Step 16 - Test steering control - old - used before mounting sensor on motor

Assumptions

* Motor is mounted, but not powered
* Angle sensor is mounted, connected to the Teensy which is connected to the RPi and basic testing has been conducted
* The sensor value for hard right and hard left is known

To be completed in this step:

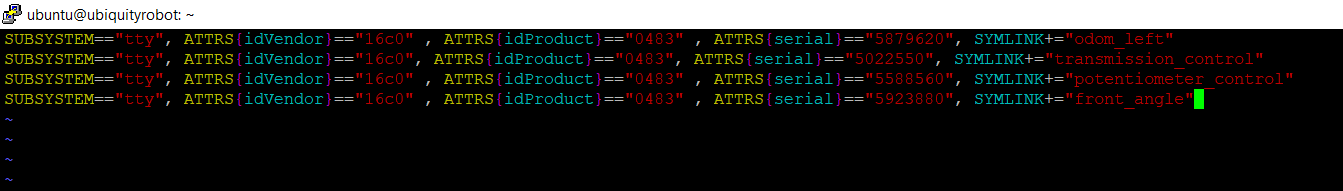
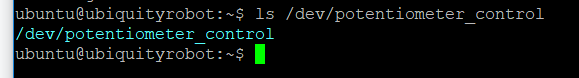
* 3D print the holder for the motor controller and Teensy
* Connect 12V power to the motor controller from battery
* Connect the motor controller (GND, Direction and PWR level) to the Teensy
* Fasten the motor controller and Teensy to the motor mount

|  |  |
| --- | --- |

* Connect the motor controller power leads to the motor
* Run sample program that reads steering sensor
  + Connect USB drive
    - Make cable long enough to connect
    - Route cable
    - See which devices are currently connected $ ls /dev/ttyACM\*
      * /dev/ttyACM0 /dev/ttyACM1 /dev/ttyACM2 /dev/ttyACM3
  + Create a UDEV rule for the front\_angle Teensy

1. Create a UDEV rule for the front\_angle Teensy
   * Background: <https://arduino.stackexchange.com/questions/3680/in-linux-how-to-identify-multiple-arduinos-connected-over-usb>; <http://www.googlinux.com/clearing-dmesg-logs/index.html>
   * Logon to the RPI; Confirm the teensy is able to be seen by RPi. $ ls /dev/ttyACM\*, then connect front angle sensor, $ ls /dev/ttyACM\*
   * Unplug / turn off power to any USB devices
   * Save the dmesg log. $ dmesg > dmesg-`date +%d%m%Y`.log
   * Clear the log. $ sudo dmesg -c
   * Plug in the Teensy to be used. Remember this has a unique serial number.
   * List the dmesg $ dmesg
   * Results:

| ubuntu@ubiquityrobot:~$ dmesg  [118616.881363] usb 1-1.3.2: new full-speed USB device number 45 using dwc\_otg  [118617.014192] usb 1-1.3.2: New USB device found, idVendor=16c0, idProduct=0483  [118617.014207] usb 1-1.3.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3  [118617.014215] usb 1-1.3.2: Product: USB Serial  [118617.014223] usb 1-1.3.2: Manufacturer: Teensyduino  [118617.014230] usb 1-1.3.2: SerialNumber: 5588760  [118617.015401] cdc\_acm 1-1.3.2:1.0: ttyACM0: USB ACM device |
| --- |

* Setup udev rule mapped to “steer\_motor ” because later you will want to update the setting: “upload port = /dev/steer\_motor ” in the platformio.ini file. Background reading: <https://linuxconfig.org/tutorial-on-how-to-write-basic-udev-rules-in-linux>
* Edit and update this line, then copy it because it will be easier to paste into the editor.
* SUBSYSTEM=="tty", ATTRS{idVendor}=="16c0" , ATTRS{idProduct}=="0483" , ATTRS{serial}=="5588760", SYMLINK+="steer\_motor"
* Change permissions so I can use FileZilla and Sublime to edit the udev files
  + $ sudo chmod -R 777 /etc/udev/rules.d/05-serial.rules
* $ sudo edit /etc/udev/rules.d/05-serial.rules
* 
* Move the cursor to the end of the last character in the last line
* Press INSERT; Move the cursor to the end and press enter
* Copy the new line above and paste into the editor by right mouse clicking at the beginning of the new line.
* Press Escape to exit the insert mode
* Save the file by entering “:w” then enter
* Quit using “:q” then enter
* Reboot the RPi for the changes to take effect $ sudo reboot
* $ ls /dev/steer\_motor to confirm the link has been made
* 

