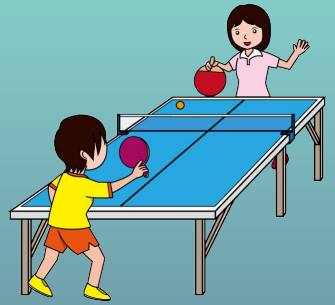
# April Tag Table Tennis Tracker

Company:

LETS BOUNCE

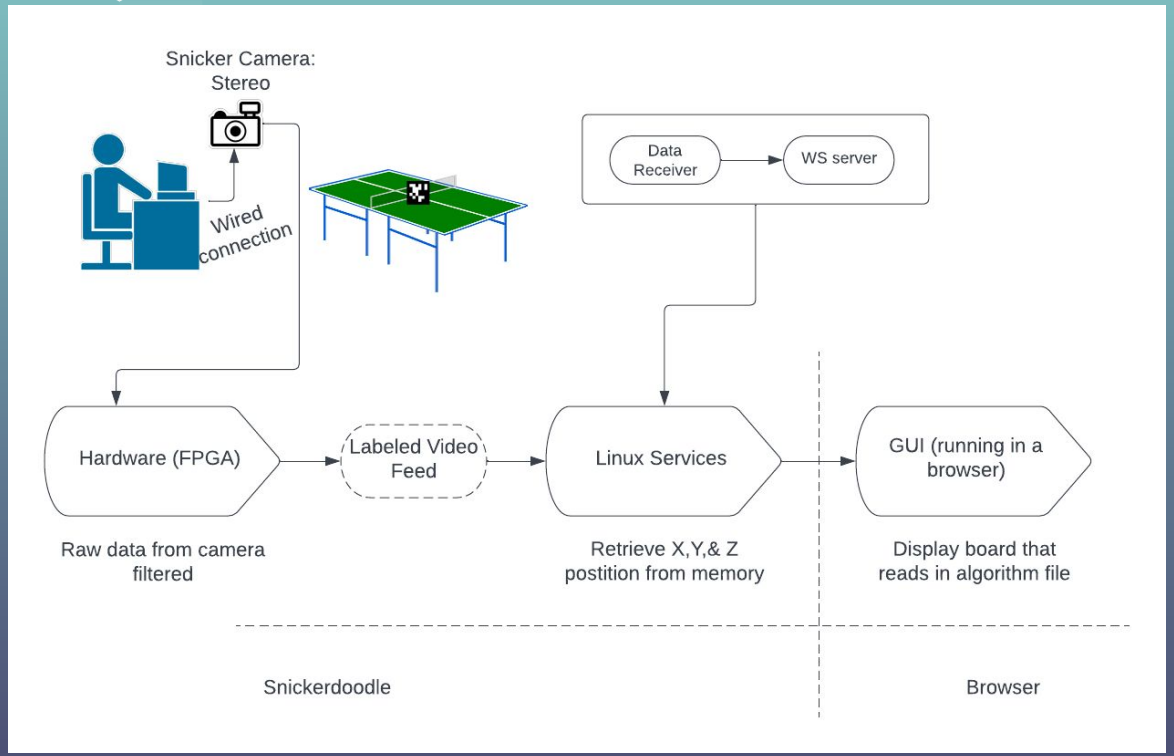
Members: Jen Ellis, Andrew Serra, Josh Skotarczak, Tessa Vincent, and Calvin Yang

# CDR Overview



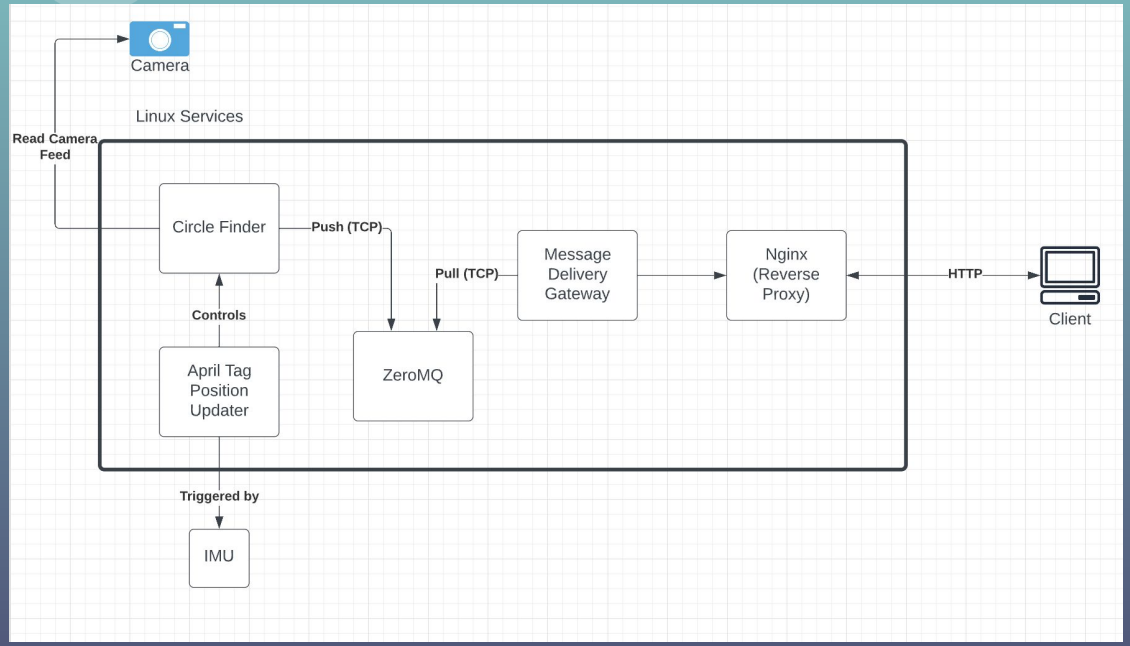
- System High Level Block Diagram
- Problems & Changes
- 3 major components within the system:
  - Hardware/FPGA
  - Algorithm
  - GUI
- Subcomponent within the system:
  - Memory Map(Communication between Hardware and Algorithm)
- Accuracy
- Requirements(JIRA)
- Total Hours /Expenses

# High Level Block Diagram




**Overall Assignment: The United States Tennis Association funded our company to develop a ping pong ball tracker system that accurately determines static ball position within 3D space even when the camera moves**

# Software High Level Block Diagram




- Same data delivery as designed
- Additional control added using the IMU, in the case the camera moves, it will reorient by restarting the Circle Finder Service.
- Change in Client and Proxy connection. The GUI polls a HTTP server.

# Problems And Changes. . .


- Problem : Blob Detection In HDL Did Not Work.

  Solution : Take what works in Blob Detector for FPGA move what's not working to python.

- Problem : Environment Variables

  Solution : Change Court Design, Change April Tag

- Problem : Reading output of the labelling filter

  Solution : Read feed as RGB but read the saved image as grayscale

# Problems And Changes . . .


- Problem : Docker Installation

  Solution : Run all services as Linux services

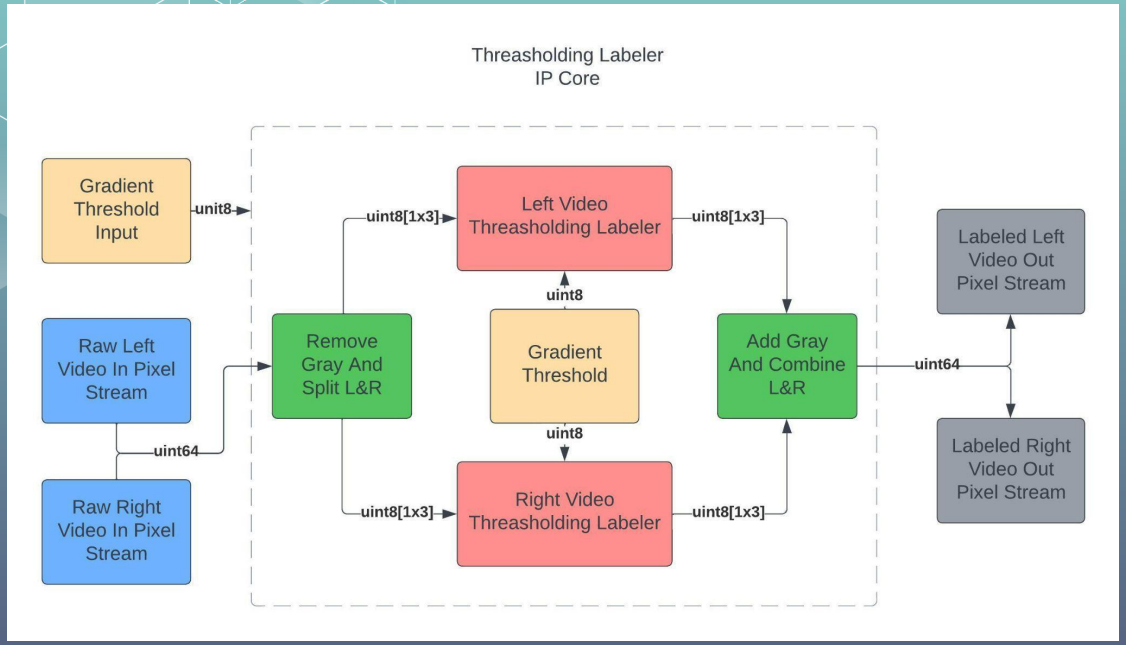
- Problem : RIT Network

  Solution : Some updates and installations got blocked by the RIT network, all updates and installations done on home network.

