Automatic Gentleman

Authors: Jean Paul Nelson, Logan Kojiro, Juan Pablo Betello: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— A system to deliver ping pong balls to the opposite side of the table into standard sized red solo cups. The system will serve as an artificial opponent in a game of cup pong, tracking the state of the game, and launching balls as appropriate.**

***Index Terms*—Design, Robot, Sensor, Object Detection, Computer Vision, Projectile Dynamics**

1. Introduction

Due to the pandemic, there have been many social distancing practices put in place to reduce the spread of COVID. Many of these guidelines restrict social gatherings like concerts, outings, parties, etc. For college students, this limits a large portion of the activities they partake in. Specifically, parties are nonexistent at this time, and because of this, popular games cup pong cannot be played. The Automatic Gentleman sets out to alleviate this problem. Our goal is to create a robotic opponent that can simulate a full cup pong game. Using computer vision, depth sensing technology, and projectile dynamics, we can create a robot that can detect cups and launch ping pong balls into them with the same accuracy of the average human.

Our robot will simulate a real human game of cup pong. It will provide real-time detection of cup variable formations, and accurate motion planning control, and a consistent ball launching mechanism. The Automatic Gentleman (referred to as AG from here on), will be able to detect the ellipse edge of the cups with 90% accuracy and no duplicates in under 1 second. It will then calculate the position of each cup in the real world coordinate system in order to do motion planning for the ball's flight path. Our launching mechanism will be able to freely rotate +/- 30 degrees to aim and shoot the ball consistently within 1.5" of a detected cups center.

2. Design Requirements

A. Cup Detection

The main requirement to the cup detection algorithm is to identify the ellipses around the edge of cups positioned across a table with 90% accuracy and no duplicates. To achieve this, the detection process is split into 3 steps:

1. Resizing and image type conversion
2. Ellipse Detection
3. Duplicate ellipse filtration

Our cup detection testbench consists of running a series of images through our detection algorithm and displaying the depicted ellipses on top of the original BGR image. We use this visual result to confirm that 90% of the cups have been detected, and that there is a maximum of 1 ellipse detected per cup.

B. Depth Sensing

In order to do the motion planning, we need the positional coordinates of each cup. Using the depth map captured by the Azure Kinect DK camera, along with the 2D-pixel coordinates of the detected ellipses. We can derive the 3D-positional coordinates of the cups in the real world. Our testbench will output the coordinates of each detected cup, which we can verify by positioning the cups in predetermined locations on the table.

C. *Game State*

The main purpose of the UI display and the game state system is to keep track of the progression of the game, particularly what cups have been made and how many shots have been made or missed. The user will be able to manually interact, and if needed, change the game state through overriding cup marking and deciding what shot to try and make next.

D. *Motion Planning*

The main requirement here is based on correctly integrating the cup detection system into the actual gameplay i.e the launcher. The Automatic Gentleman should be able to intake the location of the cups and plan shots accordingly. This will have several sub-requirements such as shooting one ball at a time and maintaining a minimum level of net accuracy.

E. *Ball Delivery*

The requirement of this system is to accurately deliver the ball to the target cup. To do this, the horizontal angle will have to be adjusted to within 1 degree of the calculated angle, and the exit velocity will need to be within 1% of the calculated exit velocity from the motion planning phase. To maintain accuracy of angle, the system will also have to keep track of its angle relative to the table at all times.

The testing for angle will consist of a series of given angles that will be verified by hand whether the machine achieved the correct angle. These will be done in batches to ensure that precision is not lost over time. At the end of a game, we should still see less than 1 degree drift from the target angle.

The velocity testing will also consist of giving target velocities to the system and directly measuring by hand. This will be done using a mobile phone slow motion camera and a grid of squares with known dimensions. Using this setup of grid squares behind the launcher, we can count the frames and squares crossed in the slow motion footage to extrapolate the velocity the ball was traveling at the time it left the launcher.

