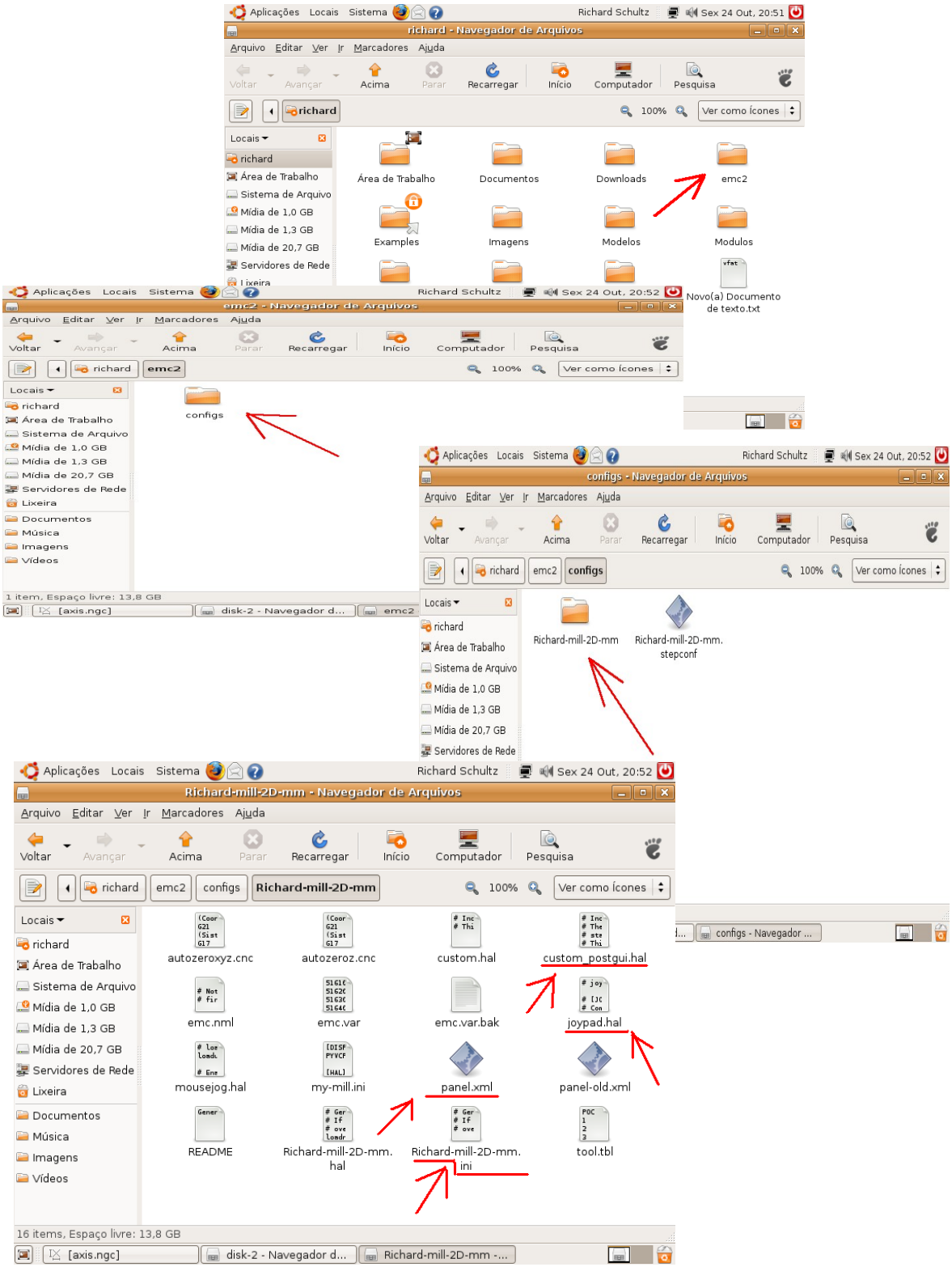
1º passo - Copiar os arquivos joypad.hal e panel.xml para o diretório onde se encontra o seu arquivo de configuração do emc (ex: //home/seudiretório/emc2/config/my-mill)



