

Currently the design we are going with for the Intake-Launcher integration is going to incorporate them into the same component. We are using a single large flywheel that is lowered close to the ground to be able to grab the ball off the floor. After the ball enters the mechanism, it will slot into a trapdoor to wait to be fired. This will allow the wheel to spin freely without moving the ball. When the player is ready to shoot, the wheel will spin up to the correct speed and the trapdoor mechanism is reversed through the use of a servo or a similar piece of hardware. Once the ball contacts the wheel again, it will continue along its path and be shot out the other end. It is important that whatever is used to reverse the trapdoor has a high enough torque rating to keep the ball and wheel from pushing back out against it. Setting up this integration this way will allow the team to utilize both passive and active mechanisms to make the overall component use less pieces.

6.3.2 Prototype and Testing

The intake-launcher integration prototyping is essentially automatic due to the mixture of the Intake and Launcher systems. The testing for this section is critical because it directly impacts the consistency and reliability of the launching system. The tests that is needed are shown in Table 71 on the next page.

Table 71 Intake-Launcher Integration Tests

Requirement	Test	Required Equipment
R.P.4	Does the intake place the ball into the correct location for the launcher?	Intake, Launcher, Power Supply
R.P.4	Can the launcher shoot without interference from the intake at any rotation?	Intake, Launcher

6.4 Camera-Arena

The Camera-Arena integration component is defined as the interface between the camera and arena. Specifically, the mounting of the camera such that the camera's field of view does not prevent the camera from seeing all of the components on the field. This directly affects the mounting height of the camera.

6.4.1 Design

Upon design of the interface for the single camera mounted in the center of the arena, it is found that the camera has to be mounted nearly 6 feet above the ground plane for the camera to see the top portions of the robot appropriately. This greatly exceeds requirements for the arena to fit in a typical room, thus efforts are taken to reduce the height that the camera must be mounted. In order to achieve this, two cameras are introduced with their field of views rotated 90 degrees from the single camera view. Figure 66 and Figure 67 shows possible configurations for single and double camera layouts respectively. There is also some overlap across the centerline because the field of view must be able to see objects that are at an increased height from the ground plane. There are two ways to deal with this overlap, one method through software and another through hardware. The software method requires us to filter out the overlap before processing the images, while the hardware method requires more precise measuring to reduce the overlap of the field of views of the different cameras. Overall the team is leaning towards using the software approach to filter the camera feeds. Figure 68 shows the projected field of views for the cameras in a single or double camera configuration. The single-camera approach is tested first without part construction to get an idea of whether or not the second camera is required. After testing, the two camera-design may be implemented depending on the result of the actual test. Regardless,

the interface is a critical risk and is monitored appropriately. Note that the field of view lines are based on theoretical values and may change based on lens focusing as well.

