Robot Basketball

Brandon Gross (EE), Suvrat Jain (CpE), Cory Ricondo (CpE), Mathew Schneider (CpE)

Dept. Of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The project is centered around the design of a Robot Basketball Arcade game. The arcade basketball arena sits on top of a tabletop and one player can pick up a controller and move 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.**

***Index Terms* — Arcade, Basketball, Computer Vision, Embedded Systems, Entertainment, Robotics**

I. Introduction

This project is proposed in the spirit of the 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, one human player can compete against the clock by controlling the robot to move and shoot the basketball. However, due to perception and coordination problems that come from remotely operating robots, the player 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 overall goal in this project is to create an arcade-style entertainment system that is both robust and intelligent. The product can fit on typical foldable tables and is playable by one person at a time. The system is designed modularly such that different subsystems can be designed, created, and tested independently without disassembling the entire system. The robot can collect and launch the ball into a scale hoop with high accuracy and precision. The robot is quick to traverse the court such that the player is always actively engaged with the game. The system assists the player by performing calculations to increase shot accuracy. The game displays information to the player including game type, score, and other useful debugging information.

II. System Overview

The Project is split into three primary systems: Arena, Robot, and Game. 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 Arena system encompasses all things related to the basketball court, basketball, physical frame structure, and computer vision. The Computer Vision subsystem is used to determine the position and orientation of the robot on the court. It also tracks the position of the ball. The Game System involves taking data in from the player and displaying information such as game and robot status, instant replays, and high-level robot control functionality. The architecture is shown in Figure 1 System Architecture.

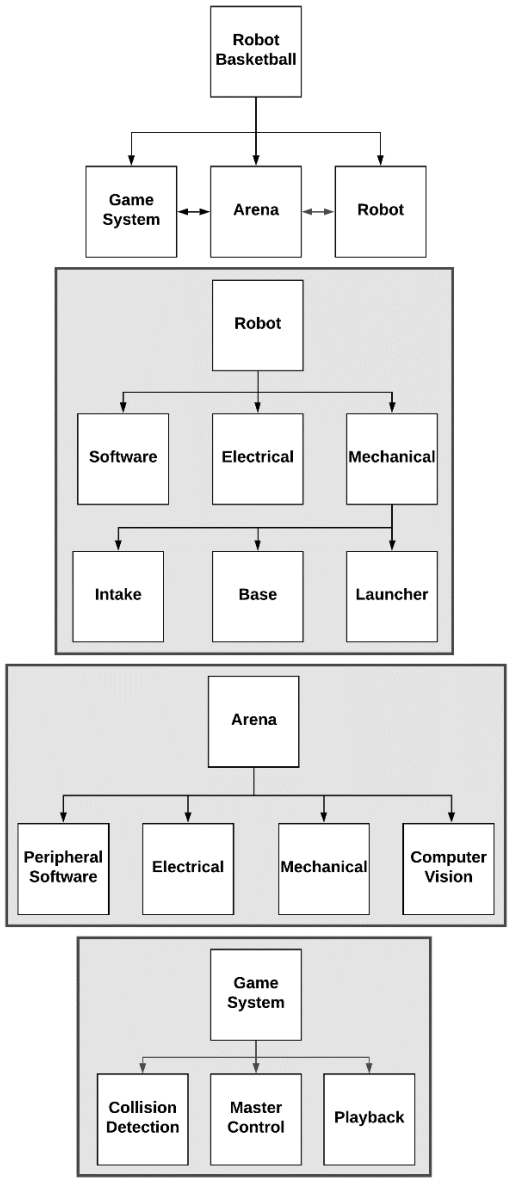

Figure 1 System Architecture

III. Robot

The Robot subsystem is comprised of all the components required to pick up and launch a ball from various places on the court. The robot acts as an Input-Output 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. The rendered model of the robot is shown in Figure 2.

