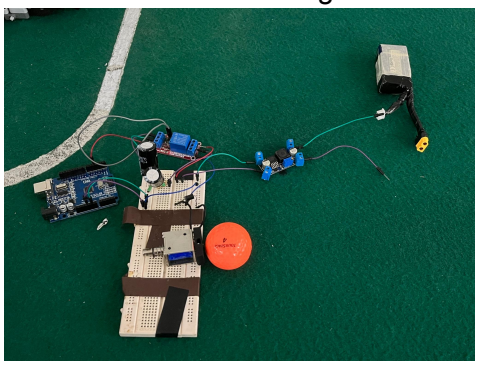
RCJ Soccer Open (Socks) Logbook

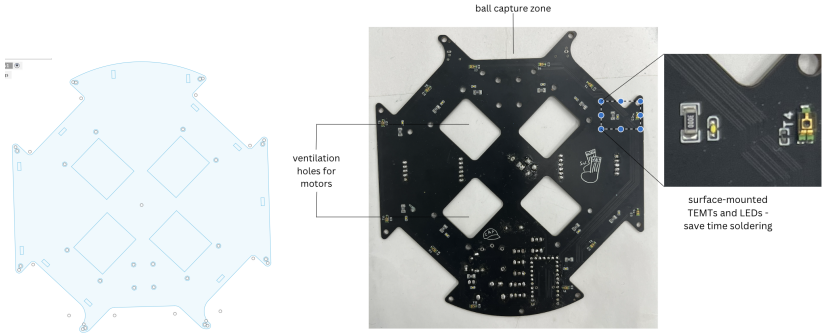
07/12/22-09/12/22

Hardware:

1. Tested 2022 drivers, motors, TFLunas, TFMini Plus
2. Decided on components
    a. Changes made
        i. Teensy 4.1 from 3.5 (main processor): 3.5 out of stock (only have 2 from 2022), has more Serial ports, also higher clock rate because last year processing bottleneck when sensors updated faster than loop time
        ii. TFMini Plus from Luna: needed the range (reliable until 0.1m vs. 0.2m) because of the smaller outer boundary, also has higher distance resolution and frame rate, but it's a bit bigger so will see if can fit
        iii. Teensy 2.0 from Pro Micro (bottom plate processor): greater number of analog ports, same clock rate, also decent price
        iv. ZKBM1 driver from Elechouse 50A: not confirmed yet, but it's much smaller
    b. Wiring plan: https://docs.google.com/spreadsheets/d/1PxFKFD5xBD2-s-LCkRITq23QFiiBBb4gck_AsuzOjyU/edit?usp=share_link
3. Tested 2022 robot
4. Tested kicker circuit using 3300uF and 2200uF 50V capacitors and JQC relay module



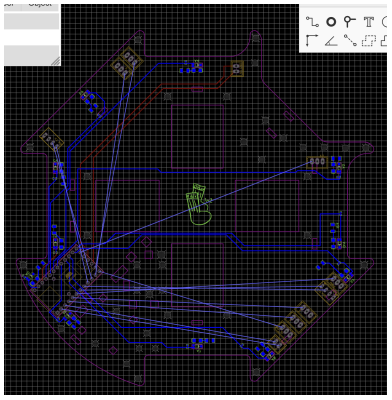
5. Made kicker test PCB with broken out relay circuit to save space (check if the transistor and diodes we have in the room specs are ok)
6. Finished bottom plate outline, will route traces



03/01/23:

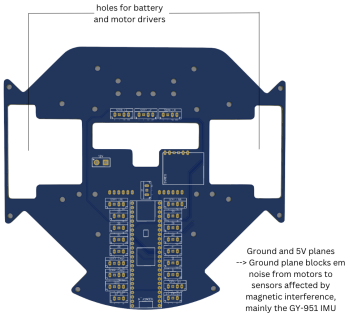
Hardware:

1. Tested new ZKBM1 drivers (smaller than 2022 ones so hopefully can use)
2. Made battery shrouds
3. Finished first draft of bottom plate PCB



a. Stuff to change: change TEMT6000/LED spacing, move broken out JSTs, add driver pins, change JST going to top plate (missing common GND pin), check LED specs, add ballcap stuff, dribbler mount holes (in case), add regulator footprint, battery shroud mounting holes, 12V from battery
b. Need to find somewhere for the boost to g
o (easily accessible in case we need to tune voltage etc.
c. Need to integrate the kicker circuit

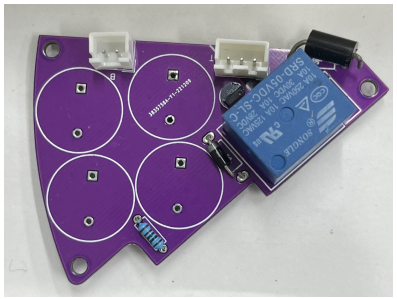
4. Overall robot layout
    a. Bottom plate: kicker circuit, teensy 2.0 at the back, SMT TEMT600s, ballcap TEMT6000 and LED across the ballcap zone (lightgate)
    b. Drivers (ZKBM1, wired through bottom plate then JST to the top Teensy) standing at the sides next to batteries
    c. Top plate: LiDARs (2 back, 1 left, 1 right, 1 front) then Teensy 4.1 at the back + switch (snap into the back shroud/tape)



    d. Camera in middle, acrylic tube + mirror on top, then GY-951 on the mirror hat
    e. 12V wiring: one battery to drivers, one to plates + boost

To do:

1. Test kicker test PCB

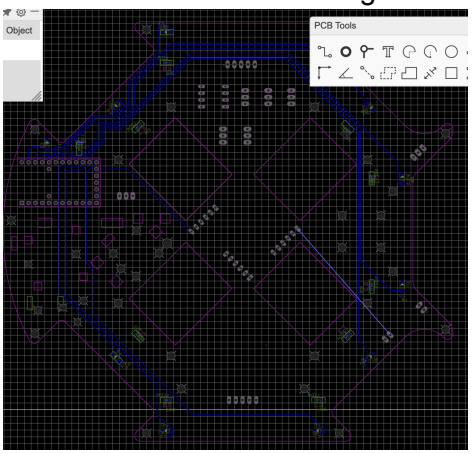


    a. Try with the components in the clubroom first - not specifically specced but from what I read in the datasheet should be fine
    b. Test with different capacitors also
2. Top PCB: LiDAR placement, breakout all the Teensy pins, 12V from battery, 5V regulator
3. 3D prints: LiDAR mounts, side shroud (mount drivers onto shroud, then slot driver onto the headers on the bottom plate), ballcap TEMT mount

04/01/23

Hardware:

1. Kick circuit
    a. Tested breadboard circuit first to see the capacitance needed, using 2200uf because it maxes out kick
    b. Tested kick test PCB, relay wasn't working because the transistor resistor (1K, using this tutorial Arduino Relay : Circuit Diagram, Wiring, Working & Its Code) was wrong (probably because our transistor is different), switched to 150 and it works
2. Top plate
    a. Laid out most of the components on Fusion
    b. Cut holes for the motors and battery
    c. Finished LiDAR placement using Fusion
3. Bottom plate
    a. Change LED to 2x brighter one, changed the resistor too
    b. Moved the TEMT6000s
    c. Finish the schematic + wiring the TEMTs, so only the JSTs + lightgate left



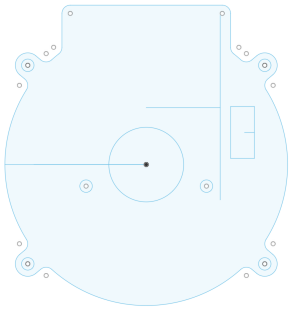
To do:

1. Test the ballcap TEMT placement (high enough to be cut by the ball, range of blocked/not blocked must be decent so it doesn't accidentally trigger)
2. Figure out where the boost is going to go
3. Bottom plate
    a. Finish routing current components (JST + driver)
    b. Add kick circuit (once fully tested on the test PCB)
    c. Add regulator
    d. Add 12V through holes
4. Figure out height of the camera (to see the whole mirror)
5. Top plate
    a. Finish LiDAR JST placement
    b. Finish regulator placement
    c. Add holes for camera plate to standoff
    d. Add 12V
    e. Cut the front of the PCB for dribbler (maybe)
    f. Add hole for the wires from bottom plate to route through

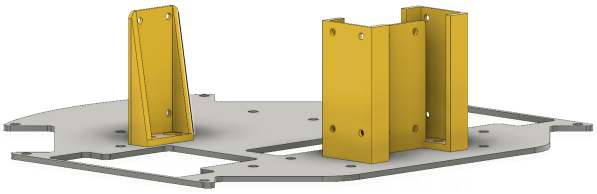
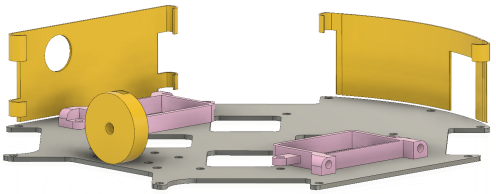
16/01/23

Hardware:

1. Finished the PCBs and sent them on 11/01 (integrated kick circuit into the bottom plate, left old camera holes on the top plate in case we need to swap it out for the 2022 camera setup)
2. Made camera plate outline for OpenMV H7



3. Made 3D prints: LiDAR mounts, boost mount, side shrouds, battery shroud, kick plate, acrylic tube mount

4. Soldered Teensy 4.1, 2.0 (solder Teensy 2.0 directly onto bottom plate without headers, so there's enough height for the boost to fit above it)
5. Solder boost output (still need the connectors for the input from the Lipo)
6. Bought some random IR LEDs from Cytron to try for the lightgate, has potential to work but still need to figure out the best distance (LEDs need to face directly head-on)
7. PCBs arrived on 16/01, will solder and test soon

To do:

1. Test PCBs
    a. Make sure nothing's shorted before testing
    b. Desolder the regulators from the old plates because we don't have that many and they're expensive (will still need to buy more though, for spares etc.)
    c. Test TEMTs and kick circuit on bottom plate (test with jumpers first)
    d. Solder top plate JSTs and test sensors/motors through the PCB

<u>17/01/23</u>

Hardware:

1. Spent an hour desoldering the regulators without destroying them, but for one of them the solder pad ripped off (still usable, but not ideal)
2. Kick circuit
    a. Soldered the kick circuit but the relay wasn't clicking
    b. The new boosts we bought only go up to 36.5V (the ones we have in the room go to 48V) but still usable maybe for goalie
    c. Tried to desolder the relay to debug the problem but it couldn't be removed and the plate sparked (probably because we were using an ungrounded iron and the capacitor was still in)
3. Tested top PCB with LiDARs and they work

<u>18/01/23</u>

Hardware:

1. Bottom plate
    a. Managed to desolder the relay
    b. Kick circuit debug
        i. Tested the bottom plate part by part and worked initially
        ii. Realised we forgot to solder the solenoid diode, but after it was soldered the circuit died, so we cut the diode but it still didn't work
        iii. Multimetered the circuit, realised the capacitor was reading 0V so desoldered it and luckily it's still working
        iv. Continued multi metering and realised the relay leads were wrong (COM was connected to NO when no power was supplied, when it should be NC), but when

we hit the relay/shorted it to GND it went back to the correct default

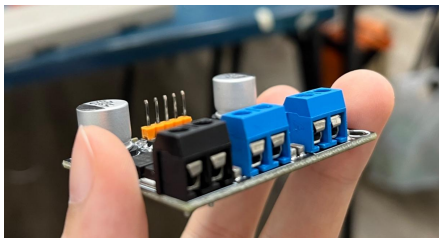
v. Suspected that it was the diode problem and compared to test PCB, realised the diode was reversed (essentially, it caused the relay to hold charge and stay 'on' until it was discharged, and the capacitor appeared to be dead because it was permanently connected to the solenoid and discharging)

vi. Soldered everything correctly and it kicks, with 48V boost it kicks until the back of the goal and rebounds a bit

vii. Video: https://drive.google.com/file/d/16rgqDHRvLwn6EW0ctfu3lywtYC6Cz0GM/view?usp=share_link

c. Tested TEMTs and they work
d. Soldered the remaining components
e. Forgot to mirror the battery shroud mounting holes on one side of the bottom plate, so will screw on the battery shroud on top of the motor holes