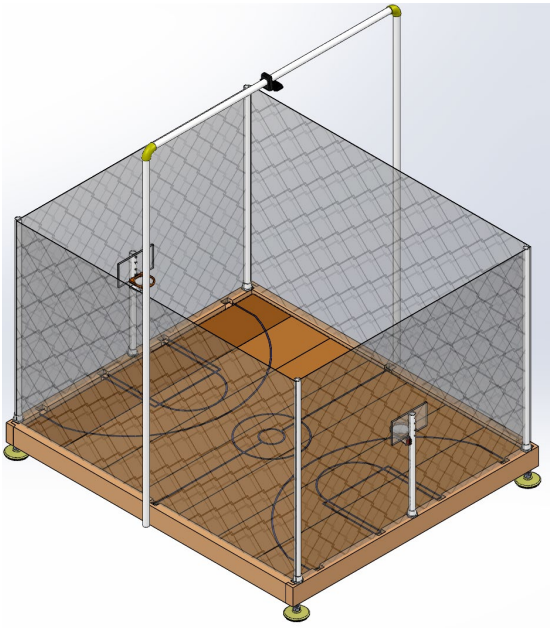


Figure 66 Single Camera configuration

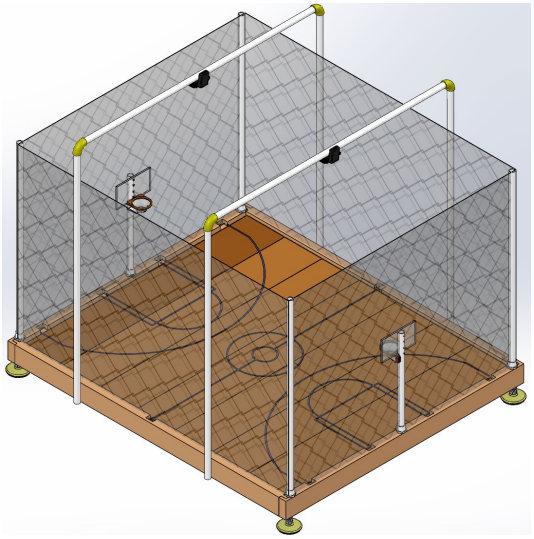


Figure 67 Two Camera configuration

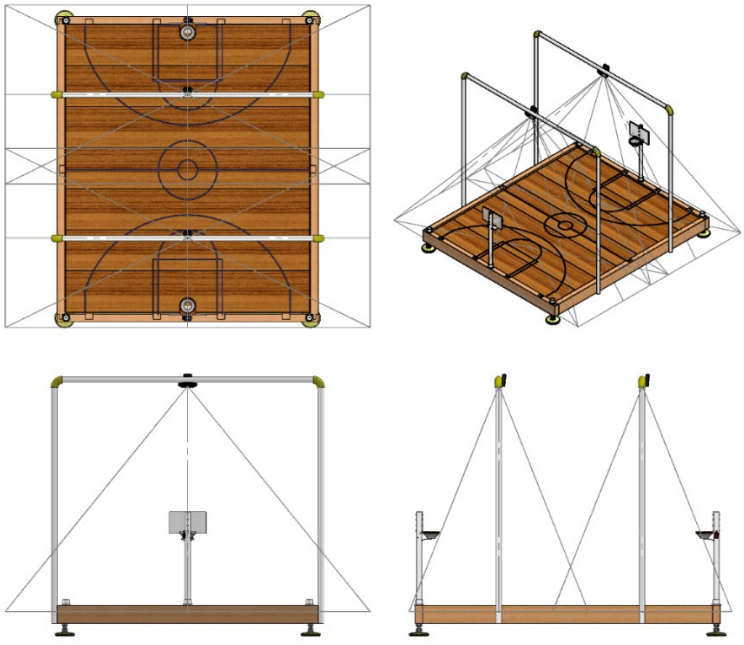


Figure 68 Two-Camera FOV

6.4.2 Prototype and Testing

The Camera-Arena prototype requires that both cameras be mounted three feet above the arena in an equal distant fashion. This will require the completion of both the frame and the camera mounts as well as either a robot or something of similar size to test the camera with. Below in Table 72 are shown the requirement to adhere to, the test to be completed, and the required equipment to complete the tests. This is one of the final tests for integration with regards to the arena and is integrally important to the computer vision working properly. Computer vision can't be implemented until the Camera-Arena integration tests pass.

Table 72 Camera-Arena Integration Tests

Requirement	Test	Required Equipment
R.A.CV.2	Verify both cameras together can view the entire court	Two Cameras, Arena frame
R.A.CV.1	Verify the entire robot is visible in the camera FOV	Two Cameras, Arena frame, Robot

6.5 Arena – Game

The Arena-Game integration involves interfacing between the Arena system and the game system. The game system requires position data of the robots and the ball from the camera stationed above the arena to accurately update the locations of the simulated versions in the game engine. Additionally, the game system must send the gamepad data to the arena system to process the player input's and send them out to the robots.

6.5.1 Design

Inter-process communication can be accomplished in a variety of ways between scripts and programs of different languages. The two options the team were most comfortable with were TCP/IP sockets and shared memory. Shared memory is overall easier with a C++ based environment while TCP/IP sockets would be better for use with something like C# and Godot. The arena will connect to a socket to be able to send the camera position data to the game, where the game will handle it. Socketing is handled rather well by Godot's API, so the team is primarily using this approach.

6.5.2 Prototyping and Testing

The Arena-Game interface prototype simply requires the Arena and game systems to be completed. Once the systems are completed, a single software section must be built and tested to interface the Arena and Game software systems. Table 72 lists the two basic tests that the arena and game must pass to be considered integrated properly.

Table 72 Arena-Game Integration Tests

Requirement	Test	Required Equipment
R.P.4	Send and verify signals between arena and game	N/A
R.P.4	Check visualization matches data sent from the arena	N/A

6.6 Robot - Arena

The Robot-Arena integration is identified as the interface between the components of the Robot and Arena. For example, the ball must interface correctly with the Intake subsystem, and launch subsystem. The fiducials for computer vision to track on the robot are designed in this section. The robot and arena share information via Bluetooth. Arena uses camera information and controller inputs, combines them into a packet and sends it to a buffer. The Bluetooth system reads the buffer periodically and sends the commands to the robot. The robot then parses the packet into useful information and carries out the commanded tasks.

6.6.1 Design

Arena and Robot Integration happens in the software. A hardware implementation would require a serial or an ethernet connection between the robot and the arena. This approach will undoubtedly provide error free data as long as the noise is attenuated. However, having cables in the field will cause troubles in Robot's movement due to which a wireless approach is taken in interfacing the two systems.

The Bluetooth comes with a variety of hardware and software components that make interfacing the two systems efficient. Bluetooth v4.2 LE is the latest Bluetooth available in the market for consumers and developers which is used in this integration. The robot runs on ATmega328P microcontroller with is available in Arduino. The firmware is open source and is flashed onto the chip. This chip also has firmware available that interacts with various Bluetooth modules available for Arduino. However, due to the requirement of a custom PCB the Bluetooth modules were researched in depth. Almost all Bluetooth modules compatible with Arduino use TI's CC254x chip which will consequently be used to send and receive Bluetooth packets to and from the robot, respectively [46]. On the Arena side, Intel's 8265NGW adapter is used to accomplish the same. Both TI's CC254x and Intel's 8265NGW are discussed in sections 3.6.1a Bluetooth and 4.11.1a Bluetooth, respectively.

The Arena generates a 14 to 18 bytes long data packet along with additional 14 bytes of overhead. The exact length is determined post experimentation of the modules however, ideally it is likely that the data length is 14 bytes at minimum.

This packet contains information regarding each of the motor velocities for driving the robot, the speed of the flywheel to shoot the ball at, the status of the fifth servo that intakes or let go of the ball, and the state determined by the user holding the controller. On the robot end, the raw data bytes are parsed for their appropriate information and run specific functions inside the loop. Each command sets a Boolean flag making writing and debugging the code easier. After sending the packet Arena Bluetooth goes into sleep mode until a response is received from the Robot. On the robot side, the Bluetooth wakes up upon receiving data packet from the Arena. It sends an acknowledgement along with its state byte so that the Arena can make appropriate decisions based on the received information and respond. This architecture assumes that both Bluetooth systems are version 4.2 and work on Serial Port Profile for communication. The structure and sizes of these packets can be seen in Communication design sections 3.6.1a Bluetooth and 4.11.2 Design of robot and arena, respectively.

6.6.2 Prototype and Testing

The communication tests are conducted using RSSI utility in hcitools available in the BlueZ stack on Linux. The RSSI value needs to be more than or equal to -30dBi for an excellent connection. The arena and Bluetooth take at maximum 7 milli-seconds to send and/or receive packet which gives an update rate of ~140Hz. This is far more than the required update rate as allows resting time for healthy packet transfers and saving energy.