F. Game Accuracy

The overall goal of the accuracy will be to be within one

standard deviation of a person over the course of a game. This will be measured by counting the number of shots it takes to complete the game. The rules of one game are as follows:

- The game will start with 10 cups in a pyramid formation at the far end of a table
- Once 4 cups are made and removed from the formation, the remaining 6 cups should be rearranged, again in a pyramid formation
- After a further 2 cups are made and removed, the remaining 4 cups should be placed in the 'diamond' formation (one single cup at the bottom, then 2, then 1)
- Finally, when only 2 cups remain they should be placed in the 'gentleman's' formation: 2 cups placed in a line at the end of the table

We will have each human participant play at least 4 games. For each game, they will count the total number of shots it takes them to make a ball in all 10 of the cups. Upon the completion of the system, we will run the AG on the same setup and count the number of shots it takes the robot to hit all the cups.

G. *Ball Storage*

We have yet to determine the number of balls the average person needs to complete a game, but the AG should be able to store at least half of that: enough such that over the course of a normal game, a player would only have to refill it once.

3. Architecture and/or Principle of Operation

A. Jetson TK1

The Jetson TK1 is the main computing source for our project. The two cameras will connect to the Jetson via USB, while the arduino will be connected via the UART serial port and the tablet display via HDMI. Our application executable will run on the Jetson and will control the capturing of images on the cameras. The image processing and gameplay control will be handled by the Jetson.

B. Arduino

As mentioned above, the arduino is connected to the Jetson via UART a serial port. The arduino will receive the positional coordinates of the targeted cup and then adjust the launcher angle to aim at the desired cup. The arduino will control the aiming of the barrel and launching of the ball by adjusting the PWM signals of the GPIO pins connected to the motors. The angle will be adjusted using a stepper motor under the launcher assembly controlled by an EasyDriver stepper motor driver. Software on the Arduino will keep track of the previous angle and steps as to not lose track of its absolute position. The launch velocity will be controlled via pwm and an L298 bridge IC. It will run for only as much time as needed for a single launch then stop.

C. Azure Kinect SDK

The images of the opponents cups will be captured by an Azure Kinect DK camera equipped with IR, Color, and Depth cameras/sensors. The camera will be positioned opposite the opponent across the table, angled down towards the cups. Using the Azure Kinect SDK, we will capture both BGR color images and depth map data. The sdk also is equipped with library functions to aid in the transformation of a 3 channel color image to a 4 channel image (depth value is the 4th channel).

D. Cup Detection with OpenCV

We are using the C++ OpenCV library to capture and process the images. Images are captured and processed at the start of each turn. We use an open-source CNC Ellipse Detection [1] library to actually detect the ellipse, because the standard ellipse detector in OpenCV is too slow for our 1 second time limit.

E. *UI Display Tablet*

We are going to be implementing a user interface on a touch LCD display through which the user can interact with the Automatic Gentleman and interfere with the game if need be. The UI will have the base function of allowing the user to manually target a specific cup, but will also contain support features such as a settings page and marking/unmarking cups.

4. Design Trade Studies

These are some of the design considerations we took while developing this or project.

A. Jetson TK1

We chose to use the Jetson TK1 as our main computing source over the Raspberry Pi because of its CPU and GPU specifications. The Azure Kinect camera requires high CPU and GPU power to efficiently utilize its advanced camera and depth sensors.

B. CNC Ellipse Detector

OpenCV has standard circle and ellipse detection libraries available in the C++ package. I tested a suite of pictures containing ellipses using the standard library, and the closer I came to our desired camera angle, the detection time increased exponentially, hitting up to several minutes in some cases.

After some digging, I found a research project, and its github project, on speeding up ellipse detection time. When I tested the same suite of pictures using this new library, I was able to get the detection time down to ~800ms, which is way below our 1 second threshold

C. Azure Kinect Camera

We chose the Azure Kinect DK camera because of its advanced cameras and sensors, primarily the depth sensors. We could have made our lives easier by mounting both cameras directly above the cup formations. This would have eliminated the need for ellipse detection since the bird's eye view of a cup is a circle and much easier to detect. However, an important requirement for our project was to simulate a real game as much as possible. Since we wanted to position our camera across the table and at an angle, accurate depth sensors were crucial in order to generate the 3D positional coordinates. The Azurer Kinect SDK also provides us with a powerful library to aid in the transformation of 3-channel images to 4-channel images.

