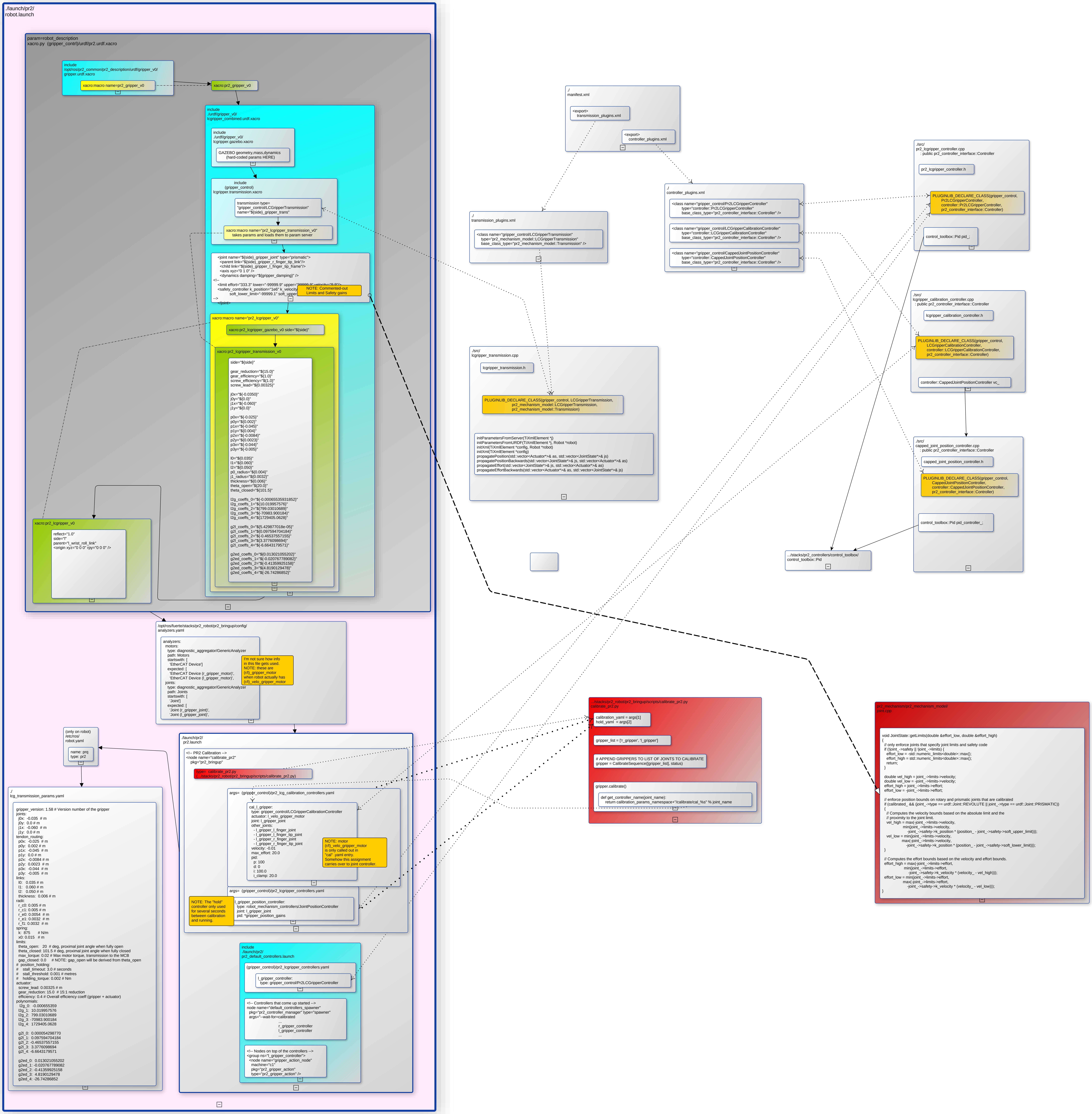

./launch/pr2/
robot.launch

param=robot_description
xacro.py (gripper_contrl)/urdf/pr2.urdf.xacro

include
/opt/ros/pr2_common/pr2_description/urdf/gripper_v0/
gripper.urdf.xacro

xacro:macro name=pr2_gripper_v0

xacro:pr2_gripper_v0

include
./urdf/gripper_v0/
lcgripper_combined.urdf.xacro

include
./urdf/gripper_v0/
lcgripper.gazebo.xacro

GAZEBO geometry,mass,dynamics
(hard-coded params HERE)

