1781 FRC Programming Documentation

Turn Over Document Programmers Training Document
How to put YAGSL on your robot!
Chicago Robotics Alliance | Post 2025 | Open Alliance
Notes on PhotonVision
How to run some pre-built autonomous paths!

2026 Pre Season (Post Roboteer Rumble)

General thoughts to get on paper

- photonVision documentation is bad, do that first, get ready for Nov 8
- navX replace with pigeon
- standard camera mounts (see huskie ideas) thinking of a camera, led, tof mounting system above the bumpers
- for localization need a deep dive into odometry settings in yagsl to tune
- for localization need a deep dive into photonVision for camera calibration and tuning
- another deep dive into mechanism control for pid of arm and elevator for fast smooth movement
- work on all three swerve drives to get working with current code base: Charged Up (2023), Ralph: Crescendo (2024), and Ava: REEFSCAPE (2025).
- simulation built into all three robot mechanisms starting with reefscape

- yass

Programmer Pit Routine

Roles
developer: DEV
logger/videographer: LOG
issue expert/drive team communicator: COM

Setup:

Take out and plugin driver-station
Take mice, 2 dev laptops
Start all laptops
Farther from the pit
Open VSCode
Update from main branch
Logging laptop:
Closer to the pit
Open AdvantageScope
Setup all the variables required to log
(Elevator Command, Arm Command, Drive Command, all the positions and duty cycles, sensation constants if needed, etc)

Take out extendable ethernet cable
Take out 2 short ethernet cables
Take out long USB extender

Long USB extender should be used by driverstation for systems check
Extendable ethernet cable setup between the laptops, 2 ethernet cables from the dev laptops can be plugged into the laptops, and interchanged at need.
(Refer screenshot for above stuff)

Programmer duty:

DEV: helps COM and LOG. Talk to other teams about how their code works. Improvises code IF POSSIBLE WITH testing.
COM: Talk to drive team about issues with driving. Ask them about what happened when stuff doesn't work. Helps out DEV about talking with other stuff. Scouts other team pits and asks them about best programming practices and records them.
LOG: Switches logs and maintains log repository after every match. Videographs all matches and tries to interlocate what went wrong.

Systems check:

Connect ethernet to log dev laptop
Connect USB to driverstation
Enable robot (duh)
Feed coral and check for collection
Test L4 and use a long object to trigger the scoring for L4
Test L3 and use a long object to trigger the scoring for L3
Test L2 and use a long object to trigger the scoring for L2
Reset the orientation if possible
Check swervedrive direction
Disable robot
Switch Batteries if needed(or don't if already done right before check)
Unplug ethernet and USB cable

Emergency:

DEV: Figure out problems with code ON DEV LAPTOP, Talk with LOG and COM about the emergency reason, along with other feedback. DO NOT CHANGE EXCESSIVELY, ONLY CHOOSE THE SIMPLEST CHANGES. Always confirm your changes with others and explain thoroughly the changes done to COM and LOG.
LOG: Open AdvantageScope and log required variables. Take out the log file from the last robot and investigate problems. Use the video if needed, and talk along with COM to drivers about what happened.
COM: Put issue in GitHub or a notepad file that can be uploaded to GitHub later. Talk with drivers about problems and quiz them on exact buttons pressed, why they think happened. Communicate to figure out the reason and help DEV.

Perform systems check, always keep eyes on LOGS. IF issues are still not fixed, go back to the last branch that was working as expected.
DEV/COM: During the occurrence of a crash, switch back to code that was not crashing. Crashing code = bad for testing + takes time to fix.
LOG/COM: During the occurrence of a crash, plug in log to robot, record 2-3 crashes, and look at console in AdvantageScope to determine the problem.
Any fixes should be created in a NEW BRANCH (example: RR1 for "Robot Rumble").

If at any point you have a doubt, confirm with others and do NOT push code that "may work or cause harm"
If emergency is fixed, continue working on new branch and note the branch to use that contains the most recent fixes and should be deployed.
(Make sure code is deployed to the robot)

2025 Reefscape

Github Spreadsheet Users Guide

Pre Roboteer Rumble (Post Twist):

Auto: fix last point by moving last path more to the left ending point.

Localization (specifically orientation)

Think about replacing navX again.
Look at logs to see what was happening.

Use AprilTags to update orientation (see discussion with Huskies and look at their code).

https://www.youtube.com/watch?v=xAmkjIHBhKA

Timestamps for matches:
Quals

Spreadsheet of timings on TWIST tab

TeleOp scoring: YOUR THOUGHTS HERE! IDEAS

Driver inhibited is not inhibiting driver. FIXXXX really
Biggest problems with 50% L4 scoring:

- bad coral pickup. Need to fix device that let's coral bounce back and keep driver from early coral pickup
- robot orientation failure: need to detect and override with apriltag orientation and verify is navX
- coral on top of coral: ensure correct ending pose with PID for moveToPoint
- bad position: same as previous for PID for moveToPoint
- went to wrong wedge: fix targeting to 50% wedge instead of 60? and investigate on logs
- descored: also PID for correct position

Solution:

- CoPilot gains complete control.
- Always uses bumper to orient robot to target
- fix target acquisition to more predictably in front of april tag (fix lights)
- fix bad orientation issue
- fix bad pickup issue
- move cameras to see aprilTag if in position
- add PID moveToPositionToScore after following path to deal with
  - distance from target
  - alignment to target
  - alignment left or right of aprilTag
  - TOF dropping off side of reef (move TOF's closer together).
- final strafe should be short and almost unnoticeable if all is done well.

When moving to target, make sure to also rotate perfectly aligned to the reef before finishing.

Also, when moving in front to align with the reef, use your TOF’s to rotate yourself towards the reef perfectly before starting strafe.

end state of

I have ideas on how to fix scoring but want your ideas here:
don't move back and forth on strafe, let driver do it.
trigger just switches scoring, the bumper does everything else. When let go of bumper stays in scoring position.
change the target arches to 50 degrees instead of 60.

driving controls 2025 post season
Next steps:
get first read this: https://pathplanner.dev/pplib-build-an-auto.html

Huskies said

If they see 2 or more april tags then they use that for orientation but if they are looking at just one they change the standard deviation poseEstimator(vision, odometry, IMU) to infinity.

from wpiLib: When incorporating AprilTag poses, make the vision heading standard deviation very large, make the gyro heading standard deviation small, and scale the vision x and y standard deviation by distance from the tag. ← irrelevant

Goals

- Meeting Wednesday and on Saturday programming summits
- code forked from basis CRA common core
- 5 core members:
  - Bryce Harris: Mechanisms, commands
  - Ian Brost: Code Architect, autonomous
  - Allison Miranda: Student Mentor
  - Brian Lara: Logging, Tuning, odometry, localization
  - Taylor Williams-Shamberger: Vision, localization
- Vision and Localization: 4 camera localization (Taylor)
- Command Based Programming on 2025 robot: (Ian, Bryce)
- 3 coral autonomous
- 1 coral one algae autonomous

can not log logs and this error:

```
Warning at com.pathplanner.lib.auto.NamedCommands.getCommand(NamedCommands.java:65): PathPlanner attempted to create a command 'CustomWaitCommand' that has not been registered with NamedCommands.registerCommand
```

```
[AdvantageKit] Logging to "/U/logs/akit_d768a6d3f6035a91.wpilog"
[AdvantageKit] Failed to open output log file.
Error at org.littletonrobotics.junction.wpilog.WPILOGWriter.start(WPILOGWriter.java:155): [AdvantageKit] Failed to open output log file.
```

Last year's logic: need to port to this year

Documentation 9/15/2025

2025 Reefscape

Github Spreadsheet Users Guide

PLEASE NOTE ROBORIO WEB PAGE HAS BEEN DISABLED!

CRT Phoenix Pro LIC-25-80827900-SP-FRC Season Pass 2025 (FRC)

Simulation Base

930
mechanical advantage

Logging

Format robot’s log files into FAT32

New logging repository- 2025logging
Notes: Remember to name each log
Sync logs with videos and check whether you can log camera movement with

Command Based Programming with LEDs

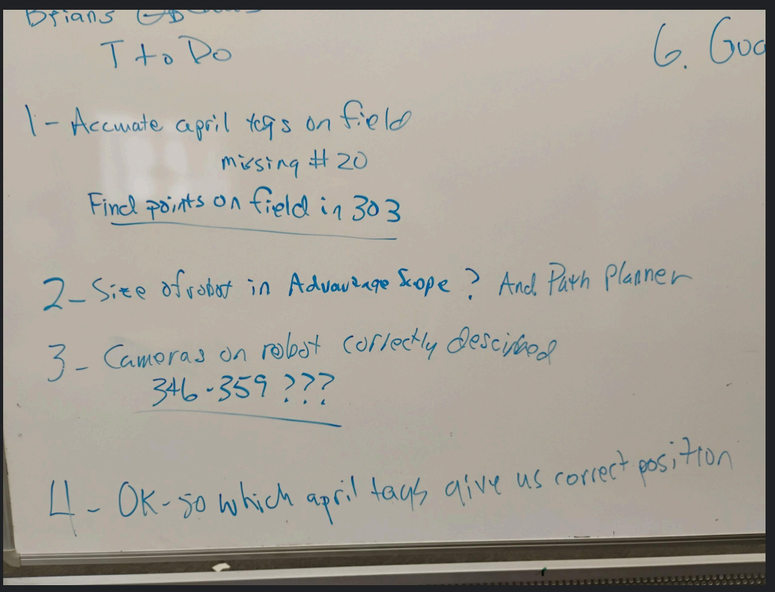
triggers:

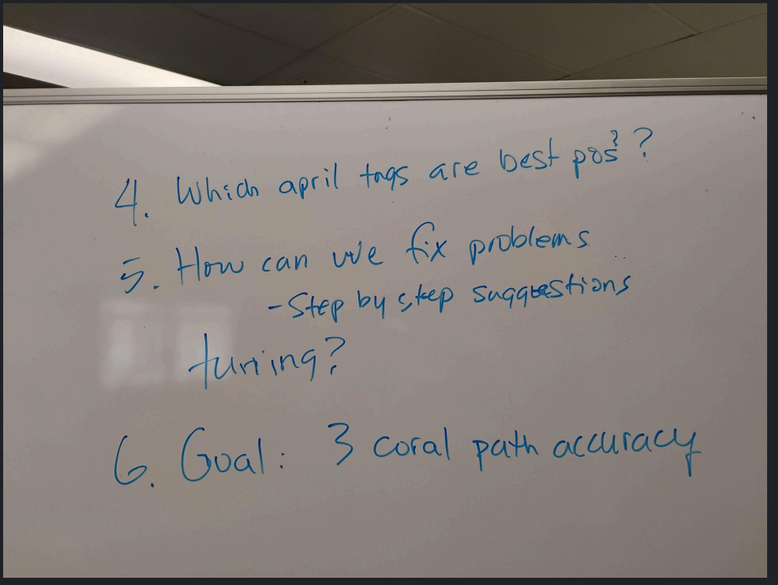
- coralInRobotSomewhere
- coralInCradel
- coralInClaw
- robotReadyToCenter
- robotSeesReefPoleInRightSpot
- robotReadyToScore
- showLEDhasCoralSomewhere: composite coralInRobotSomewhere and not coralInClaw
- showCoralInClaw: composite coralInClaw and not robotReadyToCenter and not robotSeesReefPoleInRightSpot
-

LED Commands:

- hasCoralSomewhere (blinking yellow)
- coralInClaw (solid yellow)
- readyToCenter (solid red)
- readToScore (solid green)

Localization tuning with vision





YAGSL

Constants: SEE PROGRAMMERS SPREADSHEET
https://docs.yagsl.com/configuring-yagsl/getting-to-know-your-robot
(Config Website)
Kraken x60 (CTRE PRO) Double Kracken Swerve Model
-CANcoder (Made by CTR)

Basic Config Values:
Kraken 60x
Drive Gear Ratio: 7.75:1 (Check this to make sure of it)
Diameter: 2.37 inches Effective Diameter: 2.5 inches
Length: 2.96 inches
Weight: 1.2 lbs

Specs stuff: (By Alison)
Angle Gear Ratio: 6.75
Drive Gear Ratio: 6.75
Wheel Diameter: 2
Optimal Voltage: 12v
Robot Weight: 112lbs
Current Limit Drive: 60
Current Limit Angle: 60
Grip Coefficient of Friction: 1

Step 1: Module Types
Motor: Kraken x60
AbsoluteEncoderoffset: Krackenx60foc

Encoder: Cancoder
Controller:
IMU: navx_mxp_serial

Pain: https://broncbotz3481.github.io/YAGSL-Example/
Website used to Config the robot, but its BLOCKED on CPS computers 🙃

So, here a template containing all of the Configuration values:

Swerve Drive
IMU TYPE: navx_mxp_serial
IMU ID: 4
IMU CAN Bus : null Inverted IMU: false

Module Properties
Optimal Voltage: 12v
Robot Weight (in lbs): 116
Grip Coefficient of Friction: 1
Current Limit Drive: 60
Current Limit Angle: 60
Angle/Steering/Azimuth Gear Ratio: 6.75
Angle/Steering/Azimuth Conversion Factor(optional): 0
Wheel Diameter(in): 2
Drive Gear Ratio: 6.75
Driver Conversion Factor (optional): 0
Ramp Rate Drive: .25
Ramp Rate Angle: .25

Front Right Module	**Front Left Module**
Location X: 11.25	Location X: 11.25

Inches (+x is towards the robot front)	Inches (+x is towards the robot front)
Location Y: -13 Inches (+y is towards robot left)	Location Y: 13 Inches (+y is towards robot left)
Absolute Encoder Offset: -0.324463 ONLY SPARKMAX’S ARE SUPPORTED FOR BRUSHED MOTOR CONTROL!	Absolute Encoder Offset: .38916 ONLY SPARKMAX’S ARE SUPPORTED FOR BRUSHED MOTOR CONTROL!
Drive Motor Type: krackenx60foc	Drive Motor Type: krackenx60foc
Drive Motor ID: 26	Drive Motor ID: 21
Drive Motor CAN Bus: 26 Drive Motor Inverted: true	Drive Motor CAN Bus: 21 Drive Motor Inverted: false
Angle Motor Type: krackenx60foc	Angle Motor Type: krackenx60foc
Angle Motor ID: 25	Angle Motor ID: 20
Angle Motor CAN Bus: 25 Angle Motor Inverted: false	Angle Motor CAN Bus: 20 Angle Motor Inverted: false
Absolute Encoder Type: cancoder	Absolute Encoder Type: cancoder
Absolute Encoder ID: 62	Absolute Encoder ID: 60
Absolute Encoder CAN Bus: 62 Absolute Encoder Inverted: false	Absolute Encoder CAN Bus: 60 Absolute Encoder Inverted: false

Back Right Module	**Back Left Module:**
Location X: -11.25 Inches (+x is towards the robot front)	Location X: -11.25 Inches (+x is towards the robot front)
Location Y: -13 Inches (+y is towards robot left)	Location Y: 13 Inches (+y is towards robot left)
Absolute Encoder Offset: .476074 ONLY SPARKMAX’S ARE SUPPORTED FOR BRUSHED MOTOR CONTROL!	Absolute Encoder Offset: .709473 ONLY SPARKMAX’S ARE SUPPORTED FOR BRUSHED MOTOR CONTROL!
Drive Motor Type:	Drive Motor Type:

krackenx60foc	krackenx60foc
Drive Motor ID: 28	Drive Motor ID: 23
Drive Motor CAN Bus: 28 Drive Motor Inverted: true	Drive Motor CAN Bus: 23 Drive Motor Inverted: false
Angle Motor Type: krackenx60foc	Angle Motor Type: krackenx60foc
Angle Motor ID: 27	Angle Motor ID: 22
Angle Motor CAN Bus: 27 Angle Motor Inverted: false	Angle Motor CAN Bus: 22 Angle Motor Inverted: false
Absolute Encoder Type: cancoder	Absolute Encoder Type: cancoder
Absolute Encoder ID: 59	Absolute Encoder ID: 61
Absolute Encoder CAN Bus: 59 Absolute Encoder Inverted: false	Absolute Encoder CAN Bus: 61 Absolute Encoder Inverted: false

Heading Tuning
Joystick Radial Deadband: .5
Heading kP: .4
Heading kI: 0
Heading kD: .01

Module Tuning:
Drive kP: .5
Drive kI: 0
Drive kD: 0
Drive kF: .25
Drive Integral Zone: 0
Angle kP: 5
Angle kI: 0
Angle kF: 0

Tutorial for Programming Swerve

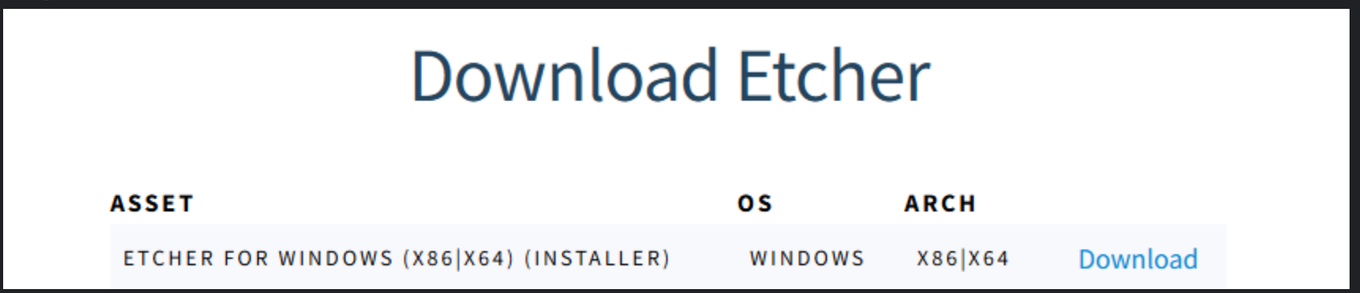
This entire process, do not do on a CPS Computer, use a personal device.

