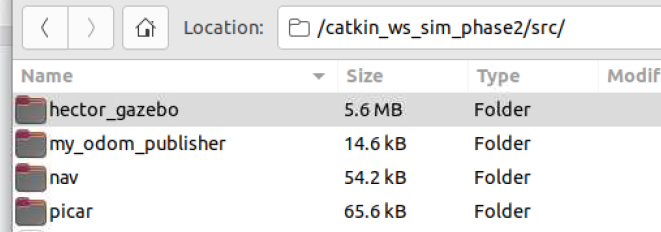
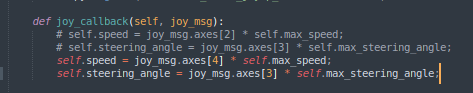
Notes on Using Jeff’s simulator

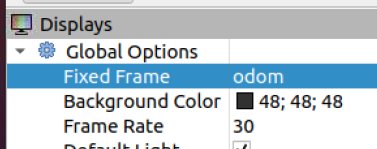
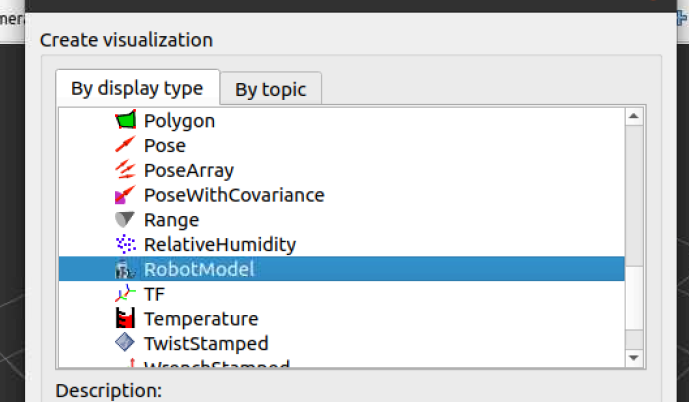
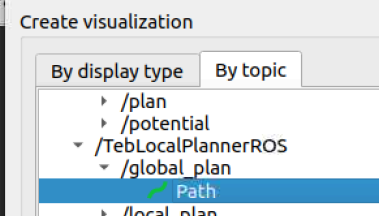
* Build a clean Ubuntu 20 laptop and install ROS using “ros\_desktop\_full install” ([link](https://docs.google.com/document/d/1iQgvms-inNsP_-TtOGxOjQipT-mIIxHGuhmcW948AYo/edit?usp=sharing))
* Go to <https://drive.google.com/drive/folders/1Lsu2ZN3aeGRUzJJqlfFOejG5v3KxXSHT>
* Download catkin\_ws\_sim\_phase2, extract and move to home directory
* Original note: Go into the resulting directory and copy the catkin\_ws\_sim\_phase2 directory to your home directory. This will be in addition (i.e. Alongside of) your existing catkin\_ws.
* 11/12/22 - I’m going to copy these folders into my catkin\_ws so I don’t have 2.
* 
* This is a list of dependencies to load. Load all of these before you do a catkin\_make.
  + $ sudo apt-get install ros-noetic-ackermann-msgs
  + $ sudo apt-get install ros-noetic-gazebo-ros-pkgs # already newest
  + $ sudo apt-get install ros-noetic-gazebo-ros-control # already newest
  + $ sudo apt-get install ros-noetic-effort-controllers
  + $ sudo apt-get install ros-noetic-geographic-msgs
  + $ sudo apt install ros-noetic-joy
  + $ sudo apt install ros-noetic-teleop-twist-joy
  + $ sudo apt-get install ros-noetic-move-base-flex
  + $ sudo apt-get install ros-noetic-teb-local-planner
  + $ sudo apt install ros-noetic-tf2-tools
  + $ sudo apt-get install ros-noetic-navigation
  + Original: Open a window; $ cd catkin\_ws\_sim\_phase2; $ catkin\_make
  + 11/12/22: Open a window; $ roscd; $ cd ..; $ catkin\_make
* Add catkin\_ws\_sim\_phase2 to bashrc $ nano ~/.bashrc
  + Add >source ~/catkin\_ws\_sim\_phase2/devel/setup.bash
  + Comment out other source statements. For example:
    - ># source /opt/ros/noetic/setup.bash
    - ># source /home/al/catkin\_ws/devel/setup.bash
* Once completed, fix the execute permissions on the Python files.
* Original: $ cd /home/al/catkin\_ws\_sim\_phase2/
* 11/12/22: $ cd /home/tractor/catkin\_ws/src/my\_odom\_publisher/src
* $ chmod +x odom.py
* $ cd /home/al/catkin\_ws\_sim\_phase2/src/my\_odom\_publisher/src/
  + $ chmod +x odom.py
* $ cd /home/al/catkin\_ws\_sim\_phase2/src/nav/nodes
* 11/12/22 - $ cd /home/tractor/catkin\_ws/src/nav/nodes
  + $ chmod +x cmd\_vel\_to\_ackermann\_drive.py
  + $ chmod +x joyop.py
  + $ chmod +x mod\_cmd\_vel\_to\_ackermann.py
* $ cd /home/al/catkin\_ws\_sim\_phase2/src/picar/src/ackermann\_vehicle\_gazebo/scripts
* 11/12/22 - $ cd /home/tractor/catkin\_ws/src/picar/src/ackermann\_vehicle\_gazebo/scripts
  + $ chmod +x ackermann\_controller
* $ cd /home/al/catkin\_ws\_sim\_phase2/
* 11/12/22 - $ roscd; $ cd ..
* $ source devel/setup.bash
* execute the launch files listed in the README.md file
  + To start the simulator:<BR>
  + roslaunch ackermann\_vehicle\_gazebo ack\_new.launch
* Joystick - just for testing connect a physical joystick. Ultimately I need to feed the lora radio control into the joystick commands