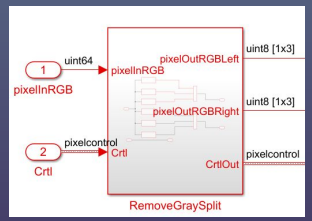
- Problem : Websocket Connections from the GUI

  Solution : Although the software design is the same, instead of using websockets, we GUI is polling an HTTP server to send the next available position data. This means we can only support 1 client at the current state.

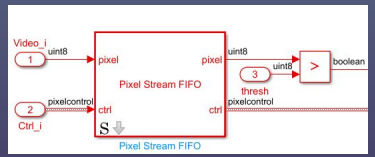
# Hardware/FPGA

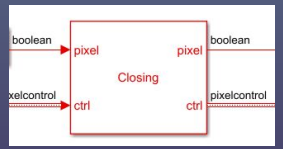


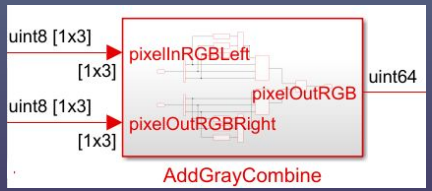

- Remove Gray & Split - Gray channels are removed and the uint64 pixel data stream is split into left and right uint8[1x3] streams.
- RGB Intensity - Gives (0-255) Scalar Value to Each Pixel Value Within Left and Right Video Streams.
- Thresholding - Produces binary image based on set threshold value using relational operator(Greater Than).
- Closing Operation - Morphological closing to remove gaps holes within the binary image.
- Add Gray & Combine- Gray channels are added back and left and right pixel streams are combined into one uint64 data stream for Fusion 2.





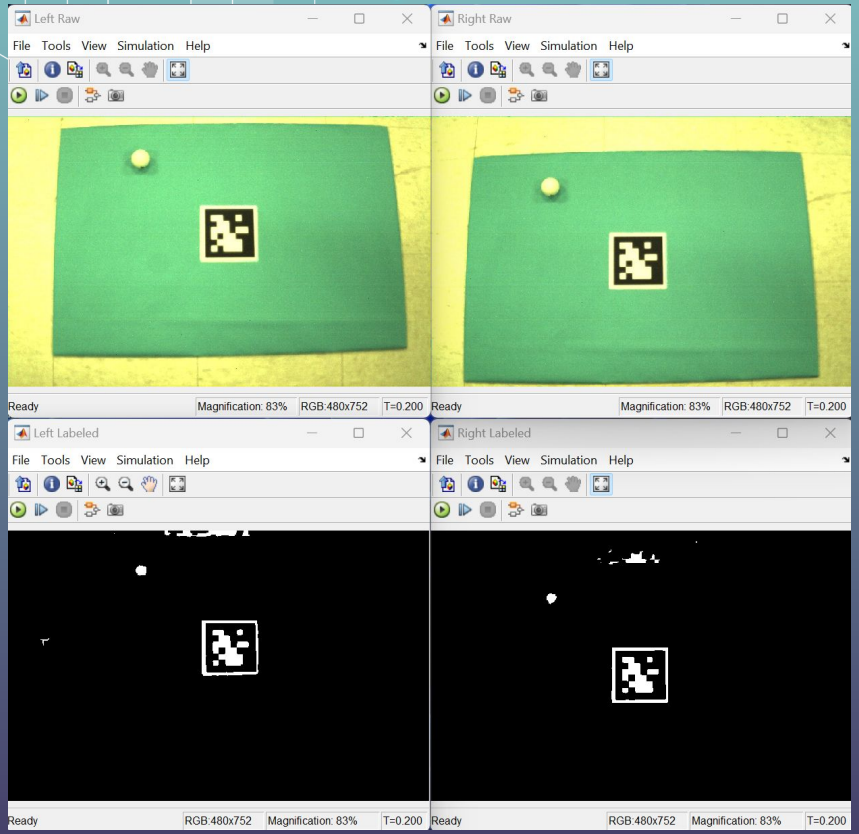


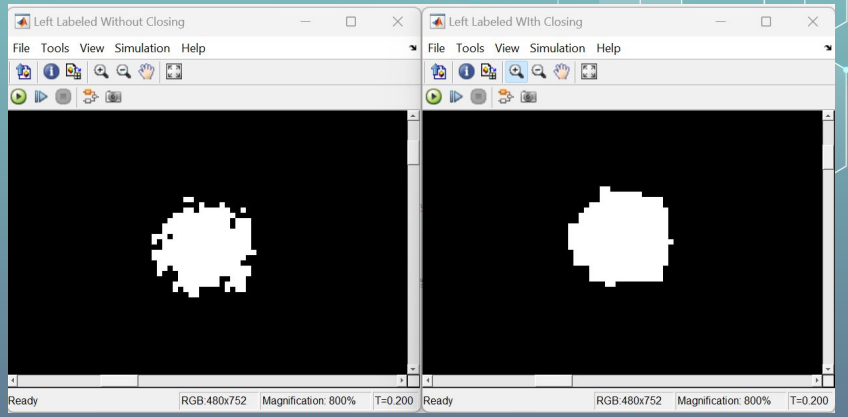



# Simulink Testing

## General Testing




## Morphological Closing



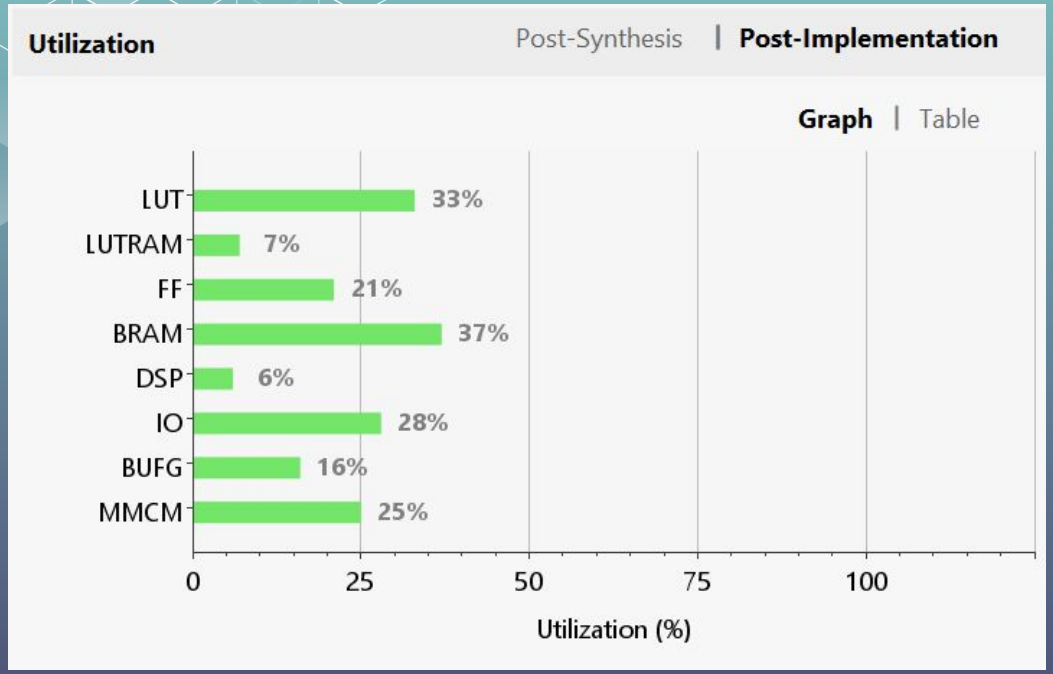

- General Testing This helps test HDL and conceptualize the real-world implementation before IP Core generation.

  Pro - Can threshold to isolate the ball !

  Con - Thresholding also detects the april tag.

- Morphological Closing - Useful for filling small holes in an image while preserving the shape and size of large holes using dilation and erosion.