1. Update PlatformIO environment ini

* Open FileZilla, Update the platformio.ini file in /home/ubuntu; Add section for front\_angle

| ;  [env:front\_angle]  platform = teensy  board = teensy31  framework = arduino  upload\_port = /dev/front\_angle |
| --- |

* “board = teensy35” is used for a Teensy 3.5

1. Update bash scripts to help PlatformIO compiling
   * Background: There are two bash scripts that have been used for compiling, compile\_a.sh and compile\_b.sh. These scripts need to be updated because you have a new microcontroller connected to the USB hub.
   * Connect all the USB cables and run $ grep 16c0 /sys/bus/usb/devices/\*/idVendor to list all the Teensy devices connected.
   * For testing, run the scripts without the platformio or roslaunch commands enabled. I had issues getting permissions correct.
   * If you reboot, to resolve permission issues run:
     + $ sudo **chown** root:plugdev /sys/bus/usb/drivers/usb/{**bind**,unbind}
     + $ sudo **chmod** g+w /sys/bus/usb/drivers/usb/{**bind**,unbind}
     + See: <https://community.platformio.org/t/multiple-boards-upload-port-works-but-crashes-2nd-board/9029/8>

| ubuntu@ubiquityrobot:~$ grep 16c0 /sys/bus/usb/devices/\*/idVendor  /sys/bus/usb/devices/1-1.3.1.1/idVendor:16c0 (joystick)  /sys/bus/usb/devices/1-1.3.1.2/idVendor:16c0 (left speed)  /sys/bus/usb/devices/1-1.3.1.3/idVendor:16c0 (transmission)  /sys/bus/usb/devices/1-1.3.1.4/idVendor:16c0 (front\_angle)  /sys/bus/usb/devices/1-1.3.2/idVendor:16c0 (steer\_motor)  **(NOTE: Put the latest device IDs in the file “Documenting the Teensy device ID’s”)** |
| --- |

| **File: compile\_a.sh**  #!/bin/bash  # this script to "turn off" the USB ports and to start the compile process  #  # Comment out the device you will be programming  #  # (1 of 2) Comment OUT the device you WILL be programming or devices not connected  #  #echo -n "1-1.3.1.1" > /sys/bus/usb/drivers/usb/unbind # diy joystick  echo -n "1-1.3.1.2" > /sys/bus/usb/drivers/usb/unbind # left speed  echo -n "1-1.3.1.3" > /sys/bus/usb/drivers/usb/unbind # transmission  echo -n "1-1.3.1.4" > /sys/bus/usb/drivers/usb/unbind # front\_angle  #echo -n "1-1.3.2" > /sys/bus/usb/drivers/usb/unbind # steer\_motor  #  # (2 of 2) UN-COMMENT the line for the platformio environment you WANT TO COMPILE  #  # Platformio environments include: [env:joystick], [env:left\_speed], [env:transmission], [env:front\_angle], [env:steer\_motor]  #sudo platformio run -t upload -e front\_angle # for [env:front\_angle]  sudo platformio run -t upload -e steer\_motor # for [env:steer\_motor]  #sudo platformio run -t upload -e transmission # for [env:transmission] |
| --- |

| **File: compile\_b.sh**  #!/bin/bash  # this script to "turn on" the USB ports after the compile process is complete  #  # Comment OUT the board that IS being programmed. It will not need the "bind" statement. All other connected devices should be active.  #  #echo -n "1-1.3.1.1" > /sys/bus/usb/drivers/usb/bind # joystick  echo -n "1-1.3.1.2" > /sys/bus/usb/drivers/usb/bind # left speed  echo -n "1-1.3.1.3" > /sys/bus/usb/drivers/usb/bind # transmission  echo -n "1-1.3.1.4" > /sys/bus/usb/drivers/usb/bind # front\_angle  #echo -n "1-1.3.2" > /sys/bus/usb/drivers/usb/bind # steer\_motor  #  # start the launch file that will initiate the Teensy's you want to test  cd /home/ubuntu/catkin\_ws/src  roslaunch teensy\_launch.launch  #roslaunch steer\_test\_launch.launch  #roslaunch transmission\_test\_launch.launch |
| --- |

To test:

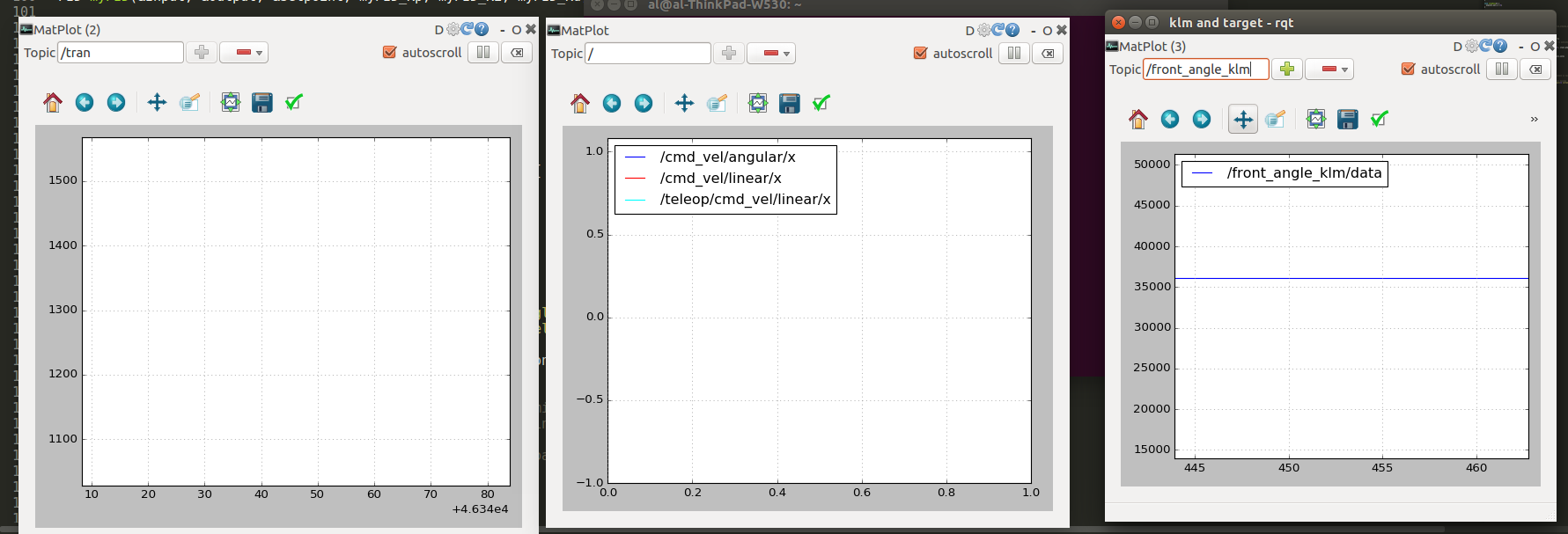
* $ grep 16c0 /sys/bus/usb/devices/\*/idVendor
* $ ls /dev/ttyACM\*
* If you have code to compile and multiple Teensy’s are connected and active, run $ bash compile\_a.sh then $ ls /dev/ttyACM\*
* Alternatively if the USB ports are still unbound (i.e. only one Teensy active), run $ sudo platformio run -t upload -e steer\_motor
* Once compile is clean and you want to test, run $ bash compile\_b.sh then $ ls /dev/ttyACM\*
* If you have seen how they appear, disappear and reappear, you can now take out the comment for the PlatformIO statement in compile\_a.sh
  + If USB ports are still unbound, run $ sudo platformio run -t upload -e steer\_motor

1. Test compiling and running blink.cpp
2. Update launch file - steer\_test\_launch.launch to only launch the angle and motor controller teensy.

steer\_test\_launch.launch

| <launch>  <?ignore - used to comment out lines  <node ns="transmission\_teensy" name="transmission\_cntrl" pkg="rosserial\_python" type="serial\_node.py" args="/dev/transmission\_control" output="screen" />  <node ns="potentiometer\_teensy" name="potentiometer\_output" pkg="rosserial\_python" type="serial\_node.py" args="/dev/potentiometer\_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" />  </launch> |
| --- |

1. Test program
   * Once you have a complete compile and before launching the programs
     + Check power to motor
     + Ensure the work area is clear; Double check the steering rods and there are no obstacles (e.g. wires) in the way
     + Remind yourself where to disconnect power should you need to do so
     + On RPi,
       - $ ls /dev # you should only see one teensy (i.e. steer\_motor)
       - Run $ bash compile\_b.sh
       - Optionally if the bind statements have already been run:
         * $ cd /home/ubuntu/catkin\_ws/src
         * $ roslaunch steer\_test\_launch.launch
   * On Laptop
     + Ctrl+Alt+T, $ rqt
     + Organize rqt screens to monitor steering PWM, cmd\_vel/angular/x and front\_angle\_klm/data. Use “dock” icon to move screens to desired position