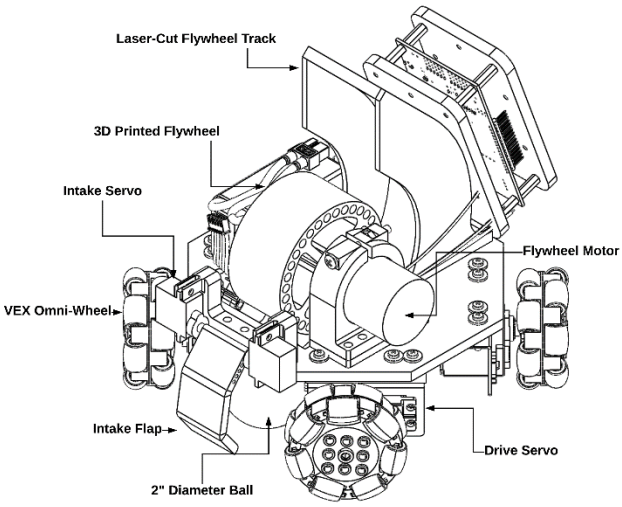

Figure 2 Rendering of robot designed in SOLIDWORKS

A. Mechanical Overview

Base

The mobile robot base is the locomotion component of the system. The design chosen is the 4-wheel holonomic design. Each of the four wheels are powered by a Parallax High-Speed Continuous Rotation servo with feedback. The servo's max speed is 160RPM. The continuous rotation servo contains a built-in open-loop velocity-control motor driver which reduces complexity in the PCB. Further, each servo contains a 910Hz PWM absolute position sensor that can be utilized for accurate positional control or verifiable velocity control. This is important because the holonomic motion requires precise speed control to achieve motion in straight lines. The Servos can be powered up to 6.5V and stall at 1.2A. The kinematics for this motion is shown below. The high-level rotational derivations can be seen in Figure 3.

$$v = V_t + w \times r$$

$$v_w = v_p = v \cdot u$$

Where v is the requested robot velocity, V_t is the translational velocity, w is the angular velocity, and r is the position vector from the base frame to the wheel frame. V_w is the wheel velocity, v_p is the velocity wheel parallel to the wheel, and u is the unit vector between the base frame and the wheel frame. These values are calculated on the robot with component calculations specific to each of the motor's wheel mounting locations and orientations. Scaling factors are also required to ensure that the output of the wheel velocity correctly matches the speeds of the wheels.