2. Top plate
    a. Soldered the remaining JSTs and components (some of the footprints for the LiDAR pins are labelled wrongly with TX/RX swapped)
    b. Tested with camera and works
    c. One of the PCBs got very badly scratched so will redo another one because the copper is exposed
3. Motor drivers



    a. Tested ZKBM1 drivers in the actual set-up, realised the speed can't be controlled, at 20 PWM and 50 PWM speed looks almost the same
    b. Also the motors take a very long time to accelerate up to speed
    c. Possibly the PWM frequency (400Hz for driver, Teensy frequency is default 4K/5K) so reconfigured the pins but still not working, need to test further to see if we can actually use these drivers or if we need to revert to the old ones

To do:

1. Figure out what's happening with the driver
2. Desolder the regulator on the scratched up PCB
3. Make camera PCB (socket OpenMV and JST to top plate)
4. Make lightgate mounts for the IR LEDs

<u>19/01/23</u>

Hardware:

1. Motor drivers
    a. Tested using jumpers and Arduino Mega and it could change the speed but was still accelerating
    b. Tested with Teensy and also could change speed (force 400Hz, need to find actual frequency from Teensy website)
        i. Video: https://drive.google.com/file/d/1dMfBrm3g-5fHtsMqGhnHNeezUX_N05zw/view?usp=share_link
    c. Could not figure out why it was accelerating though
2. Assembled the bottom and second plate (just put in the motors and motor driver), tested all 4 motors but one of them didn't stop spinning (will continue testing)
    a. Video: https://drive.google.com/file/d/1dzhx9x2iqvJHr44Zg6qaMkcB2ZWf_B15/view?usp=share_link

To do:

1. Find PWM frequency for Teensy
2. Make ballcap mount
3. Edit front LiDAR mount
4. Edit kick plate
5. Figure another way to secure the driver (heat sink is covering the mounting holes, can ziptie in worst case)
6. Make battery shroud

25/01/23

Hardware:

1. The bottom plate shorted, thought it was the kick circuit
    a. Realised we should have isolated the GND for the kick circuit
    b. Tried to debug the kick circuit but couldn't figure out the problem (the same thing happened in 2022 when we tried this set-up)
    c. In the end just soldered another bottom plate to test the movement first
2. Motor driver (tested externally because the robot set-up was shorting)
    a. 4000Hz frequency works (528MHz clock, 15 bit resolution), kind of works, acceleration wasn't too bad
    b. But when we plugged in the driver power the motor would jerk a bit before starting normally, not sure how problematic this is, but tried with another motor/driver and seemed to work normally, possibly motor problem

26/01/23

Hardware:

1. Made lightgate mounts
2. Resoldered the top plate regulator because the soldering was bad
3. Assembled components needed for movement, but motors/driver still weird
    a. Tested using 4028.32Hz and forced 400Hz but the speeds look too fast
        i. Video: https://drive.google.com/file/d/1iahx3quLunqgYqiBGUpAi8z_sztVgiBc/view?usp=share_link
    b. Suspected that we weren't actually changing the resolution so when we changed the PWM frequency for the higher resolutions it still just maxed out at 255, but checked the Teensy website which says it maps automatically so shouldn't be the problem
    c. But found out there's only 3 timers that have 32bit resolution, need to check documentation to figure out which timers can handle and use the ideal frequencies
    d. For now for testing purposes will use the 400Hz frequency
    e. Also realised the accelerating is possibly due to the capacitors on the driver being excessive (it has like twice the capacitance as our old driver)

To do:

1. Adjust front LiDAR mount
2. Plan is to get balltrack up before next week

30/01/23

Hardware:

1. Screwed on standoffs and old tube/mirror setup to test compass
2. Used an old print to mount the OpenMV and test the height it needed to be at (85-90mm from the top of the mirror to the bottom of the camera PCB)
3. Mounted the whole setup onto standoffs to simulate the robot height so can start testing balltrack

Software:

1. Can receive GY angle, but calibration is wrong
2. Tested movement with compass correction and it rotates to face the front but looks jerky (probably because speed is low and normally it wouldn't run, but because of the driver having too many capacitors it half accelerates)

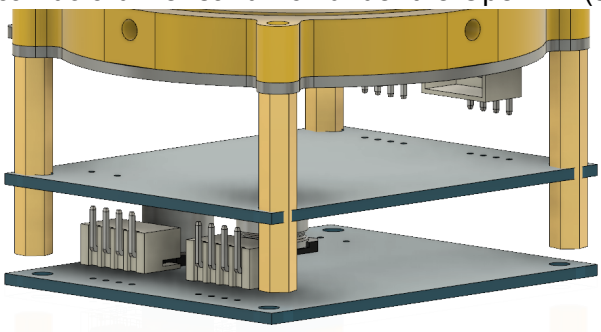
3. Ball detection
    a. Thresholds not tuned, randomly detects the goal and standoffs
    b. Couldn't detect the ball for a bit but realised we set the pixel threshold too high, but can only sense the ball when it's very close
    c. Got it to send the ball angle
    d. Managed to read from teensy side
        i. Video: https://drive.google.com/file/d/1tEVGRIawMAO1ozAP3KC3P7hxirB1J3YW/view?usp=share_link

To do:
1. Hardware
    a. Find FOV of OpenMV to know exact height of camera mounting (use calibration code)
    b. Remake mirror equation at some point
2. Software
    a. Calibrate compass
    b. Test movement at higher speed, tune gain for compass correction

---

01/02/23

Hardware:
1. Tried to centre the camera lens but can't get it exact, will try to print an centering guide
2. Camera plate: need to extend the plate so the OpenMV can socket into the plate
3. Motor drivers: will probably need to switch to the old driver because of the acceleration problem, can do a driver sandwich under the OpenMV (something like below)



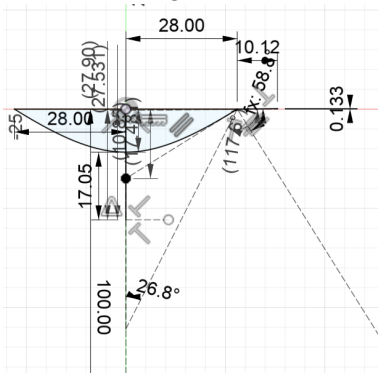
4. Mirror: since the ball is too small with the current mirror, going to try to put the robot in the corner of the field then draw a line to the top of the opposite corner to enlarge the ball without seeing over the edge (new rules say must deal with anything over the wall now)

Software:
1. GY
    a. Port: Arduino Pro/Pro Mini, ATMEGA328P 3.3V 8Hz
    b. Previously we were using Adafruit library for the HMC, but there's a lot of unnecessary stuff about connecting to other Adafruit sensors inside to tried to find another library
    c. Using this: HMC5883L Triple Axis Digital Compass Arduino Library
    d. Realised we can change the settings, changed the range (0.88 gauss), sample size (8) and polling rate (75Hz)
    e. Calibration
        i. Tried to do calibration on the field and it worked, but when I put it on the robot the calibration was wrong, probably because of the setup
        ii. So had to calibrate it with it in the exact robot setup, and calibrated by sending the raw x/y values down to the teensy and calculating the offsets on the Teensy side, and it was worked
        iii. Calibration was still a bit off, so added in our own mapping to remap the values to be more accurate
    f. Managed to send the final angle down to the Teensy, and reset the zero angle
2. Tested compass correction and it works, but need to tune the gain
3. Robot runs at >=0.05 speed

02/02/23

Hardware:

1. Mirror
    a. Tried to CAD one but still wrong, I think the math is wrong
    b. Figured out the math, basically the camera is one foci of the hyperbola so by the reflective property of the hyperbola when the ray of the camera hits the hyperbola the reflected ray passes through the other focus

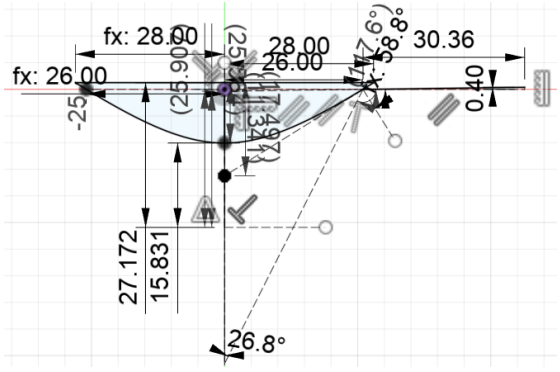


    c. Changes to make: include excess material for some buffer in the forming process
2. Redid camera PCB
3. Redid lightgate mounts to screw onto motor hole so it won't pivot

<u>03/02/23</u>

Hardware:

1. Added excess to mirror



2. Made top hat with the mirror mount, and added cavity for the GY to sit in
3. Tried to place the old drivers in the middle but there isn't enough space and mirror will need to be make smaller, so will be troublesome
    a. It can fit at the side probably, will make another mount and solder the wires onto the top plate, test troublesome way

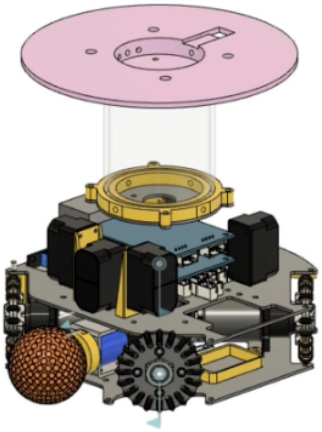
To do:

1. Thicken side shroud (keeps breaking)
2. Make another battery mount
3. Consolidate things to print and quantities
4. Check camera PCB, need to send soon

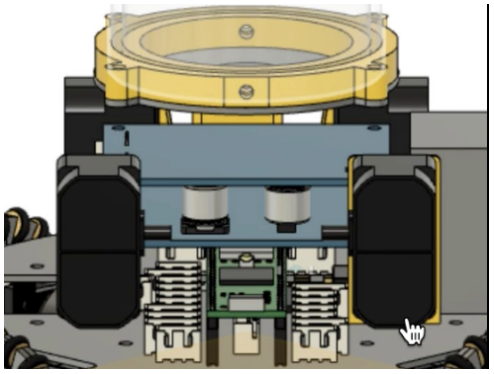
<u>06/02/23</u>

Hardware:

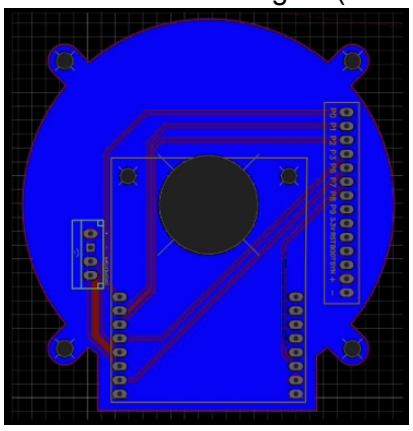
1. Robot CAD



2. 3D Prints
    a. Edited side shroud to thicken the parts that snap onto the standoff
    b. Edited tube mount (thicken the ears, made mounting holes bigger
    c. Made centering print for the OpenMV lens (no official dimensions for the measurement I need so I measured using vernier caliper), made two versions to see which fits with the printer tolerances
3. Realised that my previous calculations were wrong so the driver sandwich can actually fit underneath the camera plate

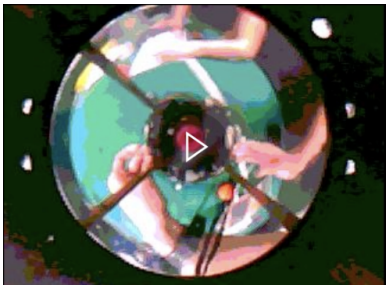


    a. But there's only 2mm space between the drivers and Teensy/Camera, so may not be a good idea
4. Made camera PCB again (old one was very wrong): broke out all the extra pins



Software:

1. Camera mask
    a. Made the code that generates the mask, not perfect but it works (basically need to capture video of the frame buffer then run it through the code, which tracks the stationary parts of the video)



b. Problems
    i. Image (bmp, pgm, jpeg) file size is too big and there's not enough memory (RAM I think, OpenMV has 512KB and the image is 78KB), might be able to avert this problem if we do more calculations on the Teensy istelf
    ii. Bitwise functions don't work with the compressed images (jpeg) so need to convert to bmp/pgm which are uncompressed and use more memory
    iii. Bitwise_and (b_and) doesn't work, not sure why but it doesn't mask, when you use nand/xor/nor there was some random mask on the frame
c. Also need to find SD card to store the mask image

To do:

1. Hardware
    a. Edit kick plate
    b. Finalize and send camera PCB
    c. Print everything and test
    d. Isolate the GND for the kick circuit on the bottom plate and resend
2. Software
    a. Figure out how to do the camera mask

<u>07/02/23</u>

Software:

1. Camera
    a. Mask
        i. Researched and found out that to get rid of the memory allocation error we need to copy to frame buffer since the heap space isn't enough
        ii. Not sure why the current way we're using doesn't work, but found a forum post that was saving the images from the frame buffer and comparing it to the updating frame buffer, which is the same idea as what we're trying
            1. Needed to make another frame buffer to store the image before doing b_and and to combine (also we were using the b_and function wrongly, the parameters are 1. what you want to put onto the frame buffer, and 2. mask which is the pixels to change, but since we put the first parameter as the frame buffer itself it was never going to work)
            2. When I added the mask it was inverted, so I just inverted the image before importing it
    b. Changed the angle calculations because they were wrong

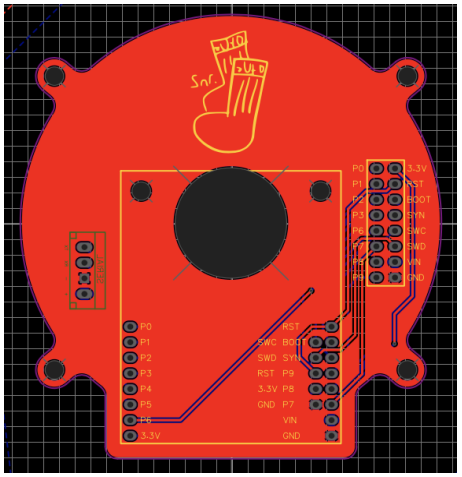
To do:

1. Hardware
    a. Edit bottom PCB: isolate kick GND, change capacitor leads spacing, change diode lead spacing, enlarge relay solder pads
    b. Test the LiDARs with the mounts
    c. Test the kick circuit externally to see if isolating the GNDs will work
2. Software
    a. Camera: add pixels to cm distance conversion, change pixel threshold to decrease as the ball moves further away (not important though)

<u>08/02/23</u>

Hardware:

1. Camera PCB



    a. Mounting holes of the plate were wrong, redid them and made the mounting holes M3
    b. OpenMV was missing some pins so made a new footprint and broke them out
2. Isolated GND on the bottom PCB
3. Edited battery shrouds
4. Sent camera and bottom PCB

Software:

1. Made graph and function to convert pixel distance to cm distance
2. Made graph for the number of pixels of the ball at each distance, and used that to find the minimum/maximum number of pixels the ball can be
3. Tried to implement the graphs but the camera has no more memory, check last year's code for the Jevois and it's basically the same, not sure why the camera is out of memory but will continue checking

To do:

1. Hardware
    a. Edit boost mount
    b. Remake top PCB (not a priority, since we probably don't have budget)
    c. Centre camera lens on the OpenMV
    d. Form mirror
    e. Test all the mounts
    f. Test lightgate
    g. Test camera with new mirror
2. Software
    a. Figure out the code for the GY that doesn't have the HMC chip

<u>09/02/23</u>

Hardware

1. 3D prints
    a. Side shroud hooks go over the standoff too much and are hard to slot on and remove
    b. Back LiDAR mount can't slot over the standoff because the lidar screws are in the way

c. Lightgate mounts the standoff doesn't fit, can just make the hole circular and screw it in instead of slotting it onto a standoff
d. OpenMV centering print can't fit in around the lens, give more tolerance (for now roughly centred it manually)
e. Front LiDAR mount one of the screws cannot be screwed in because the hole is too close to the side supports
f. Top hat GY PCB cannot screw in properly, try to deepen the holes a bit more
g. Battery shrouds seem ok
h. Prints currently mounted
    i. Battery shrouds
    ii. Camera plate (also realised some of the standoffs weren't the same height so fixed that)
    iii. Replaced the top hat with the larger one from last year to block out all the light

2. Realised that one of the motor screws on the top plate shorted the 5V and GND on the top plate (probably because the screw rests on the exposed top plane and is shorting to the bottom plate), replaced the screws with nylon ones (we really should have put VIAs around the mounting holes)
3. Emailed DAMA about the acrylic tubes (1000mm tube of ID 56mm, OD 60mm, cut into 81mm long pieces)

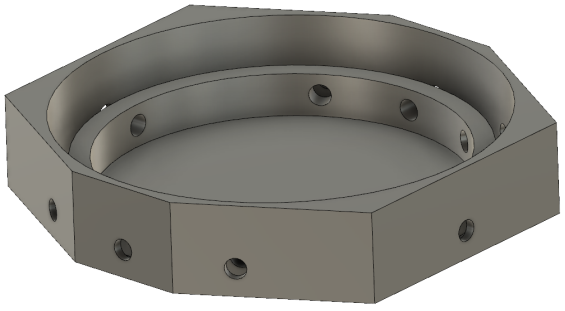
Software:

1. Camera
    a. To optimise the filtering of the ball size can make the function into a lookup instead and pre-calculate the possible ball size at each pixel
    b. Got it to detect the goals
    c. Regenerated the mask but when I tried to drag and drop it into the SD card it says there's no more space which is weird because it should have 16GB but will try again next week

<u>13/02/23</u>

Hardware:

1. Edited the back lidar mount, lightgate mount, camera centering mount, front lidar mount and side shroud
2. Made drilling guide for the acrylic tube



3. Desoldered the Teensy 2.0, 2 regulators and 2 relays from the old bottom PCB (still need to desolder another bottom PCB but currently being used to test software so will leave it alone)

Creation of timeline to ensure goals were achieved

		hardware	software
		~~by 13:~~ ~~-finish prints - back lidar, side shroud, ball cap, openmv centering thing, front lidar mount, old driver batt shrouds, drill guide (ashlee)~~ ~~by 14:~~ ~~-desolder old stuff for new bottom plate~~ ~~by 15:~~ ~~-order and collect acrylic tubes (ashlee) + drill a bunch of them~~ by 17: -test prints -test ball cap (chloe) ~~-if kenneth can print the mirror - form 1 mirror and test~~	by 15: -camera (cadence) **ORGANISE THE LIBRARIES by 17: (cadence) -tune compass correction -temts
1	13-17/02		
		by 20: **-test kick circuit, if doesnt work need to use broken out kick circuit (ashlee)** **-test camera plate (chloe/cadence)** **-ask clusterlab to print more** **by 21:** **-assemble 1st bot** ****desolder testing bot bottom plate** **by 22:** **-solder new bottom (ashlee) + top plate (chloe) + cam plate (chloe)** **by 24:** **-test movement with old drivers** **-form 2nd mirror**	**by 21:** **-tune camera to new mirror (cadence)** **by 23:** **-striker ball track (cadence)** **by 24:** **-lidars + confidence/slowdown (cadence)** **-striker boundary (temts + conf + slowdown) (cadence)**
2	20-24/02		

<u>15/02/23</u>

Hardware:

1. 3D prints
    a. Top hat the angled holes aren't tangent to the circle so the screws will go in slanted and stress the tube, so will redo, and also need to add needle hole to hold up the mirror
    b. Drilling guide it too tight, cannot fit the tubes
    c. Centering mount doesn't work, I suspect that the datasheet diagram doesn't show the lens actually being centred, but will just print a bunch with different dimensions and hopefully one will work
2. Mirror
    a. Managed to form one that is kind of warped/not cut properly but still usable

    b. Ordered + received acrylic tubes, drilled one using last year's drilling guide since the one I made doesn't fit, used the 3.2mm drill bit but it cracked (in the future use smaller 1.5mm one first)
    c. Needed to insert a needle into the mirror and print (to hold up the mirror so it doesn't fall off the print) but I forgot to put a hole for the needle in the print so we hammered out own, it's off-centre for now but will use for testing

Software:

1. Camera
    a. Retuned ball thresholds, realised with the white top hat the contrast is much better than the black one
    b. Realised we should fix the camera settings so the frame doesn't change whenever the robot runs, used the ones from 2020 camera code (tyhousevals.py) but eventually need to check and tune those
2. Movement
    a. Ball track does the correct thing
    b. Compass calibration is extremely off again so need to retune
    c. Compass correction using the new drivers on its own still very weird, have given up and will build a robot with the old Elechouse drivers

To do:

1. Hardware

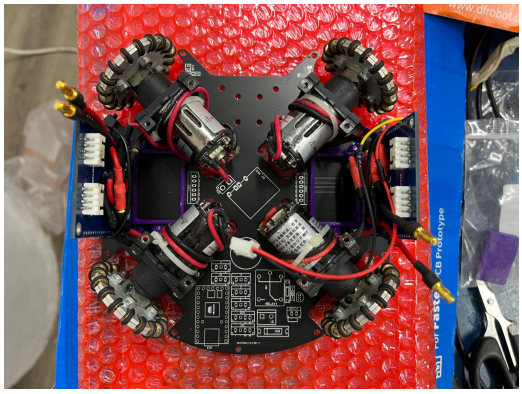
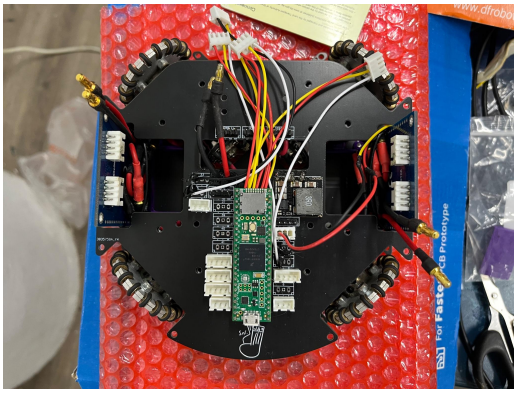
a. Fix the prints
b. Print new mounts for the driver
c. Test the lightgate
d. Make a robot with the old driver

2. Software
    a. Camera
        i. Figure out fixed settings
        ii. Tune the pixel distance to cm distance graphs
        iii. Change bot mask to a proper one (need to make it run through a changing environment, maybe checkerboard, needs a high frequency pattern to work properly since the one we have right now is kind of bad)

<u>16/02/23</u>

Hardware:

1. Robot with old drivers

    a. Got driver mount prints
    b. Soldered the top plate with JSTs for the old drivers configuration, soldered regulator
    c. Tested 5V and GND on the new top plate and works
2. Dismantled the other 2022 robot and took off the motors also
3. Also new bottom PCB and camera PCB delivered but haven't done anything yet

To do:

1. Hardware
    a. Test kick circuit
    b. Test old drivers
    c. Test camera plate
    d. Test lightgate
2. Software
    a. Recalibrate compass
    b. Test compass correction
    c. Test ball track

<u>20/02/23</u>

Hardware:

1. I accidentally reversed the polarity of a TFMini Plus
2. Added LiDARs to the robot, wiring is a mess right now but it fits (possibly wrap wires around the LiDARs so it's not too messy)
3. Tested kick circuit with 5 second cooldown, 100ms kick time at 49V but it stopped working after 2 kicks (relay clicks very softly but the solenoid doesn't activate)
    a. Turned voltage down to 30V and managed to work to increased voltage to 47V and increased the cooldown to 10s and it works
    b. After more testing 8s cooldown, 200ms kick seems to be the strongest and works
4. Tested lightgate using PCB with different resistors to maximise the range, in the end 20K resistor for the transmitter gives 900 when unblocked, 1000+ when blocked
5. Also realised that the tube mount covers the solder pads ot the camera PCB so the camera can't be soldered onto the PCB, not sure how that will work but we will figure something out

Software:

1. LiDARs work
2. Managed to get the robot coordinates, confidence
3. Need to organise libraries

To do:

1. Test functions: slowdown, go to point
2. Add rest of the components of the bottom plate
3. Finish mirror prints (see if can edit tube mount to not cover the solder pads, otherwise maybe directly wire to top plate?)
4. Adjust the lightgate mounts

<u>21/02/23</u>

Hardware:

1. 3D prints
    a. Boost mount need to cut a hole for the solder pads of the Teensy to poke through
2. Finished soldering the rest of the components on the bottom plate
3. Soldered two other drivers

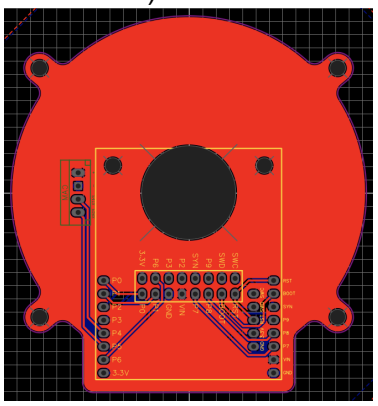
Software:

1. Tested move to point function, slowdown function, confidence
2. Reworked the camera library, neatened the libraries
3. Also spent some time trying to debug “Serial not found” error but realised the wrong Teensy was selected

<u>22/02/23</u>

Hardware:

1. Bought more solder
2. Soldered another top plate
3. Finished second camera PCB (moved the JST to the top plate, moved extra breakouts to under the camera)



4. Resoldered GY PCB because the headers were lifting off the plate
5. Soldered another battery wire
6. 3D prints
    a. Cut the boost mount on the wrong side
    b. Lightgate prints are too far out, LEDs hit the wheels
    c. Drilling guide works

Software:

1. Tested other GY and works, need to calibrate values but it can send/receive
2. Got slowdown to work on the x/y axes, need to tune with new field
3. Move to point function had a problem because the minimum speed of the motor needs to be 0.05, attempted to use an integral to fix which did not work so implemented base speed and it works now
4. Got aiming to work with the camera (also included localization aiming but haven’t tested)

5. Changed the camera code to send the angle for the goals

To do:
1. Hardware
    a. Prints
        i. To edit: lightgate mounts, boost mount, shrouds, wheel hub, centering mount
        ii. To print: top hat, back lidar mount, side lidar mount, lightgate mount, boost mount, battery shroud, side shrouds, tube mount
    b. Check if the 5V input for the driver can be soldered on the inside of the driver
    c. Solder/prepare second robot
2. Software
    a. Change the speed of the aiming so it accelerates
    b. Change the speed of the orbit to hit the ball with minimum force

<u>24/02/23</u>

Hardware:
1. Robot 1
    a. Took apart the first robot to swap out the bottom plate
    b. Resoldered the driver 5V input on the inside since it was sticking out of the plate
    c. Side shroud is a bit too tall
    d. When I reassembled the drivers I think the threads of the 3D print (battery shroud) were dead so it kept falling out, so I swapped it for new ones
    e. Finished assembling the robot, made sure the kick was still working
    f. Tried to mount the lightgate stuff, but realised it was sticking out of the plate and about to hit the wheel so decided to just redo the prints
    g. Ran the full robot without the lightgate, but the 5V was randomly cutting out, decided to stop testing and investigate tomorrow
    h. Also the LiDAR wires are a mess and blocking everything so will wrap the wires around the LiDARs themselves (got reprinted back LiDAR mounts)

Software:
1. Tuned ball track such that it stops right in front of the ball
   Added base speed to everything
2. Changed aiming to be fixed angle for 1s and increased the speed using millis() (also tried aiming until the condition failed, but condition was too strict)
3. Wrote the logic for the kick circuit, will test next week

To do:
1. Hardware
2. Software
    a. Test kick
    b. Incorporate lightgate
    c. Cap the speed for compass correction, realised when I ran it with aiming that it started turning a lot because of compass correction, but once I decreased the speed for aiming it wasn't so bad

<u>25/02/23</u>

Hardware:
1. Robot 1
    a. Thought the problem was the bottom plate with the kick so took it off since the robot was fine before I swapped out the plate
    b. Reassembled everything and multimetered, nothing was shorting but the top plate wasn't receiving 5V
    c. Regulator was outputting 1V, realised it wasn't receiving power
    d. Realised it was the 12V input wires not being soldered properly so redid them and it worked, so rebuilt robot without the kick plate to be safe (so software can have a stable robot to test with for now)
    e. But then one of the motors wasn't working, figured out the motor itself wasn't working (the

tab with the wire soldered had broken off), and swapped it out for a working motor
2. Robot 2
    a. Lightgate works but the ball has to almost be touching the kick plate to sense, so ordered some 3mm LEDs to see if they work instead
    b. Mounted the motor drivers and motors
    c. Soldered bullet plugs onto the motors that didn't have any
    d. Haven't mounted the top plate until we organise the LiDAR wires properly

To do:
1. Hardware
    a. Change top hat: thin the hat, make the hole for the needle larger and push the JST hole for the GY PCB out as far as possible so it's smaller in the mirror
    b. Buy zip ties and rubber bands for the LiDAR wires
    c. Drill another acrylic tube for robot 2
    d. Solder the battery wires for the driver for robot 2
    e. Check what other prints are needed and print
    f. Will KIV the wheel hub print we tried to do for now

<u>27/02/23</u>

Hardware:
1. Robot 2
    a. Mounted the front and back LiDARs but haven't printed the side LiDAR mounts yet
    b. Managed to wind the wires around the LiDARs so it's neater
    c. Resoldered the driver cables to length
    d. Added camera plate standoffs
2. Soldered 2 switches but tested and one is broken

<u>28/02/23</u>

Hardware:
1. Printed another set of lightgate mounts and side LiDAR mounts
2. Mounted and tested all the LiDARs and they work
3. Tested lightgate on the Teensy side and range is 20-1023
4. Managed to get it to kick based on the lightgate

To do:
1. Hardware
    a. To print: kick plate, backleft LiDAR mount, boost mount, top hat
    b. Test drivers on second robot
    c. Buy 3mm drill bit and drill the tubes
    d. Form another mirror
    e. Camera and camera plate for robot 2
2. Software
    a. Calibrate the GY on robot 2
    b. Get bluetooth working

<u>01/03/23</u>

Hardware:
1. Robot 2
    a. Tested kick using elapsedMillis switch case and it works
    b. Moved the camera setup onto the robot with the kick
    c. Tested the motors and works
    d. Added the switch
    e. One motor started running super quickly (like a short)
        i. Uploaded a blank code and it stopped running
        ii. When we ran pinMode for pin 9 it started again, and when we did it for pin 10 a different motor did it, so concluded the pairs of PWM pins were wrong
        iii. Checked the Teensy pins and they were correct so realised the motors were

plugged in wrongly into the driver and it works now

2. Robot 1
    a. Took out the unsoldered bottom plate and soldered the relay circuit
    b. Soldered the regulator (tried to desolder two of them but for one of them the part of the solder pad ripped off, but can still use if we're desperate)
    c. Soldered Teensy 2.0
    d. Tested the kick and it works, and the large capacitor can fit properly
    e. Rewired the LiDARs and swapped out the mounts, but still missing one print
3. Ordered the new camera PCB
4. Bought zip ties and a drill bit

Software:

1. Tuned robot 2 (forgot to change the config file to the correct robot, remember to in the future)
2. Robot aims all the time (possibly because the kick plate is blocking the lightgate), will check the values again tomorrow or add more conditions on the camera side
3. Got bluetooth to work (just Serial.write() and Serial.read() and it works)

To do:

1. Hardware
    a. Robot 1
        i. Mount the last LiDAR
        ii. Mount boost
        iii. Add lightgate
        iv. Make new mirror setup (drill acrylic tube, make mirror, new top hat)
    b. Both robots
        i. Add circle kick plate
        ii. Switch to 3mm IR LEDs when they arrive (need to make mount)
        iii. Swap out for the actual camera plate
2. Software
    a. Fix the kick
    b. Tune the goalie code
    c. Add bluetooth stuff

<u>06/03/23</u>

Software:

1. Previously it was aiming all the time because the wrong camera code was uploaded
2. Detecting ball in ball capture zone is purely by lightgate now
3. Changed the formula for aiming so it starts at base speed and quickly accelerates (until a speed cap) which works better than before
4. There was a problem where it was aiming for no reason but realised it was because I didn't set aiming to 0 when the ball wasn't detected (so if it was aiming and the ball disappeared it would still be considered to be aiming)
5. Now sometimes it aims straight forward, possibly because of the slowdown causing it to only move in the y-axis
    a. It wasn't printing the correct x-coordinate so figured it was part of the issue because the left LiDAR was giving wrong readings (possibly tilted)
6. Tested logic of the kick and it works (kicks at the correct time)
7. Video: https://drive.google.com/file/d/1zyk8aU9IAAPDtWeiPcmG3rXR9OfPAjsE/view?usp=share_link

To do:

1. Find root cause of the incorrect localization values
2. Start tuning the goalie stuff

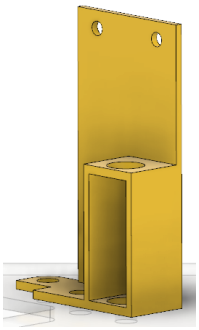
<u>07/03/23</u>

Hardware:

1. The kick died last week while we were testing the kick on the new PCB there was suddenly a spark and the capacitor was shorted, and then it happened on the fully assembled robot too so I

removed the capacitors on both of them
2. Tried to do some research and it's probably the transistor and diode being not properly specced, or could be the boost (can try putting mosfet between the boost and the capacitor)
3. Realised for the camera plate the OpenMV solder pads still clash with the tube mount so will put the camera plate underneath the camera instead of on top
    a. But the plate will cut into the side lidars so will move them out a bit
4. Made the driver cover and back shroud
5. For the side LiDARs the mounting is probably not very stable (because of the small base area it can kind of tilt)
    a. Will try to make a puzzle piece mount for the front and side LiDARs so it's more rigid (supports the entire height of the LiDAR so it shouldn't tilt up and down)

    b. Will make the back LiDAR mounts wrap around the standoff more too



6. Tried to find a large enough heat shrink to cover the LiDAR wires that are wrapped around (because the colours are illegal) but couldn't find so will eventually make some print to cover it
7. Also there has been some high pitched noise (we think it is coming from the LiDAR but will investigate further sometime)

<u>08/03/23</u>

Software:

1. Decided to just use all the old libraries because when we tried to organise them I think some of it got messed up
2. LiDARs
    a. LiDAR 2023 library was wrong (missing sendCommand and setFrameRate functions) so used the 2022 code and got it to run at 1000Hz poll rate (use serialEvent to get the LiDAR values to update asynchronously so it doesn't lag the code)
        i. To test that it is running at 1000Hz use elapsedMillis to get it to print everyth millisecond and check if the values change between each print
    b. Also to check that the LiDAR doesn't die and give a 0 reading when it's at 1000Hz put it as far away from the wall/blue goal and tilt to see how much tolerance there is before it dies, when I tested it seemed ok
3. Camera
    a. On the Teensy side the camera was doing while (Serial.available()) which will lag out the code because the whole code has to wait for the values to updated, so changed to use serialEvent
        i. For now it's assuming the entire string is received, in the future may need to change to account for any incomplete strings of data
4. Updated bluetooth library to use hardware Serial too (not altSoftSerial like last year since this year we have enough hardware Serial ports)
5. Redid LiDAR offsets and goal bounds
6. Updated GY library and tested
7. Tested camera receiving through the library

8. Tested the motors
9. Ran ball track + return to middle + slowdown + confidence + localization aiming
10. Commented out conversion of pixel to cm distance since it was already done on the OpenMV side, localization aiming function doesn't work

To do:

1. Software
    a. Camera settings
    b. Camera localization
    c. Arc towards the goal
    d. Camera ball capture detection
    e. Change debug to compile time (use #ifdef debug)
    f. Test PWM frequency for driver
    g. Change to use serialEvent for bluetooth
    h. Make sure time variables are unsigned long long
    i. Removed Dribbler stuff
    j. See if can put the frame rate under the LiDAR constructor
    k. Change POST_BACK_BR/BL/F for the new field
    l. Change ball capture detection function in the library (move to updateReading maybe)
    m. Check why x_mid is commented out in the ball track (orbit based on which side has more space)
    n. Add continue to track if the robot doesn't see the ball for a few frames since the last ball
    o. Constrain the cm distance on the OpenMV side
    p. Change the slowdown boundaries
    q. Smoothen aiming
2. Hardware
    a. May need to move front LiDAR up (it may have been seeing the ball, but only if the new LiDAR mount doesn't help)

