Initial Troubleshooting - Max PWM values

Prior to doing a full PID-V tune we'd like you to try some "safe" PID-V parameters and increase the maximum current that can flow in to the motors on the robot (Max_PWM parameter). This may in itself resolve your issue - but even if it doesn't it prepares the robot for the next stage.

You need to edit a configuration file on front and the rear pi. Do not change any characters in the file except the actual values themselves.
There is only one new line and that is with fw_max_pwm: 2000 in the red highlights below.
Do not change anything else in base.yaml please.

SSH to frontpi however you did it before. If on a laptop connected to robot WiFi it will be:

> ssh ubuntu@10.42.0.1 (password ubuntu)
Use your favorite unix text editor to edit the file for example you could use
> sudo vi /opt/ros/kinetic/share/magni_bringup/param/base.yaml
Or in the alternative I personally find pico a lot easier to use so
> sudo pico /opt/ros/kinetic/share/magni_bringup/param/base.yaml

Change the values in these lines to as shown below (these lines are right at the start, changes in red here)

```
ubiquity_motor:
  serial_port: "/dev/ttyAMA0"
  serial_baud: 38400
  controller_loop_rate: 20
  serial_loop_rate: 200
  pid_proportional: 7000
  pid_integral: 0
  pid_derivative: 0
  pid_denominator: 1000
  fw_max_pwm: 2000
```

Now from frontpi SSH to rearpi and do the same changes to it's base.yaml file
> ssh ubuntu@rearpi (password ubuntu)

Let us know how this works for the thin wheels.
If the wheels are still too weak you can go back and increase fw_max_pwm: to up to 3700 but if that doesn't work please try a full PID-V tuning.

PID-V Tuning

Summary

PID-V is a variant of the well known PID controller. PID-V tuning is a process where you adjust parameters on a proportional integral differential velocity (PID- V) motor controller to improve the responsiveness and accuracy of the motor controller. In this document we describe a set of steps to achieve PID tuning in a robot. This consists of

1) Setup to adjust parameters and view output
2) PID tuning process
3) Final testing and check out and troubleshooting

Your goal in PID tuning is to adjust the control loop to minimize the variation between the target position of the motor and the actual position of the motor.

1) Set up to adjust parameters and view output

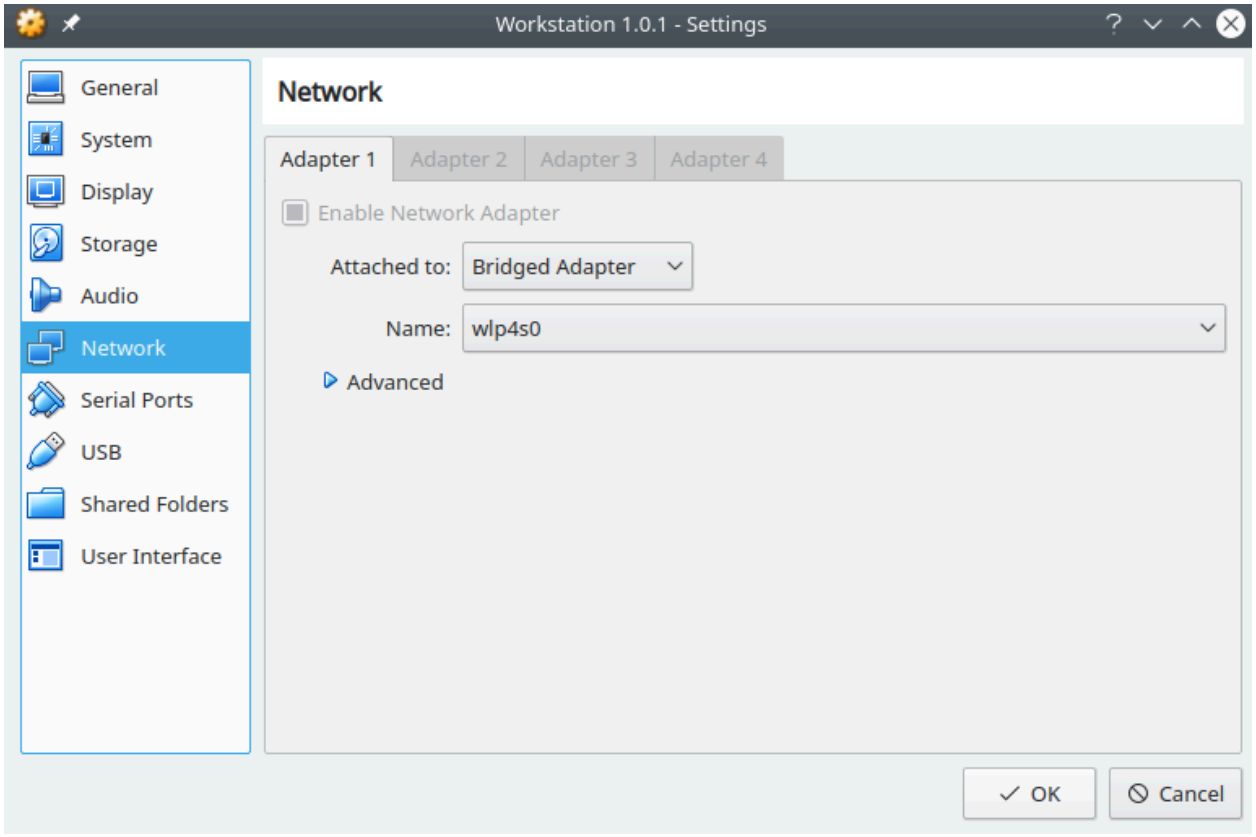
To be able to tune PID-V you need to be able to adjust the parameters and measure the output. The best way to do this is to use the ROS (robot operating system) tool suite known as RQT. There are 3 methods to make RQT operate.