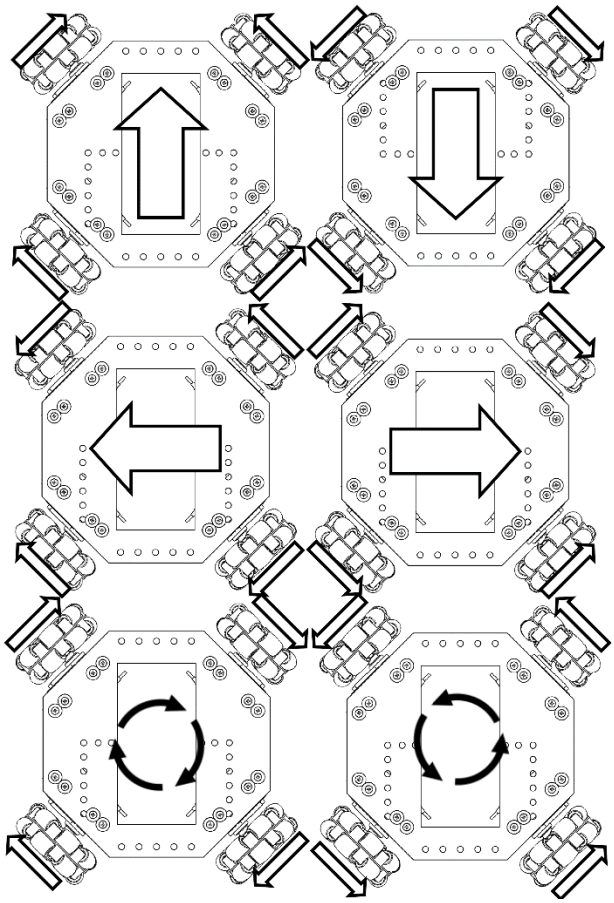

Figure 3 Robot Holonomic Drive Kinematics

Launcher

The launcher is a flywheel mechanism that grips the ball from the intake and rotates it around a track at high RPM. The flywheel is controlled by a 3600 RPM 12V DC motor with an IR break-beam sensor for RPM feedback. The required launch range for the flywheel is 1 to 5 ft but the current implementation is capable of further distances. The ball is launched at a 45-degree angle. The launcher is controlled with closed-loop velocity feedback through a PID controller with feed-forward to overcome the wheel's inertia. Velocity commands are ramped to overcome very large errors that can cause high overshoot in the system. However, the system needs to have low rise time (<1 second) and have low steady-state error (<20 RPM) to maintain responsiveness and high accuracy.

Intake

The intake for the robot must be able to pick up a ball and transfer it to the launcher mechanism. The intake mechanism is mounted on the front of the robot and is controlled by two 9g micro-servos. The intake has three operating states: disengaged at 0 degrees, holding at 110 degrees, and launch at 180 degrees. The servos are mounted oppositely of each other so particular care is taken that they receive the same (or opposite) positions at any given time. The operating procedure is to have the intake disengaged when the player is going to intake a ball. When the ball is within a close distance of the robot, the intake transitions to holding mode that contains the ball while the robot navigates around the court. Finally, when the player is ready to launch, the intake is fully engaged which pushes the ball through a holding flap and engages the ball with the flywheel.

B. Electrical

The electrical system for the robot exists primarily to support the devices required for operation of the base, launcher, and intake systems described above. That is: 4 continuous rotation servos with feedback, 2 9g micro servos, and a larger 12V DC motor with an IR break-beam sensor. Additionally, the robot is not tethered and thus requires a wireless communication system and a battery. A microcontroller unit with an additional PWM generator is included to support the logic side of the robot system. Although the servos only require power and PWM signals, the flywheel requires an additional motor controller to output varying power to achieve different launch speeds. The power and logic systems are shown in Figure 4 and Figure 5.

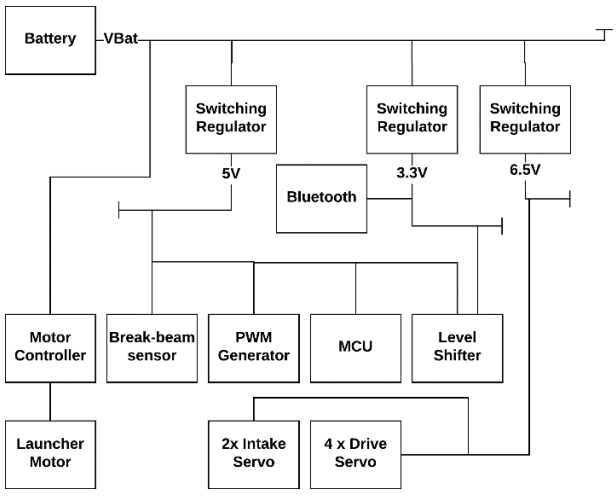

Figure 4 Robot Power Block Diagram

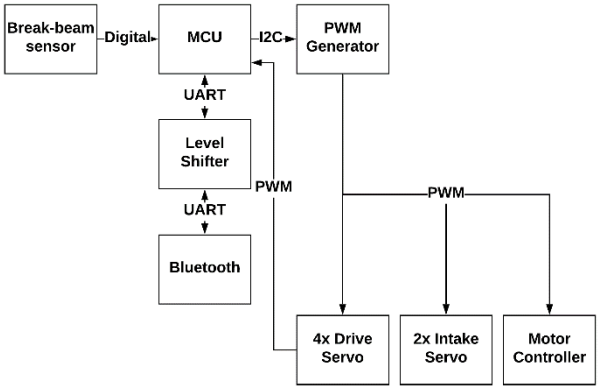

Figure 5 Robot Logic Block Diagram

Flywheel Motor Controller

A single channel of 12V input and up to 3-amp output is required to control the flywheel motor. A DRV8871 is chosen to support this device because it has an input range of 6-24V and output current of up to 3.6A with a single channel. This device utilizes two digital signals to control. The combination of the two signals results in variable output speed and direction.