<u>09/03/23</u>

Software

1. Striker
    a. Added continue to track for a bit after the last ball seen
    b. Change ball track orbit distance divisor (orbit more when closer)
    c. Change move to point distance divisor to the field inner diagonal
    d. Changed the aiming time
    e. Added acceleration for aiming
    f. Tried to run at 0.4, 0.6 then got it to run at 0.8 but with the acceleration it didn't work well because the robot wasn't correcting fast enough so it went tilted
        i. Decreased the speed until it could aim properly and it worked pretty well
        ii. Video: https://drive.google.com/file/d/1RksoZ-QKYhXCrMzc2gxZKkiDT-fQATX0/view?usp=share_link
2. Camera
    a. Realised camera FPS was only 13+
    b. Realised we weren't actually cropping the frame and setting the camera settings because we reset the sensor after we changed all the settings
    c. Added some other settings including autoExposure, and commented out the masking and got he FPD to ~30FPS
    d. Cropped the frame to exactly mirror size
    e. Also should get rid of all the functions because they're slow in Python

---

<u>10/03/23</u>

Software:

1. Did code review and went through all the functions
    a. updatePos() there was one part that was doing something to account for the back LiDARs being the goal, but didn't understand what it was doing
        i. Changed it to add when it's within the goal bounds and test if the transition is

smooth (the case where it can go wrong is if neither of the back LiDARs are seeing the wall)
    ii. Tested the transition and realised that at some point the backleft LiDAR read 0, but with a white background it worked fine
    iii. So we reduced the frame rate to 500 then 250 then 125 and then 1Hz but the dead spot still happened so it might be unavoidable, will continue to test and see if it's a code problem (just run the back LiDAR)

11/03/23

Software:
1. The problem with the backleft LiDAR was that the connection was loose, tested the LiDAR with a different wire and there's no dead spot, y-coordinate it smooth around the goal now
2. Changed the ball capture detection function to only use the lightgate
3. Something is wrong with the confidence calculation, last session we changed it to use macros (to easily change for the new field) but the value became negative even though both the numerator and denominator are positive, just reverted to old calculation for now
4. Ran ball track, realised that the line to set the angle to the ballAngle at first was commented out
    a. Sometimes the robot hits the ball at the side so increased angle_mul, but will need to tune properly to make sure it works even at higher speeds
5. Added serialEvent for bluetooth

To do:
1. Smoothen ball track for higher speed
2. Add ball track based on which side has more space
3. Check the bounds (looks like it is going quite close to the wall now)

20/03/23

Software
1. Added individual confidence powers for the goalie and striker when on the line (2.5 x power for goalie response to be fast enough when blocked, high y power so the goalie doesn't move into the goal)
2. Striker x and y confidence power is tuned such that the striker reached 0 speed in each axis when 0.75 of the field is blocked

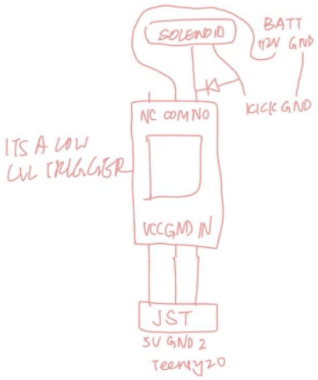
22/03/23

Software:
1. Striker
    a. Added camera for aiming
        i. For the goals the camera sends down the angle to the leftmost and rightmost point, then use (wraparound) average of the angle of the points
        ii. Then combine with LiDAR angle using weighted average (favours LiDAR angle unless blocked)
    b. Tried to turn towards goal while aiming but didn't seem to work well so not using now
    c. Robot detects which goal to aim at by seeing which goal is in front
    d. Need to tune the goal thresholds
2. Goalie
    a. Changed the ball behind goalie speed to be proportional to the angle error
3. Bluetooth heartbeat works

24/03/23

Hardware:
1. Camera
    a. Got the OpenMV H7s
    b. Tested the FOV and needs to be moved down 4.45mm which shouldn't be a problem, just need to change the LiDAR mount
    c. Camera seems fine to swap out, may need to change the mirror but will do testing first

2. Kick
    a. Tested the kick on separate PCB with a relay module instead of our own circuit and it works (kick is weak because of the maximum voltage of the relay)
    b. Managed to desoldered the old components, still need to finish soldering the rest of the components on the bottom PCB
    c. Sketch of the circuit using the relay module



    d. But I think we only have one relay module that's the correct size so will need to buy
3. Realised we only have 8 batteries between 2 teams so need to buy
4. Also the 3mm IR LEDs we ordered arrived but there were actually still 5mm so just sticking to the 5mm lightgate

Software:
1. Camera
    a. Tested new camera on the field and can see the ball further away but the colour saturation/darkness isn't as clear so orange and yellow are quite similar
    b. Checked the FPS and it was 26FPS when it should be 70+, realised that the FPS shown on the bottom corner of the screen is for the frame buffer display, so printed actual FPS and it was 77FPS
    c. Then realised the settings weren't being set (auto_gain and exposure) so when we moved to the field FPS dropped to 50+, tried to do the set register (used in 2021) but couldn't find the proper documentation
    d. In the end realised the settings had to be done in a very specific order (switch off the auto stuff, wait, then set own values)
    e. Can try to tune the settings to see if we can get the thresholds to be better

To do:
1. Hardware
    a. Robot 2
        i. Add 12V kick
        ii. Finalise camera shroud
        iii. Finalise front/back LiDAR mount
        iv. Finish assembling and pass to software
    b. Robot 1
        i. Add basically everything
    c. Both robots
        i. LiDAR wires will probably just tape to cover up the colours
        ii. Finalise driver plate
        iii. Finalise shrouds (side shroud is too tall, back shroud needs holes for the Teensy port, switch and mounting for the bluetooth)

<u>28/03/23</u>

Hardware:
1. Robot 2
    a. Back lidar mounts don't need the box on the screw side
    b. Top back shroud is too short, and need to add hole for OpenMV USB
    c. Resoldered the driver 12V input wires to be long enough

    d. Plate XT60 sometimes blocks LiDARs so needs to be plugged in properly
    e. Standoff height might be wrong because both robots are different heights (mirror on this robot is cut off), will check again
2. Tested LiDARS (got offsets), GY (recalibrated), motors and kick on second robot

Software:
1. Cleaned up camera code (changed settings, removed functions)
2. Tested camera localisation and should work even with goalie blocked
3. Added sending and receiving ball position on bluetooth
4. Calibrated ball distance for different angles around the mirror
5. GY wire cuts back of goal so only half of the goal is taken, so can check the x/y/angles of all the blobs and use the lowest/highest out of all of them

To do:
1. Add checking of the highest/lowest x/y/angles
2. Solder another switch
3. Buy another relay module

<u>29/03/23</u>

Software:
1. Camera
    a. Did the combine blobs thing on Robot 2 camera and it works
        i. Cannot merge the blobs directly because we're using the corners of the blobs to get the left/right bounds, and the corners are taken from the original blob that hasn't been merged (based from what we observed)
            1. Merging doesn't seem to affect FPS though
        ii. Initially tried to combine using angles (corrected to -120-120) and appending to a list, then comparing which is bigger/smaller, and this works for the front half but not for the back (long explanation but need to find the signs of the angles and compare them differently which is very troublesome)
            1. For the same reason also cannot use error to differentiate
        iii. So we are comparing the x-coordinate of the corners of the blobs (biggest x-coordinate for the left size because the sides are inverted) and then calculating the angles from there
    b. Checked FPS and it was 26 so changed the angle calculations to be a look up
        i. OpenMV doesn't have enough memory to store 240x240 array so we did 120x120 for the top right quadrant and then used that to calculate the angles for the different quadrants in the main code
        ii. Changed the x-coordinate to increase from right to left on the frame (because it's left to right in real life)
        iii. Changed the y-coordinate to increase from bottom to top too
        iv. Also removed the angle corrections so it's a normal graph with the origin at the robot centre
        v. Angle outputted by the look up is 0-240 to save on the remapping from 0-360 to 0-240 later on in the code
    c. Combined blob tracking for the goals (cannot combine with the ball one because need the area thresholds for the goal) and use the blob code to differentiate between yellow and blue blobs detected
    d. Also realised that the corners of the blobs (top left and right) weren't always top left and right because it just takes minimum and maximum x-coordinate, but if the maximum x is a bottom corner then it takes the bottom corner instead (happens at the back quite a bit especially when it's blocked)
        i. Tried to check based on y-coordinate but doesn't work, because sometimes the lowest y-coordinate isn't near the left/right bound of the blob and in the middle, so just reverted
        ii. Overall shouldn't make but of a difference though
    e. After all this FPS was 77, then added in UART sending and it dropped to 25, realised it was because our baud rate was 9600 so changed it to be 500000 (Teensy 0% error baud rate) and FPS is normal

    f. Sometimes FPS drops to 50 near the yellow goal, probably because there's many blobs and since we're not merging it has to check a lot of blobs
    g. Also got it to check which goal was in front (not currently used)
    h. Got the camera to send down the angles and it works
    i. Also realised that there is a reflection of the field in the tube that you can see on the camera, for now it's not interfering if we tune the thresholds properly but might need a polarising filter if the robot ends up detecting the ghost objects
2. Teensy side
    a. Added the pixel to cm distance conversion (not doing on OpenMV because it's slow) for the ball (seems symmetrical around this robot)
    b. Checked the loop time and it was 14us
    c. Tested TEMTs and calibrated, got the line to work
    d. Tested lightgate and communication between the Teensy, and the kick logic
    e. Tested whole code and robot seems to be running slowly (probably because we properly tuned the ball distance this time) so need to tune the ball power
    f. Tried to add the kick but realised it wasn't working and line wasn't sending anymore
        i. Teensy 2.0 couldn't be found and externally didn't work either (possibly something in the kick circuit causing it to die)
        ii. Ordered more which will come 4-12/04
3. Stuff changed in libraries: ball distance calculation, config file for robot 2, ball power

<u>30/03/23</u>

Hardware:

1. Checked Teensy 2 on robot 2 but it still didn't work, disconnected the top and bottom plate
2. When I tried to check the one on robot 1 I realised the LEDs weren't on on bottom plate so I multimetered robot 1 and it was shorted
    a. Disconnected top and bottom plates but it was still shorting
    b. Disassembled the whole robot and it stopped shorting
    c. Since it was already disassembled I removed all the unused leftover kick components that were still on the bottom plate to be safe
    d. Swapped out the LiDARs for the new front LiDAR mount
    e. Still need to swap out the back LiDAR mounts so waiting for that before reassembling
3. Soldered another OpenMV H7 and camera plate (need to recentre the lens)
    a. Also the back of the OpenMV says they're R1s but the packaging says they're R2s

Software:

1. Clipping LiDAR box
    a. Basically this is to use camera localization to improve confidence
    b. Tried to clip a fixed rectangle and triangle but it gave negative numbers, both examples given on the website work so not very sure, but KIV for now
    c. Tried to manually compare the lines (see where each camera intersects the LiDAR rectangle) but there are too many cases so won't do now
2. Realised that there were problems with the code
    a. Back goal coordinates are not in clockwise order
    b. Detection for any goal is using angle == 999 which isn't set anywhere, should be using the elapsedMillis anyBlueGoal or anyYellowGoal
3. Added using both goals to find the robot position
    a. Tried to just rely on front goal for the position of the robot (what we did previously) but the moment the goal was blocked at the side the position was very inaccurate)
    b. Changed to use weighted average of the x/y-coordinates from each goal based on the angular width of the goal (larger angular width, higher weight)
    c. Also accounted for no goal being seen, but right now I'm forcing the blue goal to be the front goal for testing purposes so need to change so it to work with the robot facing either goal (possible to send the front goal down from the OpenMV side)
    d. Tested with blocking of the goal and seems to be much better