Vision Setup

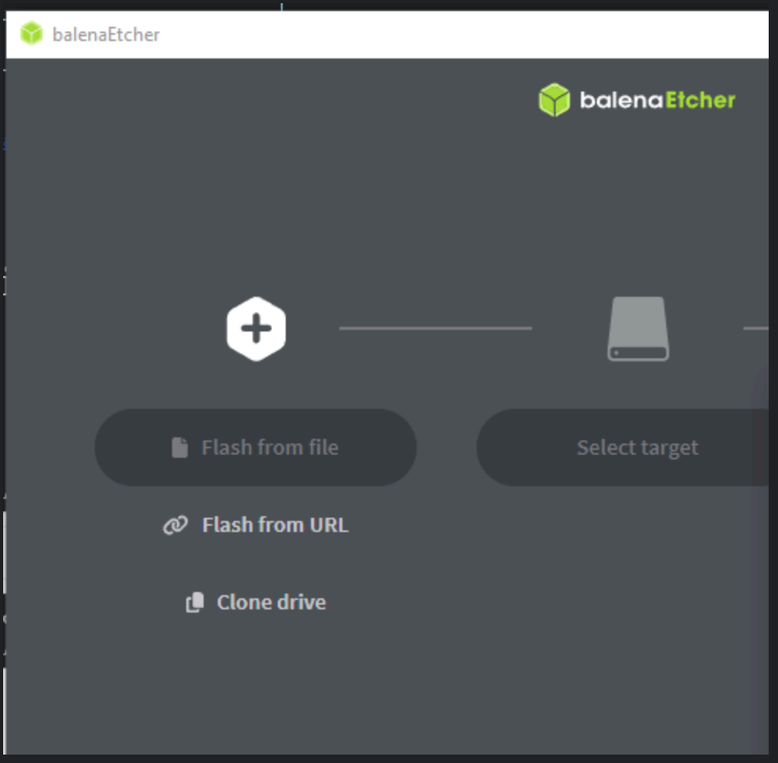
Step 1: Aquire stuff
Camera + Adaptor + Cube
(Insert Images here)

Step 2:
Photon Vision: Download
https://github.com/PhotonVision/photonvision/releases/tag/v2025.3.1

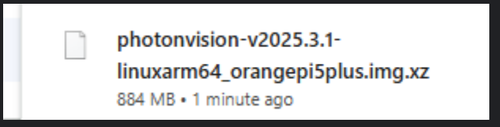
Also Download Etcher
https://etcher.balena.io/#download-etcher



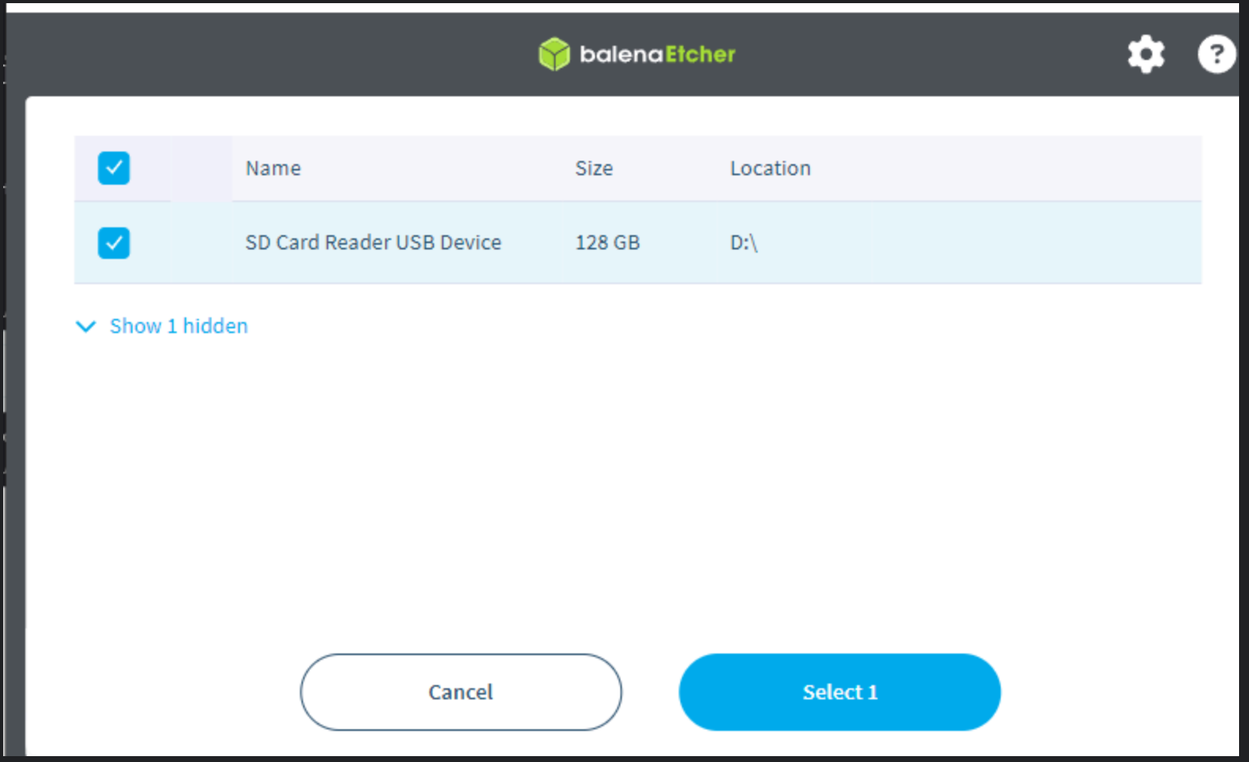
Step 3:
Open up Etcher Startup, and download by file



Input Photon Vision downloaded file



Then, input the flash drive into the computer, and select it



Then, just flash
Step 4: Once flash is complete, plug microusb into CUBE
(Video)

Now, properly set everything up.

Connect Cube to any monitor, tv, laptop, etc.
Connect HDMI into HDMI 1 on the Cube, and to the computer.

Also connect any keyboard to it, and charge it.
(img)

Once light shows it, you know you're good.

Step 5: Hacking time

Information:

Username: pi
Password: raspberry

Now, confirm that photonvision is working, to do this, type this command at the end.
sudo systemctl status photonvision

Reason why: To confirm that Photon Vision's service is actually doing it. If its enabled, its green,
(img of good)
if not, here's to solve it. If Red: Bad. If green: good!

(bad image)
Type this command to restart it

sudo systemctl restart photonvision

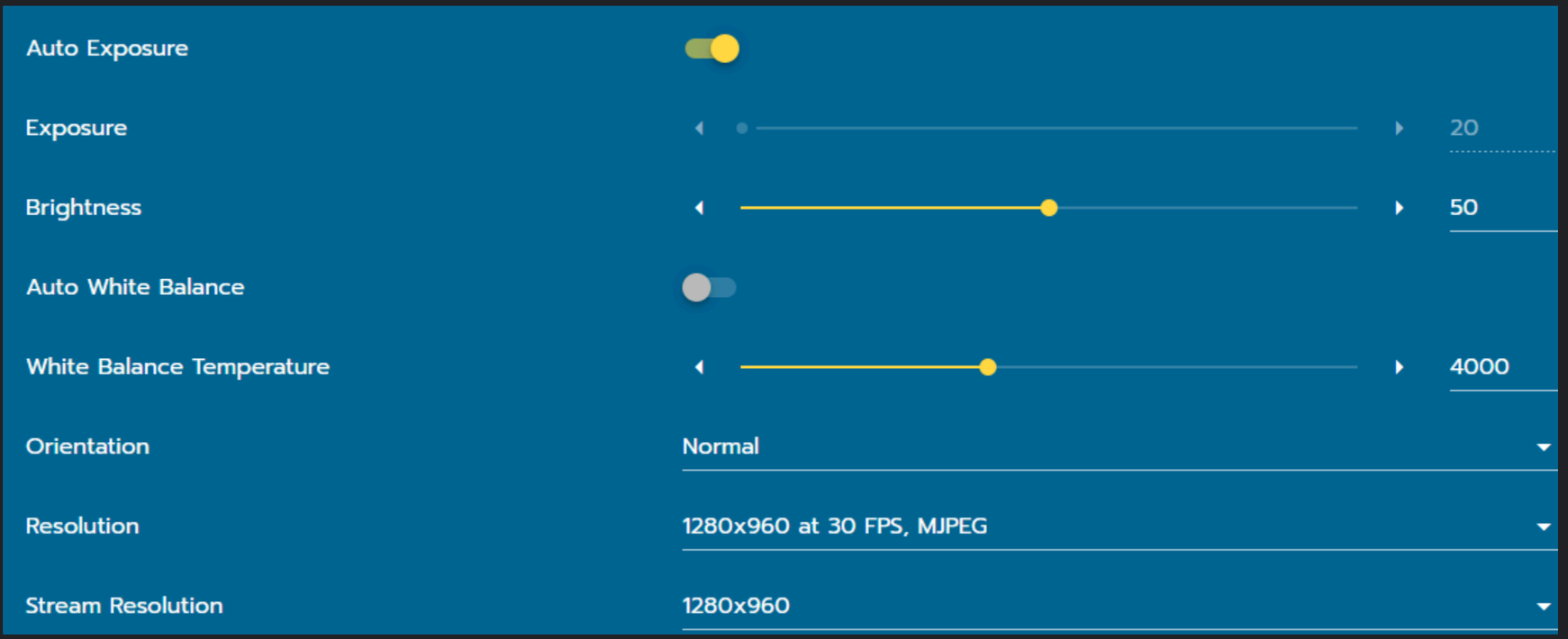
if its still failing, thats prob bc there's a error, from the system, errors will pop up, and you'll just have to debug it (Google it; sorry)