PWM Generator

A PWM Generator is included to increase modularity in the system. This reduces the number of PWM ports required on a microcontroller and allows the system to scale to increased actuations in the future. Further, a separate module used to generate the signals allows the microcontroller to focus on other tasks instead of utilizing timers for PWM generation. A PCA9685 device is chosen to create up to 16 PWM signals out of a single I2C interface. This generator is set to a constant 50Hz output to support the frequencies required by the servos. Thus, 6 PWM signals are utilized with servos, and an additional 2 signals are used to control the DRV8871. This leaves 8 PWM channels spare for additions later.

Microcontroller

The robot requires an onboard processor to perform the necessary calculations for control. These tasks require real time executions. An ATMEGA 328P is employed as the master controller for the robot. This device acquires commands from the communication module, processes inputs from sensors, and writes outputs to the PWM generator. I2C, UART, PWM inputs, and digital inputs are required to support the chosen peripheral devices. There are 4 PWM feedback lines, 1 digital feedback line, two I2C pins, and 2 UART pins. This leaves 2 digital pins spare, and 3 Analog pins spare for future additions. The chosen microcontroller has a plethora of firmware support and is one of the cheaper models available.

Communication

The robot must have a communication system on board and receive data over a wireless link. An RN-42 Bluetooth 2.1 device is chosen to integrate with the robot system. This device is chosen because of its wide support across many devices, and its relative simplicity compared to newer Bluetooth devices. SPP (UART) profile is utilized to transmit ASCII characters to and from the microcontroller. The Robot expects a specific packet design at a consistent frequency to avoid timeouts which result in the robot stopping. The packet design contains a starting character, a header section, a payload section, and an ending newline character. The data sections contain combinations of hex characters to enable high-speed parsing. Control is through an 8-bit (2 hex) command corresponding to -100% to 100% joint speed per axis. The overall packet is 32 Bytes including overhead

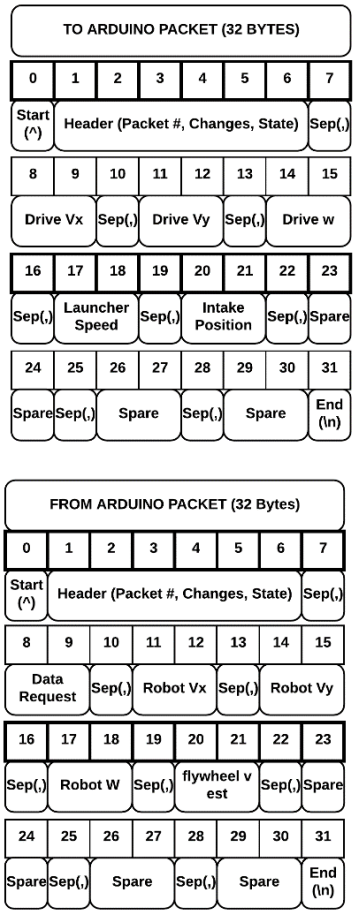

Figure 6 Packet Design to Robot and from Robot

Battery

The mobile robot requires a battery system for power. A 2200mAh 3S LiPo is chosen to power the system. A 3S LiPo has a voltage range within 11.1 to 12.6V. The chosen battery is capable of 50C discharge which is equivalent to 50A which is more than required to power all the systems when stalling. The estimated runtime for the battery within normal playtime conditions is about 2 hours. However, this runtime is dependent on playing conditions. A LiPo voltage monitor is included externally to determine the remaining charge of the battery and indicate when the voltage drops below an acceptable range.

