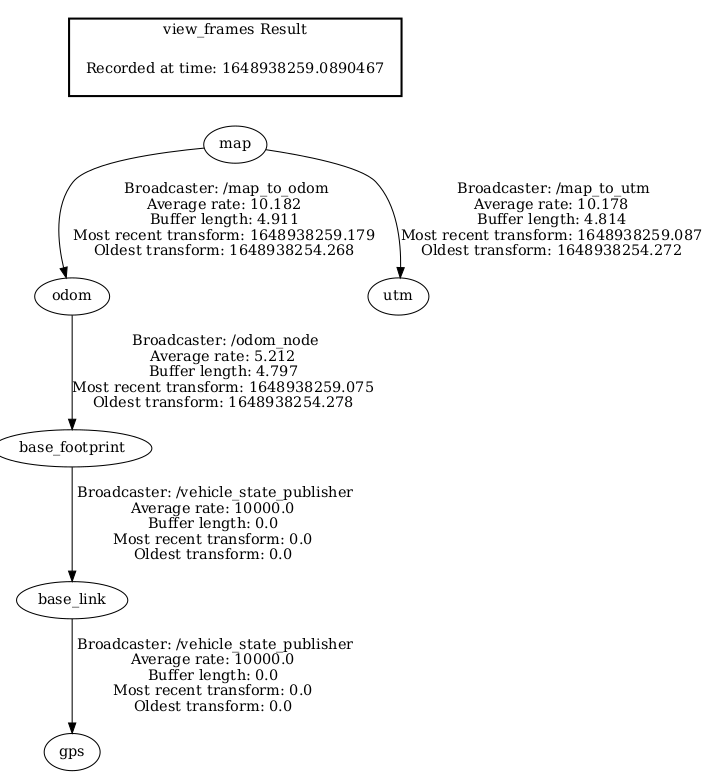
* Copy of the launch file from ROS Ag and work to get each element to load without error
* Go back and piece together your startup commands and get each of them working 1-by-1
* One of the differences is the TTGO board is controlling steering and speed. When the switch is set to be in “ROS Mode”, then it will need to listen to cmd\_vel

| Old fashioned way of starting things:  $ bash ros\_start.sh (gui) - step 1   | roslaunch teensy\_launch.launch  roslaunch nmea\_serial\_driver.launch | | --- |   $ python serial\_to\_ros.py (gui) - step 1 # takes output from IMU nano and publishes it  $ roslaunch tractor\_teleop drive.launch (gui) - step 2  $ roslaunch nmea\_serial\_driver.launch (gui) - step 3  $ roslaunch nmea\_serial\_driver.launch  /home/ubuntu/catkin\_ws/src/nmea\_navsat\_driver/launch  $ rosrun beginner\_tutorials gps\_odom.py (gui) - step 4  $ roslaunch lawn\_tractor\_sim lawn\_tractor.launch (gui) - step 5 |
| --- | --- |