D. Image Processing and Additional Filtering

The Azure kinect camera has several resolutions to capture images in. While testing the new CNC Ellipse Detection algorithm, I noticed that our results were more accurate the smaller the picture was. I eventually settled on downscaling the image to 512x512 pixels to get better results. Even so, the dictator would average 7 duplicate ellipses per cup. Duplicate ellipses in this case are defined as multiple ellipses whose center points are within 15 pixels from each other. After reading documentation of the ellipse detector, I altered several parameters to get duplicate ellipse down to an average of 2-3 per cup. This is where we came to the conclusion that an additional custom filtering step was necessary. This got us down to our requirement of 1 ellipse per cup.

E. Hardware Driver (Arduino)

We chose to use an arduino because of its ease of use and integrated GPIO. While we could have used a separate Raspberry Pi for hardware control, we did not need the additional computation power, and the Arduino is much easier to use. We also opted to have a separate hardware controller rather than using the Jetson's GPIO to make the separation of labor easier. Additionally, we were familiar with using arduinos to interact with hardware so it was one less thing for us to learn.

F. DC Motor and Driver

Initial calculations show that for a range of approximately two meters, and a driving wheel radius of 2cm, we would need an RPM of around 350 to launch at around 4 m/s. Due to the light weight of the ball, a 6v,11.5k RPM motor from Pololu will be sufficient to launch the ball. The 6v mark makes it easy to power with 4 AA batteries in series as well. In order to more precisely control the RPM, we may end up undervolting or gearing down the motor to get more fine grained control over the lower RPMs as it is unlikely we will need the higher RPMs this motor can generate. Ideally we would use closer to a 6v 1000rpm motor but most of the 1000 rpm motors are a much higher voltage.

G. Stepper Motor and Driver

We chose to use Brian Schmalz's EasyDriver stepper driver and a standard 4-wire bipolar stepper motor. This driver is well documented and will be able to run the stepper motor on a 12V power supply (either from wall power or a separate 12v battery assembly)

H. Equations

(1): v_i = initial velocity;

$$|v_i| = \sqrt{distance * g}$$

5. System Description

Our overall system is 3 main components: Cup Detection, Motion Planning, and Projectile Dynamics.

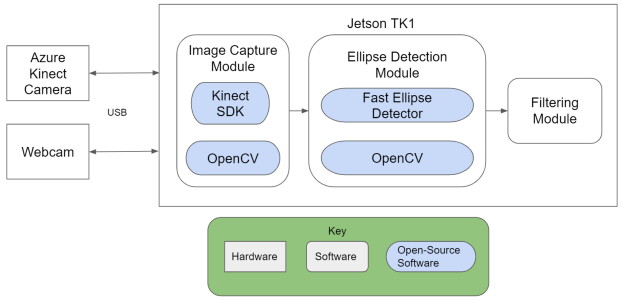


Fig. 1. Subsystem diagram of Cup Detection Modules

A. Cup Detection

As shown in Figure 2, the cup detection subsystem is split into 3 main components:

- Image Capturing
- Ellipse Detection
- Additional Filtering

In the image capturing component, our software will utilize the Azure Kinect SDK to interface with our application. With the sdk, we can stream and capture both the color and depth images from the Azure Kinect camera. We will use OpenCV to interface with the USB webcam to capture images from that camera. All captured images will be converted to OpenCV::Mat structs, which is the required input type for the CNC Ellipse Detector.

In the Ellipse Detection component, we utilize the CNC Ellipse Detector functions to efficiently and accurately detect the cup edges. Instead of using the entire executable, I stripped the open-sourced code to the main functionality that we needed. This reduced a lot of the dependency overhead. A large part of the ellipse detection is black-boxed inside the detector functions, however, many of the parameters we needed to provide took lots of testing to get accurate results.