# Final FPGA Implementation




- LUT - Look Up Table
- LUTRAM - Look Up Table RAM
- FF - Flip Flops
- BRAM -Block RAM
- DSP - DSP Blocks
- IO - Inputs And Outputs
- BUFG - Global Clock Buffer
- MMCM - Mixed-Mode Clock Manager

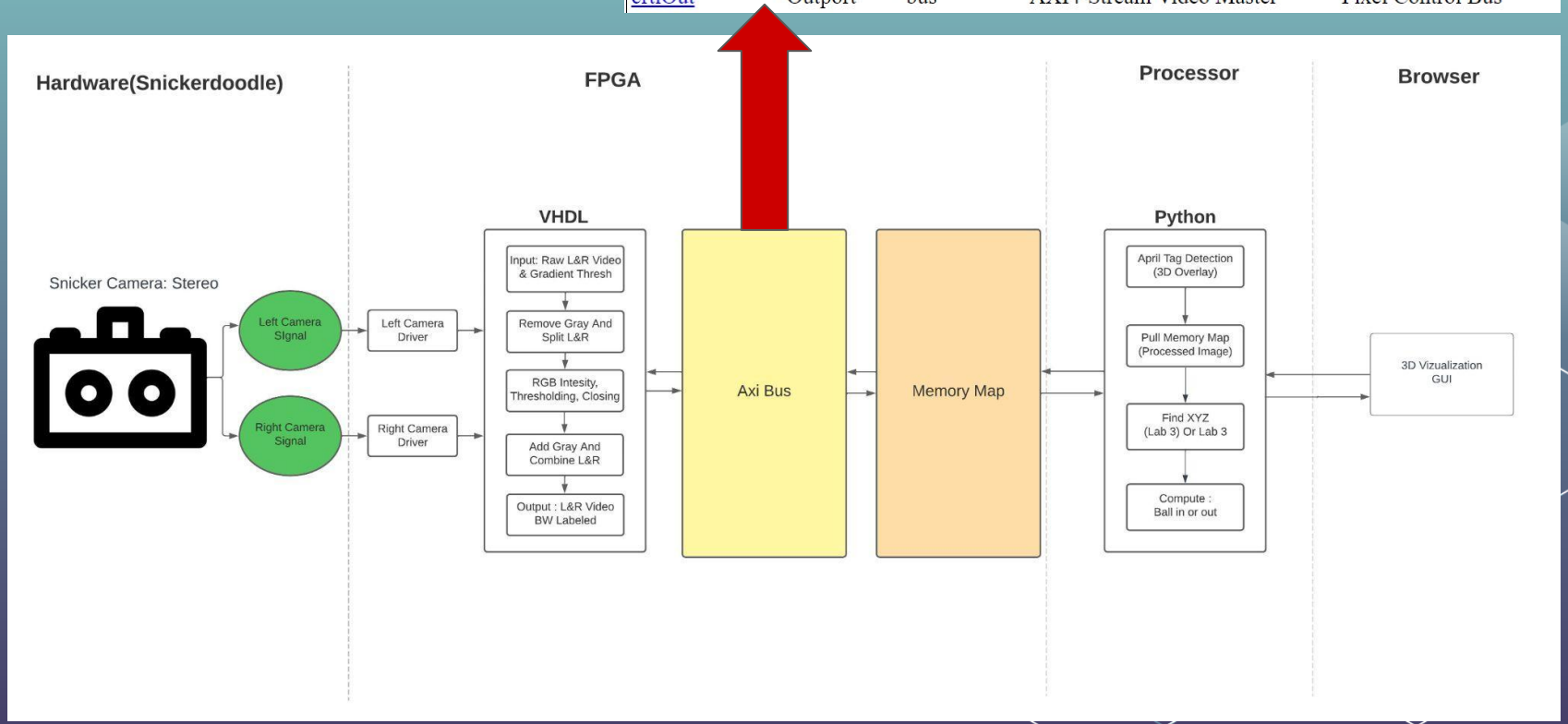
- Total FPGA Utilization - ~21.6%
- Worst Negative Slack - 8.086 ns
- Total Negative Slack - 0

Meaning On Average It Meet Timing Constraints

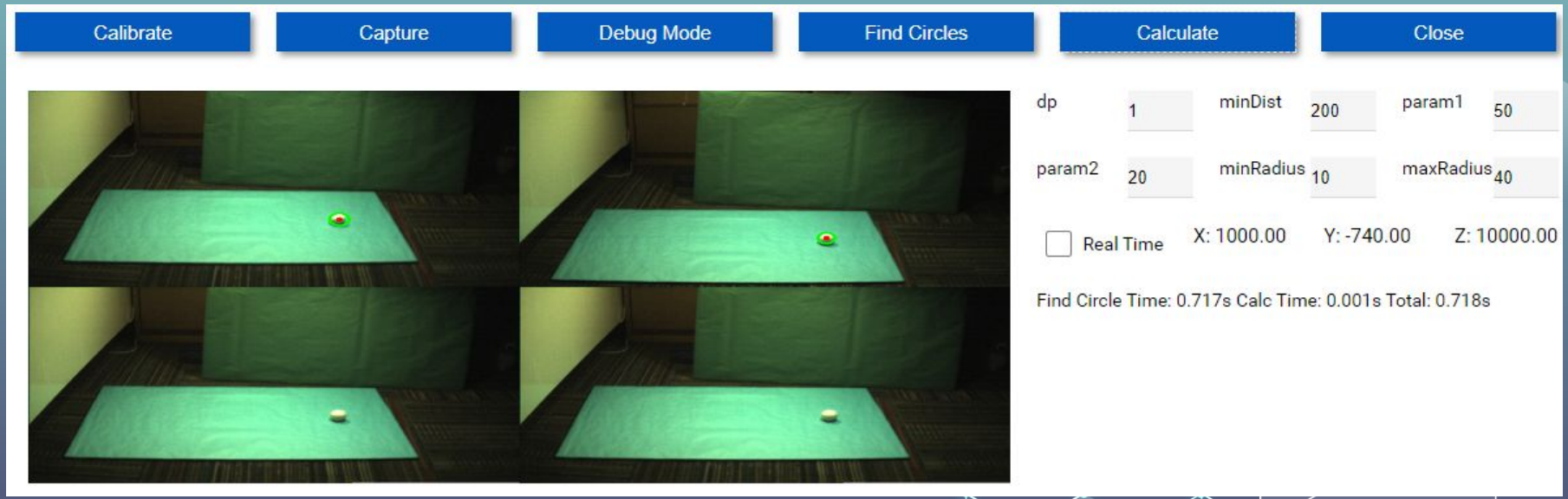
# Memory Map

Target platform interface table:

| Port Name | Port Type | Data Type | Target Platform Interfaces | Interface Mapping |
|---|---|---|---|---|
| pixelInRGB | Inport | uint64 | AXI4-Stream Video Slave | Pixel Data |
| Crtl | Inport | bus | AXI4-Stream Video Slave | Pixel Control Bus |
| GradThresh | Inport | uint8 | AXI4-Lite | x"100" |
| pixelOutRGB | Outport | uint64 | AXI4-Stream Video Master | Pixel Data |
| crtlOut | Outport | bus | AXI4-Stream Video Master | Pixel Control Bus |