2º passo – Acrescentar no seu arquivo ini (ex:my-mill.ini) na secção [HAL] os seguintes comandos:
HALUI = halui
HALFILE = joypad.hal

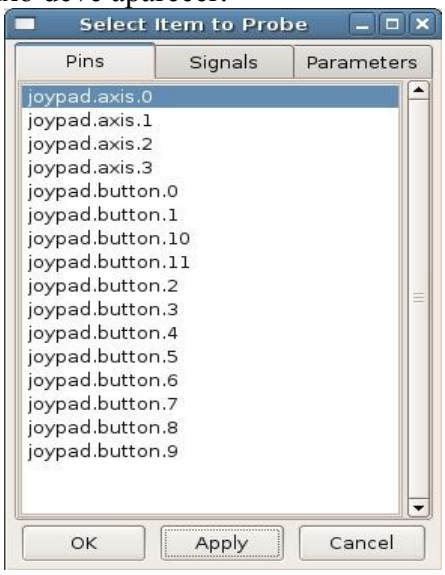
3º passo – Acrescentar uma secção chamada [HALUI] com os seguintes comandos:
#mdi-command-00 move a ferramenta para o zero em todos os eixos
MDI_COMMAND = G0 X0Y0Z0
#mdi-command-01 move o eixo de x para a posição zero
MDI_COMMAND = G0 X0
#mdi-command-02 move o eixo de y para a posição zero
MDI_COMMAND = G0 Y0
#mdi-command-03 move o eixo de z para a posição zero
MDI_COMMAND = G0 Z0
#mdi-command-04 envia os eixos para suas posições de home
MDI_COMMAND = G28
#mdi-command-05 coloca os eicos x e y em zero e move o eixo de z para 10 mm acima do zero
MDI_COMMAND = G0 X0Y0Z10
#mdi-command-06 coloca os eixos na posição que eu escolhi para troca de ferramenta
MDI_COMMAND = G0 X0Y0Z80
#mdi-command-07 move a minha máquina para a posição de limite máximo
MDI_COMMAND = G0 X539Y619
#mdi-command-08 altera o sistema de coordenadas para zeramento da ferramenta usando o meu
#dispositivo de zeramento
MDI_COMMAND = G92.3

4º passo - Acrescentar na secção chamada [DISPLAY] uma linha com o seguinte comando:
PYVCP = panel.xml

5º passo – Incluir no arquivo custom_postgui.hal (existente no diretório de config do emc) as seguintes linhas:
net remote-zero-x halui.mdi-command-01 <= pyvcp.zero-x-axis
net remote-zero-y halui.mdi-command-02 <= pyvcp.zero-y-axis
net remote-zero-z halui.mdi-command-03 <= pyvcp.zero-z-axis

6º passo – Para checar se seu joysitck funciona corretamente no emc faça o seguinte:
- Abra um terminal e digite: *halrun*
- Assumindo que o endereço do seu joysitck no emec seja /dev/input/js0, (normalmente este é o default se tiver somente um joystick) digite o seguinte comando para carregar os modulos do joystick: *halcmd: loadusr hal_joystick -d /dev/input/js0 -p joypad*
- Carregue o halmeter para checar os valores deos eixos: *halcmd: loadusr halmeter*

uma janela como o exemplo abaixo deve aparecer:



Selecione o primeiro eixo e clique em apply uma pequena janela com os valores para este eixo

_

se o valor for zero e você não estiver mexendo neste eixo esta tudo ok. Tente mover os botões do joystick para saber a qual se refere.

Atenção não esqueça de colocar o joystick no modo ANALOGICO. Senão não funciona corretamente.
Veja o exemplo do meu joystick:

7º passo – Feche o terminal.