The additional filtering component is the final step before generating the position of the each detected cup. In this step, we filter out any ellipses whose center points are within 15 pixels of another. This step proved to be vital in generating accurate position coordinates.

B. Position Coordinate Generation

This component serves as the inputs for the motion planning process. This component utilizes the array 2D pixel coordinates of the detected ellipses and the captured depth map to identify where in our 3D space each cup is located. We utilized the Azure Kinect SDK to combine the 3 BGR channels with an additional depth data channel. With this 4 channel image we can determine how far away each cup is by checking the each pixel returned from our ellipse detection component.

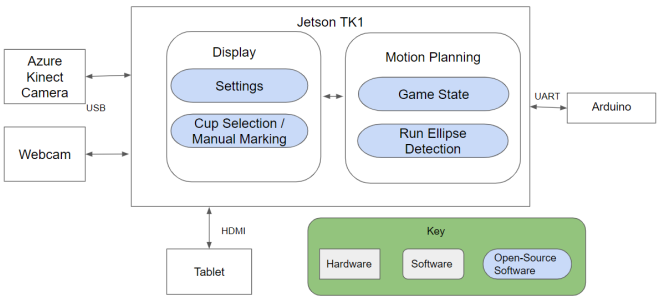


Fig. 2. Subsystem diagram of Motion Planning and Display Modules

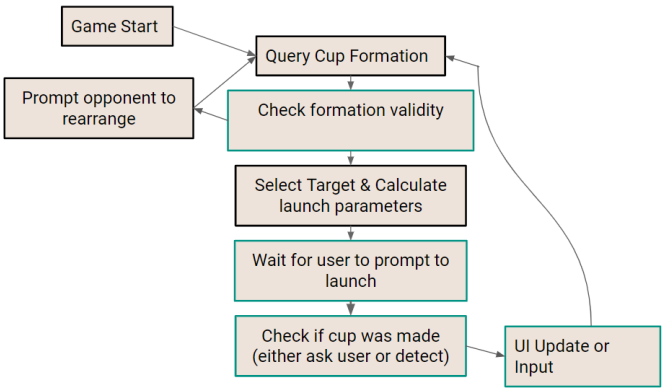


Fig. 3. FSM of game state for motion planning

C. *Motion Planning*

This system will control the overall flow of the game, ensuring proper integration and cycling of the cup detection system as well as maintaining an updated game state. This system will essentially be the software foundation through which the Automatic Gentleman's ability to read and analyze the environment is linked to it actually taking the shots. The finite state machine diagram above represents the implementation of the game progression and has the intent of keeping a game running smoothly while also giving the player opportunities to correct any missteps along the way.

D. *Display*

The display is directly connected to the game state tracking and this is what will be displayed to the user so the progression of the game can be easily seen. However, this won't just be a display but will also take user input and facilitate interaction and interference with the game and with the Automatic Gentleman's performance. The UI will include a method of manually marking or unmarking cups as made in case there are any errors or miscounts. The UI will also include settings to tune the difficulty and manually adjust shot trajectory in case the Automatic Gentleman is over or under performing.