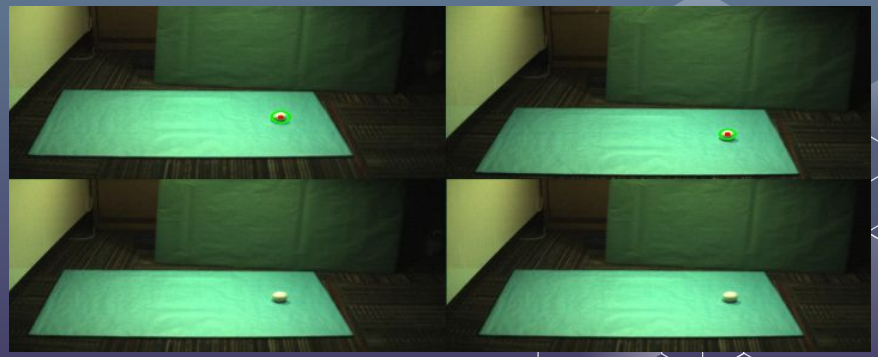
# Lab 3 Algorithm Implementation



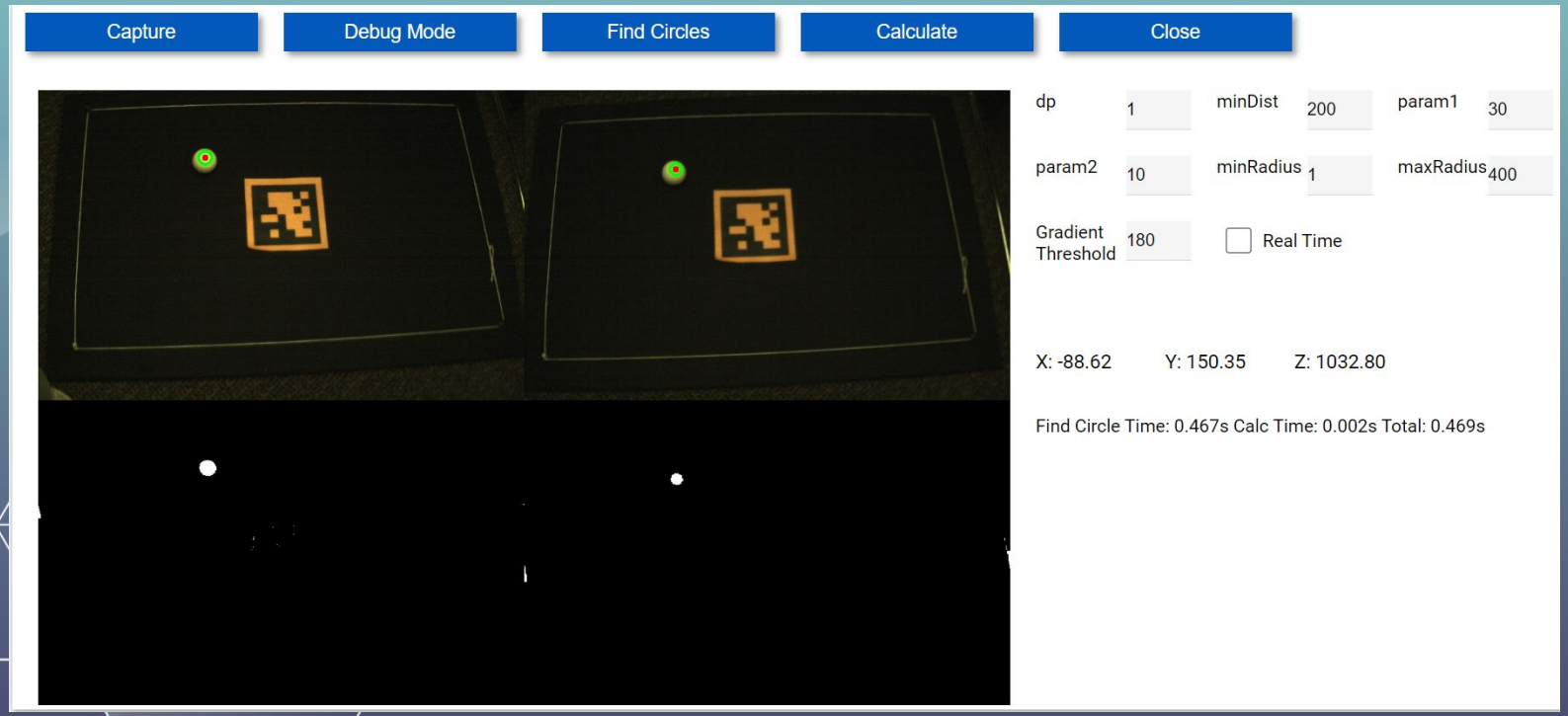
# Lab 3 Variables

- **Hough Circle Detection of Camera Feed with Overlay**
  - Inverse ratio of the accumulator resolution to the image resolution. (dp)
  - Minimum Distance between detected circles (minDist)
  - The higher threshold of the two passed to the Canny edge detector. (param1)
  - The accumulator threshold for the circle centers at the detection stage. (param2)
  - Minimum circle radius (minRadius)
  - Maximum circle radius (maxRadius)

| | | | | | |
|---|---|---|---|---|---|
| dp | 1 | minDist | 200 | param1 | 50 |
| param2 | 20 | minRadius | 10 | maxRadius | 40 |



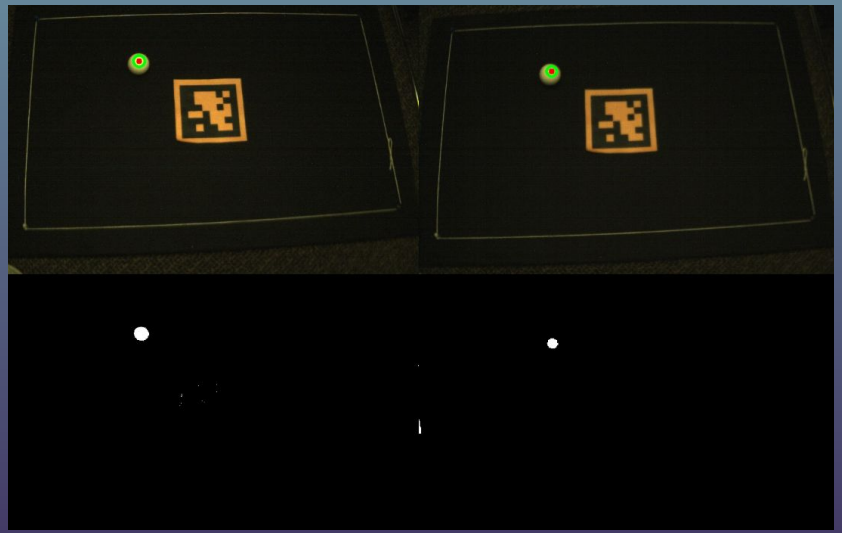
# Labelling Algorithm Implementation



# Labelling Algorithm Variables and Results

- **Hough Circle Detection of Labelling Filtered  Camera Feed with Overlay**
  - Gradient Threshold to alter the intensity of the image (whiteness or blackness).
- **Faster runtime than unfiltered Camera Feed**
  - No Labelling Filter: **0.718s** Total: 0.718s
  - Labelling Filter:  **0.469s** Total: 0.469s

Gradient Threshold 180

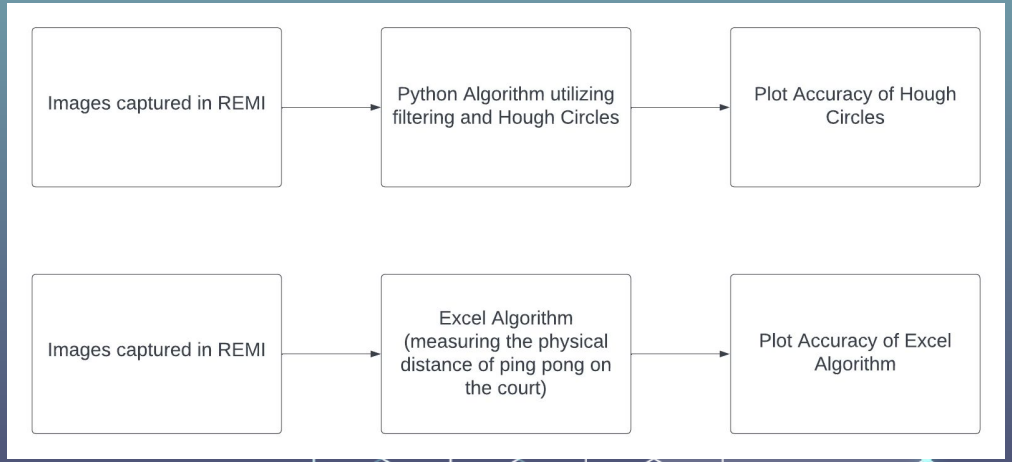


# Purpose of Remi Test Application

- **Allows for isolated testing of blob detector separate from the GUI.**
  - When blob detector in the FPGA failed a backup plan was easily be implemented.
- **Allows for parameters to be adjusted for specific area lighting.**
- **Allows for functions to be tested one at a time to decrease debug time.**
- **Camera Calibration**
- **Gives execution times to show the benefit of using the labelling filter.**
- **Can easily be structured to get data for further testing to assure accuracy.**
- **Gives functional code for the GUI implementation.**

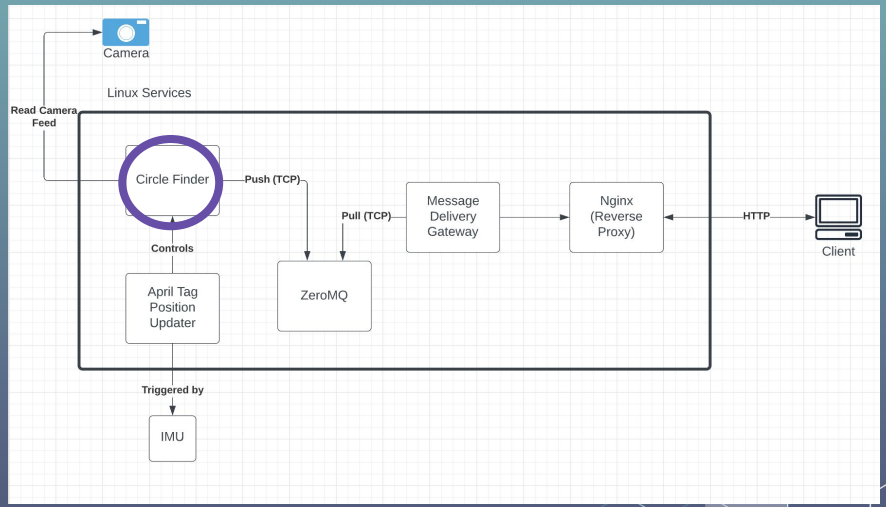
# Further Accuracy Analysis

- Simultaneously comparative data
- Camera at set distance for every test.
- April Tag is in correct orientation.
- Measure coordinate position of ball
- Environmental Variables.
    - Lighting/Shadows
    - Background Color