Once you're done with that, unplug it, then plug it into a new network, as seen here
(IMG)
Then, plug ethernet cable to the CUBE and the network

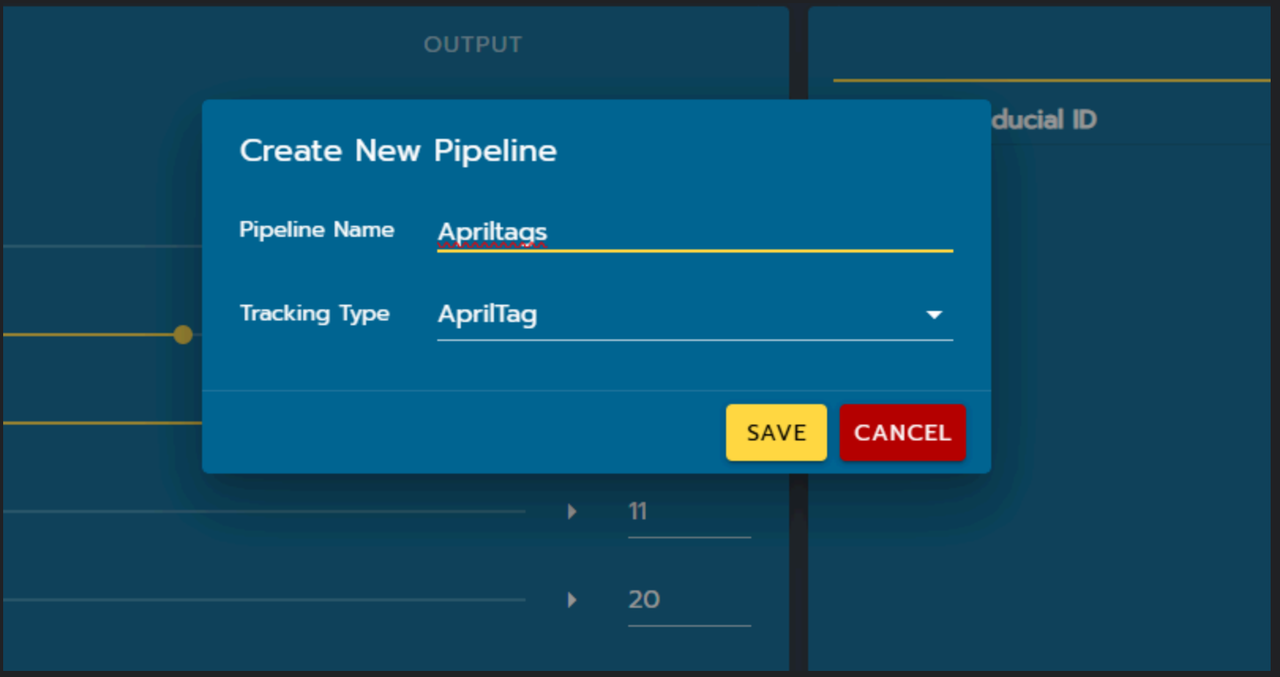
Next, connect to Robot wifi

http://photonvision.local:5800/#/dashboard
Put that link in to a new tab, and it should be good

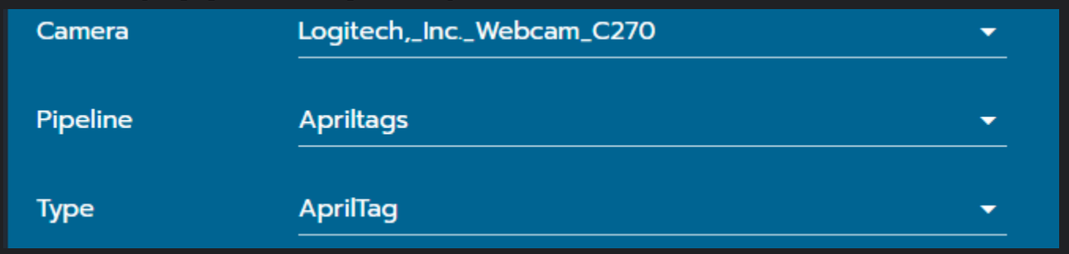
Activate it, and then just configue it with auto exposure and steam resolution



Create a new Pipeline



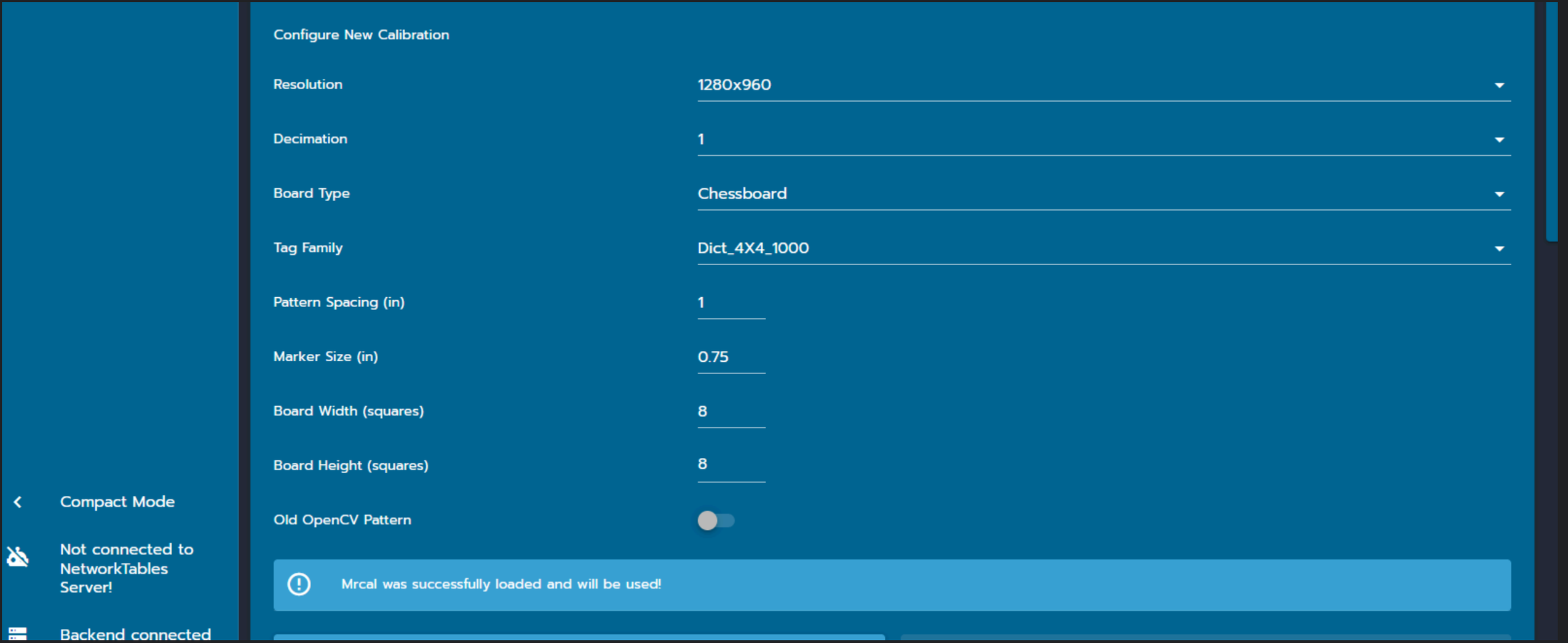
Then, change pipelin to April tag



Then, it should be good (Also reconfigure again)

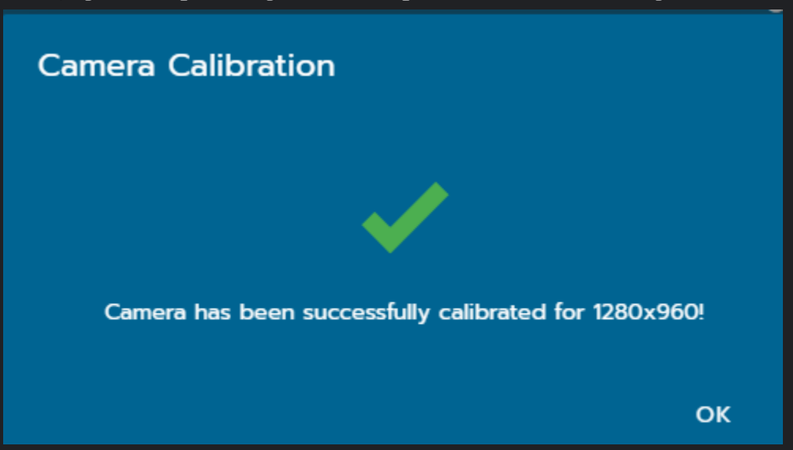
Calibration Board
Get out a Checkboard(IMG)

Go to _____ (Not Dashboard)




```
Download Checkboard, set it as Checkboard, then place it into the camera. If rbg lights apear, it is good.

Then, just repeating take snapshots whenever lights occur, and once you have enough, press end calibration.
```




```
Lastly, try 3d by clicking 3d On Dashboard
if it moves, its good.
```

If Drive is Corrupter, do this.

