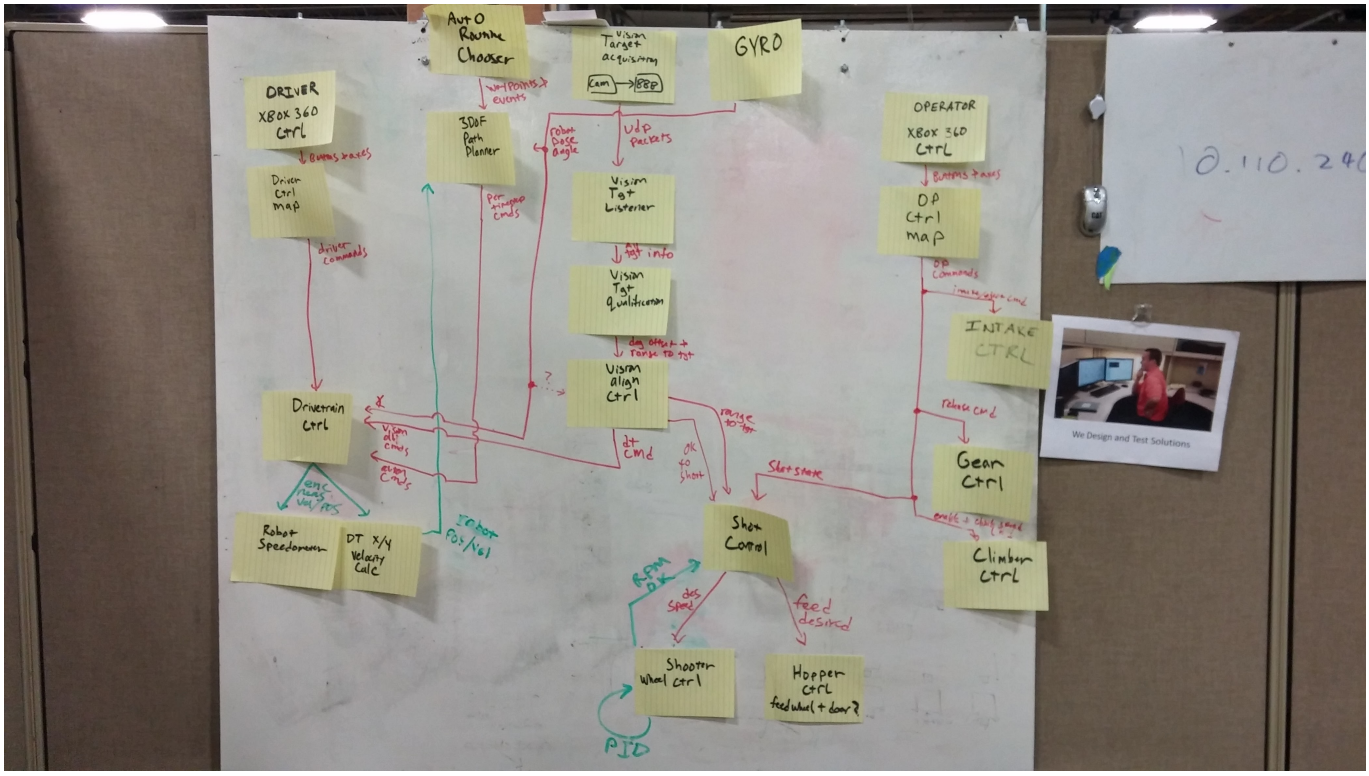
Here's a basic description of what we accomplished as a team this year.

Design Process

- Started with high-level component description of what was going to be on the robot (~week 1.5)
- Broke components into smaller chunks (Ex: vision processing became ~4 chunks)
- Write each chunk on a postit note
- Organize postit's on a whiteboard, draw lines between them to show the flow of information
  - Postit's roughly correspond to java classes
  - lines indicate what inputs and outputs each class must support



Development Process

- Git used as the version control tool
  - Track and coordinate all software changes through the team
  - Tags used to indicate tested software
- Team used github's Issues list featurs to keep track of work
  - What needs to be done
  - What is in process
  - What has already been done
- Code reviews done
  - Multiple people looking at code = more bugs caught
- Test electrical board used to test software before putting it on the robot

Autonomous

- Modes Available
  - Primary:
  - Center-peg gear place
    - Can drive left/right/backward afterward
  - Go to hopper and shoot
    - 2x for red/blue sides of the field
  - Others:
  - Just shoot (no motion)
  - Use vision to align, then shoot
  - Cross baseline
- Path Planner library written
  - Converts a set of waypoints into a sequence of wheel velocities
  - By running our motors at the calculated velocities, we traverse the path defined by the waypoints
  - Advantages:
  - Easy to define a complex set of motions
  - (Theoretically) minimizes guesswork when defining a new autonomous routine
    - You want to go three feet forward and two to the side? just punch {3.0, -2.0} into the code!
- Autonomous Sequencer class
  - Auto actions are in their own class
  - Each action conforms to a standard template
  - We have a sequencer class that runs a set of actions in a defined order
  - Ex: Drive forward, place gear, drive backward
  - Actions are reusable among many autonomous routines
  - Ex: the "Shoot" action is used in a couple different routines (both hopper modes, all "just shoot" modes)

Vision Alignment

- Divided into three parts
  - Target Identification
  - Where do we see blobs of green?
  - Used openCV & Python to find contours around the green blob
    - "Contour" = outline
    - Has info about the width, height, area, and x/y location of the blob
  - Target Qualification
  - Which blobs correspond to the high-efficiency boiler target?
  - Uses a "Heuristic" to determine which pair of blobs looks the most like the target
    - Same width, aligned in the horizontal direction, certain size, etc
  - Vision Alignment

    - Given where the target is at, how do we align the robot to it?
    - Use the target data to determine where we should move to
    - Used gyro & encoders to determine if we've moved far enough yet. (double-checked with vision processing)
        - "Vision says we need to move three degrees left! Ok! Let's start turning and watch the gyro. Turning....Now the gyro says we've moved three degrees. Ok, stop and wait for vision to confirm we're aligned"
        - PID algorithm used - Gyro is feedback device, vision used to generate the setpoint.
    - Rotation only (for now)
- Microsoft Lifecam and Beaglebone Black do the Identification
- Qualification & Alignment are done on the BBB