include
(gripper_control)
lcgripper.transmission.xacro

```
transmission type=
"gripper_control/LCGripperTransmission"
name="${side}_gripper_trans"
```

xacro:macro name="pr2_lcgripper_transmission_v0"
takes params and loads them to param server

```
<joint name="${side}_gripper_joint" type="prismatic">
  <parent link="${side}_gripper_r_finger_tip_link"/>
  <child link="${side}_gripper_l_finger_tip_frame"/>
  <axis xyz="0 1 0" />
  <dynamics damping="${gripper_damping}" />
<!--
  <limit effort="333.3" lower="-99999.9" upper="99999.9" velocity="9.9"/>
  <safety_controller k_position="1e6" k_velocity
        soft_lower_limit="-99999.1" soft_upper
-->
  </joint>
```

NOTE: Commented-out
Limits and Safety gains

xacro:macro name="pr2_lcgripper_v0"

xacro:pr2_lcgripper_gazebo_v0 side="${side}"

xacro:pr2_lcgripper_transmission_v0

```
side="${side}"

gear_reduction="${15.0}"
gear_efficiency="${1.0}"
screw_efficiency="${1.0}"
screw_lead="${0.00325}"

j0x="${-0.0350}"
j0y="${0.0}"
j1x="${-0.060}"
j1y="${0.0}"

p0x="${-0.025}"
p0y="${0.002}"
p1x="${-0.045}"
p1y="${0.004}"
p2x="${-0.0084}"
p2y="${0.0023}"
p3x="${-0.044}"
p3y="${-0.005}"

l0="${0.035}"
l1="${0.060}"
l2="${0.050}"
p0_radius="${0.004}"
j1_radius="${0.0032}"
thickness="${0.006}"
theta_open="${20.0}"
theta_closed="${101.5}"

l2g_coeffs_0="${-0.00065535931852}"
l2g_coeffs_1="${10.019957576}"
l2g_coeffs_2="${799.03010689}"
l2g_coeffs_3="${-70983.900184}"
l2g_coeffs_4="${1729405.0628}"

g2l_coeffs_0="${5.429877018e-05}"
g2l_coeffs_1="${0.097594704184}"
g2l_coeffs_2="${-0.46537557155}"
g2l_coeffs_3="${3.3776098694}"
g2l_coeffs_4="${-6.6643179571}"

g2ed_coeffs_0="${0.013021055202}"
g2ed_coeffs_1="${-0.020767789082}"
g2ed_coeffs_2="${-0.41359925158}"
g2ed_coeffs_3="${4.8190129478}"
g2ed_coeffs_4="${-26.74286852}"
```

xacro:pr2_lcgripper_v0

```
reflect="1.0"
side="l"
parent="l_wrist_roll_link"
<origin xyz="0 0 0" rpy="0 0 0" />
```

/opt/ros/fuerte/stacks/pr2_robot/pr2_bringup/config/
analyzers.yaml

```
analyzers:
  motors:
    type: diagnostic_aggregator/GenericAnalyzer
    path: Motors
    startswith: [
      'EtherCAT Device']
    expected: [
      'EtherCAT Device (r_gripper_motor)',
      'EtherCAT Device (l_gripper_motor)',
  joints:
    type: diagnostic_aggregator/GenericAnalyzer
    path: Joints
    startswith: [
      'Joint']
    expected: [
      'Joint (r_gripper_joint)',
      'Joint (l_gripper_joint)',
```

I'm not sure how info
in this file gets used.
NOTE: these are
(r/l)_gripper_motor
when robot actually has
(r/l)_velo_gripper_motor