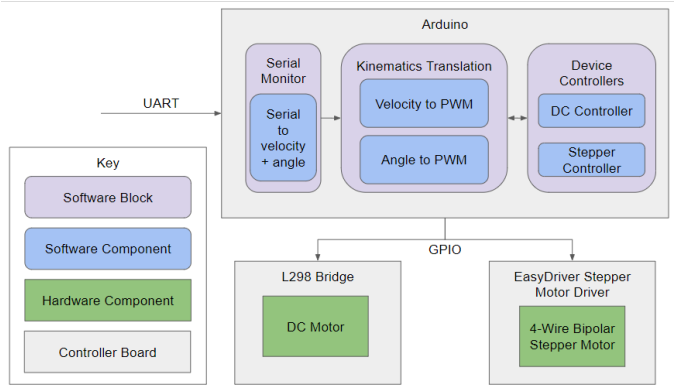


Fig. 4. Subsystem diagram of Launcher Modules

E. *Launcher*

The main computing and driving device in the delivery system/launcher is an Arduino. The Arduino is connected via UART to the Jetson. Figure 3 shows the subsystem diagram of the launching mechanism. On the Arduino, there will be a serial listener that will be ready to receive input from the UART connection. When it receives input, it first parses it into a velocity and angle. From there, the angle will be sent to the stepper controller to rotate the device. In the stepper controller, there will have to be a persistent variable that keeps track of the current position of the system relative to the table so the rotation can be appropriate to the angle we need to line up with the cup. Next, the velocity will be extrapolated to a PWM voltage to supply to the DC motor. This voltage will be adjusted such that the launching wheel angular velocity matches that of our calculated exit velocity. The physical housing can be seen in figure 5. With the balls being fed by gravity from behind the DC launching wheel, the ball will sit in contact with the wheel until it is time to launch. At this /time, the wheels will quickly spin up to the target RPM, propelling the ball out towards the target cup.

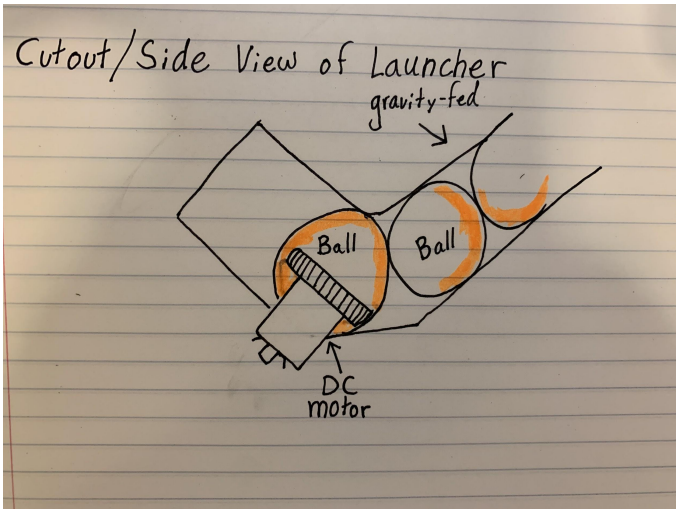


Fig. 5. Side view of launching mechanism

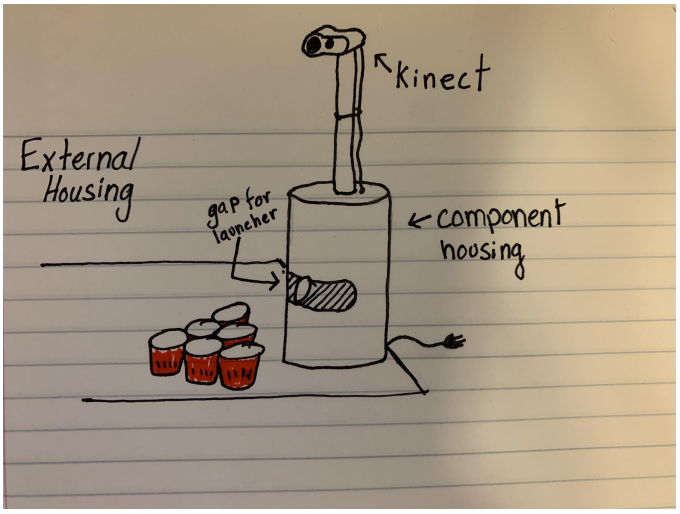


Fig. 6. Concept drawing of external housing + Kinect mount

Project Management

A. Schedule

The Gantt chart is split up into 4 sections: Project Planning, Design and Implementation, Integration and Validation, and Presentation.

During the Project Planning we focused on ensuring that it was possible to accurately detect ellipses from our desired camera angle. In this step is where we discovered the CNC Ellipse Detector. This section was also spent determining the launching mechanism that would be consistent enough for our purposes. We made sure to allot extra time for our Integration and Validation section because we are not all in Pittsburgh. By this section we aimed to have all of our individual components fully functional independently.