a) Using a laptop running virtual box
b) On the robot itself
c) Using a laptop running Ubuntu

1a) Using the virtual box.

On your laptop iInstall VirtualBox: https://www.virtualbox.org/
Download the VM from here: https://downloads.ubiquityrobotics.com/vm.html

Add the VM to your virtual box. Go into the VM settings, and under Network, make sure it is attached to “Bridged Adapter” and your computer's wifi card is selected as the name.



Make sure your laptop is connected to the robot's wifi, then boot up the VM.

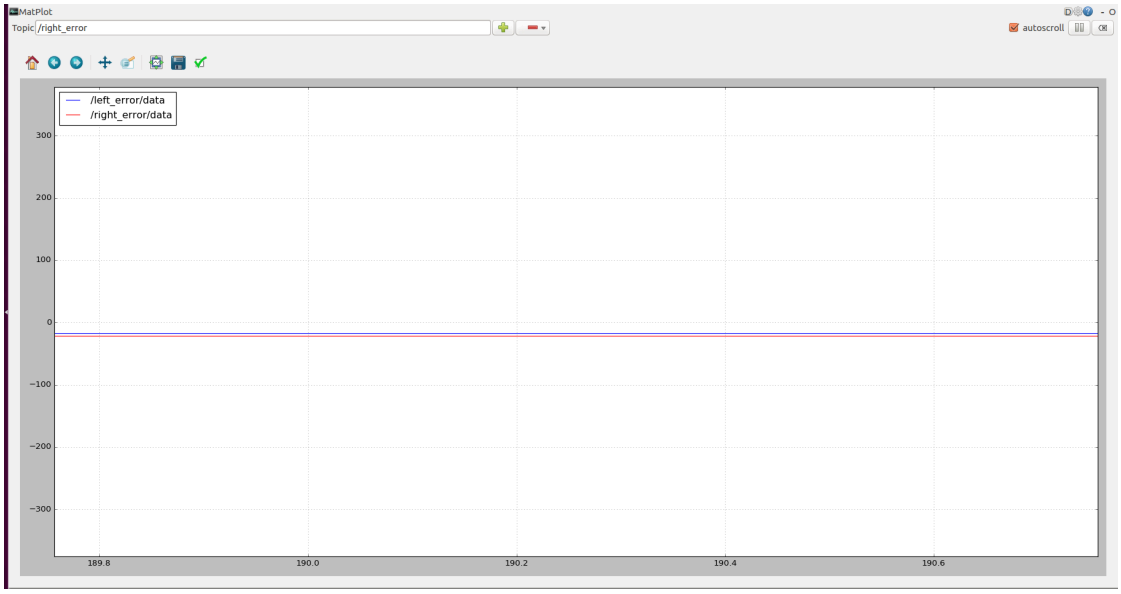
Bringing up rqt_reconfigure and rqt_plot

In the VM Use Ctrl-Alt-T to bring up a terminal and in that terminal run:
export ROS_HOSTNAME=ubuntu-VirtualBox.local
export ROS_MASTER_URI=http://frontpi.local:11311