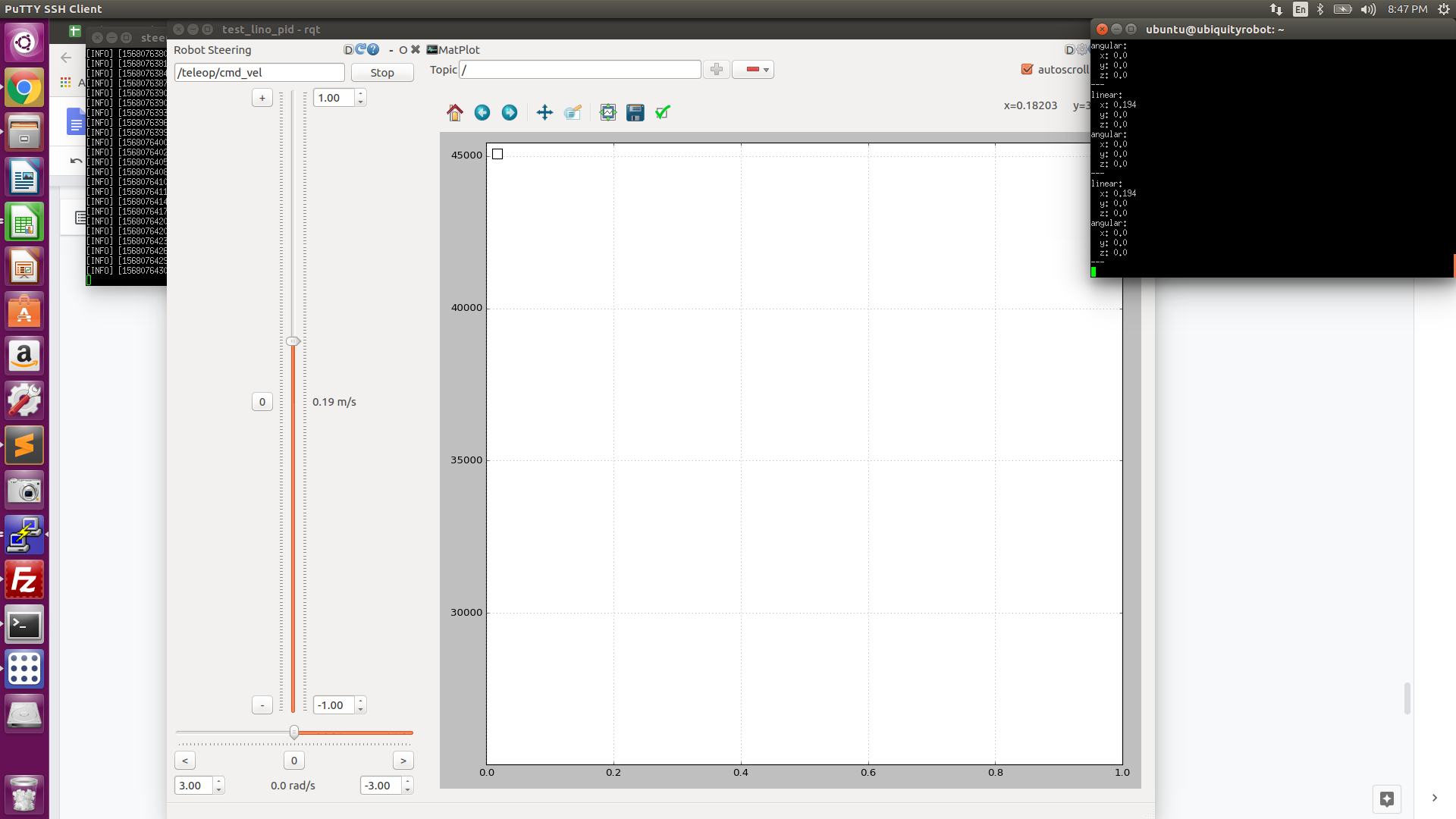


* + - Connecting the joystick - See Step 4 - Install ROS-Ag lawn tractor simulator section **Test drive using gamepad** for background and to confirm port identifier.
    - Start joy\_input to transfer data to cmd\_vel:
* Ctrl+Alt+T, $ roslaunch tractor\_teleop cmd\_vel\_mux.launch
* Ctrl+Alt+T, $ roslaunch tractor\_teleop drive\_teleop.launch joy\_dev:=/dev/input/js0
* Note: drive\_teleop.launch has as a default: <arg name="joy\_dev" default="/dev/input/js0"/> If your joystick is on js1 you can either change the launch file or adjust the argument.
* Alternatively, the command below has both of the above commands combined
  + $ roslaunch tractor\_teleop drive.launch
    - At this point you should be able to press the deadman switch, move the steering joystick and see the target angular.z adjust.

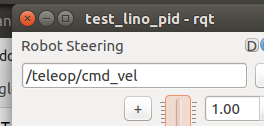
Next Step

Confirm steer straight and left right settings.