Links originais para quem quer se aprofundar mais neste tipo de programação:
http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Using_A_Joypad_To_Move_Your_CNC_Machine
http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?HereIsHowToCheck

Abaixo estão todos os arquivos modificados para que o joystick funcione corretamente

joypad.hal

```
# joypad.hal -- hal configuration file to move a cnc machine using a joypad
# [JOG]
# Components
# We will use hal_joystick to read the axis value (float) for X Y Z, we will send these values to the
# speed pin of a sim-encoder component, for X Y Z, this component outputs Phase-A and Phase-B signal,
# just like a real quadrature rotary encoder. We will decode those signals with an encoder component for X Y Z
# and will send the result counts value to the axis jog pin for X Y Z.
# Load the hal_joystick component that creates joypad.axis.<n> and joypad.button.<n> pins
loadusr hal_joystick -d /dev/input/js0 -p joypad
# Load three encoder and three sim_encoder components
loadrt encoder num_chan=3
loadrt sim_encoder num_chan=3
# Create links between the axis pins and the speed pin of the sim-encoder for X Y Z
net velX joypad.axis.0 => sim-encoder.0.speed
net velY joypad.axis.1 => sim-encoder.1.speed
net velZ joypad.axis.2 => sim-encoder.2.speed
# Create links between sim-encoder Phase-A and Phase-B and encoder Phase-A and Phase-B for X Y Z
net XA sim-encoder.0.phase-A => encoder.0.phase-A
net XB sim-encoder.0.phase-B => encoder.0.phase-B
net YA sim-encoder.1.phase-A => encoder.1.phase-A
net YB sim-encoder.1.phase-B => encoder.1.phase-B
net ZA sim-encoder.2.phase-A => encoder.2.phase-A
net ZB sim-encoder.2.phase-B => encoder.2.phase-B
# Create links between encoder counts and jog counts for X Y Z
net countX encoder.0.counts => axis.0.jog-counts
net countY encoder.1.counts => axis.1.jog-counts
net countZ encoder.2.counts => axis.2.jog-counts
# Set parameter values
setp encoder.0.position-scale          1
setp encoder.0.x4-mode         TRUE
setp encoder.1.position-scale          1
setp encoder.1.x4-mode         TRUE
setp encoder.2.position-scale          1
setp encoder.2.x4-mode         TRUE
setp encoder.capture-position.tmax          0
setp encoder.update-counters.tmax          0
setp sim-encoder.0.ppr     00000064
```

```
setp sim-encoder.0.scale           1
setp sim-encoder.1.ppr     00000064
setp sim-encoder.1.scale           -1
setp sim-encoder.2.ppr     00000064
setp sim-encoder.2.scale           -1
setp sim-encoder.make-pulses.tmax          0
setp sim-encoder.update-speed.tmax          0
# Enable jog for X Y Z
setp axis.0.jog-enable TRUE
setp axis.1.jog-enable TRUE
setp axis.2.jog-enable TRUE
# Attach realtime functions to threads
addf encoder.capture-position servo-thread
addf sim-encoder.update-speed servo-thread
addf encoder.update-counters base-thread
addf sim-encoder.make-pulses base-thread
# [BUTTON-SAMPLES]
# Here are two examples on how to attach some functions to joypad buttons. We will use Halui pins for the
# second example.
# Scale button
# we set two buttons (6 and 4) to choose the jogscale value. Pressing button 6 will set the scale to 0.01
# while pressing button 4 will set it to 0.1.
# Components
# We will use a two values selector and a flipflop component
loadrt mux2
loadrt flipflop
# Link between buttons and flipflop, flipflop will output TRUE when rising edge is detected on set pin, FALSE
# when rising edge is on reset pin.
net button4 joypad.button.4 => flipflop.0.reset
net button6 joypad.button.6 => flipflop.0.set
# Link between flipflop and mux2, mux2 will output value mux2.0.in0 when mux2.0.sel is FALSE and mux2.0.in1
# when TRUE.
net selected flipflop.0.out => mux2.0.sel
# Link between the mux2 output and the jogscale pin for X Y Z
net jogscale mux2.0.out => axis.0.jog-scale
net jogscale mux2.0.out => axis.1.jog-scale
net jogscale mux2.0.out => axis.2.jog-scale
# Set parameters values
setp flipflop.0.tmax        3750
setp mux2.0.tmax        3601
# Set the two scale values
setp mux2.0.in0         0.1
setp mux2.0.in1         0.01
# Attach realtime functions to threads
addf flipflop.0 servo-thread
addf mux2.0 servo-thread
# Flood button
# We will set a single button (button 7) to start and stop flood. We will use Halui pins for that.
```

```
# Components
# We will use simply two and2 and 1 not components
#loadrt and2 count=2
#loadrt not
# Flood-is-on halui pin is linked to the and2.0.in0 and the not-flood-is-on, generated using the not component
# is linked to the and2.1.in0. So, if the flood is on, we will have and2.0.in0 TRUE and and2.1.in0 FALSE.
#net flood-is-on halui.flood.is-on => and2.0.in0
#net flood-is-on halui.flood.is-on => not.0.in
#net not-flood-is-on not.0.out => and2.1.in0
# Link between button 7 and and.0.in1 and and.1.in1. In this way, if the flood for example is on, when the
# button is pressed TRUE will be sent to and2.0.in1 and and2.1.in1, while the in0 value for and2 components
# will be TRUE for the first and2 and FALSE for the second. So the first and2 will output TRUE.
#net button7 joypad.button.7 => and2.0.in1
#net button7 joypad.button.7 => and2.1.in1
# Link between and2 outputs and halui pin flood on and off. So, as seen above, if the flood is on, the and2.0
# will output TRUE and the flood will turn off.
#net floodOff and2.0.out => halui.flood.off
#net floodOn and2.1.out => halui.flood.on
# Attach realtime functions to threads
#addf and2.0 servo-thread
#addf and2.1 servo-thread
#addf not.0 servo-thread
#Exemplo para a programação dos meus botões
#No botão zero do joystick executa G28
net remote-home-all halui.mdi-command-04 <= joypad.button.0
#No botão hum do joystick executa G0 X0Y0Z10
net remote-tool-home halui.mdi-command-05 <= joypad.button.1
#No botão dois do joystick executa G0 X0Y0Z0
net remote-zero-all halui.mdi-command-00 <= joypad.button.2
#No botão três do joystick executa G0 X0Y0Z80
net remote-tool-change halui.mdi-command-06 <= joypad.button.3
#No botão zero do joystick executa a continuação do código paralizado
net remote-resume halui.program.resume <= joypad.button.5
#No botão zero do joystick executa a paralização do código em execução
net remote-stop halui.program.pause <= joypad.button.7
#No botão zero do joystick executa G92.3 para alterar o sistema de coordenadas do meu emc
net remote-tool-autozero halui.mdi-command-08 <= joypad.button.8
#No botão zero do joystick executa o código-G carregado no emc
net remote-start halui.program.run halui.mode.auto <= joypad.button.9
```