The packets are tested for accuracy by using serial monitor on Robot end and Command Line Interface on Arena's Linux side. Arduino's Serial monitor is used to send a string which is seen on Arena's Linux terminal using the "hcidump" utility available in the BlueZ stack. The string is displayed as a stream of raw data bytes and an ASCII table is used to decipher them. A script can also be written to parse the raw bytes. Similarly, the "hcitools" utility can be used to send raw bytes to Robot's Bluetooth which is displayed on Arduino's Serial monitor to prove accurate data transmission. A summary of such tests is summarized in Table 73 Robot-Arena Integration Tests which also shows the project requirements they satisfy.

Table 73 Robot-Arena Integration Tests

Requirement	Test	Required Equipment
R.A.2 **R.P.4**	The robot and arena can communicate bidirectionally	Robot, Arena, Terminal
R.P.4	The arena can control the robot	Robot, Arena, Terminal
R.P.4	The computer vision system can track the robot's position	Robot, Arena, Terminal, Display

7.0 Project Operation

The project is designed to be easily operated by any user. The project requires only a few scripts to be run with a quick process to come online in less than 2 minutes.

7.1 Startup

1. Remove all objects from the court
2. Power on the Jetson Nano and other powered devices by turning on the power switch
3. After boot-up, start the Computer Vision script by running: python3 RobotBasketball/Arena/CV/RealTimeDetection.py
4. Once “Ready” appears, place the robot and ball onto the court and close the walls
5. Power on the robot by connecting the power cable to the battery
6. Connect the robot to the Arena peripheral script by running: python3 RobotBasketball/Arena/Peripheral/RobotPeripheral.py
7. Make sure a controller is plugged in and start the game system by running: python3 RobotBasketball/Game/GameGui.py
8. The robot is now ready to be driven

7.2 Operation

1. The game will open in the Main Menu. Operate controller to select play or controls to go to their respective screens.
2. Once on the difficulty selection screen, selecting a difficulty will bring you straight into the game.
3. Easy Mode provides the ability to automatically adjust launcher RPM based on computer vision tracking, and automatically align the robot to the hoop
4. Hard Mode removes the assistance and makes a player automatically control the RPM
5. The game will automatically return to the main menu when the game runs out of time.
6. The player must drive the robot to the ball with the intake in the up position.
7. After the ball is close, the player captures the ball by placing the intake into the middle position
8. The player then finds a place to launch the ball from, and aligns properly
9. The player launches the ball by placing the intake into launch position
10. If a goal is made, the LED lights display flashing green and the airhorn is played

8.0 Administrative

Overhead is required as project size increases. The overhead involved for this project relates to task management, scheduling, budgeting, and communication. Each of these is necessary to achieve the requirements set forth by the team.

8.1 Budget and Bill of Materials

As per economic constraints C.ECON.2 and C.ECON.3, the team wanted to keep the cost below 300 dollars for the robot and 400 dollars for the arena. In Table 74 and Table 75 below, the bill of materials for the robot and arena are shown.

Table 74 Robot Budget

Item	Price (USD)	Quantity	Subtotal (USD)
Launching Hardware	$ 20.00	1	$20.00
Drive Hardware	$ 30.00	1	$30.00
Intake Hardware	$ 20.00	1	$20.00
Intake Motor	$ 15.00	1	$15.00
Drive Motor	$ 20.00	4	$80.00
Launch Motor	$ 20.00	1	$20.00
Controller	$ 20.00	1	$20.00
Battery	$ 30.00	1	$30.00
PCB	$ 20.00	1	$20.00
Bluetooth Module	$ 10.00	1	$10.00
Voltage Converter	$ 15.00	1	$15.00
Total per Robot			$280.00

Table 75 Arena Budget

Item	Price (USD)	Quantity	Subtotal (USD)
Frame Hardware	$ 100.00	1	$ 100.00
Camera	$ 40.00	1	$ 20.00
Controller	$ 100.00	1	$ 100.00
Power Supply (AC-DC)	$ 20.00	1	$ 20.00
Bluetooth Module	$ 10.00	1	$ 10.00
Court Hardware	$ 25.00	1	$ 25.00
LEDs	$ 25.00	1	$ 25.00
Gamepad	$ 25.00	2	$ 50.00
TV Display	$ 70.00	1	$ 70.00
Total			$ 445.00

Table 76 and Table 77 below contain the actual bill of materials for the robot and arena. The team was unable to be on budget for the robot overall but were under budget for the arena. The total spent on both parts was 711.69. Thus, the team was able to fulfill constraint C.ECON.1 by being below 1000 dollars total. The single component that went over budget the most was the 360-degree servos chosen to be used for the drive motors. Another thing that we originally didn't plan on using was a Lexan sheet. Polycarbonate is usually not cheap so that added a late cost to the robot section. While, excluding the polycarbonate, the budget would've been met using the motors that had originally been planned, we were able to significantly reduce the complexity of our robot PCB by using the servos instead. However, if it is determined that the servos purchased will not meet the requirements that have been set, the motors that were originally being considered is purchased and used instead. *The total* cost for development was $325.47for the robot and $386.22 for the arena.

Table 76 Bill of Materials for Robot development

Item	Budget Item	Price ($)	Quantity	Subtotal (USD)
Arduino Uno	Robot Controller	11.86	1	11.86
L298Nx5	Motor Controller	13.99	1	13.99
PCA9685	Motor Controller	6.99	1	6.99
MCP23017	Motor Controller	7.95	1	7.95
Servo	Launcher Motor	16.88	1	16.88
Stepper	Launcher Motor	12.95	1	12.95
Stepper Controller	Actuator Control	21.95	1	21.95
Robot Kit	Drive Hardware, Drive Motor	95	1	1
DC-DC Convertor	Voltage Converter	13.99	1	13.99
Bluetooth module	Bluetooth Module (Robot)	5.22	4	20.88
Level Shifter x10	PCB	9.89	1	9.89
Motor	Launcher Motor	14.99	1	14.99
360 Servo	Drive Motor	26.99	4	107.96
Power Supply	Battery	21.99	1	21.99
Encoder	Launcher	8.95	1	8.95
Battery	Battery	33.25	1	33.25
Lexan Sheet	Launcher	138	¼	34.50