| Store list of devices connected with power off; $ ls /dev/input > dev\_list\_1.txt |
| --- |
| Connect the joystick |
| $ ls /dev/input | diff --suppress-common-lines -y - dev\_list\_1.txt   | al@al-HP-ProBook-645-G1:~$ ls /dev/ | diff --suppress-common-lines -y - dev\_list\_1.txt  serial <  ttyACM0 <  al@al-HP-ProBook-645-G1:~$ | | --- | |

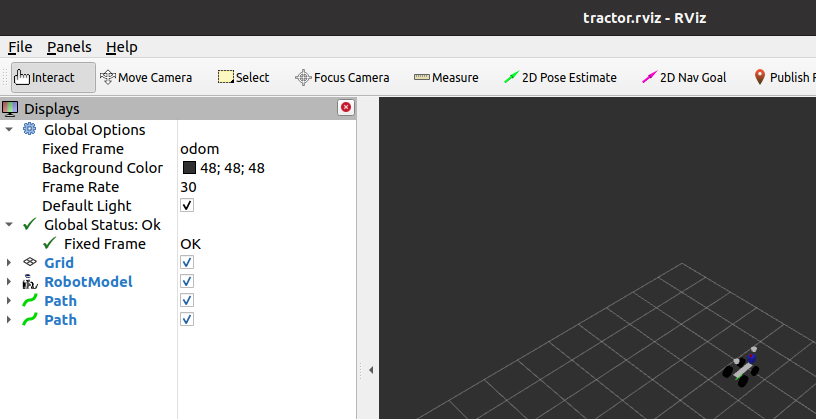
Adjust to match your joystick and drive around

* 
* 
* Plug in your joystick
* Window 2: $ roslaunch nav joyop.launch # should now be able to drive around using the joystick

Start Tebplanner and use RVIZ to send navigation goals:

* Window 3: $ roslaunch nav nav\_newteb.launch
* Window 4: $ rviz -d ~/catkin\_ws\_sim/src/nav/ack\_new.rviz
* Change the Fixed Frame to odom
* 
* Select Add; By Display type: Robot Model; OK
  + 
  + Select Add; By Topic:
    - /move\_base/TebLocalPlannerROS/global\_plan
    - 
    - /move\_base/TebLocalPlannerROS/local\_plan