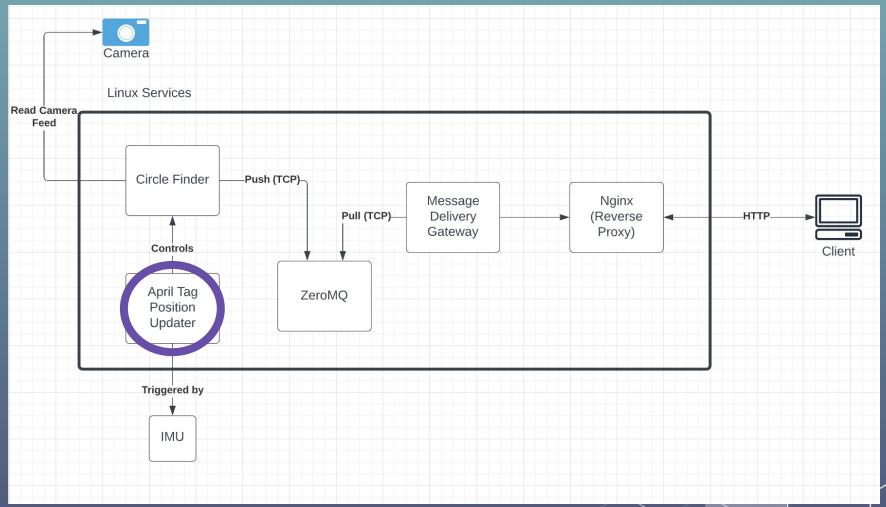
# Linux Services – Circle Finder

- Read the FPGA Labeled Video
- Determine X, Y, and Z coordinates of the ball.
- Determine field bounds by reading the output file written by the Position Service.
- Will restart when needed.
- Detect if ball is in/out.
- Send serialized object containing position and in/out data to ZeroMQ using TCP over port 5555



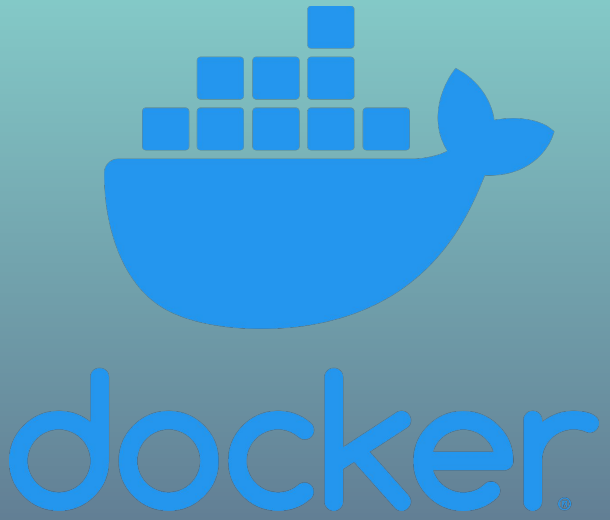
# Linux Services – Position Service

- Independent service that controls the Circle Finder Service
- If acceleration threshold (1 m/s2 for x and y, 11 for z) is exceeded:
  - Update the boundary file. Writes corner points to file.
  - Restarts the Circle Finder service to use updated boundary data.
- Continuously runs, and finds the april tag when repositioning is needed.



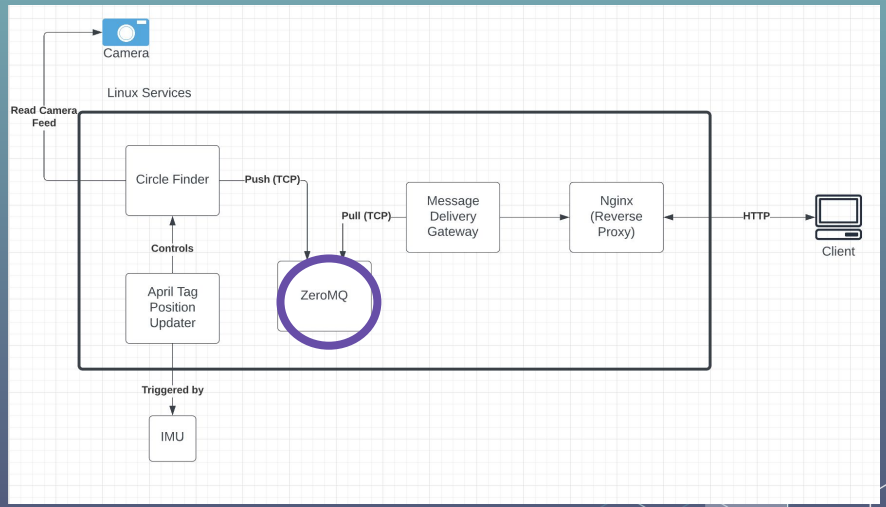
# Docker

- Used as a solution to assure that our code can run on any board
- Resolved all dependencies that were required in our projects and all environment variables would be same
- Why didn't it work?
  - 'Glasgow'
  - Manual installation was a bust
- Possible use of more standard version of Ubuntu can solve our issue and allow this to be implemented
- Moved onto ZeroMQ



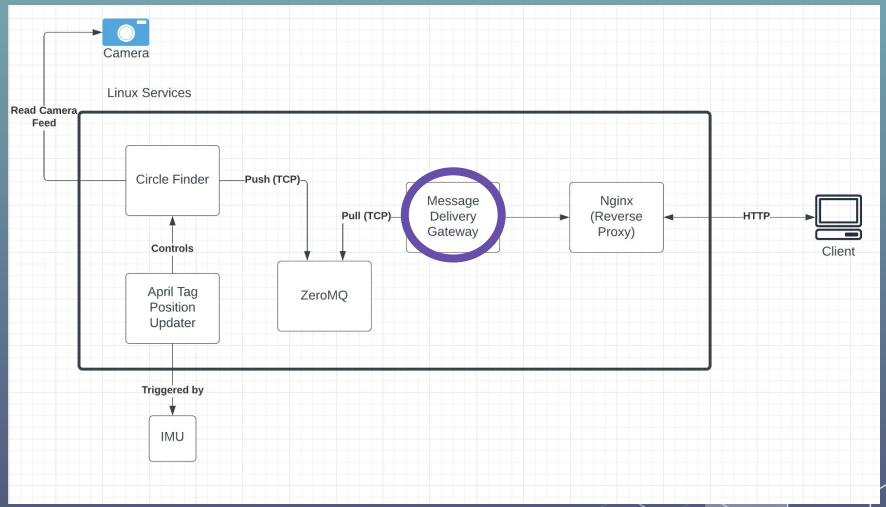
# Linux Services – ZeroMQ

- Message broker used in place of Kafka
  - Problems with Docker, ZeroMQ is lightweight alternative
  - Minimal code and configuration to get it running
- Supports TCP communication between a Flask HTTP server and the Circle Finder
- Not a service, runs independently and manages messages by their topics.



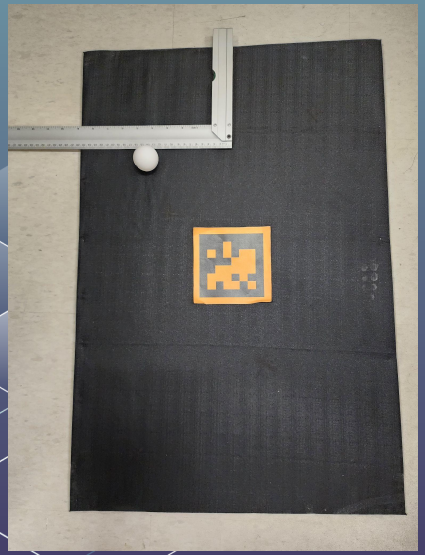
# Linux Services – Message Delivery

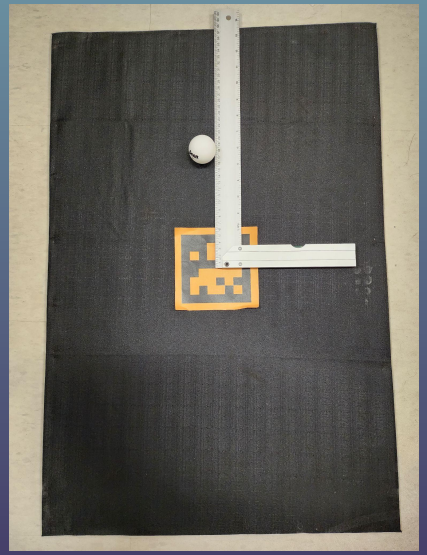
- Flask HTTP Server
- Contains one route “/position”, handles a GET Request
- If a request is received from the GUI, it sends the latest message stored in ZeroMQ
- Pub/Sub implementation - Requires a websocket connection to serve multiple clients.
    - Running a websocket server and waiting for ZeroMQ messages block each other’s operations.