Table 77 Bill of Materials for Arena development

Item	Budget Item	Price ($)	Quantity	Subtotal (USD)
DC-DC Convertor	Voltage Converter	13.99	1	13.99
Intel Module	Bluetooth Module (Arena)	24.89	1	24.89
Jetson Nano	Arena Controller	100	1	100
SD Card	Arena Controller	11.99	1	11.99
SD Card Reader	Arena Controller	6.99	1	6.99
Arena hardware	Frame Hardware	20	1	20
Frame Material	Frame Hardware	20	1	20
Limit Switches	Arena	8.39	1	8.39
Logitech C920	Camera	59.99	2	119.98
Pixy Cam	Camera	59.99	1	59.99

Table 78 and Table 79 below are the respective bills of materials for reproducing the robot and arena if another version is needed. As for the arena, a lot of the original spending was trial and error and the team now has a better understanding of what amount of materials is needed to construct and assemble the arena. The cost for manufacturing the robot is $327.65 and $284 for the arena.

Table 78 Bill of Materials for Manufacturing and Reproducing Robot

Item	Budget Item	Price ($)	Quantity	Subtotal (USD)
PCA9685	Servo control	2.30	1	2.30
Servo	Launcher Motor	16.88	1	16.88
Parallax Servo	Drive Motor	26.99	4	107.96
PCB	PCB	2	1/2	1
Bluetooth module	Communication	4.77	1	4.77
ATMega328	Microcontroller	1.50	1	1.50
Base Mechanical	Base	20	1	20
Launch/Intake Mechanical	Launcher	20	1	20
Battery	Battery	33.25	1	33.25
Motor	Launcher Motor	14.99	1	14.99
Encoder	Launcher	8.95	1/2	4.50
Omni-Wheel	Base	15	4	60
Servo Plate	Base	.5	12	6
Lexan Sheet	Launcher	138	¼	34.50

Table 79 Bill of Materials for Manufacturing and Reproducing Arena

Item	Budget Item	Price ($)	Quantity	Subtotal (USD)
SD Card	Arena Controller	11.99	1	11.99
Jetson Nano	Arena Controller	100	1	100
Wi-Fi / Bluetooth module	Bluetooth Module (Arena)	24.89	1	24.89
Frame Mechanical	Arena Frame	20	1	20
Logitech C920	Camera	59.99	2	119.98
Limit Switches	Arena	8.39	1/2	4.2
Mesh	Frame hardware	6.34	1/2	3.16

8.2 Milestones

Figure 69 is a Gantt chart that shows the various major milestones and project timelines required to successfully complete the project. Our first major milestone is the completion of this paper at the end of July. There are a few check-in points along the way during the semester to keep us on track in the form of 50% and 75% deadlines where the paper must be at a specific page count. The major critical path for the production of the robot is that of the PCB design, purchase, and fabrication due to the long lead time to purchase and build the PCBs. The major parts and subsystems of the robot such as the launcher, intake and base, however, can be separately built and tested without the robot being fully built and assembled. This allows for progress to still be made in the form of design revisions and placement tweaks of major components without having to get a new PCB entirely. The major critical path for the arena is actually getting the frame built and being able to check the height the camera(s) need to be mounted at in order to see everything. On top of that, with a physical arena built, we is able to verify that the other aspects of the design such as where to mount the hoops, and the height of the walls, are correct and do what they need to do. As of the time of this paper, the arena base frame is 90% done and is complete by the time the team resumes meetings at the beginning of the fall semester in August. The computer vision aspect of the project could potentially cause major problems down the line if we find the tracker, we plan on using doesn't work the way we want it to. On the other hand, the game system can be created mostly in parallel with the other components and shouldn't cause a critical delay unless it is put off as most of the issues that could arise with the game system require it to be integrated with the arena, robot, and computer vision in order to test and resolve and most of its components can be tested independently via input from a gamepad or numbers generated from a file.