(only on robot)
/etc/ros/
robot.yaml

```
name: prq
type: pr2
```

./
lcg_transmission_params.yaml

```
gripper_version: 1.58 # Version number of the gripper
joints:
  j0x:  -0.035  # m
  j0y:  0.0 # m
  j1x:  -0.060  # m
  j1y:  0.0 # m
tendon_routing:
  p0x:  -0.025  # m
  p0y:  0.002 # m
  p1x:  -0.045  # m
  p1y:  0.0 # m
  p2x:  -0.0084 # m
  p2y:  0.0023  # m
  p3x:  -0.044  # m
  p3y:  -0.005  # m
links:
  l0:   0.035 # m
  l1:   0.060 # m
  l2:   0.050 # m
  thickness:  0.006 # m
radii:
  r_c0: 0.005 # m
  r_c1: 0.005 # m
  r_e0: 0.0054  # m
  r_e1: 0.0032  # m
  r_f1: 0.0032  # m
spring:
  k:  875       # N/m
  x0: 0.015   # m
limits:
  theta_open:   20  # deg, proximal joint angle when fully open
  theta_closed: 101.5 # deg, proximal joint angle when fully closed
  max_torque: 0.02 # Max motor torque, transmission to the MCB
  gap_closed: 0.0     # NOTE: gap_open will be derived from theta_open
#  position_holding:
#    stall_timeout: 3.0 # seconds
#    stall_threshold: 0.001 # metres
#    holding_torque: 0.002 # Nm
actuator:
  screw_lead: 0.00325 # m
  gear_reduction: 15.0  # 15:1 reduction
  efficiency: 0.4 # Overall efficiency coeff (gripper + actuator)
polynomials:
   l2g_0:  -0.000655359
  l2g_1:  10.019957576
  l2g_2:  799.03010689
  l2g_3: -70983.900184
  l2g_4:  1729405.0628

  g2l_0:  0.000054298770
  g2l_1:  0.097594704184
  g2l_2: -0.46537557155
  g2l_3:  3.3776098694
  g2l_4: -6.6643179571

  g2ed_0:  0.013021055202
  g2ed_1: -0.020767789082
  g2ed_2: -0.41359925158
  g2ed_3:  4.8190129478
  g2ed_4: -26.74286852
```

./launch/pr2/
pr2.launch

```
<!-- PR2 Calibration -->
<node name="calibrate_pr2"
   pkg="pr2_bringup"
```

type= calibrate_pr2.py
(.../stacks/pr2_robot/pr2_bringup/scripts/calibrate_pr2.py)

args= (gripper_control)/pr2_lcg_calibration_controllers.yaml

```
cal_l_gripper:
  type: gripper_control/LCGripperCalibrationController
  actuator: l_velo_gripper_motor
  joint: l_gripper_joint
  other_joints:
    - l_gripper_l_finger_joint
    - l_gripper_l_finger_tip_joint
    - l_gripper_r_finger_joint
    - l_gripper_r_finger_tip_joint
  velocity: -0.01
  max_effort: 20.0
  pid:
    p: 100
    d: 0
    i: 100.0
    i_clamp: 20.0
```

NOTE: motor
(r/l)_velo_gripper_motor
is only called out in
"cal" .yaml entry.
Somehow this assignment
carries over to joint controller.

args= (gripper_control)/pr2_lcgripper_controllers.yaml

```
l_gripper_position_controller:
  type: robot_mechanism_controllers/JointPositionController
  joint: l_gripper_joint
  pid: *gripper_position_gains
```

NOTE: The "hold"
controller only used
for several seconds
between calibration
and running.

include
./launch/pr2/
pr2_default_controllers.launch

(gripper_control)/pr2_lcgripper_controllers.yaml

```
l_gripper_controller:
  type: gripper_control/Pr2LCGripperController
```

```
<!-- Controllers that come up started -->
node name="default_controllers_spawner"
  pkg="pr2_controller_manager" type="spawner"
  args="--wait-for=calibrated
                    ...
                    r_gripper_controller
                    l_gripper_controller
                    ...
```

```
<!-- Nodes on top of the controllers -->
<group ns="l_gripper_controller">
  <node name="gripper_action_node"
    machine="c1"
    pkg="pr2_gripper_action"
    type="pr2_gripper_action" />
```

./
manifest.xml

```
<export>
   transmission_plugins.xml
```

```
<export>
   controller_plugins.xml
```

./
transmission_plugins.xml

```
<class name="gripper_control/LCGripperTransmission"
   type="pr2_mechanism_model::LCGripperTransmission"
   base_class_type="pr2_mechanism_model::Transmission" />
```

./src/
lcgripper_transmission.cpp

lcgripper_transmission.h

```
PLUGINLIB_DECLARE_CLASS(gripper_control, LCGripperTransmission,
                        pr2_mechanism_model::LCGripperTransmission,
                        pr2_mechanism_model::Transmission)
```

```
initParametersFromServer(TiXmlElement *j)
initParametersFromURDF(TiXmlElement *j, Robot *robot)
initXml(TiXmlElement *config, Robot *robot)
initXml(TiXmlElement *config)
propagatePosition(std::vector<Actuator*>& as, std::vector<JointState*>& js)
propagatePositionBackwards(std::vector<JointState*>& js, std::vector<Actuator*>& as)
propagateEffort(std::vector<JointState*>& js, std::vector<Actuator*>& as)
propagateEffortBackwards(std::vector<Actuator*>& as, std::vector<JointState*>& js)
```