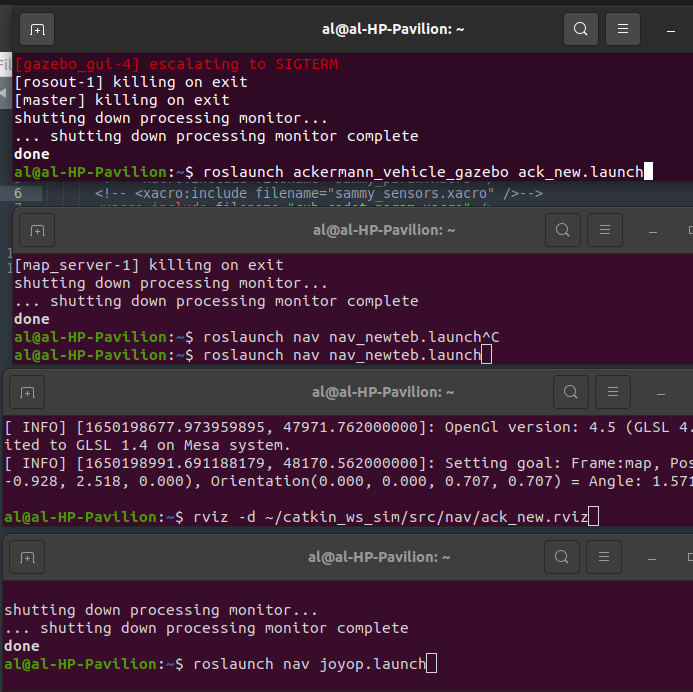
and save the configuration in local directory.

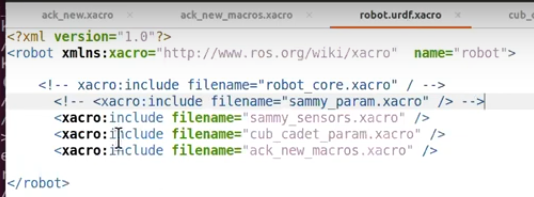
* 
* Use 2D Nav Goal to send navigation commands

Change robot model

* Xacro file is in directory:
* /home/al/catkin\_ws\_sim\_phase2/src/picar/src/ackermann\_vehicle\_description/urdf
* Open: robot.urdf.xacro

| <?xml version="1.0"?>  <robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="robot">  <!-- xacro:include filename="robot\_core.xacro" / -->  <!-- <xacro:include filename="sammy\_param.xacro" /> -->  <!-- <xacro:include filename="sammy\_sensors.xacro" /> -->  <xacro:include filename="cub\_cadet\_param.xacro" />  <xacro:include filename="ack\_new\_macros.xacro" />  <xacro:include filename="sammy\_sensors.xacro" />  </robot> |
| --- |





Older Notes are below this point………….

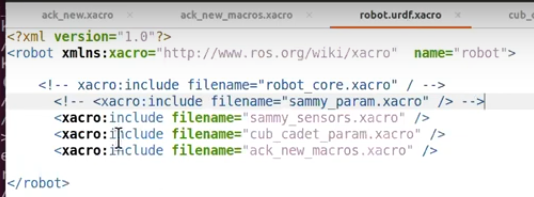
Start navigation using move\_base with teb\_local\_planner:

$ sudo apt-get install ros-noetic-navigation

$ sudo apt-get install ros-noetic-teb-local-planner

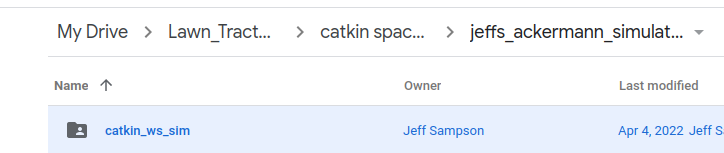
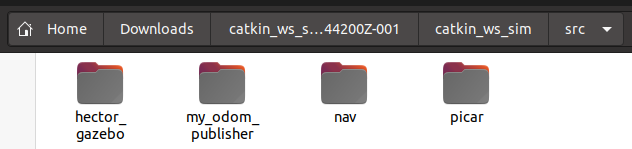
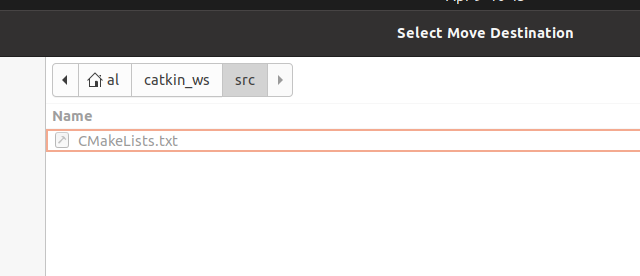
$ roslaunch nav nav\_newteb.launch

Change robot model

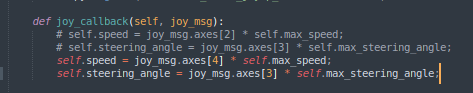


* + You can run a joystick with either of these commands:<BR>
  + roslaunch nav joy.launch<BR>
  + roslaunch nav joyop.launch<BR><BR>
  + Or to start navigation using move\_base with teb\_local\_planner:<BR>
  + roslaunch nav nav\_newteb.launch<BR>
  + To view the vehicle and send navigation goals:<BR>
  + rviz -d ~/catkin\_ws\_sim/src/nav/ack\_new.rviz<BR>

| This assumes you already have a catkin\_ws directory in your home directory.  This is intends that you will install the catkin\_ws\_sim\_phase2 along side the existing catkin\_ws. See README.md to switch the "source" between catkin\_ws and catkin\_ws\_sim\_phase2.  source /opt/ros/noetic/setup.bash  # source ~/catkin\_ws/devel/setup.bash  # source ~/catkin\_ws\_sim/devel/setup.bash  source ~/catkin\_ws\_sim\_phase2/devel/setup.bash  View the OriginalInitialInstallNotes.txt that was originally posted to Slack. (that was for the first phase, catkin\_ws\_sim) |
| --- |
| This is for the first URDF experiment; labeled catkin\_ws\_sim.  The second one is called catkin\_ws\_sim\_phase2.  So everywhere below where it says catkin\_ws\_sim assume it should be catkin\_ws\_sim\_phase2.  =========================================================================================  Then go here:  https://drive.google.com/drive/folders/1t2P2sfiranPe10fmUBUnnVvKNgrfybAV  And go into jeffs\_ackermann\_simulation you should see a catlin\_ws\_sim directory. If you click on that and do a download (under the 3 dots by the trash can in upper right hand corner) it will generate a ZIP file and down load to your downloads directory. You can unzip it there. Go into the resulting directory and copy the catkin\_ws\_sim directory to your home directory.(Along side of your existing catkin\_ws.)  I assume your .bashrc has:  source ~/catkin\_ws/devel/setup.bash  and replace it with:  source ~/catkin\_ws\_sim/devel/setup.bash  Restart all of your terminal windows.  The README.md file has a list of dependencies to load. (If you know what you are doing you could probably fix these with rosdep. But I just loaded them.) Load all of these before you do a catkin\_make. (If you have already loaded these, you don't need to do this again.)  Open a window and go to the catkin\_ws\_sim directory and do:  catkin\_make  If that works then I noticed the zipped file transfer deleted the execute permissions on the Python files. So you have to fix those.  catkin\_ws\_sim/src/my\_odom\_publisher/src/odom.py  catkin\_ws\_sim/src/nav/nodes/cmd\_vel\_to\_ackermann\_drive.py  catkin\_ws\_sim/src/nav/nodes/joyop.py  catkin\_ws\_sim/src/nav/nodes/mod\_cmd\_vel\_to\_ackermann.py  catkin\_ws\_sim/src/picar/src/picar/src/ackermann\_vehicle\_gazebo/scripts/ackermann\_controller  If you get that far then you can execute the launch files listed in the README.md file. Or, if you don't want to waste a lot of time and just want to look at the files, you can look at them on Google drive. Or down load them to your computer and dig through them there.  There is no guarantee this is going to work... |
| # test1  <b>Test to see how this works</b>  To fix missing/bad keys  https://answers.ros.org/question/379190/apt-update-signatures-were-invalid-f42ed6fbab17c654/  <b>You may need these dependencies:</b>  sudo apt-get install ros-noetic-ackermann-msgs<br>  sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-ros-control<br>  sudo apt-get install ros-noetic-effort-controllers<br>  sudo apt-get install ros-noetic-geographic-msgs<br>  sudo apt install ros-noetic-joy<br>  sudo apt install ros-noetic-teleop-twist-joy<br>  sudo apt-get install ros-noetic-move-base-flex<br>  sudo apt-get install ros-noetic-teb-local-planner<br>  sudo apt install ros-noetic-tf2-tools<br>  sudo apt-get install ros-noetic-navigation<br>  <b>Notes:</b>  Since the workspace is named catkin\_ws\_sim\_phase2 instead of catkin\_ws you may need to update the source line in the .bashrc file.  To start the simulator:<BR>  roslaunch ackermann\_vehicle\_gazebo ack\_new.launch<BR>  You can run a joystick with either of these commands:<BR>  roslaunch nav joy.launch<BR>  roslaunch nav joyop.launch<BR><BR>  Or to start navigation using move\_base with teb\_local\_planner:<BR>  roslaunch nav nav\_newteb.launch<BR>  To view the vehicle and send navigation goals:<BR>  rviz -d ~/catkin\_ws\_sim/src/nav/ack\_new.rviz<BR> |