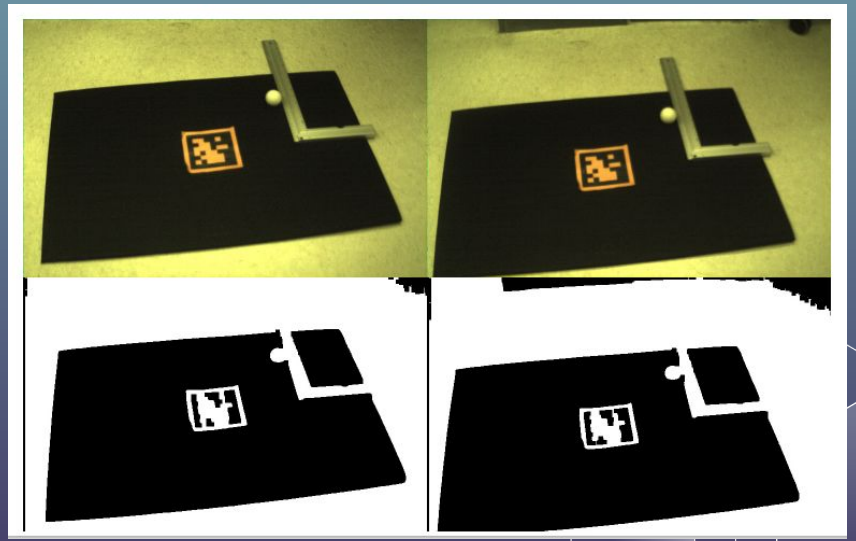


# System Accuracy Based on Ball Position

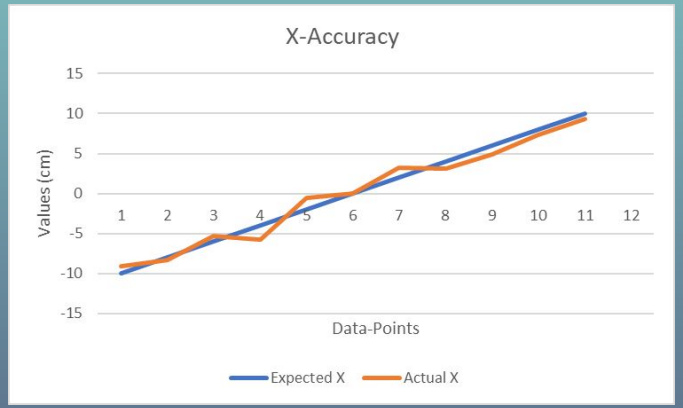
- Similar to the “Model-Based Design for Visual Localization via Stereoscopic Video Processing” by Bryan Mah
- Uses real measurements and pins it against the X, Y, and Z values of the camera
- Major shift in ball location causes slight variance in dataset until everything can catch up

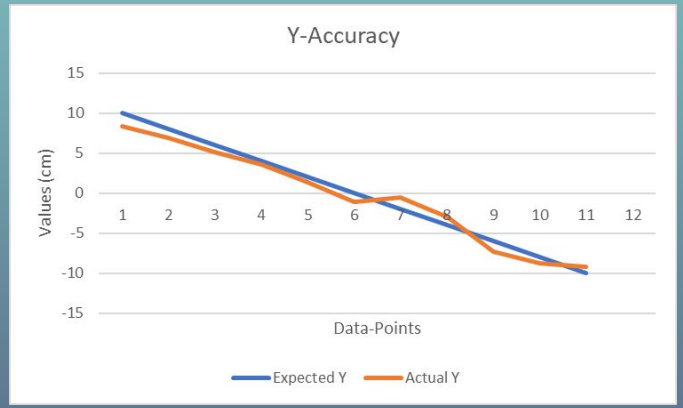


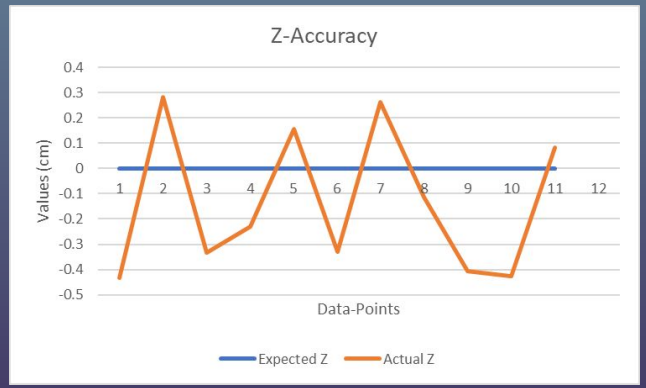




# System Accuracy Based on Ball Position



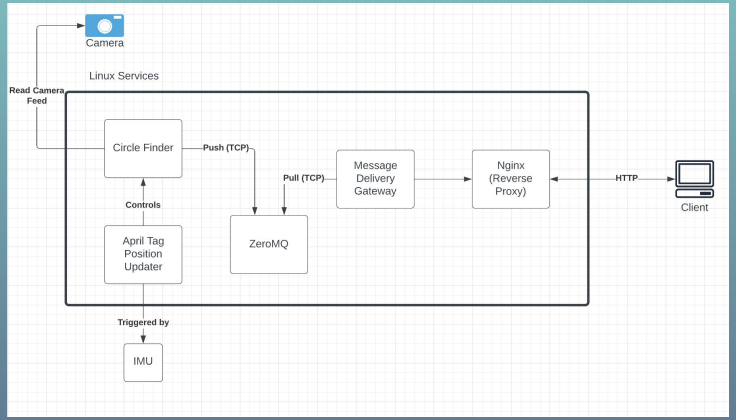



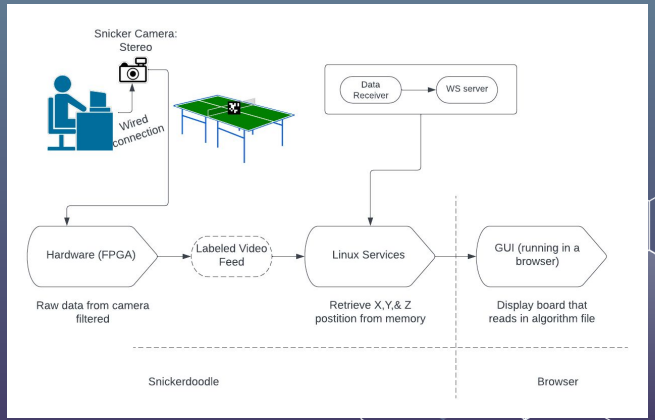
# System Accuracy Based on Camera Position

- No real way to test this because of the nature of our system
- Restarts entire service when the camera position is shifted
- On average 3 seconds for service to return
- We acknowledge that this can be a problem when it comes to really windy days in the stadium
- Considering the accelerations when thinking about the total accuracy of the system
- Ways to increase speed on the side of delays
    - Better hardware
    - More efficient software
    - More efficient lines of communication

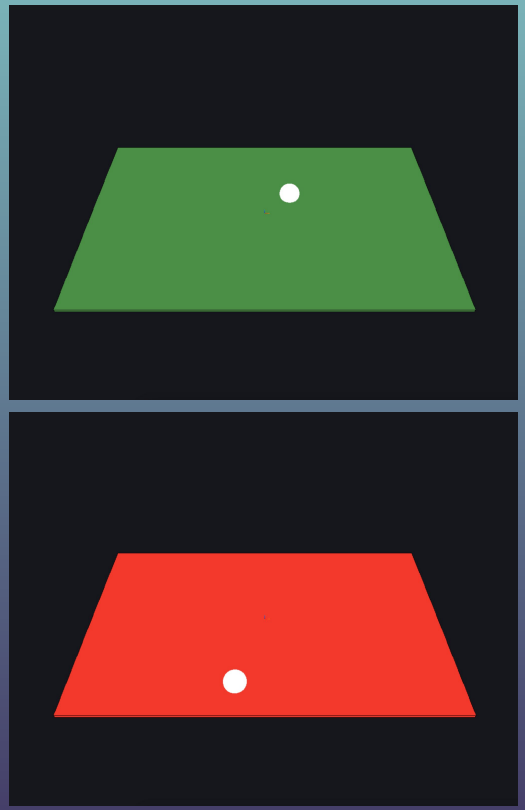
# Multi-Level System Integration Problems