| Previous way of starting things:   | roslaunch teensy\_launch.launch  *# used to set parameters and start rosserial\_python to get data from Teensy boards to ROS*  *publish*  *# steering angle, speed and control the steering motor and transmission control motor*   | <rosparam param="steering">[-14000.0, 13500.0, 0.0, 60.0, 3.0]</rosparam>  <rosparam param="speed">[1000.0, 1290.0, 1600.0, 135.0, -100.0, 70.0, 5.0, 0.0, 20.0]</rosparam>  <node ns="transmission\_teensy" name="transmission\_cntrl" pkg="rosserial\_python" type="serial\_node.py" args="/dev/transmission\_control" output="screen" />  <node ns="left\_speed\_teensy" name="left\_speed" pkg="rosserial\_python" type="serial\_node.py" args="/dev/odom\_left" output="screen" />  <node ns="front\_angle\_teensy" name="front\_angle" pkg="rosserial\_python" type="serial\_node.py" args="/dev/front\_angle" output="screen" />  <node ns="steer\_motor\_teensy" name="steer\_motor" pkg="rosserial\_python" type="serial\_node.py" args="/dev/steer\_motor" output="screen" /> | | --- |   roslaunch nmea\_serial\_driver.launch  *# used to start* nmea\_navsat\_driver   | <arg name="port" default="/dev/ttyUSB0" />  <arg name="baud" default="4800" />  <arg name="frame\_id" default="gps" />  <arg name="use\_GNSS\_time" default="False" />  <arg name="time\_ref\_source" default="gps" />  <arg name="useRMC" default="False" />  <node name="nmea\_serial\_driver\_node" pkg="nmea\_navsat\_driver" type="nmea\_serial\_driver" output="screen">  <param name="port" value="$(arg port)"/>  <param name="baud" value="$(arg baud)" />  <param name="frame\_id" value="$(arg frame\_id)" />  <param name="use\_GNSS\_time" value="$(arg use\_GNSS\_time)" />  <param name="time\_ref\_source" value="$(arg time\_ref\_source)" />  <param name="useRMC" value="$(arg useRMC)" />  </node> | | --- | | | --- | --- | --- |   $ python serial\_to\_ros.py # takes output from IMU nano and produces ros topics from a serial data stream  $ roslaunch tractor\_teleop drive.launch   | <launch>  <!-- Joystick Input Node -->  <include file="$(find tractor\_teleop)/launch/drive\_teleop.launch"/>  <!-- Multiplex Motor Commands Node -->  <include file="$(find tractor\_teleop)/launch/cmd\_vel\_mux.launch"/> {this let the joystick take over if needed}  </launch> | | --- |   $ rosrun beginner\_tutorials gps\_odom.py  *# Subscribes to* heading and fix and then publishes odom, plus broadcasts a transform between odom and base\_footprint   | self.heading\_sub = rospy.Subscriber("heading", QuaternionStamped, self.heading\_callback)  self.fix\_sub = rospy.Subscriber("fix", NavSatFix, self.gps\_callback)  self.odom\_pub = rospy.Publisher("odom", Odometry, queue\_size=1)  odom\_broadcaster.sendTransform((\_xg, \_yg, 0.0), self.odom\_quat, rospy.Time.now(), "base\_footprint", "odom") | | --- |   $ roslaunch lawn\_tractor\_sim lawn\_tractor.launch  *# Set the URDF, pkg="robot\_state\_publisher", node pkg="mbf\_costmap\_nav", node pkg="lawn\_tractor\_navigation", pkg="map\_server" and fake localizations:*  *<node pkg="tf" type="static\_transform\_publisher" name="map\_to\_odom" args="0.0 0.0 0.0 0 0 0.0 map odom 100"/>*  *<node pkg="tf" type="static\_transform\_publisher" name="map\_to\_utm" args="0.0 0.0 0.0 0 0 0.0 map utm 100"/>* |
| --- | --- | --- | --- | --- | --- |

I think you want a tree similar to below



So I’m going to:

* $ roslaunch /home/tractor/catkin\_ws/src/nmea\_navsat\_driver/launch/nmea\_serial\_driver.launch
* $ cd /home/tractor/catkin\_ws/src/beginner\_tutorials/scripts
* $ chmod +x gps\_odom.py
* $ rosrun beginner\_tutorials gps\_odom.py
* $ rosrun /home/tractor/catkin\_ws/src/beginner\_tutorials/scripts/gps\_odom.py
* $ roslaunch lawn\_tractor\_sim lawn\_tractor.launch (execute this a piece at a time and work through the issues)
* $ source /home/tractor//catkin/devel/setup.bash

$ python3

>>> help("modules")

$ geonav\_transform not listed

This url helped with the gps\_odom.py tuple issue

<http://www.tcrobots.org/Lawntractor/slack2html/html/lawntractor2020.html>