* Choose Plugins -> Robot Tools -> Robot Steering



In the rqt topic put /teleop/cmd\_vel in the topic field



On the RPi echo the topic and move the slider in rqt to see the topic being published

* Drag the window all the way to the right

|  |
| --- |
|  |

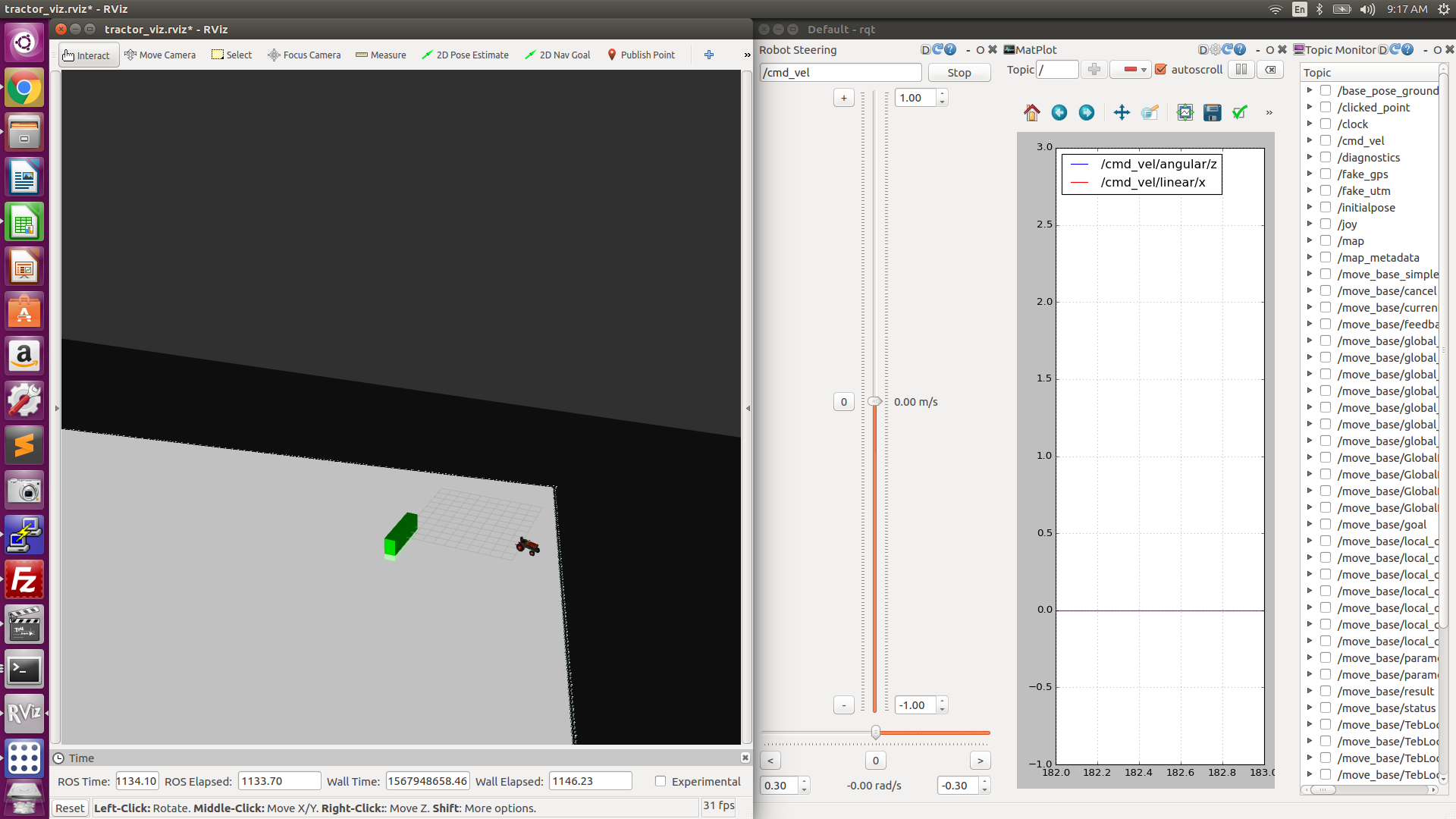
* You can now adjust the slide bars and send cmd\_vel command using this interface
* An alternative is to run a bash script written for this purpose.

| #!/bin/bash  timeout 10 rostopic pub -r 10 teleop/cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.5}}'  timeout 10 rostopic pub -r 10 teleop/cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'  timeout 10 rostopic pub -r 10 teleop/cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.5}}' |
| --- |