- Understanding where things are going on and troubleshooting as needed
- Separation of correct FPGA values, CPU values and GUI output
- X, Y, and Z values vs what was showing on the GUI



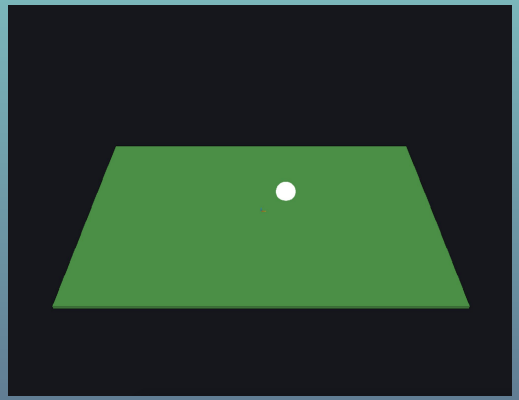



# 3D Visualization GUI

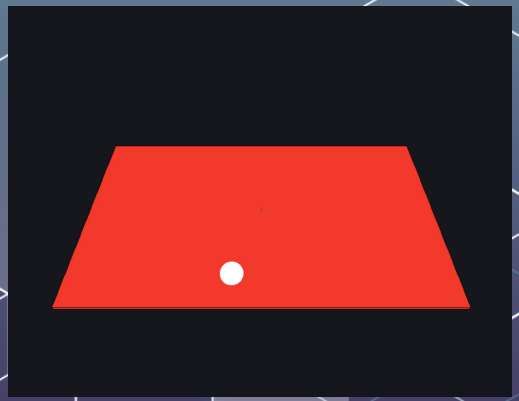


- Visualization is hosted on device, accessible through port 80
- No React – Vanilla JS and ThreeJS
  - One HTML and one Javascript file
- Polling vs Subscribing
  - HTTP polling, 500ms intervals
- Minimal file size with data payload
- Dimensions are the same as the physical field size
  - Prevents unit and scaling operations to be done
- Field center is at the center of coordinate system. Received position can be mapped easily.

# In and Out of Bounds Detection

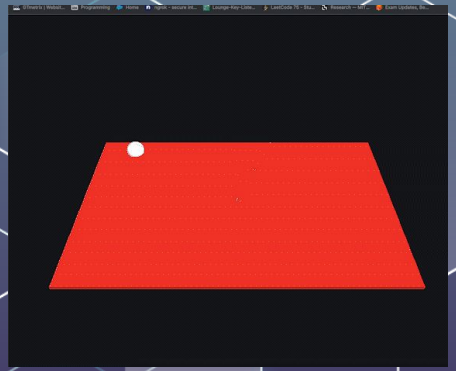


**IN VIEW**

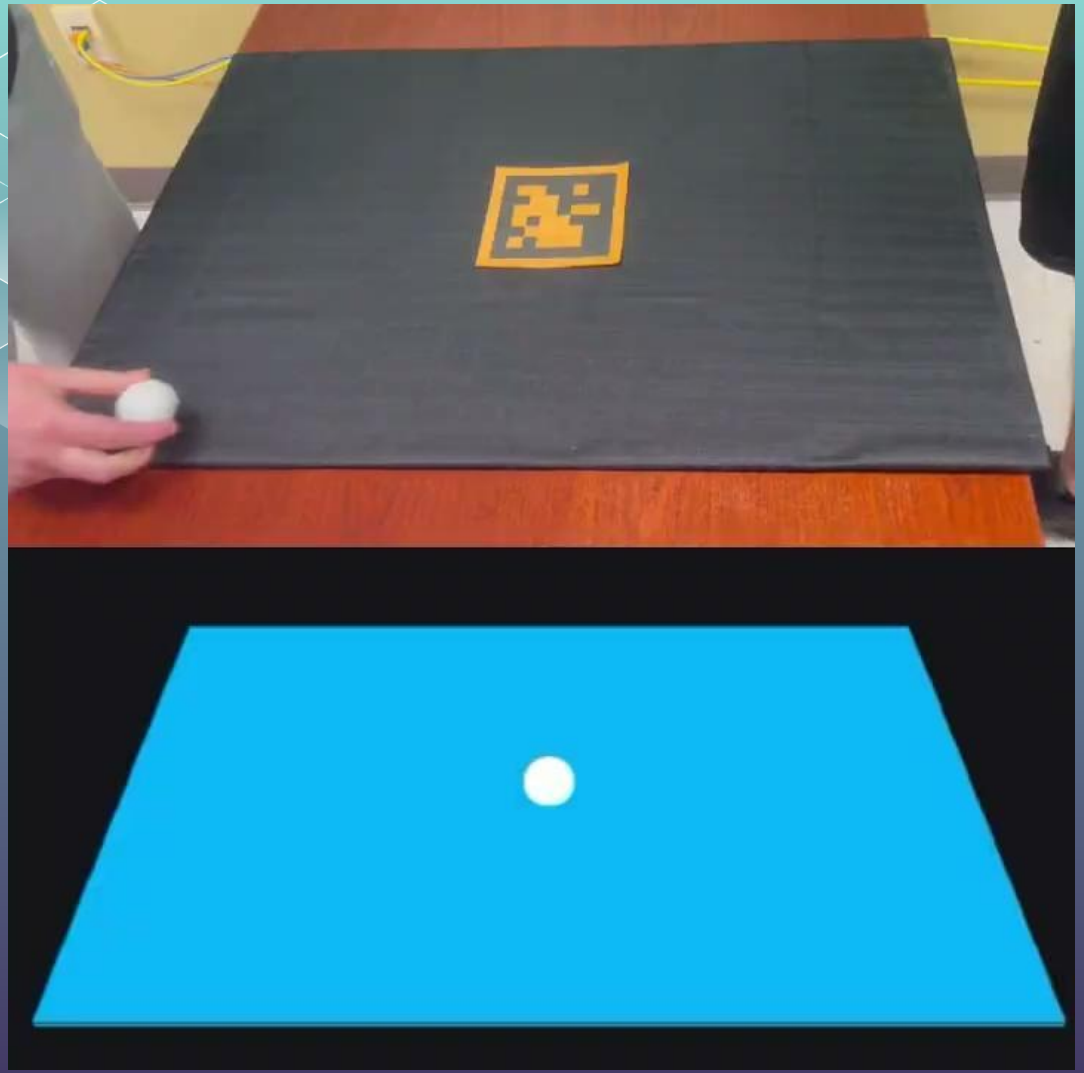


**OUT View**

- Bounds determined are 20cm x 20cm around April tag
- If the ball is on the April Tag it is considered “IN”
- Field color changes depending on calculations sent from the backend.