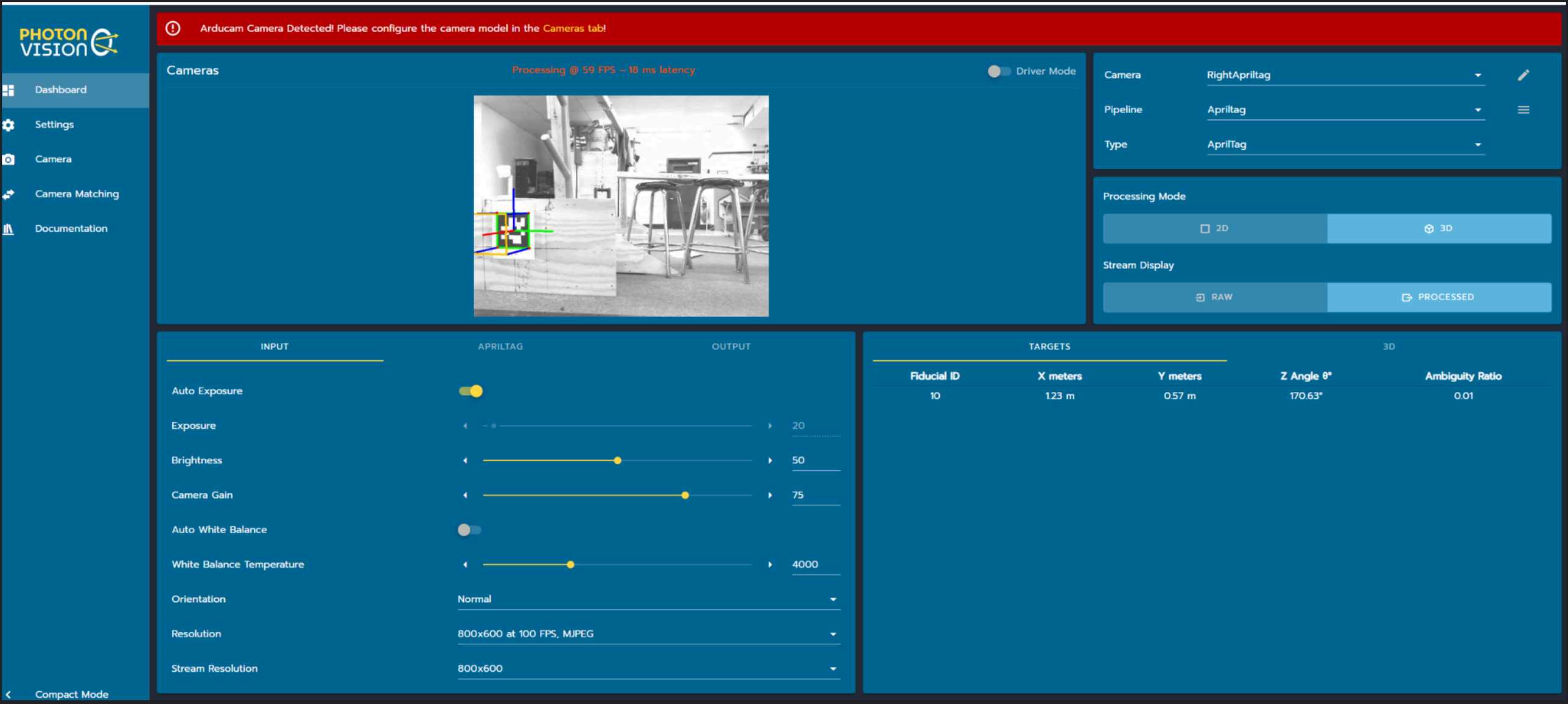
1.Plug flask drive into computer
2.Go to Disk Manager
3.And delete all garbage from the disk(bottom)

Then, just repeat the steps again, yay!

PhotonVision

orange pi's addresses:
photonvision-a.local 10.17.81.202/24
  RightAprilTag
  LeftAprilTag

photonvision-b.local



Uncontrolled Movement

```
Drive is controlled from:
      driveController.centerOnReef -> getCenteringChassisSpeeds
      followPath
      idle
      driveUsingVelocities -> driverDriving
            driverDriving(inputHolder.driverLeftJoystickPosition, inputHolder.driverRightJoystickPosition);
                  driverInputHolder.driverLeftJoystickPosition = getControllerJoyAxis(ControllerSide.LEFT, 0);
                  driverInputHolder.driverRightJoystickPosition = getControllerJoyAxis(ControllerSide.RIGHT, 0);

Consider adding in getControllerJoyAxis:
       if (!selectedController.isConnected()) {
```

```
        return new Translation2d();
    }
```

L4 Diagnosis

Arm will not transition to L4 so what does it mean "relies on others" is that just each subsystem relying on the previous subsystem setting of the same subsystem, or each listed state in sequence regardless of subsystem?

Here are some dependencies:
(assuming Arm is in POLE state): Arm matches POLE state: Arm/MotorEncoder (19 - 31)
(assuming Elevator is in POLE state): Elevator matches POLE state only if
Drive Controller matches state (hasFinishedCentering()):
reachedDesiredDistance
(0) leftTOF.isRangeValidRegularCheck()
(1) rightTOF.isRangeValidRegularCheck()
(3) 240 < (leftTOFDistance + rightTOFDistance) / 2.0 < 340

```
note: last attempt to get to L4 I commented out alignment check
if (/* inputSpeeds.omegaRadiansPerSecond <  && */ inputSpeeds.vxMetersPerSecond == 0) {
    reachedDesiredDistance = true;
}
```

hasFoundReefPole:
(4) armTOF.getRange() < 700
(5) armTOF.isRangeValidRegularCheck()
(6) robotController.isArmInPoleState(): POLE state: Arm/MotorEncoder (19 - 31)
(7) robotController.isElevatorInPoleState(); POLE state and:
(8) 700 < firstStageTOF < 800
(9) -50 < secondStageTOF < 50

But arm would not move past 40, because pid is too weak or
robotController.isSafeForArmToLeaveIdle() was false?
secondStageTOF < 150?

So probably the PID was too weak and stayed around 40?

```
armMotorConfig.closedLoop.pid(0.005, 0, 0.000, ClosedLoopSlot.kSlot2);
armMotorConfig.closedLoop.outputRange(-0.3, .3);
```

which I was changing when we had to leave…

Auto

Always thinks it is at blue alliance, despite multiple measures to make sure its supposed to be at red

Climber

Notes:
just to remember, im posting some info here

climber operates on one motor, every 125 rotations the axle spins once
axle should spin approximately 2 - 2 and a half times to complete the deep climb

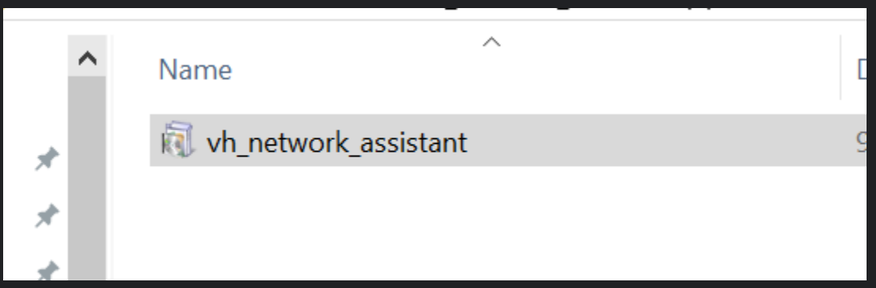
Wiring Diagram

L25 Robot Wiring diagram

New Radio

Notes on getting radios to work:

Connect to radio with ethernet cable. Run as administrator:

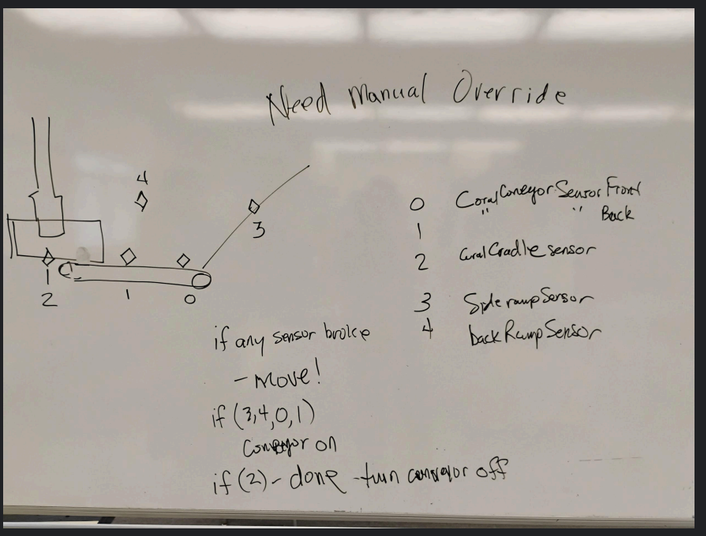


Configure team number 1781 and key is AVA

https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/radio-programming.html
Make sure to change switch #3 for here and competition is different.
AVA drive base radio: or frc-1781-ava (not the one with ap in it).
password: password

https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/radio-programming.html

Conveyor



Twist

[Paths from 2024 in spreadsheet]

Required Programs for DriverStation Laptops

- WPILIB 2024
- DriverStation/Game Tools
- Shuffleboard
- AdvantageScope
- REVClient
- Phoenix X
- LimeLightFinder
- RoboRio Imaging Tool

Pre-Season Planning

- Rev absolute encoder issues
- poseEstimation issues
- Logging
- Auto for Twist using only paths
-

Photonvision vs Limelight | Photonvision is better

https://photonvision.org/
So the goal here is to have very accurate odometry using vision. Here I'm looking for a vision that can see far, be accurate, and be performant (less).