Then in that terminal type

rosrun rqt_plot rqt_plot

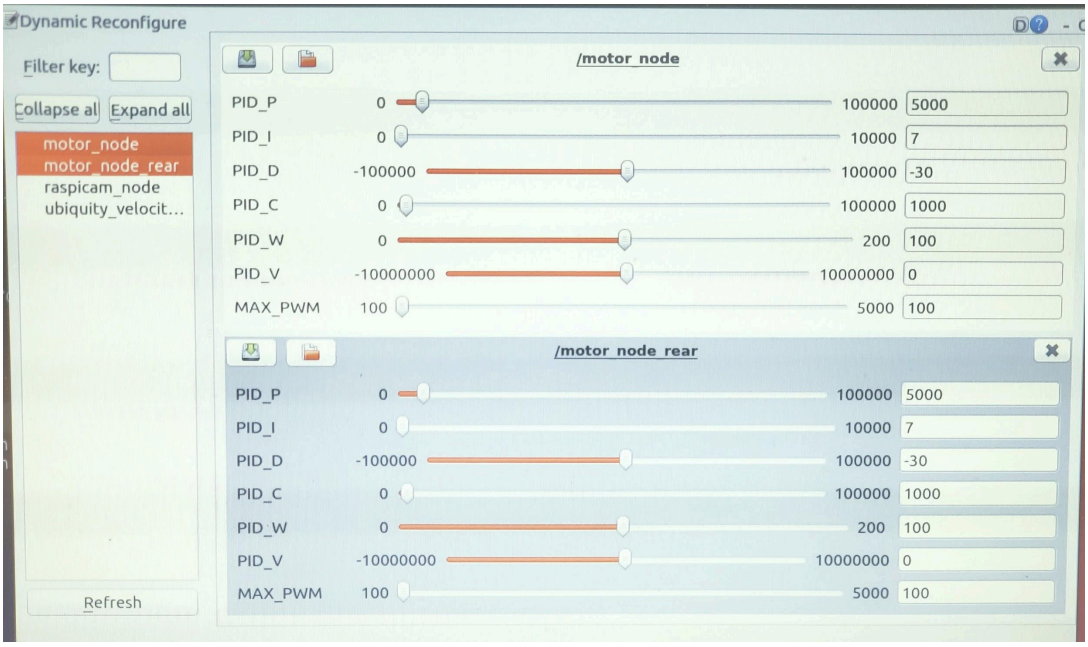
This should bring up the below program. You can then add the error topics by typing /left_error and clicking the plus sign, and then /right_error and clicking the plus sign.



Open another terminal, run:
export ROS_HOSTNAME=ubuntu-VirtualBox.local
export ROS_MASTER_URI=http://frontpi.local:11311

Then:

rosrun rqt_reconfigure rqt_reconfigure
Click on motor_node and on motor_node_rear on the left column to get to the pid settings
To set values you can move the sliders or manually type in values in the fields to the right.



You will see two sets of parameters. The frontpi parameters are labeled /motor_node and the rearpi parameters are labeled /motor_node_rear. Our system has 2 motor controllers in it - you should adjust the parameters for both front and rear simultaneously. So if the front Pi has an I value of 7 you should set the rear pi to have the same value.

You should find proper parameters using process explained in a few pages.

1b) Using the robot itself

Ok, so first off they need to plug in a wireless keyboard/mouse in to the raspberry pi on the robot.

i) Press esc on the keyboard to quit the touchscreen interface
ii) Use the mouse to navigate to start and click on it and then open lxterminal
iii) In the terminal type: **rosrun rqt_plot rqt_plot**
iv) Repeat step ii with a separate terminal and type: **rosrun rqt_reconfigure rqt_reconfigure**

You should get a similar set of windows to those described in 1b

1C) Using A Ubuntu Laptop

Set up a Ubuntu 16.04 laptop using a partition

Instructions for how to do this are contained here:

https://learn.ubiquityrobotics.com/workstation_setup

Specifically for the Ubuntu set up you can use this guide
https://ubuntu.com/tutorials/tutorial-install-ubuntu-desktop-1604#1-overview

Once you've done this follow the instructions in 1a) from ***Bringing up rqt_reconfigure and rqt_plot*** *in short the method is the same -*

In a terminal type: **rosrun rqt_plot rqt_plot**

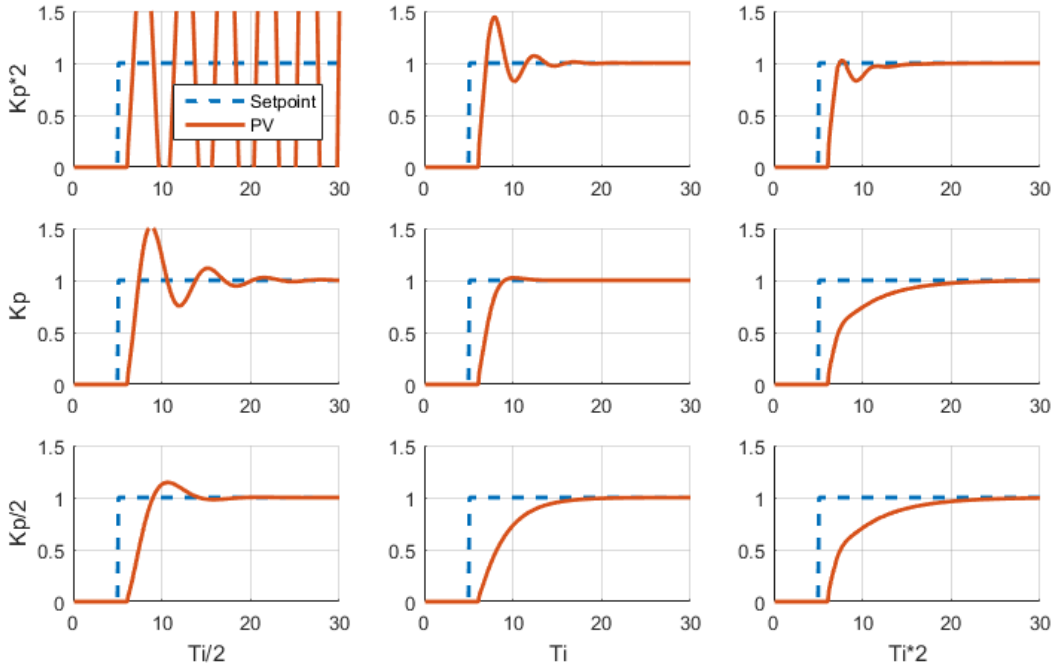
Open a separate terminal and type: **rosrun rqt_reconfigure rqt_reconfigure**