One change that we made was to add a couple items related to changing the design in the launcher. This didn't put us too far off track, but it did add a bit of extra work in the next two weeks modeling a physical housing for the new launching system.

B. Team Member Responsibilities

Our responsibilities were split up into our 3 system components:

JP - Implement image capturing and ellipse detection algorithm to detect ellipses around each cup. Use the depth data to generate 3D positional coordinates of each detected cup.

Logan - Extrapolate a relationship of voltage to exit velocity for our DC motors. Design and assemble our physical launching component. Implement arduino application to control and trigger launching mechanism, specifically to rotate the mechanism to point directly at the target cup and send the ball out of the front at the appropriate velocity. Design and implement motor controller circuitry.

Juan Pablo - Design overall software gameplay foundation. Integrate cup detection component to run each turn. Monitor game state and progression. Integrate launching component to fire a single ping pong ball per turn. Create an interactive UI display.

C. Budget

We tried to use as many items that we could get from the inventory or personal inventories as possible. The camera and Jetson were the most expensive items but we were able to secure both of those for free. Figure 7 shows a detailed list of our materials.

A	B	C
Part	**Use**	**Price**
iPong launcher	Launcher	39.95
Stepper motor	Launcher	12.99
Stepper motot bridge SN754410NE	Launcher	7.66
Jetson Nano	Main Compute	95
tablet	UI Display	65
Different stepper Motor	Launcher	15.95
Micro-stepper driver	Launcher	11.78
DC motor driver	Launcher	8.89
DC motor	Launcher	13.53

Fig. 7. Bill of Materials

D. Risk Management

Our biggest risk was in the consistency of our launching mechanism. Logan focused on mitigating this risk by making it his first task in our Project Planning phase. We initially tried to use a real ping pong ball launcher, and strip it down its main components that we would be able to control with arduino PWM signals. This proved to be inadequate. The real ball launchers used a spring loaded mechanism which we would

not have precise control over using an arduino. Additionally, this mechanism had a degree of uncertainty in the exit velocities that was much too high. We had to pivot to design our own custom ball launcher.

Another tisk was the computing power of the SoC we chose. We initially did testing on a personal computer and then a Raspberry Pi. We quickly found out that a Jetson TK1 would be the easiest SoC to get our hands on that had the necessary computing power.

Finally, one last risk was the accuracy of detecting ellipses from an angle using the standard OpenCV libraries. To mitigate this risk, JP focused on this in our first Project Planning phase. We ran multiple tests of ellipse detection from various angles. Based on the detection time and result accuracy, we came to the conclusion that an external ellipse detector was necessary in order to reach our requirements.