Photonvision was recommended by team 7525 and since they had a really good auto. I would need to optimize the NVIDIA Jetson a bit here since I'm still running it using a GUI which isn't gonna be true on a comp bot where we want all the performance, might wanna mess with the Java GC too.

Links
Anand's Coprocessor Roundup
7525's vision code
Removing GUI from Ubuntu

Summer 2024

end goal - recreate working autonomous with good localization (vision+odometry) and auto routines

fixing arm:
refresh rate was too slow (every 200 ms vs every 20)
fix - add `mRightMotor.setPeriodicFramePeriod(PeriodicFrame.kStatus5, 20);`
also, removed usage of relative encoder and replaced with absolute encoder in arm for more smooth/accurate arm movement

2024 Crescendo | SEASON OVER

Board 2024 Crescendo spreadsheet CAN Diagram Pit Checklist Processing 101 scouting doc project and training grid

REV conversation

Should have said the can frame rate 20 milliseconds for the absolute recruiter. Also checking for faults to guard against the zero and limit switch down problem we had
Also he told us to update the Rev library and he was surprised that nobody had done that even though they don't tell anybody do that
How many other libraries have been updated? We should check for that during the season

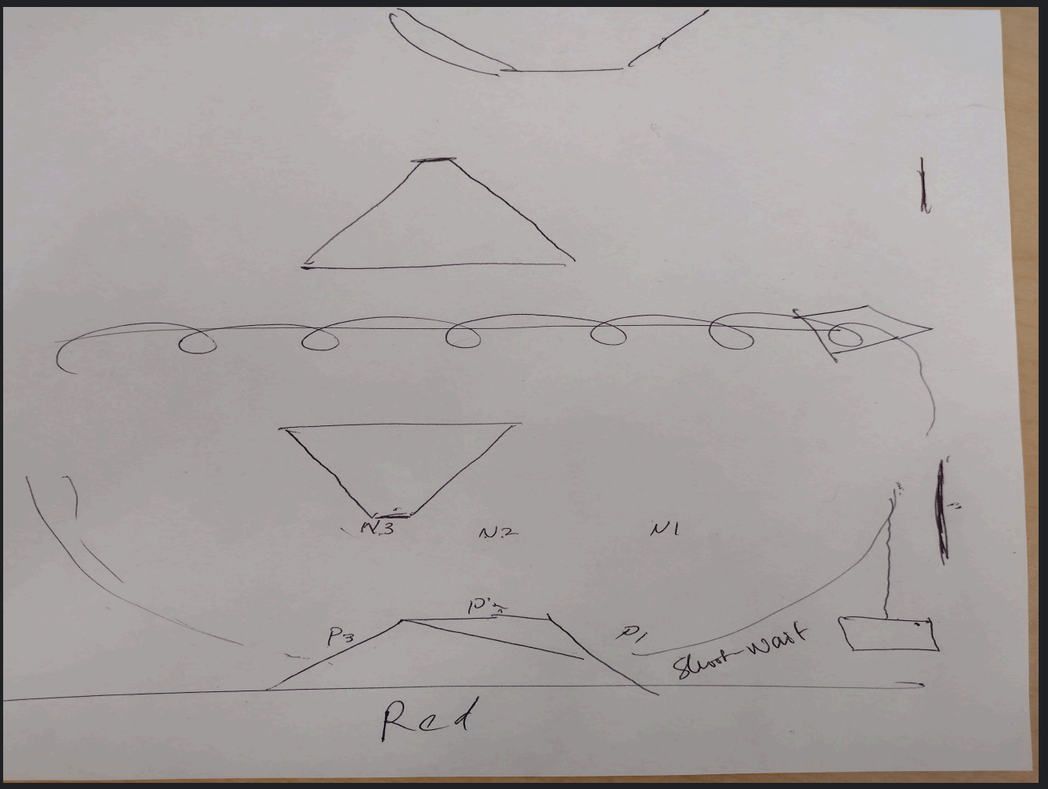
Profiling Code for overruns

Path problems

- testing on Mikey with code from pathTestingRufus
- does not reach desired end point, changed max velo, still not there
- upping P from .1 to .5 in setTrajectory
- helped a lot upping to 1, very good, now 2, then 4 to no negative effects.
- that was on strait, now diagonal path…but with 0 orientation, no problem.
- that was on blue, try red.prefect but all on 1.0 velo and 1.0 acc.
- now, try 90 degree start, on blue, strait. perfect

- now same thing red, perfect
- now, for 60 degree going strait out, blue, then red. perfect
- now measuring actual movement. requested 3 meters, went 2.8 in real life.
- carpet pile effects the odometry adding 3% with the pile, -3% without.

Auto plan



Arkansas List

- Inspection
- 9:40 to 10:00 practice on speaker 11:40 to 12:00
- In pit test kickstand comes off for autonomous consider alternative
- p3;n3 on practice field if possible
- calibrate vision (9 - 11 am)
- encrypt radios (bring in to encrypt)11:40
- try t1 and t1F in arkansasTest in p1
- running with limelights on except for p1 and p3 on red
- practice matches start at 12

Stephen's Autonomous Routines

Folder

ARM Encoder issue

```
state SAFE arm encoder: 63.74 desiredSetPoint: 64.00
63.74 64.00 0.01
state SAFE arm encoder: 63.74 desiredSetPoint: 64.00
63.74 64.00 0.01
state SAFE arm encoder: 63.74 desiredSetPoint: 64.00
63.74 64.00 0.01
state SAFE arm encoder: -0.00 desiredSetPoint: 64.00
Hit reverse limit on arm resetting encoder to 0
63.74 64.00 0.01
state SAFE arm encoder: 63.74 desiredSetPoint: 64.00
0.00 64.00 3.20
state SAFE arm encoder: 0.00 desiredSetPoint: 64.00
0.00 64.00 3.20
state SAFE arm encoder: 0.00 desiredSetPoint: 64.00
Loop time of 0.02s overrun
Warning at edu.wpi.first.wpilibj.IterativeRobotBase.printLoopOverrunMessage(IterativeRobotBase.java:412): Loop time of 0.02s overrun
0.37 64.00 3.18
state SAFE arm encoder: 0.37 desiredSetPoint: 64.00
1.45 64.00 3.13
state SAFE arm encoder: 1.45 desiredSetPoint: 64.00
1.45 64.00 3.13
state SAFE arm encoder: 1.45 desiredSetPoint: 64.00
3.04 64.00 3.05
state SAFE arm encoder: 3.04 desiredSetPoint: 64.00
4.71 64.00 2.96
```

Shooter Positions

Position	x	y	
Podium blue	3.0	4.7	
Podium red	13.6	3.6	
Amp blue	1.8	7.7	

Amp Red	14.7	7.7	
Subwoofer shot	calculated	calculated	

Bosch motor

Looking to see if we can use the built in Hall sensor in the Bosch seat motor to control the position of the shooter. You've been spoiled with the build in encoders of the brushless motors. This one may take some work. (unless we decide to use the CANcoder instead)

WPI info on Hall circuit and wiring with Andymark (am-3812) DIO Kit https://wpilib.screenstepslive.com/s/4485/m/63630/l/679357-bosch-seat-motor#hall-circuit-interface

Notes from ChiefDelphi
https://www.chiefdelphi.com/t/java-code-for-bosch-seat-motor-with-dio-kit/164325
https://www.chiefdelphi.com/t/frc-hall-sensor-question-for-bosch-motor-about-java-program/345481
https://www.chiefdelphi.com/t/bosch-seat-motor-programming-help/342365

Motor spec sheet
https://media.screensteps.com/attachment_assets/assets/000/276/722/original/2016-12-21_spec_sheet_-_Bosch_FRC_motor_6_004_.pdf

Jeremiah soldered one of the DIO kit so we'll be able to give you the motor today to test out.