2) Set up to adjust parameters and view output

For more information on PID tuning see the following detailed guide.

https://www.pid-tuner.com/pid-control/

The short summary is that you are trying to get a set of parameters that gets close to the center of the following 9 possible choices:



The error **- which equals the difference between the blue line and the red line** is something that can be plotted using rqt_plot as described above. A well tuned PID-V loop should see a small error spike which quickly returns to close to zero without oscillations. Your goal is to get a set of parameters that minimizes the error in all situations - particularly if there is a large change in speed. Ideally you have a low error while also having no oscillations. To find the correct set of parameters use the following process.

While plotting the rqt_plot and looking at the output and with the robot in a normal driving condition (on the ground normally loaded) then:

i) Set I, D and V to zero, set P to a small value (a small value for p would be 1000) attempt to drive the robot
ii) If the error plotted shows oscillations reduce P until it doesn't show oscillations
iii) if the error shows no oscillations increase P until you can just start to see oscillations
iv) reduce P by about 1/3rd from where you can just start to see oscillations in the error term

You may find that this set of parameters is just fine for your robot - however its often possible to improve further if so carry on.

v) increase i slowly (start i at 1 then increase to 2 etc) until you can just see oscillations on the error

vi) increase the magnitude of d until those oscillations are damped away remember you may need to make d more negative (a typical value of d would be -30 and you might make it -15 or -40)

vi) repeat steps v) and vi) increasing i to see oscillations and increasing d to damp those oscillations away until you can't increase i or d any more without there being oscillations.

vii) back down i by 1/3rd and back down d until you can get stable performance without oscillations

viii) increase v to reduce the height of the maximum error (a small value for v is around 1000)

ix) if increasing v causes oscillations increase d to damp them out

x) continue with small steps until you get a stable performance without oscillations

As soon as you have a stable set of parameters note them down - if things go wrong you should go back to the last set of good parameters

xi) try making small variations on p, i, d an v to get a lower error and a shorter period where there is an error sometimes you can increase v and reduce i and d and you get better performance.

On Max PWM

Our motor controller has a safety limiter that limits the amount of current that can flow through the motors - this is known as the max PWM (pulse width modulation). By default this is set to 250. The new motors would be expected to need to increase this number in order to get adequate torque.

After step iii) above **Please check to see whether you are getting a max PWM error** if you are then you need to cautiously increase the max PWM until you no longer get the max PWM error.

You do this by opening another terminal (you may have several open already) using the terminal command

> **rostopic echo /diagnostics | grep -A 4 Limit**

If there is no problem you should see

values: [] for no errors

If you see something about max_pwm limit reached that implies that this is a problem and we will need to look at it further

3) Final testing and check out and troubleshooting

You may find that with highly tuned PID-V parameters the robot oscillates, vibrates or wobbles a little - this is the robot trying to find the zero point perfectly - some motion is inevitable. The small zero point vibrations are not harmful - but they can be annoying. In those circumstances reduce I, D and V until it stops - it can happen that you have to reduce these down to 0 that may be OK.

Once you have parameters its wise to check their performance on the set of maneuvers that you think the robot is likely to undertake. If in real life the robot will drive up ramps, then consider getting the robot to drive up ramps, etc.

After you find parameters that suit your needs you will on BOTH the frontpi and rearpi have to edit the base.yaml file on BOTH units and save the new parameters. As before log in to the robot and pick your favorite text editor for example:

> sudo vi /opt/ros/kinetic/share/magni_bringup/param/base.yaml

Or maybe

> sudo pico /opt/ros/kinetic/share/magni_bringup/param/base.yaml

Please also tell us what the parameters are and we will make them the default parameters in your distributions!!!!!!

Thanks