6. Related Work

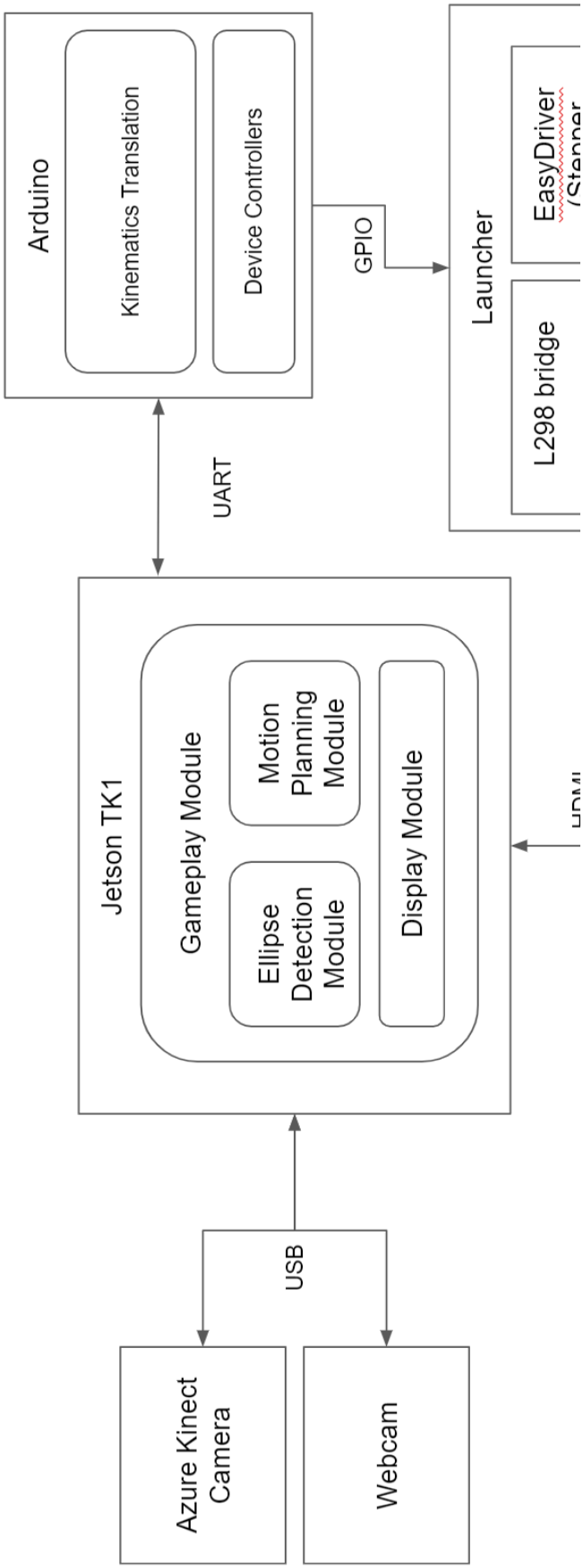
We have found similar projects that touched on independent components of our project. For example, the Stanford ball launchers featured in their 'battle of the bots' 2015 competition. As a class project, students from Stanford were tasked with designing robots to shoot balls into a hoop at a short distance. Many of the designs appear to use a similar approach of a spinning wheel to deliver the balls. With the size of the hoop and the size of the balls they are using it looks like a similar degree of accuracy that we need in our project. The CNC Ellipse Detector was a project centered around speeding up ellipse detection processes. This proved to be vital in ensuring our application ran within our time constraints. The ellipse detector project is open source which made it very useful in our project development. While there is little documentation on a hackathon project from 6 years ago, the Stanford robots are a good indication that this design is on the right track and capable of delivering the accuracy we are looking for.

References

[1] CNC Ellipse Detector Github,. https://github.com/dlut-dimt/ellipse-detector
[2] CNC Ellipse Detector Research Paper,. https://www.sciencedirect.com/science/article/pii/S0031320314001976
[3] Stanford Robots,. https://www.youtube.com/watch?v=fXsB7fXcWO8

Automatic Gentleman	x	x	x	x												
Tasks:	**Week 1**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**	**Week 7**	**Week 8**	**Week 9**	**Week 10**	**Week 11**	**Week 12**	**Week 13**	**Week 14**	**Week 15**	**Week 16**
	Project Planning			**Design and Implementation**						**Integration & Verification**				**Presentation**		
Object Detection																
Build and Run Azure Kinect Camera																
Detect Ellipses from saved images																
Filter ellipses from real time captured images																
Test detection from various cup formations																
Combine real-time image capture with ellipse detection E2E																
Motion Planning / gameplay logic																
distance to velocity																
x/y position to angle																
package data for sending to arduino																
gameplay loop:																
user input to trigger shot (test w button or switch)																
user selects cup number to hit																
Select only from cups that have been detected																
Detect/monitor game state (who's turn is it, what cups have been made, etc.)																
UI screen																

JP
Logan
Juan Pablo
Everyone


Arduino
Kinematics Translation
Device Controllers
GPIO
Launcher
L298 bridge
EasyDriver
UART
Jetson TK1
Gameplay Module
Motion Planning Module
Ellipse Detection Module
Display Module
USB
Azure Kinect Camera
Webcam