Meu arquivo de configuração

```
# Generated by stepconf at Thu May  1 14:28:21 2008
# If you make changes to this file, they will be
# overwritten when you run stepconf again

[EMC]
MACHINE = Richard-mill-2D-mm
```

```
NML_FILE = emc.nml
DEBUG = 0

[DISPLAY]
DISPLAY = axis
POSITION_OFFSET = RELATIVE
POSITION_FEEDBACK = ACTUAL
MAX_FEED_OVERRIDE = 1.2
INTRO_GRAPHIC = emc2.gif
INTRO_TIME = 5
PROGRAM_PREFIX = /home/richard/emc2/nc_files
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm .005mm
PYVCP = panel.xml

[TASK]
TASK = milltask
CYCLE_TIME = 0.010

[RS274NGC]
PARAMETER_FILE = emc.var

[EMCMOT]
EMCMOT = motmod
SHMEM_KEY = 111
COMM_TIMEOUT = 1.0
COMM_WAIT = 0.010
BASE_PERIOD = 100000
SERVO_PERIOD = 1000000

[HAL]
HALFILE = Richard-mill-2D-mm.hal
HALFILE = custom.hal
POSTGUI_HALFILE = custom_postgui.hal
HALUI = halui
HALFILE = joypad.hal

[HALUI]
MDI_COMMAND = G0 X0Y0Z0
MDI_COMMAND = G0 X0
MDI_COMMAND = G0 Y0
MDI_COMMAND = G0 Z0
MDI_COMMAND = G28
MDI_COMMAND = G0 X0Y0Z10
MDI_COMMAND = G0 X0Y0Z80
MDI_COMMAND = G0 X539Y619
MDI_COMMAND = G92.3

[TRAJ]
AXES = 3
COORDINATES = X Y Z
LINEAR_UNITS = mm
ANGULAR_UNITS = degree
```

```
CYCLE_TIME = 0.010
DEFAULT_VELOCITY = 12.00
MAX_LINEAR_VELOCITY = 30.00

[EMCIO]
EMCIO = io
CYCLE_TIME = 0.100
TOOL_TABLE = tool.tbl

[AXIS_0]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 25.0
MAX_ACCELERATION = 700.0
STEPGEN_MAXACCEL = 707.5
SCALE = 188.976377952
FERROR = 1
MIN_FERROR = .25
MIN_LIMIT = -0.001
MAX_LIMIT = 540.0
HOME_OFFSET = 0.000000
HOME_SEARCH_VEL = -6.000000
HOME_LATCH_VEL = 2.645833
HOME_IGNORE_LIMITS = YES
HOME_SEQUENCE = 1

[AXIS_1]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 25.0
MAX_ACCELERATION = 700.0
STEPGEN_MAXACCEL = 707.5
SCALE = 188.976377952
FERROR = 1
MIN_FERROR = .25
MIN_LIMIT = -0.001
MAX_LIMIT = 620.0
HOME_OFFSET = 0.000000
HOME_SEARCH_VEL = -6.000000
HOME_LATCH_VEL = 2.645833
HOME_IGNORE_LIMITS = YES
HOME_SEQUENCE = 2

[AXIS_2]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 20.0
MAX_ACCELERATION = 1000.0
STEPGEN_MAXACCEL = 1007.5
SCALE = 188.976377952
FERROR = 1
MIN_FERROR = .25
```

```
MIN_LIMIT = -210.0
MAX_LIMIT = 0.001
HOME_OFFSET = 0.000000
HOME_SEARCH_VEL = 6.000000
HOME_LATCH_VEL = -2.645833
HOME_IGNORE_LIMITS = YES
HOME_SEQUENCE = 0

[FILTER]
PROGRAM_EXTENSION = .bmp,.png,.gif,.jpg Imagens
bmp = image-to-gcode
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
PROGRAM_EXTENSION = .cnc,.nc Arquivos Gcode
```