Arm angles with new Scollector

Highest angle under 4 feet: 70 degrees
Angle to be in safe mode:

Practice 2/3/2024 notes

1. subwoofer angle too high maybe
2. weird note sticking out when collected, probably fixed by adjustment of centering plastic parts
3. steven's excellent notes

Offsets

Mikey:

```
public static final double FRONT_LEFT_MODULE_STEER_OFFSET = -0.039;
public static final double FRONT_RIGHT_MODULE_STEER_OFFSET = -0.018;
public static final double BACK_LEFT_MODULE_STEER_OFFSET = -0.857;
public static final double BACK_RIGHT_MODULE_STEER_OFFSET = -0.872;
```

Donnie:

```
public static final double FRONT_LEFT_MODULE_STEER_OFFSET = -0.772;
```

```
        public static final double FRONT_RIGHT_MODULE_STEER_OFFSET = -0.427;
        public static final double BACK_LEFT_MODULE_STEER_OFFSET = -0.238;
        public static final double BACK_RIGHT_MODULE_STEER_OFFSET = -0.419;
```

Diagnosing Shooter Inconsistency

Why flop shots? First, make sure to keep collector on once we get a switch to

Time of Flight Playing with Fusion

See this how to (local version, may want to update after 2024): Playing with Fusion ToF
This is how you id a time of flight sensor:
- Open 10.17.81.2:5812 when connected to the robot to access Playing with Fusion web server running on roboRio
- Warning! Do not reverse the polarity of the power connections. This will cause permanent damage to
- the sensor
-

Controls

Driver Right bumper pressed: arm goes down and collector motors on to collect until ToF registers a note or rightBumper is no longer pressed, then it stops the motors and moves the arm back to safe position.

Driver Right Trigger pressed: arm goes down into collecting position (if moving under stage).

Driver Left bumper pressed: collector motors run in reverse.

Copilot trigger: Shooter motors start PID to optimal velocity, arm moved to aim at speaker (based on aprilTag distance) and robot rotates to aline april tag to center. When all three conditions are met the collector motors start. Process ends when copilot releases the trigger.

Field Dimensions

Each FIELD for CRESCENDO is an approximately 26 ft. 11¼ in. (~821 cm) by 54 ft. 3¼ in. (~1654 cm) carpeted area bounded by inward facing surfaces of the ALLIANCE WALLS, SOURCES, AMPS and AMP pocket walls, and guardrails.

April Tags 2024

Shooter

2 NEO

Need speed Feedback

conveyor @ speed X

Reach → adjust Arm according to distance
- Auto Aim
- Raw Aim

AND
Robot Pose
chassis speeds

Collector

1 Neo

1 ToF

Aim down

In - Right Bumper

Out Left Bumper
- ~~reject until not set~~

Intake until ToF senses Note

Provide Drive Feedback

Reject Intake override
In - Right trigger - Arm
Out Left Trigger

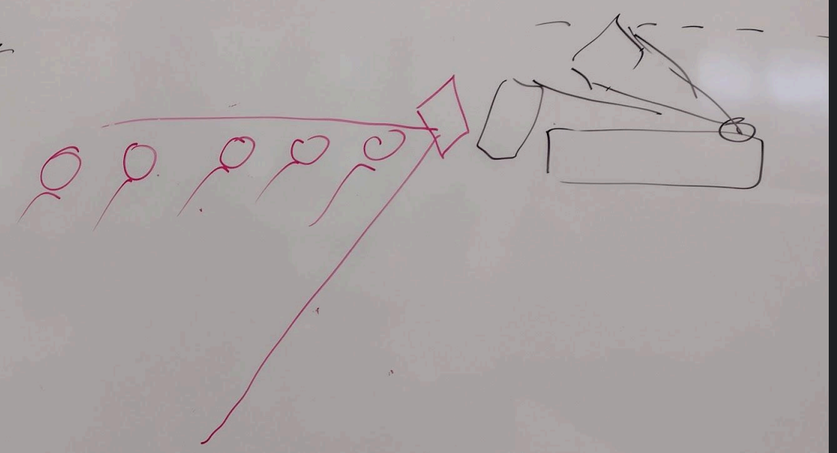
*Arm down in all not Pos always

Arm
2 Neos & Limit switches
Set 5 positions

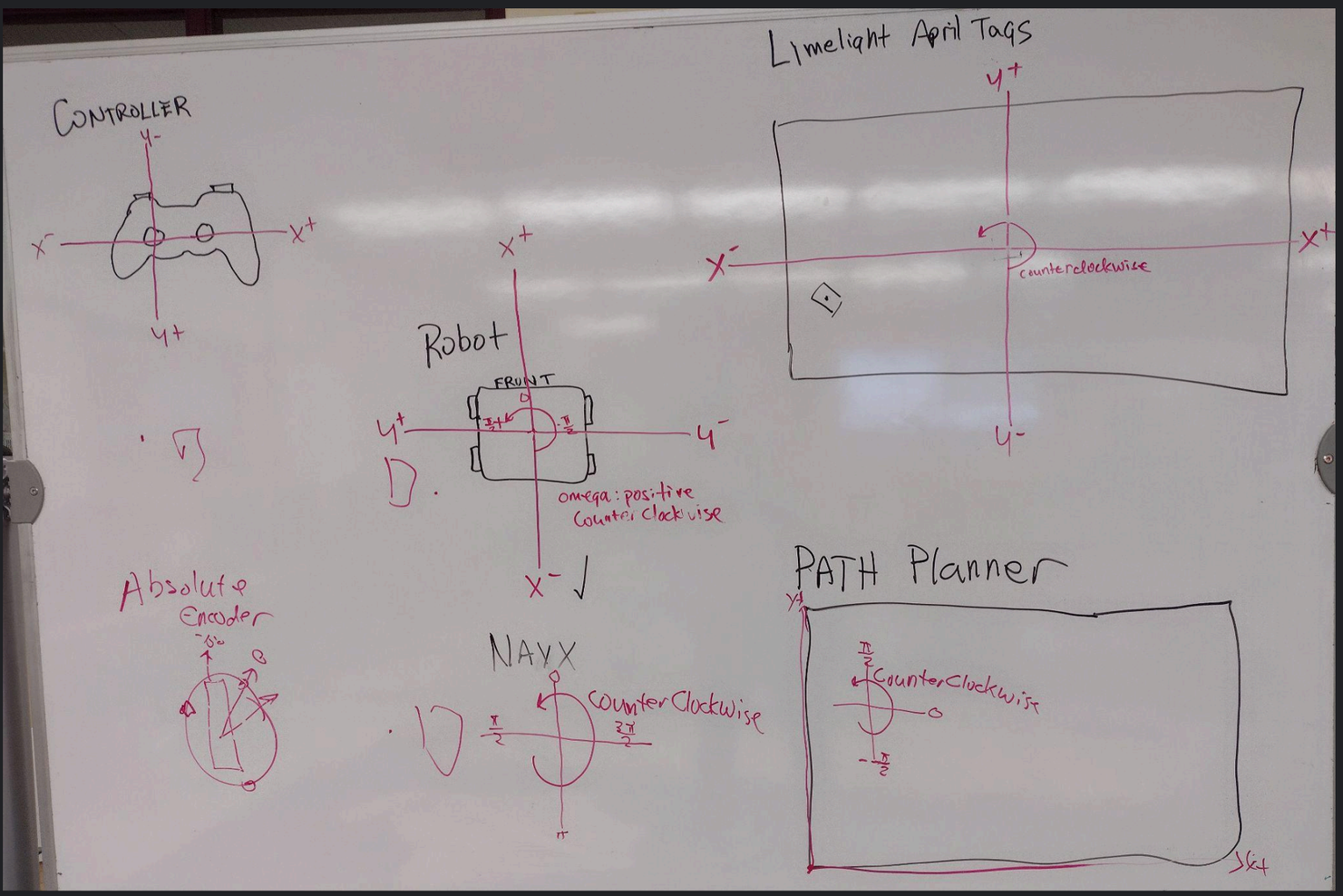
Default to shooting pos

Trap / all way up
Amp / a little down
Sub

Fold Later



Coordinate Systems



Coordinates we get from megatag Limelight april tag localization should be converted to path planner coordinates which have (0,0) in the corner of the field instead of the center:
robotX = limelightX + 8.27 // half field width in meters
robotY = limelightY + 4.1 // half field height in meters

Collector Notes

PathPlaner Notes

2024 Crescendo spreadsheet

Path Planner

Easy to install and make a path. I already made a simple 1 meter forward path

Crescendo Preseason

Programmers

- Alison Miranda: April tag and Limelight train for node detection
- Vincent Dizon: Autonomous Framework and paths
- Cameron Sanders: scollector base
- Analysa Nunez: limelight train for node detection
- Aditya Singharoy: path planning
- Joaquin Herrera: climber

Priorities

- Odometry: evaluate why odometry is off, when does it go off most, this is mostly done but could be evaluated more in-depth.
- pathplanning: create through points for path planning: look at the incorporation of the path planning tool into our autonomous framework.
- control direction of rotation in auto: this would be solved by following trajectories with path planner.
- control mechanisms simultaneously in auto (also solved by path planner probably)
- april tag vision improved and evaluated:
- improve ramping with swerve drive (Cameron?)
- Localization: Odometry and april tags for improved localization.
- Path Planning and Trajectories: We should get trajectories into autonomous so that we can more easily plan autonomous routines reliably.
- Autonomous Framework: revamp autonomous so it makes sense and is easy to create new reliable routines with future mechanisms.
- New Swerve Modules: We can't start working on this with actual components but we can look at how the code would be changed with the new components.

Swerve Constants

Calculating the constants for the swerve modules and their PIDs and encoders is not trivial
[google spreadsheet with calculations of constants]

Autonomous Program Concepts and Design Ideas

Add image of whiteboard ideas here.
Describe key terms, classes here

Yet Another Generic Swerve Library (YAGSL)

- https://www.chiefdelphi.com/t/yet-another-generic-swerve-library-yagsl-beta/425148
- https://github.com/mjansen4857/pathplanner/releases/tag/v2024.0.0-beta-5

AprilTag distance fix (from Axiom)

- Change screen to black and white
- Only change senor gain and exposure

→ This allows limelight to look farther and fix the distance problem for the apriltag

Pit Routines and responsibilities

We will always have about 2 programmers in the pit when the robot comes back from a match. We need to insure we always have these programmers and take turns if needed or if we have enough programmers.