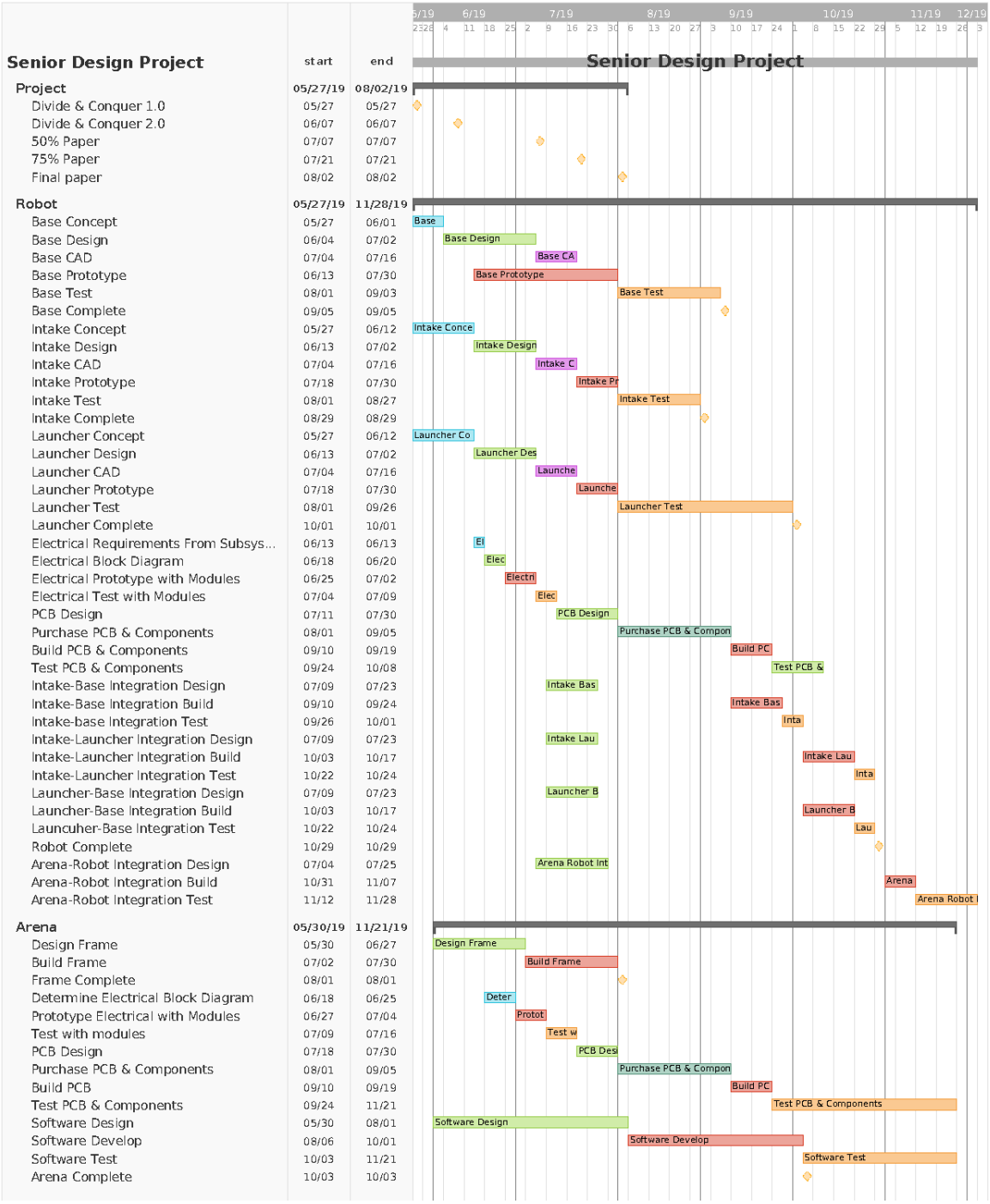


Figure 69 Gantt Chart indicating critical milestones and work timelines

8.3 Communication

Communication is critical to the project team's success. A thorough use of documentation and sharing tools allows the team to work at peak efficiency regardless of physical location or project timeline. Although there are many tools available to achieve this, three critical tools that can seamlessly link together are utilized to reduce the number of sites or applications to download.

8.3.1 Microsoft SharePoint

SharePoint and other Microsoft products are used for this project because it can act as a one-stop shop where all the materials required for the project can be found. SharePoint itself is a website platform that has various pages and plugins. Each research topic has its own page that the team fills out as the research and design is completed. This allows all our research to be compiled real time and is organized such that information can be retrieved when necessary. The plugins utilized within SharePoint include Microsoft Planner, a tool that allows users to add tasks with information like assignee, due date, and relevant files. The tasks are tracked as cards that can be moved around with order of importance, or have reminders set so that things are finished on time. Everything with the SharePoint is synced and stored on OneDrive, Microsoft's cloud storage platform. This allows for version control of all the documentation required for the project.

8.3.2 Discord

Discord is a free VoIP software that provides chat, screen-sharing, file-sharing, voice and video calls in an easy to use platform. Discord is chosen over Slack, Skype, and other chat software because it provides the required features for free, it is stable, and the team has utilized it for other projects in the past. This tool provides us a way to store any text messaging between members of the group and return to it at any point in the future.

8.3.3 GitHub

GitHub is a cloud application that integrates with the git version control scheme. The team can work on their local machine and develop any files or software required, and then when finished, upload the file to the cloud that other members can update from. The tool is very powerful when simultaneously working on the same file because git can merge different versions of the file based on changes made. This is particularly helpful in software that are modularized into functions or blocks that multiple members can work on simultaneously without losing progress in another block.

9.0 Project Summary and Conclusion

Over the course of the last semester, we researched and designed a basketball-based arcade game. In order to do this, we had to figure out a few different parts. The different parts of this project consisted of the base robot, and arena, and a game system to manage it all. Each part of the project presented a different and unique challenge. First, the robot was a huge mechanical undertaking that none of us were really prepared for. Although some of the group has robotics experience in the past, none of us are mechanical engineers. Another big part of the robot was learning about PCB design in order to get everything to integrate in a clean and acceptable way. Second, the arena presented us with the challenge of building a scaled arena in a modular and easy to transport way, while still being able to be large enough to give the player a range of challenge during the game. Another challenge of the arena was setting it up in such a way that we can use computer vision, as it is the main brain behind the project. This leads into the last part, the game system. This consists of computer vision, and a GUI to relay data to the player. The computer vision is the brains of the robot. This component finds the range between the robot and the rim, assisting the player in making shots. The GUI component shows the location of the field components; the robot, ball and hoop. It also contains other data such as time left, and score obtained so far in the current game.

Overall, we have been successful in creating a design that we believe meets all the requirements that we set for ourselves at the beginning of the summer. During our design phase, overall, we stayed on schedule. Sometimes we were ahead of schedule and other times we weren't; however, we were able to pick up the pace to meet deadlines on time. Part of the initial setback time towards the beginning of the project was adjusting to how the other people in the group operate and accomplish work. Once this hurdle was overcome, work was able to progress smoothly.

The plan moving forward is to first get our PCB layout tested and ordered. After this, we must construct the robot. The arena is already mostly built and only requires a few finishing touches and potential tweaks. Most of the work left is the computer vision and game system. Both are still in their infant stages up to where they needed to be to hammer out a design strategy for both, but the operational function isn't there yet. We must be able to fully track the robot, ball, and hoop in order to consistently get the correct measurements. After we can successfully track the positions, we must "teach" the robot the correct power to put on shots for different lengths through testing and adapting the formulas used by the robot.

Appendix I Copyright Permissions

You have received a message from the Amazon Seller - HENENG

Of course, you can use our pictures.