* Download Jeff’s WS from [Google](https://drive.google.com/drive/folders/1Lsu2ZN3aeGRUzJJqlfFOejG5v3KxXSHT?usp=sharing)
* 
* Extract src folders
* 
* Move folders to catkin\_ws\src …
* 
* 
* $ sudo apt-get install ros-noetic-geographic-msgs
* $ sudo apt-get install ros-noetic-ackermann-msgs
* $ sudo apt-get install ros-noetic-effort-controllers
* $ cd /home/al/catkin\_ws/src/my\_odom\_publisher/src
* $ chmod +x odom.py
* $ cd ~/catkin\_ws
* $ catkin\_make
* $ cd
* $ roslaunch ackermann\_vehicle\_gazebo ack\_new.launch
* Close the process

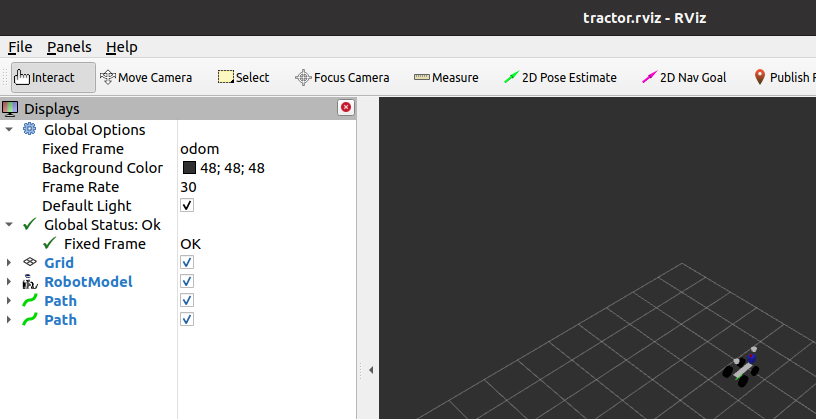
Adjust to match your joystick and drive around

* 
* 
* Plug in your joystick
* $ sudo jstest /dev/input/js1 # to confirm your joystick is publishing on js1
* Determine which axes you want to control speed and steering and update values in joyop.py
* $ sudo jstest /dev/input/js1
* Use Sublime; Open joyop.py and adjust speed and angle axes reference to match your joystick
* $ sudo apt install ros-noetic-teleop-twist-joy
* $ cd /home/al/catkin\_ws/src/nav/nodes
* $ chmod +x cmd\_vel\_to\_ackermann\_drive.py
* $ chmod +x joyop.py
* $ chmod +x mod\_cmd\_vel\_to\_ackermann.py
* $ cd
* $ cd ~/catkin\_ws
* $ catkin\_make
* $ cd
* Window 1: $ roslaunch ackermann\_vehicle\_gazebo ack\_new.launch
* Window 2: $ roslaunch nav joyop.launch # should now be able to drive around using the joystick

