**Link and references:**

IR receiver datasheet: <https://www.vishay.com/docs/82479/tssp58038.pdf>

LJStand (lightweight champions 2018): <https://bitbucket.org/ljstand/>

Angular drive: <https://www.cs.cmu.edu/~pprk/physics.html>

Jing Heng’s github: <https://github.com/hcbozo>

Cup’18 posters: <https://photos.google.com/share/AF1QipPh5H2HzPVjt2wpaWPG-9Mnrah6Dkoe2Ghs8lTklUj96C2vK_ss1Fg4HTWEEaDdhg?key=dHA1TUNIdHFCR1ZnMnN1T1o4S0NZaGlhMzE0OHpn>

Cup’17 posters: <https://photos.google.com/share/AF1QipNzufjNPruF_a2l2uguBTrDln-8qcq0lzRuGnhFGGmJst7W1q2AmL1tlUa5VyGCiw?key=aVNHM0UtbTFEZFRUc1lISkV5OEpyN05NWEt2Q0hB>

**IR program**

* initIR()
  + Easy enough to understand, just initialize the IR sensors
  + Set them to INPUT mode
* readIR(int)
  + Type in an integer into the bracket, which denotes the number of times it repeats the for-loop in the function
  + Store the IR readings into an array; tempValues after digitalReadFast through another for-loop to cycle through all the IR sensors
    - Used a compound addition and a bitwise operator XOR ([https://www.arduino.cc/reference/en/language/structure/bitwise-operators/bitwisexor](https://www.arduino.cc/reference/en/language/structure/bitwise-operators/bitwisexor/))
    - Essentially, in layman terms, each digitalread returns 0 or 1 (LOW or HIGH) and after reading all the IR sensors, add everything by the number of times you decide is best

NOTE: (PLEASE read this) ok so essentially, the IR sensors are **extremely sensitive** and you can’t perfectly single out the angle after 1 loop cycle (under my circumstances, the reason was due to lag time etc;sensor readings will display, array = [ {1,1,1,1,0,0,0,0,1,0,1,1,0,0,1…} ] However, the reason why LJStand got it to work so well was that they had **JUST ONE** Teensy to process the IR sensors and they set it read the IR after 3333 microseconds (or 3 millis) Thus, assuming their loop time is like 150 microseconds, they would have compounded their readings by like 20 times (their ideal sensor readings, array = [ {20,18,13,10,10,8,6,6,5,4,3,3,3,0…} ] thus clearly, the angle can then be determined easily after assigning an angle to each individual IR sensor.

So TLDR, unless your looptime is as short as LJStand, don’t copy their code wholesale.

However, the problem that we faced was that we only had 1 Teensy to handle every single function of the robot (from movement to sensor readings etc) Thus identifying after the problem

~~wayyyy too late~~, our loop time turned out be 2~3 millis (extremely slow due to i2c and serial comms with other sensors), essentially meaning that after 1 loop, the condition to compute the IR data will just trigger immediately instead of waiting for the data to compound and add up to give more reliable data. Hence, the angle cannot be determined accurately and overall motor output is VERY jerky and nonfunctional.

There are 2 ways to solve this (but personally pls try and go with option 2 if there are enough resources, tho option 1 works fine for me?)

1. Force the IRring to read the data multiple times (i set mine to 15 and increase the time condition for it to compute the data by alot. I set a (38 x 833)us cycle, meaning that it will only compute the data after about 30 millis) Definitely not the best option as if the ball **flies at v fast speed, the robot may not be able to respond fast enough to the ball - tho tbh 30 millis should be sufficient)** You can probably can experiment with it more, but 35 cycles is the minimum no. which gives a good enuf set of data to determine the angle accurately. There will ofc be some cases where the data returns ‘no ball detected’ (disclaimer: probably is i didn’t clean up and refine the code completely and i spaghetti coded it somehow/ So, to whoever is reading this, pls experiment on how else we can refine this so that the robot can be more responsive and decrease the overall loop time)
2. OR HAHA LMAO GO FULL LJSTAND and just use another Teensy (or a powerful alternative microcontroller that has enuf digital pins to read >16 IR sensors and leave it to run the IR program separately) and only send data after the data is computed on the slave microcontroller over to main controller to determine movement.

* processIR()
  + Has 2 other functions inside, IRsortvalues() and IRcalculateanglestrength()
  + First, it multiplies each value in the array by 100 then divide it by number of times it read the sensors
  + Resets IR counter (no. of times it read the IR)
  + It then goes on to sort values then calculations
* IRsortvalues()
  + sort the TSOP values from greatest to least
  + also sort the TSOP indexes from greatest to least strength in indexes (or array positions)
* IRcalculateanglestrength()
  + Sum up all (strength) values in the array and sum up as 's'
    - If s=0, there is either no ball, or dodgy data due to spaghetti code
  + As a result, everytime s = 0, then add 1 to ‘loopcounter’
  + If loopcounter is more than 1 in the next loop, check again if the first index in array ‘sortvalues’ is > 0, then assign the angle = 0 (meaning that somehow the code fked up even tho the ball is in front), else assign the no ball angle
  + If s = 0, assign the (output) angle as the last angle (i wrote my program to do this constantly) to ensure that the motor is always moving in the direction where the ball was last seen (prevents jerky movements)
  + To calculate the angle:

for (int i = 0; i < 3; i++) {

angle\_kp[i] = indexes[i] \* sortedValues[i] \* 22.5 / (sortedValues[0] + sortedValues[1] + sortedValues[2]);