* Plugins -> Visualizations -> Plot will allow you to subscribe and visualize (i.e. plot) rostopics

|  |
| --- |

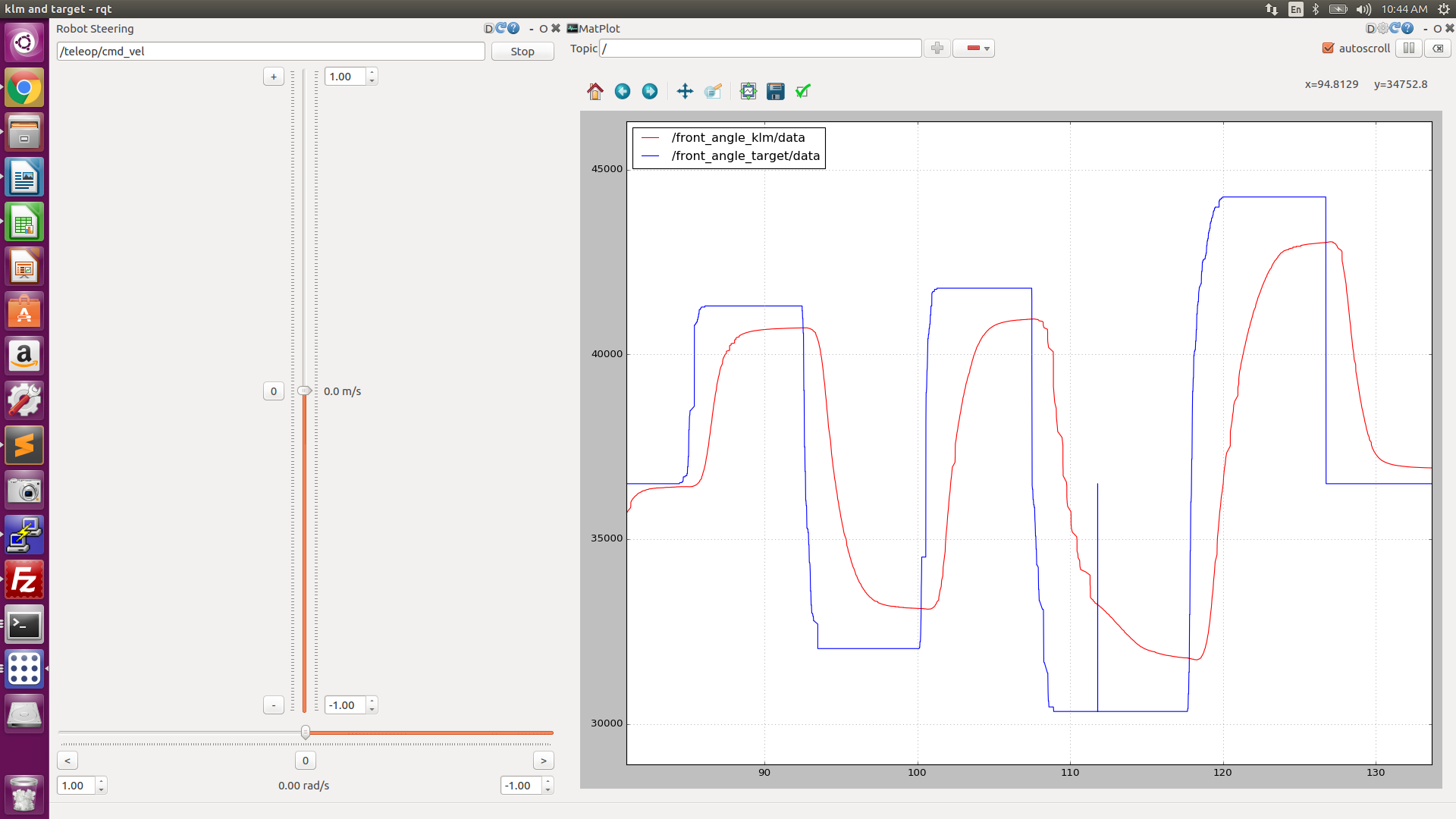
* Plugins -> Topics -> Topic Monitor lets you drag and drop the topics you are interested in



Testing protocol

| Cmv\_vel angular.z | Points to look for |
| --- | --- |
| 0.0 | Is the steering steady and straight; Is the steer effort zero |
| 0.03 to -0.03 | Does the steering react and move to these smaller increments |
| Move from 0 to +/- .5 | Does the steering reach the target and steer effort go to zero? What there overshoot? How long did it take to achieve the target? |
| 1 and -1 | Does the steering reach the target and steer effort go to zero? What there overshoot? How long did it take to achieve the target? |

