# Design Logic:

In this problem, I started from centralizing code part where it will be used delicately. I have extracted function for start the competition, function for start the conveyor (Which sends out a service call asking conveyor power to be 100%), and function for stopping the conveyor (Which sends out a service call asking conveyor power to be 0%) once these function are all established, I can easily control the conveyor start or stop by just calling stopConveyor(n) and startConveyor(n). Note that since start service call needs node handle, and I ame technically using node handle from callback. Therefore by using ros::NodeHandle & passedNodeHandle so that the nodeHandle can be shared.

In the main body of the code, I first start by calling the competition to start. After that, the program start by moving the conveyor belt, and start camera detection. The program first detect by checking the list length of the logic camera model. If I have a model size larger than 1, I can say there is a shipping box in range, other wise, the shipping box is not in range. Once a box in range, another loop is started so that the camera can focus on monitoring the shipping box z position. Since at this point, there are no computation limitation, therefore, I ask the scan frequency to be as high as possible. I did offset the stop trigger a bit earlier so that the shipping box can pause under the camera accurately. Once the box is stop, I wait 5 seconds as asked and restart the belt. If this is not the first shipping box, I have the drone to be called during which the belt is stopping to pick up the extra box at the loading station.

# Observation

During testing and debug, I noticed that the belt does not stop in a realtime manner. When I start the stop belt function callback, the belt still goes forward for a bit. After offsetting the belt not stopping on time issue by setting the stop belt trigger negative 0.1 a head, I can achieve the shipping box to be stopped between +0.1~+0.2 at positive Z position.

What’s more, I also noticed that the shipping box is spaced differently. There are not uniformly distance between the first and second shipping box vs. the second and third shipping box. I think this issue will be addressable with an additional sensor.

Final observation from CWRU’s team coding I realized instead of spam sending the message using a while loop, another way to call competition start while I am not sure if the competition is ready is to us serviceClient.waitForExistence()