I hope I can help you~

I wish you a successful project~

Happy life~~~

Figure 70 Heneng Permissions

Brandon,

Thank you for your product image inquiry. We are happy to grant you permission to utilize the images, citing Parallax Inc. We hope the product met your needs and that your final project is a success!

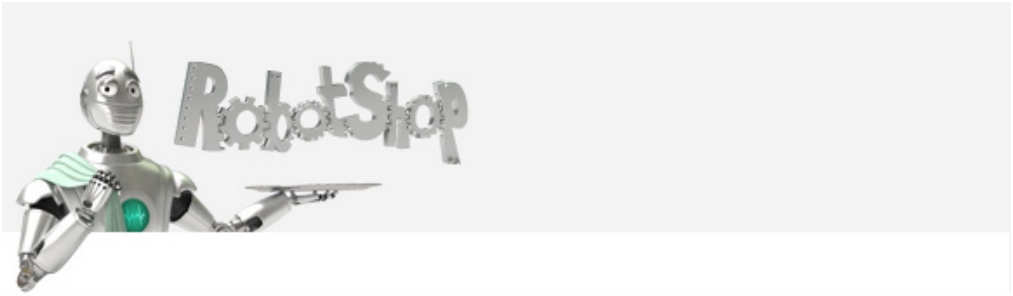
Feel free to contact Parallax Inc. at any time. Please know that if you ever have questions or require technical guidance we have engineers available through our toll free Tech Support line. We also have excellent support materials including downloads and tutorials available on our website.

Thank you,

Julia
Parallax Sales | (916) 624-8333 | (888) 512-1024 | Fax (916) 624-8003 |
www.parallax.com

In reply to:
Hello, I am an Engineering Student at the University of Central Florida. I am working on my final project for my degree which requires research and project documentation. I would like to utilize some images you have posted for the product: Parallax 900-00360 Feedback 360° High Speed Continuous Rotation with your permission. The image will be cited in the document appropriately. Thank you very much!

Figure 71 Parallax Permissions



Your request (T558168) has been updated. To add additional comments, reply to this email.

Félix Antoine (RobotShop)

Jul 23, 09:16 EDT

Hi Brandon!

Thank you for contacting RobotShop. You can utilize our images.
We wish you the best on your project.

Have a nice day!

Félix Antoine
RobotShop inc.

Visit RobotShop.Community

Brandon Gross

Jul 22, 17:45 EDT

Hello, I am an Engineering Student at the University of Central Florida. I am working on my final project for my degree which requires research and project documentation. I would like to utilize some images you have posted for the following products: 60mm Aluminum Omni Wheel, Lynxmotion Aluminum A4WD1 Rover Kit, and 4WD 58mm Omni Wheel Arduino Robot with your permission. The image will be cited in the document appropriately. Thank you very much!

Figure 72 RobotShop Permissions

KI

Kathleen Ives <Kathleen.Ives@kongcompany.com>
Mon 7/22/2019 5:57 PM
Brandon Gross

Hi Brandon,

You can use them! Do you need the high res images of anything specific?

Kathleen Ives
Marketing & Brand Impact

Meet my sidekick Lilly!

KONG
Dogs Need to Play.

16191 Table Mountain Parkway
Golden, CO 80403
KONGcompany.com
Main | 303.216.2626

Figure 73 KONG Basketball Tennis Ball Permissions

CR

Cory Ricondo
Tue 7/23/2019 11:00 PM
rrt@we-worldwide.com

Good evening,

I am a Computer Engineering student enrolled at the University of Central Florida. I am currently working on a paper for my senior design project. I'd like to request permission to use two photos of an Xbox One controller available on the Xbox website.

Links are below,

https://compass-ssl.xbox.com/assets/d3/d8/d3d83d26-d0df-49b8-9ad2-f213425a091d.jpg?n=X1-Wireless-Controller-Black_Gallery_1056x594_02.jpg

https://compass-ssl.xbox.com/assets/e3/34/e3348687-9b7e-48dc-a33a-be4f22dade88.jpg?n=X1-Wireless-Controller-Black_Gallery_1056x594_04.jpg

Thank you and have a great day,

Cory Ricondo

MR

Microsoft Media Relations <rapidresponse@we-worldwide.com>
Wed 7/24/2019 10:47 AM
Cory Ricondo; Microsoft Media Relations <rapidresponse@we-worldwide.com>

Hi Cory,

Any images you find may be used. In order to get permission to use Microsoft copyrighted content, please see below for instruction. Also, check out this site for more details: Microsoft News Center

- You must reference the full product name
- Link to the original Microsoft content (if applicable)
- Include the following statement "Used with permission from Microsoft"
- The content must not be used offensively.

Please reference the Use of Microsoft Copyrighted Content if you have any questions, and to learn about Microsoft's copyright content.

Best regards,
Joel

...

Figure 74 Gamepad image permissions from Microsoft [47]

References

[1] "IEEE Code of Ethics," IEEE, 2019. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed 30 7 2019].

[2] *IPC-2221,* Northbrook: IPC, 1998.

[3] *IEEE 802.15.1,* IEEE, 2005.

[4] M. Barr, *Barr C Coding standards,* Barr Group, 2018.

[5] "Google Python Style Guide," Google, 2019. [Online]. Available: http://google.github.io/styleguide/pyguide.html. [Accessed 29 7 2019].

[6] *IEEE 829:2008,* IEEE, 2008.

[7] *IEEE 1540,* IEEE, 2001.

[8] *IEEE 1872-2015,* IEEE, 2015.

[9] *ISO/IEC/IEEE 42010:2011,* IEEE, 2011.

[10] *IEEE 1012,* IEEE, 2016.

[11] "Robocup," [Online]. Available: https://www.robocup.org/.

[12] " Youtube VEX," [Online]. Available: https://www.youtube.com/watch?v=A8daR6qBw3M.

[13] "Youtube Stanford," [Online]. Available: https://www.youtube.com/watch?v=fXsB7fXcWO8..

[14] R. A. Brooks, "A Robust Layered Control System," Massachusetts , 1985.