Use RVIZ to send navigation goals:

* $ rviz -d ~/catkin\_ws\_sim/src/nav/ack\_new.rviz
* Add:
  + Robot Model,
  + /move\_base/TebLocalPlannerROS/global\_plan
  + /move\_base/TebLocalPlannerROS/local\_plan
  + Change the Fixed Frame to odom

and save the configuration

* 

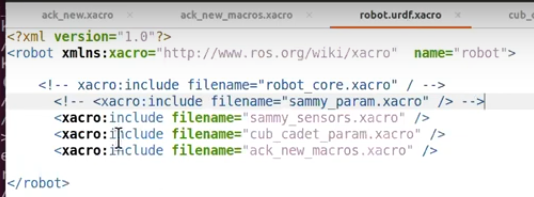
Start navigation using move\_base with teb\_local\_planner:

$ sudo apt-get install ros-noetic-navigation

$ sudo apt-get install ros-noetic-teb-local-planner

$ roslaunch nav nav\_newteb.launch

Change robot model



Other references:

<https://github-wiki-see.page/m/heechul/picar/wiki/ROS-PiCar-Setup>

| $ history |
| --- |
| 1 sudo apt install chrony  2 sudo systemctl enable chrony  3 sudo systemctl start chrony  4 sudo systemctl status chrony  5 sudo apt install curl  6 curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -  7 sudo apt update  8 sudo apt upgrade  9 wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -  10 sudo apt-get install apt-transport-https  11 echo "deb https://download.sublimetext.com/ apt/stable/" | sudo tee /etc/apt/sources.list.d/sublime-text.list  12 sudo apt-get update  13 sudo apt-get install sublime-text  14 sudo apt install git  15 git clone https://github.com/linorobot/rosme  16 cd rosme  17 ./install  19 sudo rosdep init  20 rosdep update  21 cd  23 source /opt/ros/noetic/setup.bash  24 mkdir -p ~/catkin\_ws/src  25 cd ~/catkin\_ws/  26 catkin\_make  27 source devel/setup.bash  28 echo $ROS\_PACKAGE\_PATH  29 roscd  cd  30 nano ~/.bashrc  31 history  32 history > historyrosinstall20220409.txt |

11/16/22 - exploring the setup

$ rosservice call /gazebo/get\_world\_properties "{}"

$ rosservice call /gazebo/get\_model\_properties ackermann\_vehicle

$ roslaunch ackermann\_vehicle\_gazebo ack\_new.launch

| <launch>  …  <arg name="world\_name" default="worlds/empty.world"/>  <arg name="cmd\_timeout" default="0.5"/>  <!-- Vehicle pose -->  ….  <include file="$(find ackermann\_vehicle\_description)/launch/ack\_new.launch">  …  <include file="$(find gazebo\_ros)/launch/empty\_world.launch">  …  <arg name="world\_name" value="$(find hector\_gazebo\_worlds)/worlds/sick\_robot\_day\_2012\_20m.world"/>  …  <node name="spawn\_vehicle" pkg="gazebo\_ros" type="spawn\_model"  args="-urdf -param robot\_description -model ackermann\_vehicle  …  <node name="controller\_spawner" pkg="controller\_manager" type="spawner"  …  <node name="ackermann\_controller" pkg="ackermann\_vehicle\_gazebo"  type="ackermann\_controller">  …  <rosparam file="$(find ackermann\_vehicle\_gazebo)/config/em\_3905\_ackermann\_ctrlr\_params.yaml" command="load"/>  …  <include file="$(find my\_odom\_publisher)/launch/start\_odom.launch"/>  </launch> |
| --- |

$ roslaunch nav joyop.launch # should now be able to drive around using the joystick

