Previous Meeting:

- Build simulink k-controller, using Jie's LQR code.
- Controller should collect current position and desired position, and output required angular velocities.
- Controller should feature 'K' and 'K' dependents as ROSparams.
  - 'K' can be changed directly, or can be computed if 'K'' dependents are updated.
- Controller should abstract as much as possible from user.
  - Assume the user is ME-145 student.
- Controller dir should be cleaned up, annotated, and callbacks should be investigated.

Saturday Chat:

- Goodgle drive slx file (suspected of being the right file) was generated into a rosnode, to ensure working state. Synthetic data was published, ensuring working state.
- Updates to Jie's work were noted:
  - Trimmed dir by removing redundant, or unnecessary files.
  - Callback structure was kept, serving as an easy location for users to initialize rosparams. Users assign values to 'K' and/or 'K' dependents, values are pushed to parameter server.
  - For new (current) rosnode workflow, see attached drawing.
- Codegen errors propagated from redesigned controller. Primarily from unsupported functions:
  - mat2str, str2mat

Today:

- Walk through correct simulink file.
- Define variables (l, L, I, V, theta)
- Decide on controller architecture
  - How much do we want to distribute this? Separate nodes? Separate machines?
  - How much of this distributed controller will be exposed to users?
  - How costly is the 'k' calculation upon updates?
  - How much overhead is on current simulink codegen errors?
- Design with following in mind (not currently in order of importance):
  - Difficulty of use
  - Runtime
  - Development overhead (not that I mind coding, just because I know were behind)
  - Abstract as much work form user as possible; update 'K', or compute 'K' based on other updates.