PCB

The PCB design primarily exists to connect all the discussed components in a clean, easy to use, and robust package. The final PCB design is printed as a 2-layer 100x60mm board which contains many surface-mount devices and an array of header pins to interface with the peripheral devices such as motors, servos, sensors, and batteries. An FTDI device is not included on the PCB, but the PCB contains a header-port for easy access to program and debug the device.

C. 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 hard real time to ensure that inputs and outputs are processed in a context that does not affect the fidelity of the system. The design does the following: updates inputs, processes the system's data, and updates the outputs. 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. The system layer contains classes defining robot-specific systems such as the intake, launcher, and drive systems. The object layer contains abstract code for actuators, sensors, and state processing. 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 layering system is shown in Figure 7.

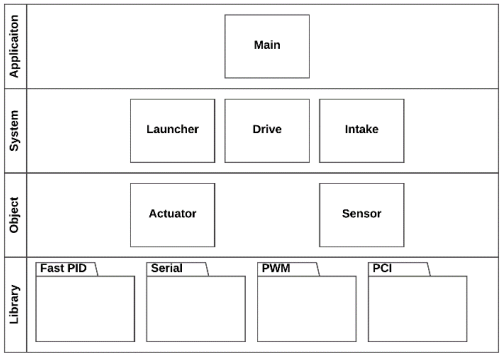

Figure 7 Robot Software Architecture design

IV. Arena

The Arena subsystem is the frame that the robot can be placed on and all the components required for basketball gameplay. It contains the computer vision component of the project for robot and ball tracking. The subsystem contains all the player experience including lights, sounds, and display. The model is rendered in Figure 7.

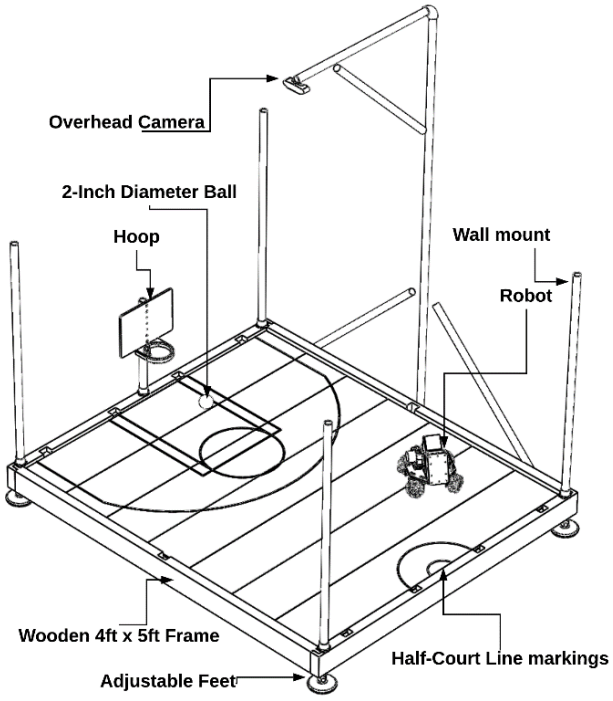

Figure 8 Arena model render from SOLIDWORKS

A. Mechanical Overview

The size of the arena is 4 ft width by 5 ft length by 3 ft height which is scaled from a real court. Walls surround the arena to prevent a rogue object from flying in or out of the arena and hitting someone or something it is not supposed to. The court component involves the actual floor of the arena that the robots drive around on and it contains basketball court markings. The ball for this project is a 2-inch diameter tennis ball themed as a basketball. The hoop is mounted to the frame and is set to a diameter that is feasible for the launcher to remain accurate under most conditions. Each time a basket is made, the score for the game must be updated, thus the hoop must sense 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).