Lead developer

- Creates match branch (initially identical to master), builds it and is ready for development on developement laptop.
- insures that current master branch is present and built on every laptop (develop and driver stations).
- Insures current master is on the robot and tested before leaving for competition.
- Watches every match and is accessible by the pit if any issues appear
- Modifies code in pit.
- Conducts code review with mentor, driver, and driver's programmer if there is a change in code.
- Saves and commits (an pushes if online) to competition branch. Responsible for that branch and documentation of any changes during competition.
- reviews log files
- Runs through check with tech programmer before each competition.
- Insures all software is up-to-date with required inspection checklist

Driver's programmer

- With driver, responsible for setting up and testing driver's station.
- Liaison with drive team, documenting any issues.
- Expert on autonomous routines and author of routines and maintains list of routines.
- Responsible for driver's station when it is in the pit (charging) and driver's station is set up for driver.
- Can drive the robot and runs through testing routines when in pit.
- Documents outcome of all autonomous routines and issues during competition.

Tech programmer

- Responsible for transferring log files from driver's station to develop laptop.
- Reviews log files when finished with return of the robot.
- maintains USB cables and assists lead developer when testing in pit.
- Expert at reviewing log files and works on logging in code.
- Maintains programmer's box, double checking checklist of what needs to be brought to a competition to make sure everything is there.
- Responsible for returning all equipment to programmer's box.

Return to the pit should be a choreographed routine that includes all the members in the pit, something like:

- robot enters pit
- Driver's programmer picks up driver's station and places it on the bench, plugging it into the charger that is already positioned there (not the charger that stays on the robot cart) then moves out to the front of the pit to talk to the driver about any issues.
- Tech programmer has USB extension ready (spare, again, not the one on the robot cart, that stays there) and plugs it into the robot and then to the driver's station.
- Tech programmer copies log file from laptop onto a jump drive and then transfers it to the development laptop (folder should be created in documents folder for logs from each competition or training)
- Lead developer programmer is available for modification of code. If there has to be a modification the lead programmer, sitting in the back left corner of the pit at the development laptop starts to modify code. Tech programmer would use the other usb cord (normally left on the cart) to connect the development laptop to the robot and then stay be the robot ready to switch between cables on the robot. The driver's programmer talks with the driver and developer to discuss change in code. The driver and developer agree on the code change, save, and commit on
- The tech programmer is responsible to getting the cable back onto the cart and stowing the other cable too and is responsible for the USB jump drive.
- If there is time the driver's programmer runs through tests of the robot with tech programmer and mechanical.
- Driver's programmer moves driver's station back to the robot, insuring it is ready to go and double checking the joystick and controller are plugged in correctly and operating.

- battery guy?

Testing Procedures

- All issues should be worked on in a branch specified on the first line of the issue description.
- Review code changes with other students and mentors
- Create test script (see below).
- Insure that the branch is up to date with master.
- Run through test scripts by another student and mentor.
- After successfully completing test submit merge request.
- After another review of changes with a mentor the branch will be merged into master.
- Fetch new copy of master and deploy, running through all tests again.
- Close issue.

Test scripts

- Make sure to take in as many possibilities as you can forsee.
- Develop a set of tests for the specific robot.
- Test on two different robots.
- Script should include reboot of code and start in Practice Mode, start in teleOp, start in autonomous, teleOp then autonomous

Crazy Robot Revealed

The problem with the crazy spin was very obvious and should not have happened if I had adhered to our policies (non stated but need to be stated) for coding, specifically thorough testing, non-redundant code, and fail-safe coding. The missing line of code that should have been there is this:

```
      @Override
        public void autoInit() {
                mAimRotation = 0;
                mIsAiming = false;
                resetOdometry();
                setFieldOriented(true);

                mDriveAutoController.reset();
                mRotateAutoController.reset();
                holdCurrentOrientation = false;
                mRotateTeleopController.enableContinuousInput(0, Math.PI * 2);
        }

        @Override
        public void teleopInit() {
                mAimRotation = 0;
                mIsAiming = false;
                setFieldOriented(true);
                mNavx.setAngleAdjustment(180);
                holdCurrentOrientation = false;
                mRotateTeleopController.enableContinuousInput(0, Math.PI * 2);
        }

Testing:
```

If the robot was started in Autonomous without being first in teleOp the continuous input for the rotation controller was not set, making the robot spin when drive(0, 0, 0) was called at the end of an autonomous session that was not timed out. Since autonomous routines rearly have idle time at the end and we normally start the robot in teleOp before switching to practice, we never saw this issue at aim. But, if we tested all cases it would have come up. Testing should always be done with a restart of robot code and then practice just like it would be done at a match.

Redundant code:
We should not be calling initialization code in both teleopInit and autoInit. We should have a robotInit method for each sub system where we call code that needs to be done each time the robot is started. In fact, it would probably be a good idea to force all sub systems to define teleOpInit, autoInit and robotInit. Anything that needs to be done in either teleop or auto should be in robotInit. Only code that is specific to auto or teleop should be in each of those. We already had another issue based on this problem with not initializing states in both teleop and auto.

Fail-Safe:
We removed code that kept the robot from rotating quickly when the arm is extended. That code needs to be added. We need to anticipate any dangerous or destructive operating modes and make them inpossible to happen. We are going to put back into driveSystem that a rotational velocity above a certain safe threshold will not be allowed if the arm is not in a safe position.

Logging:
As we implement more complete logging of the robot state problems can be identified faster. During TWIST there was not time to diagnose this problem because we did not know what was happening during autonomous, although, in retrospect, we have seen this problem with continuous input controllers before and should have recognized the spin as a problem with that immediately.

Fixing errors:
We fixed this problem by removing drive(0, 0, 0) from autonomous but never figured out the root cause of the error until this morning. That was a mistake, because without finding out why the robot was spinning we would not have learned from this error the lessons we need to know for next time. Also this error was present for some time and was seen but never addressed because it was difficult to recreate. Still, careful examination of the code would have revealed the error.

Lessons from TWIST 2023

- Development laptop: always have a development laptop ready separate from driver's station that has current master that has been built on that machine (otherwise it might have old libraries that need to be updated online that will keep the build from progressing). Having that laptop online is preferable but not always possible.
- autonomous logging: autonomous is only 15 seconds long, with about 50 calls of autonomousPeriod() per second, that's about 750 times. We should know what has happened during those 750 calls. At TWIST we had an uncontrolled spin during an autonomous but I could not figure out why and looking at the log did not help. We need to look into 6728 Mechanical-Advantage's level 2 or 3 logging to be able to always know why something happened during a match. But for robotRumble we need to figure out what happened and prevent it. At least we have to look at a safety feature in the code that would not allow the robot to spin quickly if the arm is extended (added issue). We had that at one point (for teleOp) but probably not for autonomous. I am going to look at the code today. The symptoms of the issue was as follows:
    - The robot would begin spinning while it is stopped.
    -

  Here is a list of the steps taken to troublespeeshoot so for
    - Checked ethernet and usb connections, all looked ok
    - Navex swapped Issue persisted
    - Verified no leads were touching on the navx and cut any leads that looked suspicious.

  Here is a list of ideas of what to look into:
    - Look into auto code is continuously updating requests to drive system.
    - Can we determine when we are having trouble with navx and turn off any auto code that is relying on navx to work correctly.
    - Look at the 180 thing when is it done. I think at beginning of autonomous.

- Log review after each match: After creating more extensive logs we need to copy the log(s) from the driver's station after each match (get a working jump drive as standard equipment) and move it to a review laptop (so at least 3 working laptops: driver's, development, review) where a programmer(s) review the log looking for any issues or hints to what went right or what went wrong.

- AprilTags: I believe switching pipelines has not been done correctly in our autonomous routines and needs to be addressed before robotRumble, or just do away with limelight completely. It would be better to get this working reliably for robotRumble so we are more confident in using vision for next year. We need to log when switching pipelines and display the current pipeline in shuffleboard (may already be doing that). April tag identification during TWIST was not working reliably (probably because of not switching pipeline in a timely manner). We should consider making an autonomous step that can be added to routines just for switching pipelines since the switch has to be done before looking for values in code, not in the same routine, but a second before expecting values.

- Need to capture code version in the log file a versioning system for code and snapshots of code, naming versions.

- For robotRumble we should restrict autonomous routines to return to the grid, lift arm and stop, not dropping the cube.

- For next year we need a rewrite of driveSystem to tighten down access to the wheels so there is only one routine (not auto and tele) that sends meters per second for x, y, and radians/second for rotation and keeps robot facing a certain way and detects and plans for navx failure, recalibration or unexpected reset.

Other simple improvements (NJA)

- For slow joystick control implement both the joystick deadzone compensation. (Check if this is different for each joystick)
- Also have a button that cuts the speed in half when pressed.
- The combo should help for alignment and balance.

[Shuffleboard Layout]