Zeppelin programming plan

# RoboRio programming (Java)

* Driving
  + Teleop drive
    - pick a driving mode (tank drive, Xbox controller drive, ?)
    - Finish this part first, to give drivers as much practice time as possible
  + Autonomous drive
    - Tank drive mode
    - Get the next action to do from the Raspberry Pi
      * SetSpeed LeftWheel RightWheel (that is, act like we’re using the joystick positions for tank drive)
      * Stop (put on the brakes)
      * Shoot
      * Place gear
    - Do what the action says.
* Shooter
  + Command: shoot!
  + Subsystem: Set flinger speed
  + Subsystem: Open ball release servo
  + Question: How do we know when we’re out of balls?
* Gear placer
  + Is there anything to do in the new gear placing design?
* Climber
  + Command: Climb
  + Subsystem: Set climber motor speed
  + Anything else?

# Raspberry Pi programming (Python/C++)

* + Autonomous service (Python)
    - A service that responds to requests from the roboRio
    - Gets requests and send responses over USB connection
  + Read/write data from USB connection (Python)
    - roboRio will send a “start autonomsous” command to turn on all the vision/gyro/etc.
    - roboRio will send a “stop autonomous” to turn this stuff off
    - When autonomous is on, RasPi will send commands to roboRio
      * SetSpeed leftWheel rightWheel
      * Stop
      * Shoot
      * Place gear
    - Use JSON for encoding data
  + Find vision target (C++)
    - Uses OpenCV image processing
    - Expose as a function or wrapper that can be called from Python
    - Find the gear target in the image
    - Find the ball target in the image
    - Takes input: which target to find
    - Returns a result that tells:
      * Width and height of the whole image (camera resolution)
      * Which target it found (or none)
      * Rectangle containing the target (top left, top right, bottom right, bottom left)
  + Figure out target location (Python)
    - Input: Result from “Find vision target”
    - Response tells the size of the image (pixels), and gives the location of the target within that image as a rectangle (a list of 4 x/y coordinates, one for each corner)
    - From the response, figure out the direction and distance to the target.
    - Returns direction (in RADIANS, with positive being to the right and negative to the left) and distance (in meters, with positive being forwards and negative being backwards)
  + Make autonomous drive command (Python)
    - Given a target location and target type, create the next thing the robot should do, at a time scale of a few tens of milliseconds. (Whatever the time scale for the robot is.)
    - Do we turn or move or stop?
    - Returns a command.