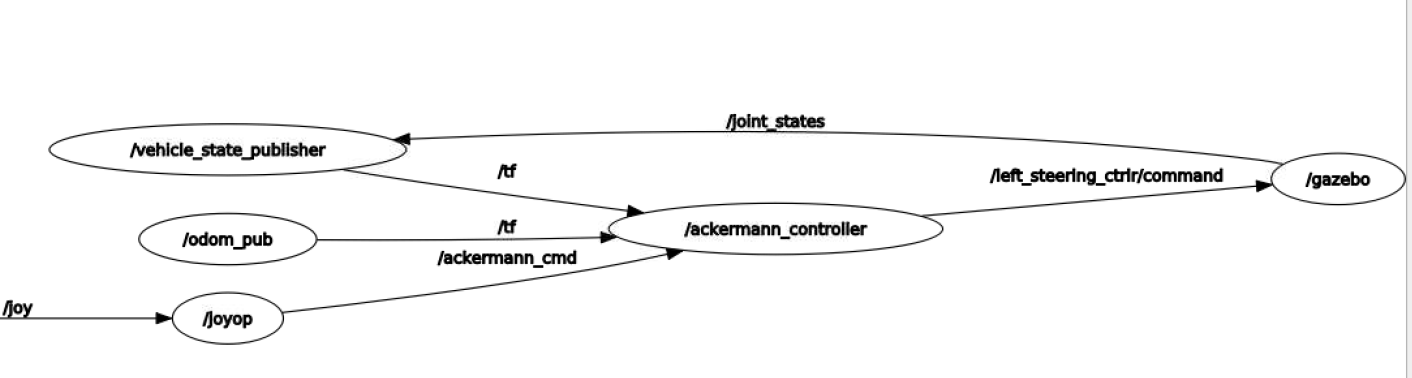
| /home/tractor/catkin\_ws/src/nav  <node name="joy" type="joy\_node" pkg="joy">  <node name="joyop" type="joyop.py" pkg="nav" output='screen' args="$(arg max\_speed) |
| --- |

$ roslaunch nav nav\_newteb.launch

| <launch>  ..  <node name="map\_server" pkg="map\_server" type="map\_server" args="$(find nav)/maps/farm.yaml"/>  …  <node pkg="tf2\_ros" type="static\_transform\_publisher" name="map\_to\_odom" args="0 0 0 0 0 0 map odom" />  …  <node pkg="move\_base" type="move\_base" respawn="false" name="move\_base" output="screen">  <rosparam file="$(find nav)/config\_newteb/costmap\_common\_params.yaml" command="load" ns="global\_costmap" />  <rosparam file="$(find nav)/config\_newteb/costmap\_common\_params.yaml" command="load" ns="local\_costmap" />  …  <!-- Set "cmd\_angle\_instead\_rotvel" to "True" and use mod\_cmd\_vel\_to\_ackermann.launch -->  <include file="$(find nav)/launch/mod\_cmd\_vel\_to\_ackermann.launch" />  … |
| --- |

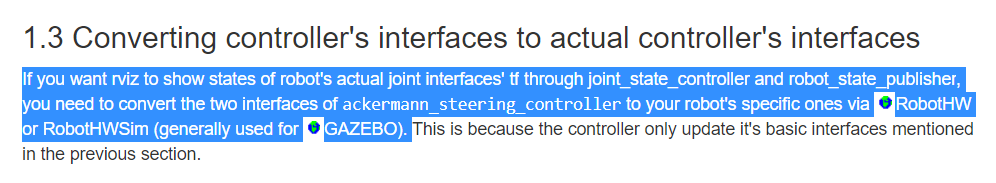
$ rviz -d ~/catkin\_ws\_sim/src/nav/ack\_new.rviz

So lets take steering as an example. At the moment I move the steering joystick and the wheels turn left and right in Gazebo. In the physical world the wheels have not moved. Gazebo is getting the state of the wheels from somewhere other than the actual steering sensor. Where is that?



$ rosrun rqt\_graph rqt\_graph

http://wiki.ros.org/ackermann\_steering\_controller



Starting from the beginning

<https://github.com/nickcharron/waypoint_nav>

* fuse odometry data with IMU and GPS data