./
controller_plugins.xml

```
<class name="gripper_control/Pr2LCGripperController"
      type="controller::Pr2LCGripperController"
      base_class_type="pr2_controller_interface::Controller" />
```

```
<class name="gripper_control/LCGripperCalibrationController"
      type="controller::LCGripperCalibrationController"
      base_class_type="pr2_controller_interface::Controller" />
```

```
<class name="gripper_control/CappedJointPositionController"
      type="controller::CappedJointPositionController"
      base_class_type="pr2_controller_interface::Controller" />
```

./src/
pr2_lcgripper_controller.cpp
  : public pr2_controller_interface::Controller

pr2_lcgripper_controller.h

```
PLUGINLIB_DECLARE_CLASS(gripper_control,
      Pr2LCGripperController,
      controller::Pr2LCGripperController,
      pr2_controller_interface::Controller)
```

```
control_toolbox::Pid pid_;
```

./src/
lcgripper_calibration_controller.cpp
  : public pr2_controller_interface::Controller

lcgripper_calibration_controller.h

```
PLUGINLIB_DECLARE_CLASS(gripper_control,
      LCGripperCalibrationController,
      controller::LCGripperCalibrationController,
      pr2_controller_interface::Controller)
```

```
controller::CappedJointPositionController vc_
```

./src/
capped_joint_position_controller.cpp
  : public pr2_controller_interface::Controller

capped_joint_position_controller.h

```
PLUGINLIB_DECLARE_CLASS(gripper_control,
      CappedJointPositionController,
      controller::CappedJointPositionController,
      pr2_controller_interface::Controller)
```

```
control_toolbox::Pid pid_controller_;
```

.../stacks/pr2_controllers/control_toolbox/
control_toolbox::Pid

.../stacks/pr2_robot/pr2_bringup/scripts/calibrate_pr2.py
calibrate_pr2.py

```
calibration_yaml = args[1]
hold_yaml  = args[2]
```

```
gripper_list = ['r_gripper', 'l_gripper']
```

```
# APPEND GRIPPERS TO LIST OF JOINTS TO CALIBRATE
gripper = CalibrateSequence([gripper_list], status)
```

```
gripper.calibrate()
```

```
def get_controller_name(joint_name):
   return calibration_params_namespace+"/calibrate/cal_%s" % joint_name
```

pr2_mechanism/pr2_mechanism_model/
joint.cpp

```
void JointState::getLimits(double &effort_low, double &effort_high)
{
  // only enforce joints that specify joint limits and safety code
  if (!joint_->safety || !joint_->limits) {
    effort_low = -std::numeric_limits<double>::max();
    effort_high = std::numeric_limits<double>::max();
    return;
  }

  double vel_high = joint_->limits->velocity;
  double vel_low = -joint_->limits->velocity;
  effort_high = joint_->limits->effort;
  effort_low = -joint_->limits->effort;

  // enforce position bounds on rotary and prismatic joints that are calibrated
  if (calibrated_ && (joint_->type == urdf::Joint::REVOLUTE || joint_->type == urdf::Joint::PRISMATIC))
  {
    // Computes the velocity bounds based on the absolute limit and the
    // proximity to the joint limit.
    vel_high = max(-joint_->limits->velocity,
                   min(joint_->limits->velocity,
                       -joint_->safety->k_position * (position_ - joint_->safety->soft_upper_limit)));
    vel_low = min(joint_->limits->velocity,
                  max(-joint_->limits->velocity,
                      -joint_->safety->k_position * (position_ - joint_->safety->soft_lower_limit)));
  }

  // Computes the effort bounds based on the velocity and effort bounds.
  effort_high = max(-joint_->limits->effort,
                    min(joint_->limits->effort,
                        -joint_->safety->k_velocity * (velocity_ - vel_high)));
  effort_low = min(joint_->limits->effort,
                   max(-joint_->limits->effort,
                       -joint_->safety->k_velocity * (velocity_ - vel_low)));
}
```