[15] M. Mukherjee, "An Introduction to the Common Types of Medieval Catapults," [Online]. Available: https://historyplex.com/types-of-catapults.

[16] "Bluetooth Basics," [Online]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics/all.

[17] introbotics, "The Guide to Bluetooth Modules for Arduino," [Online]. Available: https://www.intorobotics.com/pick-right-bluetooth-module-diy-arduino-project/.

[18] SparkFun, "Bluetooth Basics," [Online]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions.

[19] H. Soffar, "Wi-Fi Direct uses, advantages and disadvantages," 25 June 2016. [Online]. Available: https://www.online-sciences.com/technology/wi-fi-direct-uses-advantages-and-disadvantages/.

[20] F. Member, "Arduino Forum," [Online]. Available: https://forum.arduino.cc/index.php?topic=434811.0.

[21] ElectronicDesign, "BLE v4.2: Creating Faster, More Secure, Power-Efficient Designs—Part 1," [Online]. Available: https://www.electronicdesign.com/communications/ble-v42-creating-faster-more-secure-power-efficient-designs-part-1.

[22] Genstattu, "The LiPo Battery Characteristics and Applications," [Online]. Available: https://www.genstattu.com/blog/the-lipo-battery-characteristics-and-applications/.

[23] "Electropedia," [Online]. Available: https://www.mpoweruk.com/nicad.htm.

[24] "Battery and Energy Technologies," [Online]. Available: https://www.mpoweruk.com/leadacid.htm.

[25] Recom, "Recom DC/DC Converter," Recom, [Online]. Available: https://recom-power.com/pdf/Innoline/R-78E-1.0.pdf.

[26] [Online]. Available: https://www.build-electronic-circuits.com/kicad-vs-eagle-2018-comparison/.

[27] "Arduino IDE," Arduino, 2019. [Online]. Available: https://www.arduino.cc/en/main/software. [Accessed 29 7 2019].

[28] "Atmel Studio," Microchip, 2019. [Online]. Available: https://www.microchip.com/mplab/avr-support/atmel-studio-7. [Accessed 29 7 2019].

[29] M. Matera, "Fast PID," Github, 2017. [Online]. Available: https://github.com/mike-matera/FastPID. [Accessed 29 7 2019].

[30] "PID Library," Arduino, 2019. [Online]. Available: https://playground.arduino.cc/Code/PIDLibrary/. [Accessed 29 7 2019].

[31] "Adafruit Servo Driver Library," 2012. [Online]. Available: https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library. [Accessed 29 7 2019].

[32] "Arduino servo library," Arduino, 2019. [Online]. Available: https://www.arduino.cc/en/reference/servo. [Accessed 29 7 2019].

[33] "Servo Easing," Github, 2019. [Online]. Available: https://github.com/ArminJo/ServoEasing. [Accessed 29 7 2019].

[34] "Adafruit NeoPixel Überguide," [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-uberguide.

[35] vsar, "NVIDIA Forums," [Online]. Available: https://devtalk.nvidia.com/default/topic/1050237/jetson-nano/jetson-nano-wifi-usb-adapter/.

[36] kangalow, "Jetson Nano + Intel Wifi and Bluetooth," [Online]. Available: https://www.jetsonhacks.com/2019/04/08/jetson-nano-intel-wifi-and-bluetooth/.

[37] doxygen, "wpa_supplicant / hostapd," [Online]. Available: https://w1.fi/wpa_supplicant/devel/p2p.html.

[38] metageek, "MetaGeek," [Online]. Available: https://www.metageek.com/training/resources/understanding-rssi.html.

[39] Amazon, "Amazon," [Online]. Available: https://www.amazon.com/Belkin-BE112230-08-12-Outlet-Power-Protector/dp/B000J2EN4S/ref=sr_1_4?keywords=power+surges&qid=1563049819&s=gateway&sr=8-4.

[40] Amazon, "Amazon," [Online]. Available: https://www.amazon.com/AmazonBasics-6-Outlet-Surge-Protector-2-Pack/dp/B014EKQ5AA/ref=sr_1_3?keywords=power%2Bsurges&qid=1563054089&s=gateway&sr=8-3&th=1#HLCXComparisonWidget_feature_div.

[41] Amazon, "Amazon," [Online]. Available: https://www.amazon.com/gp/product/B01LXN7MN3/ref=ox_sc_act_title_1?smid=AA0YO4F2UD50F&th=1.

[42] Amazon, "Amazon," [Online]. Available: https://www.amazon.com/ALITOVE-Converter-5-5x2-1mm-100V-240V-Security/dp/B078RT3ZPS/ref=sr_1_1_sspa?keywords=5V+4A+%284000mA%29+switching+power+supply&qid=1563063546&s=gateway&sr=8-1-spons&psc=1.

[43] "Pybluez," Github, 2018. [Online]. Available: https://github.com/pybluez/pybluez. [Accessed 29 7 2019].

[44] "Jetson Nano GPIO in C," Nvidia, 23 5 2019. [Online]. Available: https://devtalk.nvidia.com/default/topic/1052379/jetson-nano/how-to-use-gpio-in-c-language/post/5342398/#5342398. [Accessed 29 7 2019].

[45] "Jetson GPIO," Github, 2019. [Online]. Available: https://github.com/NVIDIA/jetson-gpio. [Accessed 29 7 2019].

[46] M. Currey, "Bluetooth Modules," 18 February 2016. [Online]. Available: http://www.martyncurrey.com/bluetooth-modules/.

[47] "Pixy2 Overview," [Online]. Available: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:overview.

[48] Amazon, "Amazon," [Online]. Available: https://www.amazon.com/ALITOVE-Converter-5-5x2-1mm-100V-240V-Security/dp/B078RT3ZPS/ref=sr_1_1_sspa?keywords=5V+4A+%284000mA%29+switching+power+supply&qid=1563063546&s=gateway&sr=8-1-spons&psc=1.