}

angle = angle\_kp[0] + angle\_kp[1] + angle\_kp[2];

Essentially, its a proportionality kind of thing between the IR ‘strength’ of the 3 strongest individual sensors and adding them all up to get your final angle

Program flow in main loop (just IR sensors):

1. initIR() (in setup part)
2. updateIRtime (start the timer in setup)
3. If (IRtimer >= *time set*)
4. ProcessIR
5. Obtain ball strength
6. Reset timer

* processballstrength()
  + Ok lmao i nvr actl tried and tested this function fully
  + Idea is that: when ball is far away, the values array = [20,20,0,0,0,0,0,0,0,0…] thus there are only 2 values that are non-zero (pls play around with the values and observe changes at certain distances)
  + Generally, the function returns ballstatus = 1 when the IR data tells me that the ball is **‘far away’**, else gives 0 (define and experiment for yourself what u deem as far away)
* Chaseball()
  + Lmao kek this was so ghetto
  + processballstrength()
  + If ball is in front
    - robotAngle = ballAngle (just charge straight forward)
    - RunningSpeed = maxSpeed
  + If ball is behind,
    - If ball is far away, robotAngle = ballAngle + 25
    - Else, RobotAngle = ballAngle \* 1.3 (idea is that u want the robot to move around the ball)
  + If ballAngle is somewhere in between, then manually tuned the multiplier to go around the ball such that it is in front
  + If ballAngle = 400 (basically the angle i assign it to when there is no ball detected), robot will return to base
* For out detection, rely solely on ultrasonic to move back onto the line accordingly; do on a case by case basis

Phew ok, now i'm gonna list the mistakes i have made for anyone who are trying out lightweight for the first time / just want to laugh at my programming

1. Play with the FULL IR ring more. There is so much UNTAPPED potential in there. Sure, you can argue that the compound eye is better (easier standard code that you can copy from Kaicong in his 2018 Canada code) but sometimes, after observing all the teams there. It was clear that the IR ring is superior in its detection range. Even when the (shit) refs lift up the ball, the robot can still detect it. Only situation where you can’t detect it is when you hide the ball behind the ball. TLDR, the IR ring is very sensitive
2. Speaking more on that, pls clean up my code more. As you can probably see if you have past coding experience, i spaghetti coded many parts of the program. I sincerely apologize and pls dont kill me
3. For lightweight, **JUST STAY IN.** tbh, you stay in, u automatically secure top 2 finish (nanyang sensors are shit and raffles just went a weee bit too fast at times for this year, admittedly their line detection program wasn’t that much better than ny)
4. Thus, fix your light sensors (hardware side)
   1. Experiment with kc’s line detection program and using line angle to move back in (pls ask him he’s a god) This could very well eliminate the need for ultras and thus, saves weight
   2. Or sure stick with ultras to help you move back in, but pls ensure that u have 4 ultras for all sides (my robot kept going out at the front part as we couldn’t stay in at that side without a reliable source to determine the direction to move back into the field if it goes out)
5. Have a function to stop on the line AND is able to move back in to track the ball; cos it's just very inefficient for robot to just keep jerking on the line/return to base (didn’t have time to code this part, go analyse the kids’ 2018 Canada code or ask Kaicong for help)
6. Initially thought that Serial communication was a bad idea but turns out it was ok? Relative to I2c, it is slightly faster, tho it is much more easier to lose synchronization between Teensy and nano (due to diff clock speeds) and to debug it so yeah, pick your poison
7. Ensure connections are solid, alot of time was wasted on securing loose connections between motors, sensors etc / debugging software before realising it was a hardware issue
8. Communication between everyone and making sure all knows wtf is going on at all times
9. Try to cross-roles and generalise more (even tho it might be your first and last year doing cup); this ensures that progress is never stalled if 1 person isn’t around and makes your journey in cup more fulfilling at the same time
10. Haha yes camera. The good thing about OpenMV is that they have a very supportive development team, thus **DON'T BE AFRAID TO ASK FOR HELP ON THEIR FORUMS**
11. And no i didn’t get the camera to work **so don’t bother copying my camera code**. Pretty sure it is just my retardness in wiring/coding as we got the communication to work in the Dec hols but it just doesn’t work when i tried again nearing the final lap
12. If yall can get the camera to work + can stay in, u win
    1. All the competitors there just blindly whack for the goal once they got possession of the ball, sooooooooooooooooooooooooooo many wasted opportunities

Kek, thats it. Beanstalk signing off :(