B. Electrical

The electrical system for the Arena subsystem is a straightforward design that utilizes as many off-the-shelf components as possible to eliminate the need of an additional PCB. The electrical system is composed of power sourced from a standard wall outlet using a power strip / surge protector. This device powers the TV, and two 5V AC-DC convertors that in turn power the Jetson Nano, and LED Lights. The remaining devices source power from the Jetson Nano's GPIO pins. The electrical block diagram is shown in Figure 9.

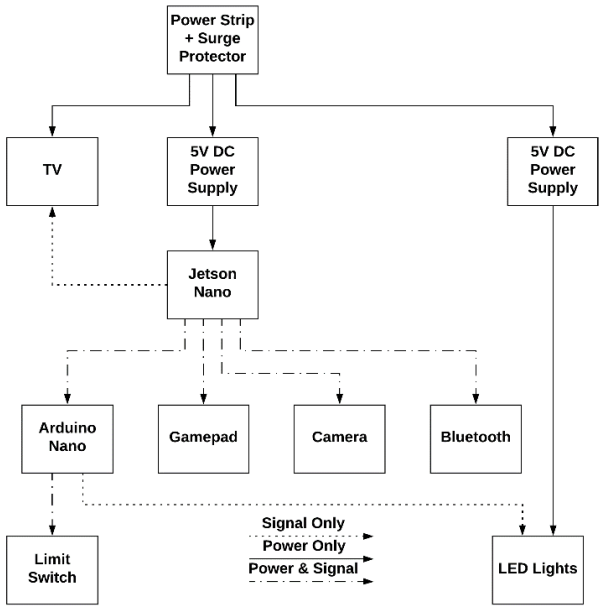

Figure 9 Arena Electrical System Block Diagram

Display & Sounds

The arena utilizes a TV for displaying information to the player through the game engine and playing sounds to enhance the experience. The resolution of the TV is 720P and the size is 32-inches. The specific TV is a Sony Bravia KDl-32BX310 display which is chosen due to high availability and low cost.

Camera

The camera for this project is used for computer vision to track the robot and ball. The camera is placed 5 feet above the arena so that it may see the entire arena, robot, and goal. The camera is utilized to analyze locations of the robot and ball such that the assistive features of the game can be employed. A high framerate is critical to the application. Additionally, a high resolution is important such that tracking is accurate. Thus, a Logitech C920 1080P widescreen camera is utilized. The 1080P mode provides 30 frames per second at a high resolution which can achieve the locational precision required for robot tracking.

Gamepad

The gamepad is the first thing a player touches when playing the game, so it's important to utilize a gamepad that feels familiar. 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. An Xbox One controller is utilized due to the driver availability, popularity of the controller, and tactile feedback functions.

LED Lights

NeoPixel RGB LEDs are used 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 has also been used for testing integration between all the objects involved with the arena. The design uses NeoPixel Strips around the walls of the arena that are connected to the microcontroller and they receive signals from the arena about what to display. The LED lights attracts viewers from far away and engages the player when achievements are made.

Controller

This controller performs calculations for computer vision, Bluetooth communication from arena to robot, and runs the game engine which performs calculations for robot location, calculations for force to launch the ball, and shows video on the display. The controller must interface with the additional peripherals in the system such as the TV, Gamepad, Camera, Bluetooth, and LED devices. This is a significant undertaking for a traditional microcontroller; a Jetson Nano is chosen to employ a traditional operating system to support the large amount of data processing. The Jetson nano contains a quad-core ARM 1.43GHz processor and a 128-core GPU. The team quickly discovered that the nature of a non-Realtime operating system prohibits the control of LED-lights and hoop sensors despite the availability of GPIO pins. Thus, the system also incorporates an Arduino Nano to control the NeoPixels and hoop sensors. A UART protocol is utilized between the Jetson and the Arduino to set LED states and determine whether a goal has been made. The gamepad and Camera connect to the Jetson Nano through a USB interface, and the Bluetooth module is an M.2 Intel 8265 WIFI/Bluetooth module.

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 is established through an Intel 8265 device which contains Bluetooth capabilities. The packet design is discussed in section III, Robot-Electrical-Communication.