custom_postgui.hal

```
# Include your customized HAL commands here
# The commands in this file are run after the AXIS GUI (including PyVCP panel)
# starts
# This file will not be overwritten when you run stepconf again
net remote-zero-x halui.mdi-command-01 <= pyvcp.zero-x-axis
net remote-zero-y halui.mdi-command-02 <= pyvcp.zero-y-axis
net remote-zero-z halui.mdi-command-03 <= pyvcp.zero-z-axis
```

panel.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!--
Include your PyVCP panel here.
The contents of this file will not be overwritten when you run stepconf again.
-->
<pyvcp>
	<vbox>
	<labelframe text="Mover p/Zero">
	<button>
		<text>" X "</text>
		<font>('Helvetica',12)</font>
		<halpin>"zero-x-axis"</halpin>
	</button>
	<button>
		<text>" Y "</text>
		<font>('Helvetica',12)</font>
		<halpin>"zero-y-axis"</halpin>
	</button>
	<button>
		<text>" Z "</text>
		<font>('Helvetica',12)</font>
		<halpin>"zero-z-axis"</halpin>
	</button>
```

```
        </labelframe>
        </vbox>
</pyvcp>
```

Como podem notar os arquivos acima são apenas exemplos que usei para configurar a minha máquina. Na sua máquina você tem que adaptar os arquivos atualmente em uso para conter estas modificações.

Atenção:

Não me responsabilizo por qualquer dano causado a seu equipamento ou máquina pelo uso das informações aqui demonstradas. USE POR SUA CONTA E RISCO!!!

Tela do meu Emc² depois das alterações