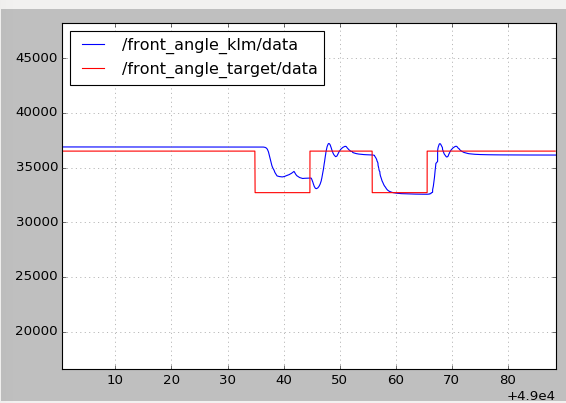
Example output going from: 0.0 -> -0.5 -> +0.5 -> -0.7 -> +0.7 -> 0.0



$ timeout 3 rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.5}}'

$ timeout 3 rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.75, y: 100.0, z: 0.0}, angular: {x: 200.0,y: 0.0,z: 0.0}}'

10/15/19



Using rosparam to change settings

The command <rosparam param="steering">[28900.1, 46300.0, 36500.0, 50.0, 2.0]</rosparam> is located in the teensy\_launch.launch and steer\_test.launch files. This sets the steering parameters:

steer\_left\_max = 28900;

steer\_right\_max = 46300;

steer\_straight = 36500;

max\_steer\_eff = 50;

steer\_Ki = 2;

The command that will also do the same thing is $ rosparam set /steering "[28900.0, 46300.0, 36500.0, 50.0, 2.0]"

I adjusted the steer straight to be 36800

10/18/19

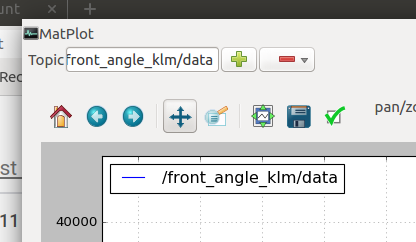
* $ rosparam set /steering "[28900.0, 46300.0, 38000.0, 70.0, 2.0]"

Once you have the transmission running you can further tune the settings. For example, in the grass, get the steer\_effort adjusted