C. Computer Vision

The computer vision portion detects and tracks the location of all moving objects on the court. Prior to any detection, background subtraction is performed to remove most of the frame noise. The background is static, so prior to startup, the objects are removed from the court and several frames are added to a background subtraction model. After the model is created, the robot and ball are placed onto the court. The frame that the remaining calculations are performed on is created through masking the foreground that is separated from the background model (Figure 9 top right & middle left). Two color patches are placed onto the robot to reduce complexity of this system. In this case, blue and pink patches are put on the top of the robot that the computer vision looks for to determine robot position and orientation (Figure 9 middle right). After color filtering, the software detects various contours for the squares (Figure 9 bottom left). It also filters for orange and then performs a Hough transform operation to detect the ball. After the patches are detected and the centroids are found, the position and orientation of the robot is found through vector arithmetic (See Figure 9 bottom right). The computer vision software utilizes the various functionality available with OpenCV. If additional computational time is required, the OpenCV libraries can be CUDA-enhanced to take advantage of parallel processing. The entire data flow is described in Figure 10.

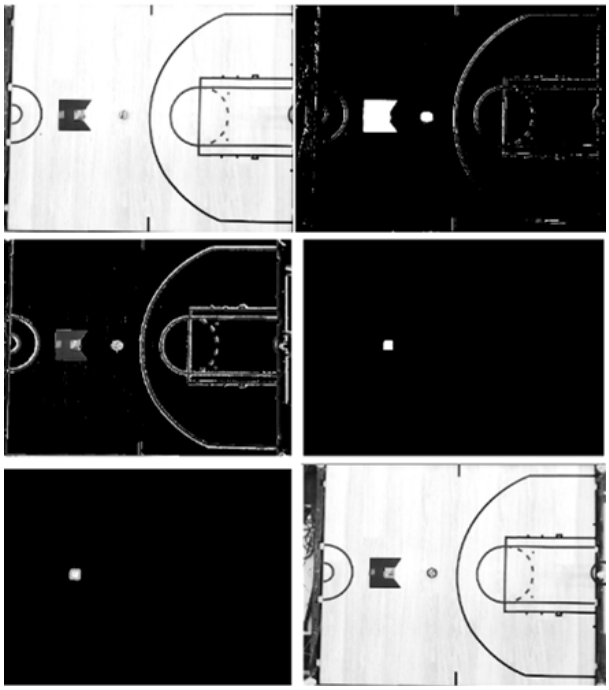

Figure 10 Computer Vision Results

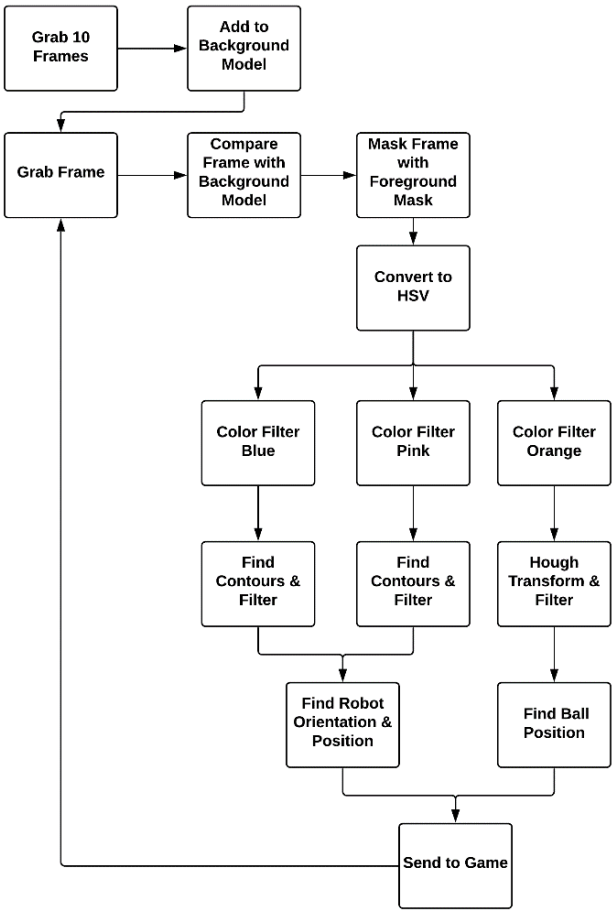

Figure 11 Computer Vision Flowchart

D. 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. The peripheral software's primary job is to map data from the Game to the Robot through sockets. The Game sends data through a local socket which is then parsed and passed to the robot through a Bluetooth RFCOMM socket. The software actively connects to the robot, and ensures a valid connection is available. Due to the nature of asynchronous communication, flow control is required to avoid buffer overflow on the Robot or the peripheral software. The peripheral software handles sending & receiving data by verifying when the robot is ready to receive data, and when valid data is available to send. The peripheral software also controls LED states by listening to the Game system and sending commands to the Arduino Nano through a hardware serial interface. Finally, it listens to the Arduino Nano for changes in the hoop-sensor status and passes that data to the game system.

V. Game

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 aid the user by showing a 2D virtual representation of an environment. The game system is the primary software arm on Arena system, but it can act independently from the Arena system. The three main components of the game system are the collision detection, video playback and robot control.

A. Engine

The game engine handles a few tasks overall and acts as a hub for data to flow in and out of. It handles data visualization such as showing a mockup of where the physical components are such as the robot and ball are on the field or playing an animation when a shot is made or missed. The game engine will also need to be capable of both 2D and 3D animation to carry out its tasks. The chosen engine, PyGames, is picked primarily because of its availability on the Jetson nano AARCH64 architecture.