[49] "www.cnet.com," 27 July 2019. [Online]. Available: https://www.cnet.com/products/sharp-lc-32lb150u-32-class-31-5-viewable-led-tv/.

[50] "TrackerKCF Class Reference," [Online]. Available: https://docs.opencv.org/3.4.6/d2/dff/classcv_1_1TrackerKCF.html#afc69677f498909662ceeb87476b7ebf7.

[51] J. Jongerius, "Measuring Lens Field of View," [Online]. Available: https://www.panohelp.com/lensfov.html.

[52] A. Saha, "Read, Write and Display a video using OpenCV (C++/ Python)," [Online]. Available: https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/.

[53] A. Rosebrock, "Find distance from camera to object/marker using Python and OpenCV," [Online]. Available: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/.

[54] A. Rosebrock, "Measuring distance between objects in an image with OpenCV," [Online]. Available: https://www.pyimagesearch.com/2016/04/04/measuring-distance-between-objects-in-an-image-with-opencv/.

[55] "How to measure distance between 2 objects in a video?," [Online]. Available: https://answers.opencv.org/question/177732/how-to-measure-distance-between-2-objects-in-a-video-edited/.

[56] "Image Moments," [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/moments/moments.html.

[57] S. Mallick, "MultiTracker : Multiple Object Tracking using OpenCV (C++/Python)," [Online]. Available: https://www.learnopencv.com/multitracker-multiple-object-tracking-using-opencv-c-python/.

[58] S. Mallick, "Object Tracking using OpenCV (C++/Python)," [Online]. Available: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/.

[59] S. Mallick, "GOTURN : Deep Learning based Object Tracking," [Online]. Available: https://www.learnopencv.com/goturn-deep-learning-based-object-tracking/.

[60] "About," [Online]. Available: https://opencv.org/about/.

[61] "OpenCV," [Online]. Available: https://en.wikipedia.org/wiki/OpenCV.

[62] "TensorFlow," [Online]. Available: https://en.wikipedia.org/wiki/TensorFlow.

[63] "Torch (machine learning)," [Online]. Available: https://en.wikipedia.org/wiki/Torch_(machine_learning).

[64] "PyTorch," [Online]. Available: https://en.wikipedia.org/wiki/PyTorch.

[65] "Tensor," [Online]. Available: https://en.wikipedia.org/wiki/Tensor.

[66] "Caffe (software)," [Online]. Available: https://en.wikipedia.org/wiki/Caffe_(software).

[67] "Convolutional neural network," [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[68] "Long short-term memory," [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory.

[69] "CMake," [Online]. Available: https://en.wikipedia.org/wiki/CMake.

[70] "OpenCL," [Online]. Available: https://en.wikipedia.org/wiki/OpenCL.

[71] "CUDA," [Online]. Available: https://en.wikipedia.org/wiki/CUDA.

[72] "OpenCV Contributions," [Online]. Available: https://github.com/opencv/opencv_contrib.

[73] "OpenCV with CMake," [Online]. Available: https://gist.github.com/SSARCandy/fc960d8905330ac695e71e3f3807ce3d.

[74] "Artificial neural network," [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.

[75] "cv::TrackerKCF Class Reference," [Online]. Available: https://docs.opencv.org/3.4.6/d2/dff/classcv_1_1TrackerKCF.html#afc69677f498909662ceeb87476b7ebf7.

[76] "90 Degree Locking Hinge," [Online]. Available: https://www.amazon.com/dp/B07BHK87PD.

[77] "180 Degree Locking Hinge," [Online]. Available: https://www.amazon.com/dp/B07CTGDY3C/ref=psdc_511240_t1_B001DT4Y8G.

[78] "Laminate Flooring," [Online]. Available: https://www.homedepot.com/p/TrafficMASTER-Natural-Hickory-7-mm-Thick-x-8-03-in-Wide-x-47-64-in-Length-Laminate-Flooring-23-91-sq-ft-case-360731-10249/305171172.

[79] "Metal Cup Hooks," [Online]. Available: https://www.amazon.com/ECKJ-Pieces-Screw-Hooks-Plated/dp/B07DN72KXC/ref=sr_1_1_sspa?keywords=cup+screw+hooks&qid=1563663157&s=gateway&sr=8-1-spons&psc=1.

[80] "1/4" Mesh," [Online]. Available: https://www.amazon.com/Clear-Mesh-Netting-Material-Aquarium/dp/B07G765TSK/ref=sr_1_3?keywords=clear+mesh&qid=1563662334&s=gateway&sr=8-3.

[81] "Nvidia Jetson Nano vs. Raspberry Pi," [Online]. Available: https://www.maketecheasier.com/nvidia-jetson-nano-vs-raspberry-pi/.

[82] "Nvidia Jetson Nano Review," [Online]. Available: https://www.arnabkumardas.com/platforms/nvidia/nvidia-jetson-nano-review-and-benchmark/.

[83] "Jetson Nano Developer Kit," [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[84] "How to Choose Sunlight-Readable LCD Monitors," [Online]. Available: http://nauticomp.com/choose-sunlight-readable-lcd-monitors/.

[85] N. Fedorov, "DisplayPort vs HDMI vs DVI vs VGA," [Online]. Available: https://www.avadirect.com/blog/displayport-vs-hdmi-vs-dvi-vs-vga/.

[86] "Get to know your Xbox One Wireless Controller," [Online]. Available: https://support.xbox.com/en-US/xbox-one/accessories/xbox-one-wireless-controller.

[87] "Xbox Wireless Controller - Black," [Online]. Available: https://www.xbox.com/en-US/xbox-one/accessories/controllers/xbox-black-wireless-controller.

[88] "BasketBall Court Markings," [Online]. Available: https://courtfloors.com/basketball-volleyball-game-markings.html.

[89] "Aluminum Sheet - 3003, 5052, 6061," [Online]. Available: https://www.metalsdepot.com/aluminum-products/aluminum-sheet.