To do:

1. Hardware
    a. Robot 1

i. Finish second plate
ii. Print another top hat (with a bigger needle hole) and finish mirror setup (form new mirror since the current one is very warped)

b. 3D prints
i. Top back shroud: fix height, add OpenMV port hole, chamfer so it doesn't break
ii. Thicken side shroud (leave space for driver capacitor)
iii. Thicken back shroud
iv. Fix the kick plate
v. To print: top hat, side shroud, bottom back shroud, back LiDAR mount, driver plate, top back shroud

c. Bottom plate
i. Robot 1 still works (LEDs switch switch on, Teensy 2.0 can be detected)
ii. Robot 2 redo bottom plate and remove the kick, then make sure line works
iii. Random idea: use an electromagnet to pull back a spring and kick the ball (found a 40kg spring but decreases to 0.7kg when it's 2mm away)

d. Final touch-ups for competition
i. Add all the shrouds
ii. Tape up the LiDAR wires and cover other colours in tape
iii. Tape up bottom plate (to prevent the static from the field from shorting the robot)
iv. Make sure all the nuts are tightened so nothing drops off

2. Software
a. Change camera code to work for either goal being in front
b. Add in weighted average of the LiDAR and camera positions based on confidence (so it jerks less if the LiDARs get blocked) and then test on goalie to make sure the position is more stable
c. Add compass correction for the camera goal angles
d. Extra improvements for the camera localisation
i. Currently we're using 4 angles from camera (blue right/left, yellow right/left), and we are using the weighted average of the blue right-blue left and yellow right-yellow left angles
ii. But we can expand on that to take the weighted average of each pair of angles (br-bl, yr-yl, br-yl, br-yr, bl-yr, bl-yl) so each of these pairs will give one robot point, and then take the weighted average of all these points (but not sure if it's more accurate, and requires seeing both goals)
iii. Also possible to use this method to get confidence (rectangles of the points) but same problems
iv. But try to figure out the clipping of the polygon first to improve confidence

<u>31/03/23</u>

Hardware:
1. Swapped out back lidar mounts
2. Added camera standoffs and recentred camera mostly

Software:
1. Edited camera code to adjust the cropping of the frame
2. Camera localisation
a. Changed to work regardless of which side it's facing (I used structs which I realise is unnecessary so will change it later but it still works)
b. Edited the logic for which goal is being seen to make it less messy
c. Added compass correction for goal angles
d. Polygon clipping fixed, realised the equation to calculate the intersection point for the edges was wrong
i. Added the logic for the clipping, so if there is an intersection it calculates the new x and y confidence based on the maximum/minimum x and y-coordinates of the polygon vertices
ii. Tried to find the centroid of the polygon as the new coordinate but it's not very accurate so currently using weighted average of LiDAR and camera coordinate

To do:

1. Hardware
    a. Robot 2: print top hat, form new mirror and assemble
    b. Finish the rest of the 3D prints
2. Software
    a. Work on logic for the localization based on the different combinations of the goal angles

11/04/23

Video: https://drive.google.com/file/d/1jnEx-p6YMxcIeiz-R1pCh4kNsMHK-Wpr/view?usp=share_link

14/04/23

Video: https://drive.google.com/file/d/1VgHG2tEL00A2mcp-n-eBJ9o9Q-S9emvD/view?usp=share_link

15-16/04/23 (National Competition)

Video:

03/05/23

Hardware:

1. Kick circuit
    a. Since both our own relay circuit and the relay module circuit failed, I tried to use random motor drivers I found to drive the solenoid (https://www.ti.com/lit/pdf/slvae59), basically just wired how a motor would be (no need diode, just use the h bridge to handle the back e.m.f.)
    b. Jumpered the red driver (fan one) and tried to test by shorting PWM pins to 5V/GND but didn't work, everything seemed to be working and wired correctly but the driver wasn't outputting correctly (gave 0V)
    c. Tried to use CJMCU driver and it worked (rated for maximum 27V though)
    d. Added a 50V 470uF capacitor on the driver input for enough current (seems good enough, when we tested at 48V it has enough charge to kick more than once, but doesn't do that for 27V probably because the voltage is too low)
    e. Tried at 48V and also works but decided to keep it at 27V since both manage to reach the opposite goal from our penalty area, so I won't risk shorting the robot
    f. Kick seems the same for 25ms as 100ms so kicking for 25ms now and using 7s cooldown, works for multiple continuous kicks
    g. Soldered the circuit into the spare bottom plate (will probably put the capacitor behind the driver, or fold it onto the driver itself)
    h. Works on the PCB (using the breakouts), but tested using Arduino UNO powered from the computer instead of the Teensy 2.0 and regulator
    i. Video: https://drive.google.com/file/d/19Ec2gjpfInYkw2ufejvWAhYucZt4xDfq/view?usp=share_link

08/05/23

Hardware:

1. Soldered the whole kick circuit into the bottom PCB, tested with Teensy 2.0 and it works
    a. After assembling tested sending kick command from Teensy 4.0 and it works
    b. But cannot upload while the kick is activated because the power keeps cutting out (computer power is back powering the regulator)
2. Assembled the rest of the robot
    a. Extended the bottom and top plate 12V wires to plug across robot, realised it's easier to extend the XT60 side of the cable so will do that instead
    b. Tested all the sensors and motors

Software:

1. Camera
    a. Changed the camera code to only detect 1 set of goal blobs (based on width of angle, if angle is more than threshold then don't switch, else switch to detecting other goal)

because when we tried to detect both goals at the same time the FPS dropped
        i. FPS is ok after switching to this
    b. GY wire was blocking back goal which caused some problems, but can probably tape it to the side so it doesn't block the goal
    c. Need to tune the angle threshold (right now it's 20-25º out of 360)

To do:
1. Software
    a. Tune angle threshold
    b. Possibly add a case to not switch to the other goal if the other goal cannot be seen (might happen since the mirror is uneven, but will see if needed)

<u>09/05/23</u>

Hardware:
1. Got new drivers for kick, but not soldering a new plate until the current one is completely tested
2. Changed the lightgate mounts for robot 1 to the ones that have space to add mounting tape to pad the ball capture zone

Software:
1. Camera
    a. Changed switching goal condition to use the distance from goal (if distance > 80cm), using the x/y-coordinate of the middle of the top line of the goal
        i. Added pixel to cm distance conversion (but it was only calibrated only for the front of robot, may need to do the weighted one)
    b. Added the rejection of the angles because at certain extreme angles at the sides the lines of the angles become too parallel and the coordinate reported is very inaccurate (may need to recheck these angles)
    c. Switching between the goals works even if other goal cannot be seen (made it switch back in next loop)
        i. But switching doesn't always happen in the centre because of the uneven mirror
2. Teensy
    a. Changed the camera library to the shorter string length (since only sending one goal's angles down down, so it's 2 ball, 2 goal, 1 end character)
        i. Added a struct to take in the left/right angles instead of angle and distance
    b. Sending down to Teensy works
    c. Added camera to update reading for the left/right angle
    d. Added calculations for angle width and middle goal angle

<u>10/05/23</u>

Hardware:
1. Realised lightgate wasn't working and that the emitter wasn't turning on (used camera to check)
    a. After desoldering realised it was because there wasn't a resistor for the emitter
    b. Since it was also desoldered I just transplanted the working one from another plate, and tried to do the same for the receiver but the solder pad came off so need to resolder the whole plate (use this one as a backup, can use a wire to solder the leg of the LED directly to another breakout)

Software:
1. Added kick condition but couldn't test because the lightgate wasn't working (conditions are robot has enough speed, ball is in ball capture zone and the goal is in front)
    a. For enough speed condition I made it 0 speed at front striker bound, full speed at back striker bound, printed speeds seemed reasonable but need to test with the robot running to see if it kicks often enough

<u>15/05/23</u>

Hardware:
1. Made another bottom plate and transplanted all the lightgate components

2. Resoldered the Teensy 2.0 on bottom plate (used headers so it's removable)
3. Desoldered the boost off the old kick circuit (only one that can go up to 50V) and will use for striker (not necessary right now because we're kicking at 27V but just in case we decide to increase the kick voltage)
4. Tested lightgate and it works

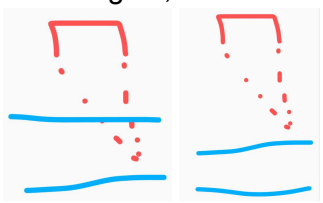
To do:

1. Hardware
    a. Resolder the battery wires XT60 side (if I increase the plate side length the bullet plugs will be sticking out into the battery hole and the battery won't be able to fit)

<u>16/05/23</u>

Software:

1. Camera coordinate largely reliable but when it switches between the goals the values may jump (~10cm difference between the coordinates) so need to check
    a. But could be because of the camera thresholds so will tune and check again
2. Added the pixel to cm distance conversion for the distance to the goal (same logic as ball)
    a. But using the camera to find coordinate of robot and then finding the distance would be more reliable since the mirror is uneven
3. Tried clipping the LiDAR bound using the camera point for y-axis and seems to work
    a. Basically we update one LiDAR bound depending on which goal we're using (back bound for front goal, front bound for back goal)



    b. So for front goal if camera y-coordinate is more forward than LiDAR front bound then make the LiDAR back bound the LiDAR front bound (maximum confidence), else if the camera y is more forward than the back LiDAR bound make the back LiDAR bound the camera bound (left picture), else don't use the camera y (right picture)

<u>17/05/23</u>

Hardware:

1. Resoldered the battery wires to plug across the robot and seems to work
2. Finished assembling the robot
3. One of the LiDAR wires broke (the one I soldered on my own because we ran out of the female header connectors) but resoldered it again and it works
    a. Also we can't use the male side on the LiDAR and female side on the plate because its too tall to fit under the camera PCB
4. Also for the blue battery shroud the driver pops out if you put the yellow batteries in

Software:

1. Previously I shifted the front striker bound back to be before the robot touches the line, but reverted back to the previous one which was when the centre of the robot was on the line (otherwise it'll be very slow when scoring)
2. For kick condition widened the kicking goal angle to be +-10º, also may need to power/tune the minimum kicking speed but for now seems ok
3. Added the camera localisation but the robot runs slower with it (not sure why) so taken out
4. Also added aiming using the camera when seeing the front goal and it works when the goal is blocked (aims towards empty space)
    a. But might be messing with the aiming because when we tested it on the right side of the field it was running at too steep an angle and lost the ball sometimes (had this problem previously too, so may just revert to using the localisation aiming
5. Robot tilted counterclockwise slightly whenever it kicks but doesn't seem to be affecting anything

To do:

1. Hardware
    a. Make purchase list
    b. Make kick plate
2. Software
    a. Transfer the variables that are different for each robot to all be in the config file (there's stuff like aiming acceleration, aiming angle multiplier that we put in during nationals)
    b. Add turning to face the goal

22/05/23

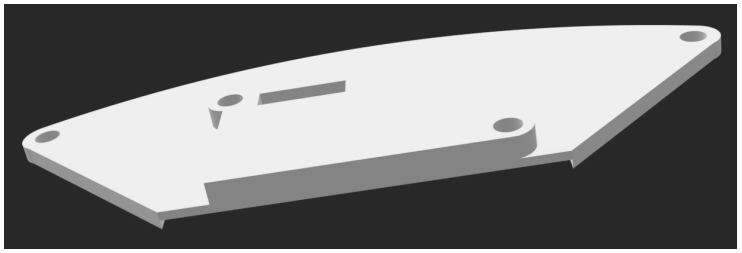
Software:

1. Printed the speed of the robot with and without the camera localisation and they seem the same (previously the camera localisation looked like it was making the robot slower, but could have just been battery level)
2. Moved the constant/tuning variables for both robots into the config/tuning files
3. Changed aiming back to localisation aiming and added in tilting towards goal, the robot no longer loses the ball
    a. But realised the robot tilts too little on the left side of the field so the kick condition doesn't trigger as much (because of goal angle condition), and on the right side it tilts way too much so sometimes it doesn't score
4. Changed move to point angle to be corrected for the GY angle (seems to help the movement, no long arcs out of control)
5. Robot spazzes out sometimes at the sides (looks like the robot thinks it's at the slowdown boundary when it's not)
    a. Tried taking out camera localisation and it seemed better
6. Then when the robot goes to the penalty and sees the line it moves back in for a very long time (because for the outFlag to be turned off all the LiDARs have to be within bounds)
    a. Tried to change the confirm in to be not seeing the line and it no longer lags, but not sure if that will work reliably to make sure the robot doesn't go out of bounds
7. After a bit the robot started dying and the ball kept leaving the ball capture zone, will continue next session
8. Video: https://drive.google.com/drive/folders/1I2vXPuYBBGPP_UWymSHtPG1PdNmcoolr?usp=share_link