B. 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. 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. The collision detection will be set up in such a way that as the robot moves closer to the designated wall area of the arena, the controller will begin to vibrate, and the intensity of the vibration will increase the closer that the robot's position to the wall is. The system also fully prevents controls that drive the robot into a wall

C. 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 holding 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 knocked down pins.

D. Master 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. 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. The Robot Master control accumulates all the data relevant to robot function and determines the high-level functionality of the robot.

VI. Conclusion

Over the course of the last two semesters, the team researched, designed, built, and tested a basketball-based arcade game. The robot is an agile holonomic drive capable of picking up a 2-inch diameter basketball and launching it on a 1:15 scale half-size basketball court. Controlling the robot can be difficult, so player assistance is added through automatic speed calculations and angle adjustments. The robot's position and orientation, as well as the ball's position, is found using computer vision processing on video data from an overhead camera. The data from the robot and computer vision systems are transferred to a game system that displays information to the player. The game system also ingests data from the player through a gamepad to pass through to the robot. The final product is a fun and engaging game that players can pick up the gamepad and start driving the robot to pick up and launch a ball to score as many points as possible prior to the clock time running out

Acknowledgement

A special thanks to Professor Chan and Professor Sukthankar for their insight into the project at various stages. An additional thank you to the aides in the Innovation lab for their help and resources, with a specific thank you to Elizabeth Nogues. Thanks to Nicole Foust for her assistance in creating the basketball court markings on the Arena. Finally, thank you to Birket Engineering for resources and expertise in PCB manufacturing and testing.

References

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

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

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

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

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

[6] B. Gross, S. Jain, C. Ricondo and M. Schneider, "Robot Basketball," 2019.

Biography

Brandon Gross *is an Electrical Engineering Major with a minor in Intelligent Robotic Systems. He will be interning at Walt Disney Imagineering in Spring 2020.*

Suvrat Jain *is a Computer Engineering Major with a minor in Intelligent Robotic Systems. He will be joining in Birket Engineering in Spring 2020.*

Cory Ricondo *is a Computer Engineering Major. He will be entering the workforce in Spring 2020 and is currently looking for Full Time employment.*

Mathew Schneider *is a Computer Engineering Major. He will be joining NAWCTSD full-time in Spring 2020.*