3/1/20

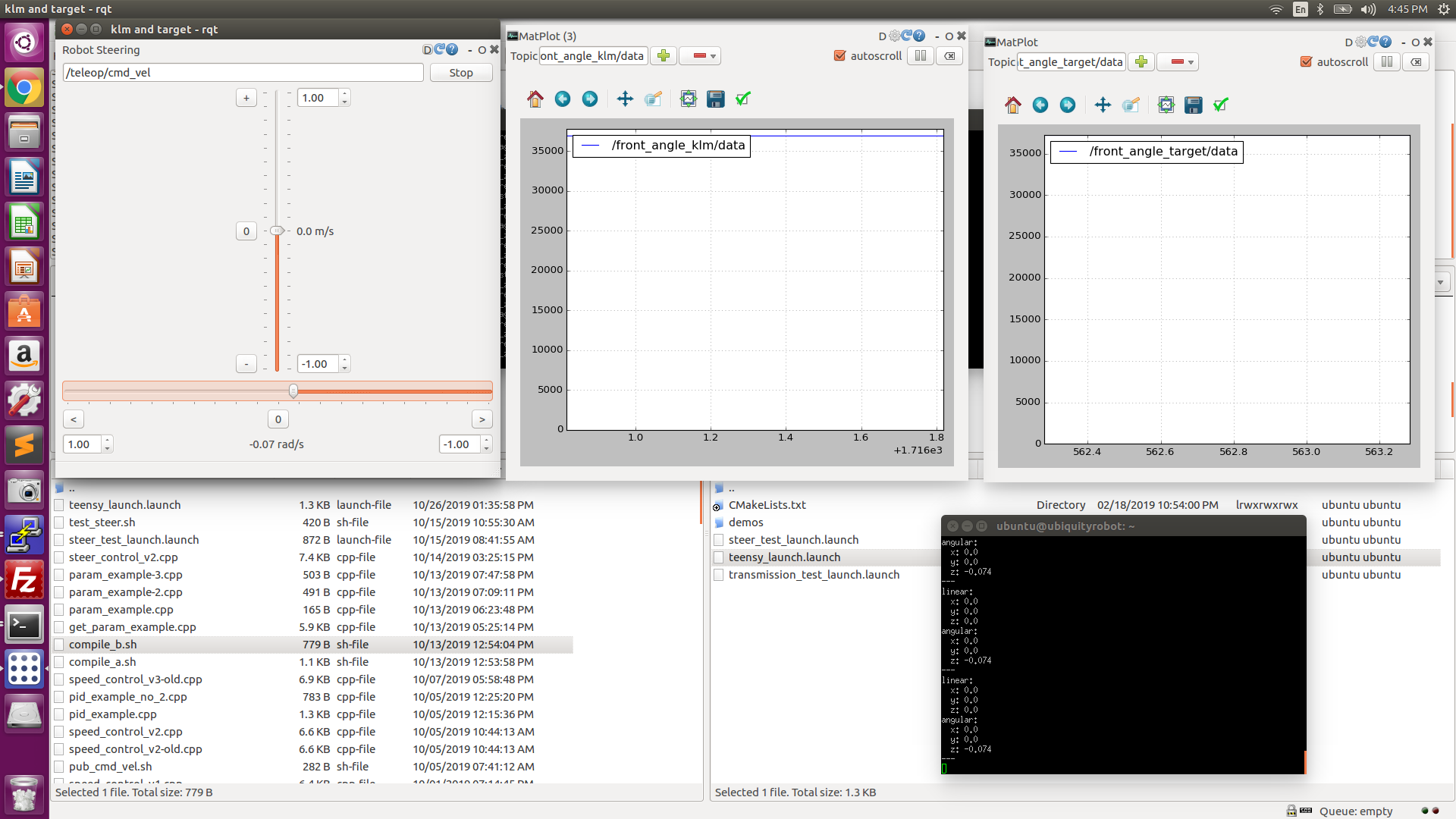
Import KLM perspectives in rqt

Dragging from right to left with the right mouse button down will slow down the speed of the plot.

Press the up/down arrow pan axis first

Doing the same up and down to zoom in on the preferred range.

| Straight ahead  position\_klm: 36102 |  |  |
| --- | --- | --- |
| Full left  position\_klm: 27184  Full right  position\_klm: 47415 |  |  |



The steering is not making the full rotation; Check variables being passed.

5/17/20

$ timeout 3 rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.5}}'

No movement at all - This issue was the signal wire to the motor controller had come loose.

$ rosparam set /steering “[27900.0, 45798.0, 36874.0, 60.0, 3.0]”

* $ rosparam set /steering "[27900.0, 45798.0, 36874.0, 60.0, 3.0]"
* $ rosparam set /steering "[23600.0, 44300.0, 33319.0, 60.0, 3.0]"

<rosparam param="steering">[24300.0, 44000.0, 34000.0, 70.0, 3.0]</rosparam>

5/18/20

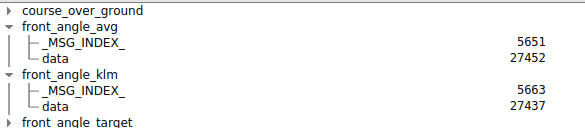
The wheels banged into a big ant mound and turned the wheels hard one direction. I need to re-confirm left, right, center.

Current:$ rosparam get /steering

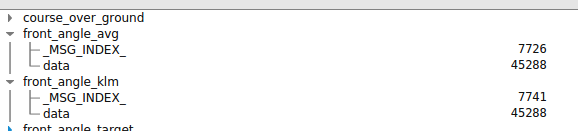
[24300.0, 44000.0, 34000.0, 70.0, 3.0]



Left



Right



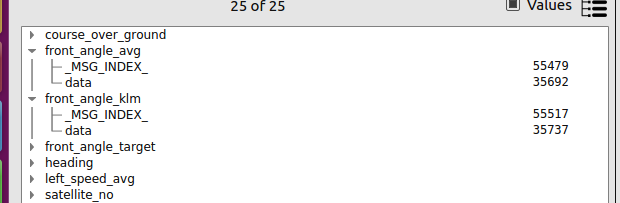
* $ rosparam set /steering "[27400.0, 45200.0, 35600.0, 70.0, 3.0]"

0.5, -0.3, 60 - produces a circle about 15’ in diameter

0.5, -0.2, 60 - produces a circle about 17’ in diameter

Positive still produced a right hand turn!

0.5, 0.005, 60



After 30 minutes of testing the center seems to be off which is leading to the inability to turn left.

I need to recenter.: 29209

Full left: 19200

Full right: 42000

42022

* $ rosparam set /steering "[19200.0, 42000.0, 29209.0, 70.0, 3.0]"

Pulling left when going straight

* rosparam set /steering "[19200.0, 42000.0, 30000.0, 70.0, 3.0]"
* rosparam set /steering "[19200.0, 42000.0, 30400.0, 70.0, 3.0]"

Center has become unadjusted again. It is near 40000!

Just to get the tractor back

* rosparam set /steering "[38200.0, 46000.0, 42000.0, 70.0, 3.0]"

5/22/20

2 rotation potentiometer keeping track of the rotations

Continuous rotation potentiometer

<https://www.usdigital.com/products/encoders/absolute/magnetic/MA3>