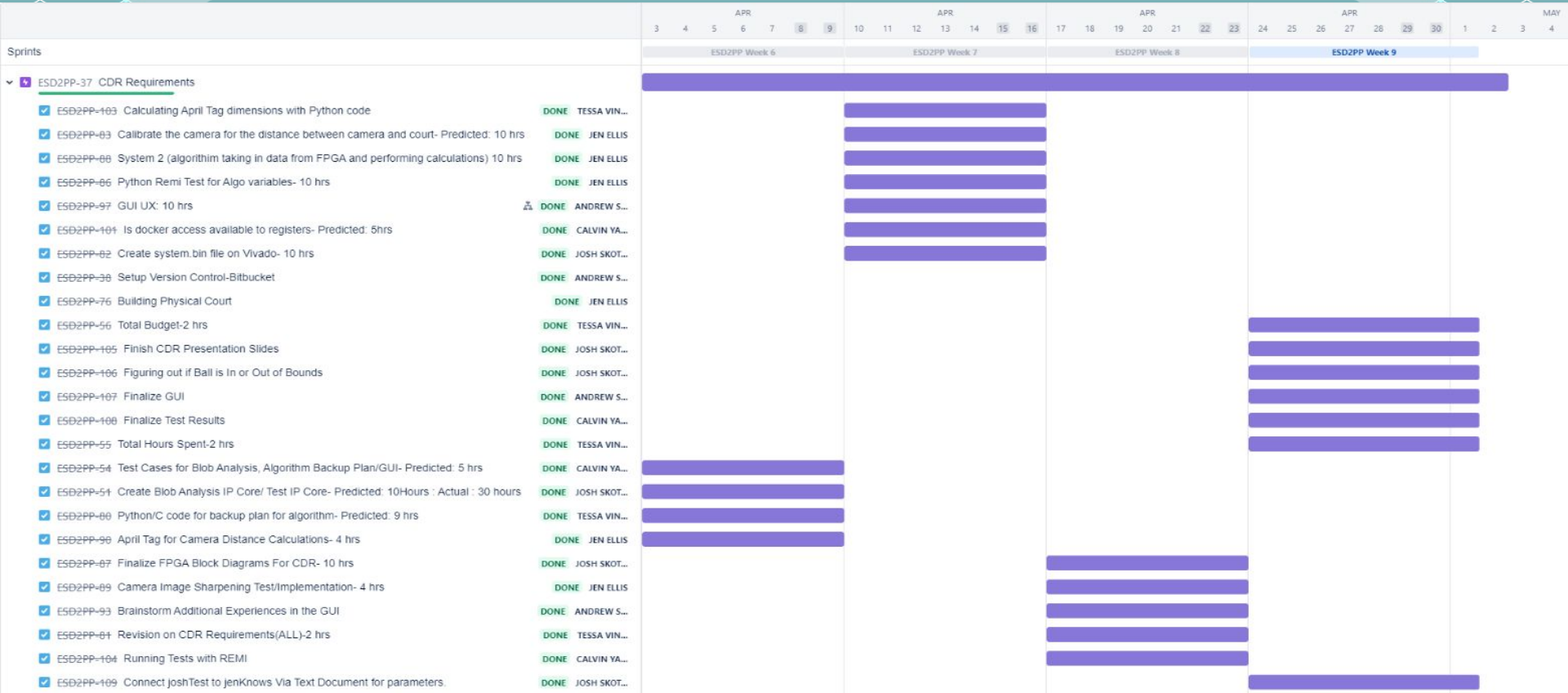


# DEMO VIDEO

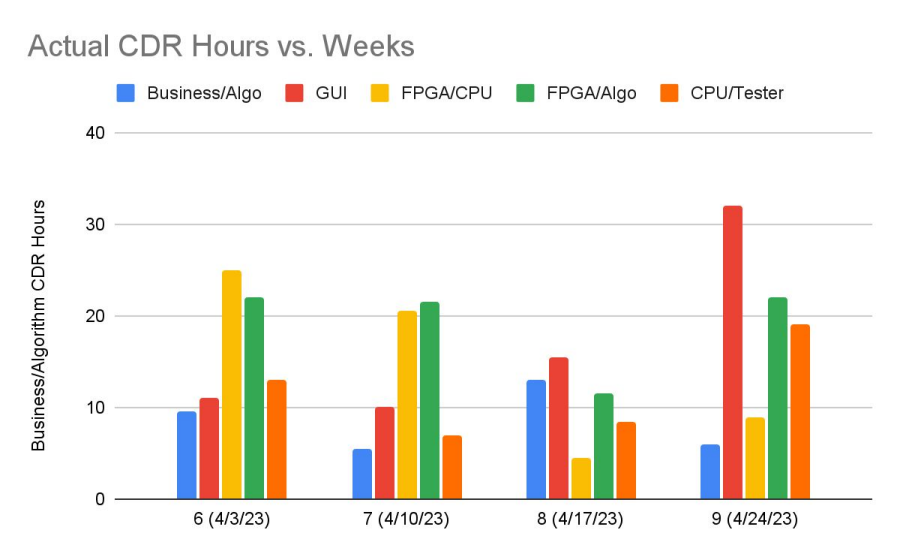


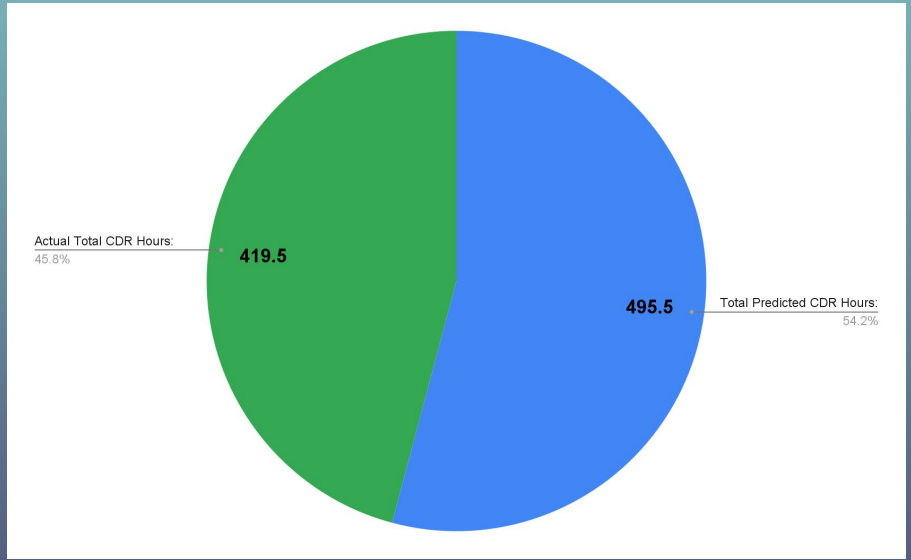
# Requirements

## Used backlog feature to make sure weekly goals were met

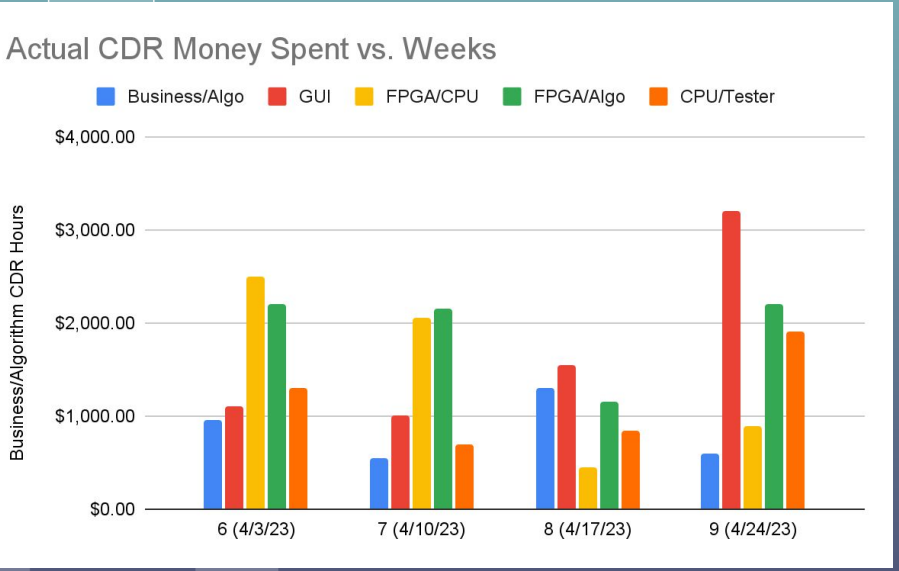


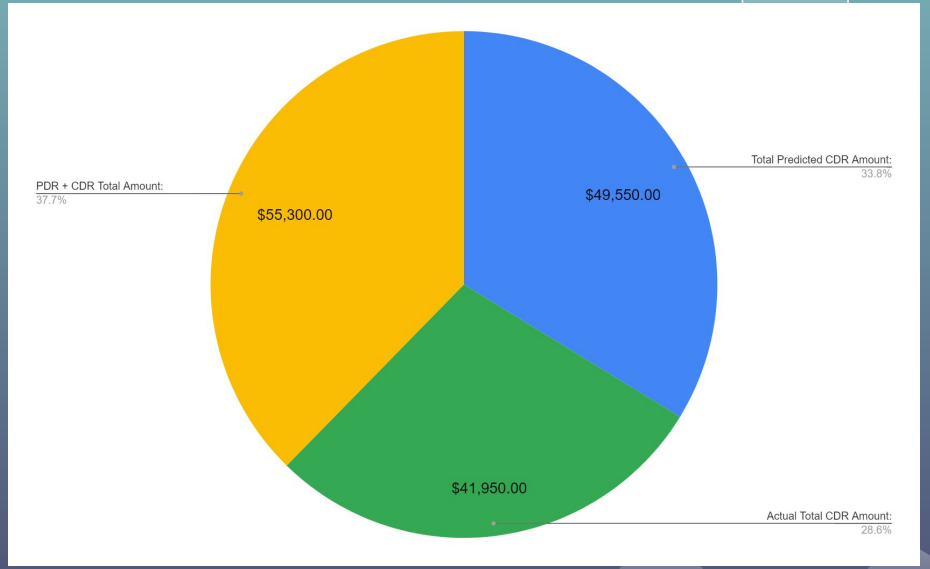
# Total Hours


Actual CDR Hours vs. Weeks
Business/Algo
GUI
FPGA/CPU
FPGA/Algo
CPU/Tester
Business/Algorithm CDR Hours
40
30
20
10
0
6 (4/3/23)
7 (4/10/23)
8 (4/17/23)
9 (4/24/23)


Actual Total CDR Hours:
45.8%
419.5
495.5
Total Predicted CDR Hours:
54.2%

# Total Expenses



# Thank You for Watching!

# Questions?