29/05/23

Hardware:

1. For the new boost mount the boost is too far in so part of it is clashing with the capacitor on top of the driver



2. Tried new kick plate (brass insert into the print) and it works
3. Robot 2
    a. Soldered bottom plate, tested kick and it works (the boost is the 36V one)
    b. Added test point for the kick voltage on the solenoid wire using bullet plugs (to test the voltage keep the solenoid permanently in the kick state)
    c. Resoldered the battery wires to wrap around the robot
    d. Assembled until the middle plate

30/05/23

Hardware:

1. Robot 1 kick plate came out when the robot was running, fell onto the regulator and shorted it, it

still works so just taped over the regulator
2. Robot 2
    a. Assembled and tested LiDARs, camera, and GY
    b. Realised the bottom plate wasn't lighting up
        i. Disassembled everything and removed wires connecting the two plates and it was still shorting
        ii. Removed middle plate, tested Teensy 2 externally and it works
        iii. Also the boost wire was ripped out so resoldered that and hot glued over it
        iv. Tried to assemble until the middle layer but it started shorting again
        v. Disassembled the whole robot and tested everything but couldn't find the short
        vi. Reassembled everything and tested all sensors again, tested motors and kick and all work
3. Resoldered two spare drivers and tested

Software:
1. Changed the x-axis bounds to the new field, removed line trigger for now
2. Changed camera frame offset
3. Tuned the ball track function/orbit ball (base speed, minBallDist, maxBallDist)
4. Changed aiming point to (910, 2430) if the robot is between goal bounds, otherwise robot aims to 5cm in from the goal post (since the robot always aimed at too great an angle at the sides)
5. Tuned aiming acceleration
6. Changed how lower/upper bounds are calculated for x so if there is overlap between the two bounds, use the bound of the LiDAR that is closer to the wall (because left LiDAR values were unreliable)
7. Tuned kick condition so the robot kicks both angles pass through 0 (right angle between 2-180º, left angle between 180-358º)
8. Tuned tilt to face the goal so right side doesn't tilt and left side tilts up to 40º (since the robot is always naturally tilted towards the left)
9. Kick cooldown time reduced to 3s
10. Changed ball capture detection so the first time it must be triggered by the lightgate, but if it's just refreshing the condition can use either lightgate/camera conditions (which are looser)
11. Sometimes the robot stops detecting the ball but if the battery is unplugged and replugged it becomes fine, but the problem isn't the whole robot hanging because it still runs move to point

To do:
1. Hardware
    a. Test kick voltage point
    b. Add kick voltage test point to robot 1
    c. Measure and buy spare O rings for the wheels
    d. Check bullet plugs size (should be 3/3.5mm) and buy
    e. Get consumables
    f. Robot 3
        i. Get prints
        ii. Make bottom plate
        iii. Make middle plate
        iv. Make camera plate and solder camera
        v. Test spare GY
2. Software
    a. Add camera localisation
    b. Tune robot 2

<u>31/05/23</u>

Hardware:
1. Tested kick voltage point on robot 2 (run the driver continuously)
2. Robot 3
    a. Desoldered components no longer being used off the bottom plate
    b. Soldered on light gate, kick circuit (solder pads are damaged for light gate receiver so had to wire it manually)
        i. The other spare plate is more unusable since the regulator solder pad is gone

    c. Boost is missing the clippy cable to the battery (either buy or use bullet plugs)
    d. Resoldered the middle plate (wires to driver were wrongly soldered, added wires to bottom plate, shortened the battery wires)
    e. When I plugged the battery into the top plate the regulator was on fire (I think this was the one where I has to resolder the capacitors because they fell off so it was probably already dead) but soldered on a brand new regulator and the middle plate is fine
    f. Tested LiDARs and one isn't working so we only have one spare (will try to complain to DFRobot and request for a replacement)
    g. Soldered new OpenMV H7 (need to recentre the lens) and a camera plate
3. Soldered 2 spare Teensy 2.0s (swapped 1 out with one of the robot ones)
4. Counted everything used on the robot including the connectors and consumables

Software:

1. Camera
    a. Took out the rejecting angles cases because can be used now to update the x-axis
    b. Made an exception to return 255 when the bot is aligned with the goal on x axis (left and right angle are the same) because the calculations have zero error
2. Camera localisation
    a. Tried to solve a problem where the camera coordinate was always less than the LiDAR lower bound (for front goal)/more than the LiDAR upper bound (for back goal) so it didn't improve confidence
        i. Tried changing the camera coordinate point to a bounding box based on the angle error from the goal, accounting for pixel resolution to find the minimum and maximum possible x and y-coordinates
        ii. But it got complicated when the goal was blocked so reverted
    b. Changed the goal post points we were using for the camera localization to be the front of the goal post, not the back of the goal post
    c. Reverted back to the old camera localisation
        i. For y-axis, clip lower bound if using front goal, else clip upper bound
        ii. For x-axis, clip lower bound if on the right side of the field, upper bound if on the left side of the field, both if in the middle of the field
        iii. Clip by finding the intersection of the upper y-bound (for front goal)/lower y-bound (for back goal) and use the points as new bounds

To do:

1. Hardware
    a. 3D prints
        i. Top back shroud (widen OpenMV hole)
        ii. Bottom back shroud (lower hooks below the boost mount, add bottom hooks)
        iii. Side shroud (add hooks to the bottom so the driver can't flex it out)
    b. Robot 3: solder boost, battery wires, prints, mirror setup, test the plates
    c. Solder remaining spares (OpenMV H7, drivers, spare plates, spare cables, motors)
    d. Buy O rings and bullet plugs
    e. Make sure we have enough standoffs and screws
2. Software
    a. Make sure the clipping works for all cases
    b. Make sure striker doesn't get pushed out
    c. Goalie
        i. Add compass correction for ball track movement
        ii. Add movement out of bounds if striker has ball in the ball capture zone
    d. Tune other robot

<u>01/06/23</u>

Hardware:

1. Robot 1
    a. While testing the robot there was a weird smell
        i. Multimetered all the power but nothing was shorted
        ii. One of the motors wasn't working so disassembled and tested but both drivers and all the motors were fine

        iii. Disassembled the whole robot and tested middle plate
        iv. Tested Teensy 2.0 and multimetered the bottom plate but there was still no short
    b. Added the kick voltage test point (swapped out the spare solenoid), added the boost and kick circuit
    c. Then the bottom plate wasn't lighting up, the regulator was shorted but the rest of the plate was fine so just disassembled and reassembled the robot part by part but couldn't figure out the source
        i. Tested kick, TEMTs, LiDARs and GY and they all work
    d. While running kick plate kicked itself off the solenoid and the plunger fell into the robot again , tried to remove the bottom plate and managed to get the plunger back in
        i. But I couldn't reassemble the bottom plate, so going to wait until I make a print to limit the plunger then reassemble the whole robot because too risky to run
2. Robot 2
    a. While software was running the robot realised the LiDAR bounds weren't aligning because the LiDAR was seeing the floor/ramp when it was far from the wall
        i. Puzzle piece mount was shaky so will add base to the side puzzle piece so it can't tilt down and see the floor
3. Soldered spare battery cables and boost
4. Tested the GYs in the cupboard drawer, 2/4 are HMCs but only 1 HMC works
5. Tested middle plate ports and those currently in use are working

Software:

1. Realised we were using the wrong point to calculate the camera robot coordinate but this only affects the distance to goal calculation (shifted goal point forward)
2. Tuned the longer side ball orbit for robot 2 (when the robot goes closer to the boundaries it orbits inwards to avoid going out of bounds)
3. Robot 2 seems slow
4. Changed the LiDAR offset to use linear scaling with fixed points on the sides of the field (robot 2 right LiDAR seems like it is seeing the floor)
5. Camera localisation is bugged somewhere

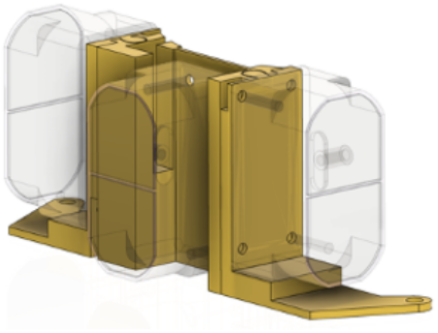
To do

1. Hardware
    a. 3D prints
        i. Plunger limiter (to prevent the piston from falling back into the robot in case the kick plate falls off during the match)
        ii. Shrouds (bottom back shroud, top back shroud, side shroud)
        iii. Side puzzle piece (add base to prevent tilting down towards the floor)
    b. Reassemble robot 1 once plunger stopper is done
    c. Robot 3: mirror set up, solder older another solenoid with bullet plugs (which was swapped out for robot 1) and test bottom plate
    d. Change the O rings
    e. Make other spares/cables
2. Fix camera localisation

<u>21/06/23</u>

Hardware:

1. 3D prints
    a. Plunger limiter and top back shroud prints work
    b. Bottom back shroud was made to replace the stand-offs at the back but still need to extend hole for the Teensy port
    c. Added base and screw were added to the side puzzle piece (fixes seeing floor problem), but still need to edit it to include the shroud for the wires



   d. Need to edit the back LiDAR mounts to include the shroud also
   e. Don't need the box to cover the LiDAR wires because I managed to fit all the coloured LiDAR wires into the space above the camera PCB
   f. Also extend the lightgate mount to reach the top of the middle plate so there's more padding and the LED won't be in contact with the ball
2. Robot 1
   a. Reassembled everything and tested LiDARs, GY and camera
   b. Bottom plate wasn't lighting up but it stopped shorting when the motor screws were removed (part of the copper around the motor holes is exposed) so I just replaced them with nylon screws
   c. Tried to test the motors but one wasn't spinning, realised the motor bullet plug fell off so we resoldered that and it works
   d. Resoldered the GY wire to be black
3. Robot 2
   a. Tried to put on the plunger limiter but the screw for the middle lidar mount was too long so the print couldn't fit, will wait until the LiDAR prints are ready then disassembled and replace everything at once

Software

1. Camera localisation
   a. Reverted to old version for clipping x-bound so it cuts to the goal post x-coordinates instead, and for corner case we cut the x-bound without cutting y-bound
   b. Included case for if the left and right angles are swapped (happens near edge when values are close, but shouldn't happen because the robot won't enter that area)
   c. Constrained front/back angles in case it was seeing the goal from behind the goalpost (also shouldn't happen), robot doesn't know which goal it is facing in the corner case but ignored for now because it shouldn't reach this case when running
2. Aiming
   a. Tried making the threshold for aiming closer to the left/right side of goal larger (falls into this case more often), but it doesn't work since angle is too big when robot is aiming from closer to centre (also it misses sometimes so it's too risky)
   b. Tried having only 2 aim zones but ineffective due to same reason as above
   c. Tried driving forward in centre which works well only if compass is not tilted so will see on the competition field
   d. So in the end using the left/right aiming points if the robot is beyond those points, otherwise aim to the centre (but if during the competition the opponent's goalie is permanently in the centre we can just make it always go to the nearest left/right aiming points to try and score)
3. Camera
   a. Changed the lookup table with pixel to angle conversion to be more precise, angle values are now only rounded when sending to Teensy
   b. Changed the switching goals distance calculation to calculate from the edges of the goal (take minimum distance) instead because it kept switching goals at the sides
4. When testing camera localisation I realised that deg2rad and rad2deg used floats and ints so changed them all to double

To do:

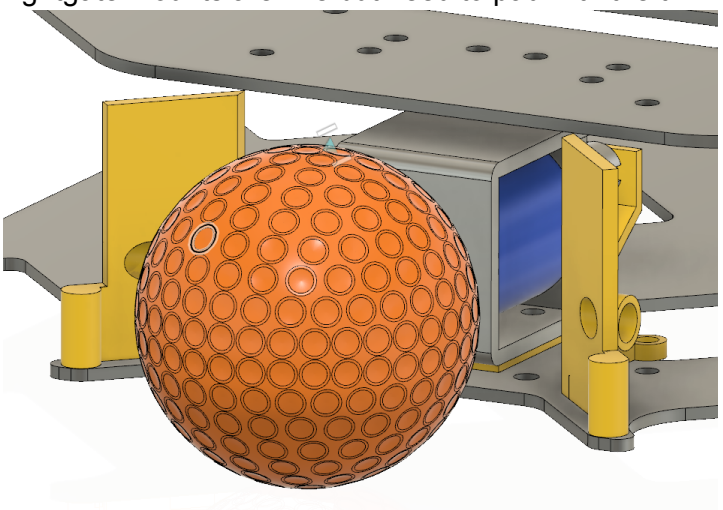
1. Hardware
   a. 3D prints (see above)

    b. Replace LiDAR mounts for sides and back on robot 1
    c. Rewire LiDAR wires and replace all the prints on robot 2
2. Software
    a. Compass corrected movement for goalie ball track
    b. Increase power of confidence of striker since camera localisation increases confidence

22/06/23

Hardware:

1. Robot 1
    a. 5V GND shorted while robot was running and the bottom plate turned off
    b. Disassembled bottom plate only but nothing was shorted
    c. Dismantled everything and put it back together bit by bit but still didn't short
2. Robot 2
    a. Organised the LiDAR wires but still waiting on the prints before assembling
3. Can consider placing a capacitor on 5V and GND (put one on input from battery and one on output for regulator) to smooth the voltage (may be why the lidars are unstable sometimes or why the bottom plate flickers)
4. 3D prints
    a. Lightgate mounts are fine but need to pad with the thin mounting tape before swapping



    b. Tried new stand-off back shroud but looks close to touching wheel so will edit
    c. Need to edit the LiDAR mounts shroud to extend towards the camera PCB (triangle) to cover all the wires

Software

1. Added compass correction for ball track
2. Changed some parts of the functions that weren't using constants
3. Added lightgate detection constants
4. Changed the order of boundary calculations so it calculates the camera position first, then uses that to clip the LiDAR bounds
    a. Made it so that the back LiDAR goal correction will still be added even if the left/right LiDARs are blocked and the robot doesn't know if it's between the goal posts
5. Camera code rejects angles close to 90/270 by calculating the error from 90, if the minimum error is smaller than certain value then the angles are rejected (to avoid the very inaccurate points when the angle lines are too close)
6. For striker back to goalie transition, the robot runs striker move to point until striker home point is reached because it moves too fast during the immediate transition to goalie (due to slowdown pushing it back) and the robot spazzes out
7. Realised wrong config file was being used (but LiDAR offsets were retaken today, just need to remember to use the correct one in the future)
8. May need to do different LiDAR mapping for both robots (or just remove it if not needed)

23/06/23

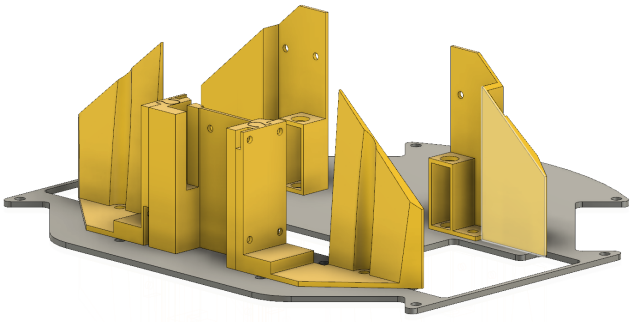
Hardware:

1. Robot 1
    a. Replaced the IR LED mounts and added thin tape all around it to pad it
    b. Covered the exposed red battery wires with wire shroud
2. Need to edit the LiDAR shroud to cut below the driver screw (in the way right now)

Software
1. Goalie
    a. Slowdown confirmed out case changed to consider both lower and upper bound, previously only considered lower bound
    b. Added goalie tilt away from angle (very small, ~5º max)
    c. For ball track changed the goalie position (angle) to be weighted between ball angle and the angle to the back of the goal (instead of back of the wall)
        i. We also tried to find the ideal goalie position in any case (the goalie blocks the entire goal)
            1. Cadence: https://www.desmos.com/calculator/wb4ggdmbe5
            2. Ashlee (mine is much more complicated because I tried to do it using geometry and offsetting the tangents)
                a. Simulation: https://www.desmos.com/calculator/jktbuayq4m
                b. C++ code: https://replit.com/join/htrrnzkrsj-acyt
            3. But there wasn't enough time to test and also looks like there'll be a lot of cases to consider (e.g. if the robot keeps getting pulled forward) so for now will not be using this
    d. Shifted the arc forward by 9cm if there's no striker
    e. Tried to reduce the angle error 'dead zone' (if error <= 3 then the robot doesn't ball track and runs pure rotation) but robot keeps oscillating so changed back
    f. Sometimes charges forward but not sure why
2. Robot 2 LiDARs are seeing larger than the length of the field (probably because of the walls)

<u>26/06/23</u>

Hardware:
1. Robot 2
    a. Swapped out all the LiDAR prints



    b. Realised the new 36V boosts have a taller capacitor and cannot fit above the Teensy 2.0 so edited boost mount to push boost to the right and not above the Teensy (for now I put the boost mount in to secure the boost and make sure it doesn't short anything and removed the Teensy 2.0)
    c. Swapped out the lightgate mounts
    d. Bottom back shroud works
2. Tried to replace the LiDAR mounts but got one stuck, will continue fixing tomorrow

To do:
1. Hardware
    a. Build robot 1 (LiDAR mount then reassemble)
    b. Change boost mount for robot 2 and test the kick
    c. Make cricut for bottom of robot to prevent shorting
    d. Make side shroud

<u>27/06/23</u>

Hardware:

1. Robot 1
    a. Swapped out the LiDAR prints and also the top back shroud because the previous one was warped
    b. Finally discovered why robot 1 was shorting all those random times
        i. When I reassembled the robot the bottom plate was shorting, so we disassembled it and I found out that the bottom plate only shorts when the GNDs of the two plates are connected
        ii. So the top plate GND was shorting to the bottom plate 5V, and after multimetering it was because the underside of the top plate around the motor mounting holes has exposed copper (top GND plane), and the top of the bottom plate also had exposed copper around the same area (bottom 5V plane) and the motor was shorting them both together
        iii. To fix this I put heat shrink around the motor mounting holes and used nylon screws, and also put nylon washers between the plates for the standoffs
    c. The left LiDAR print that didn't fit yesterday also fit today
2. Robot 2
    a. Replace GY wire with a black one
    b. Used the boost mount that causes the boost to stick out of the plate for now so that we can test the robot with the kick
3. Managed to cricut some plastic sheets but the software didn't let us cut arcs and also forgot to add holes for the TEMTs and LEDs

Software:

1. Tuned striker confidence power to 4 so the robot can catch itself with line detection
2. Goalie
    a. Added line detection if the robot is guaranteed to be past the goal posts (to prevent detecting penalty area lines)
    b. Can try to decrease the max speed point to cap the overall speed (instead of putting a cap in the main code), decreased to 0.8 but increased it back to 1.0 because the goalie looks like it might stop on the line
    c. For slowdown when the robot was moving leftwards it was never slowing down because the angle was < 0 but the condition was for 180 < angle < 360 so fixed that
    d. Changed arc to be more curvy (set the point such that at the goal post the robot would sit 11cm in front of the goal post), and made the y-value be 9cm in front of the line if the value outputted by the arc was less than that (basically the boundary is a hat)
    e. Changed line detection so it triggers when the back LiDAR is less than a certain value (cannot see it on the old field but should help in the new field)
    f. Changed return to home to use intermediate points to avoid going into penalty area (using x-coordinate to decide which intermediate point it should go to)
    g. Increased the speed cap and robot is much more responsive

To do:

1. Hardware
    a. Change boost mount
    b. Print circle marker
    c. Solder spare battery wire (get new clippy cable) and solenoid

<u>28/06/23</u>

Hardware:

1. Swapped out new boost mounts and they work well (still taped over the capacitor to be safe in case it falls out during the match)
2. Bottom plate of robot 2 looks like it's turning off sometimes so might heat shrink around the motor mounting holes if it gets worse
3. Redid the cricut (just took a picture of the CAD so it would cut the plate shape properly) but forgot the holes for the LEDs so cut those manually

Software:

1. Took out the kick condition (kick if the robot is far front enough) because it should already be covered in the other condition (enough speed) and was causing the robot to kick when it wasn't in front of the goal
2. Widened the arc (pushed the point in front of the goal post out by 1cm) because it looked like the robot was going to go into the penalty area
    a. Removed the intermediate point because not necessary and wastes time
3. Also we tried to do #define MID_X FIELD_X / 2 and when we printed it it was correct but when we used it in calculations it didn't work (discovered when we were trying to figure out why the tilting to face goal wasn't working, realised this was the problem) so changed it to use 910.0 and it works
4. Made the tilting near the goal asymmetric
5. Changed slowdown to not apply front slowdown in goalie move to point (prevent it from going out of bounds)
6. Added acceleration to move to point so it doesn't start running at max speed and tilts (accelerates from rest based on time, then decelerates based on distance from the point)
7. Added back goalie to striker transition
8. Added striker back to goalie transition if the ball was too far away (>30cm) to try and prevent robots from chasing/clashing into each other
9. Changed the striker back to goalie transition to pass through striker (robot moves back to striker home point first before the state switches to goalie) if the y-coordinate is greater than the striker home point
10. There were some problems with the striker back to goalie transition when the otherHeartbeat was dead but will test next session

<u>30/06/23</u>

Hardware:

1. Added side shrouds, the clip on ones are kind of fragile but usable until they break, standoff ones are more reliable

Software:

1. Removed striker to goalie transition if the ball is too far away
2. Striker back to goalie transition (for when the other robot has no heartbeat)
    a. Basically the state is always being set to goalie since the other robot is dead, but this can be overwritten so the goalie transitions to striker whenever the ball is stationary
    b. So now the striker will only transition back to goalie is the ball is removed or if the ball stops being stationary
    c. And last session we had two problems
        i. The robot should have immediately transitioned back to goalie once it touched the ball (because then the ball isn't stationary anymore)
        ii. When it did transition back when the ball is moved (not removed) the robot would speed back to the goalie home position and it looked like slowdown was causing it to move super quickly (which it shouldn't because we turned off the goalie front slowdown if it was doing move to point)
    d. After testing we realised
        i. The ball stationary condition was very loose so sometimes even if the robot has the ball as long as it moves slowly enough the robot still senses it as stationary (because it only ever compares it to the loop before)
        ii. It would speed back because the moment the ball isn't stationary it immediately switches to goalie (it only goes through striker if the ball is removed), but because the goalie noBall timeout is very long, for 500ms it still thinks it is seeing the ball and runs ball track which has the front slowdown applied, so it runs at a very high speed for a bit because of slowdown, then when noBall is over it runs the move to point
3. Then we realised that if the striker transitions to goalie while the robot is behind it (no heartbeat case) it will almost always score an own goal
    a. We tried to make it do ball orbit (always inwards) but realised it wouldn't work in the normal goalie position because it would keep going in the wrong direction/out of bounds
    b. In the end we put a cap on the speed when the ball is behind (can narrow down the ball is behind angles to be the back 90º but not important) so that the robot slows until it stops in front of the ball (essentially blocking the ball from the opponent)

4. Made it run striker if it's in the front ¾ of the field (1215 - 270) so that it won't run striker to goalie and just block the ball, and will still have a chance to score
5. Changed the bluetooth to not send if the ball is in the robot's ball capture zone so that if the goalie gets the ball and transitions the striker won't be chasing after it
6. Also the robots clashed in the case when the goalie transitions and the ball is removed, because both robots are trying to go to the same point, so changed the goalie point to be the mirror of the striker's and they don't clash as much