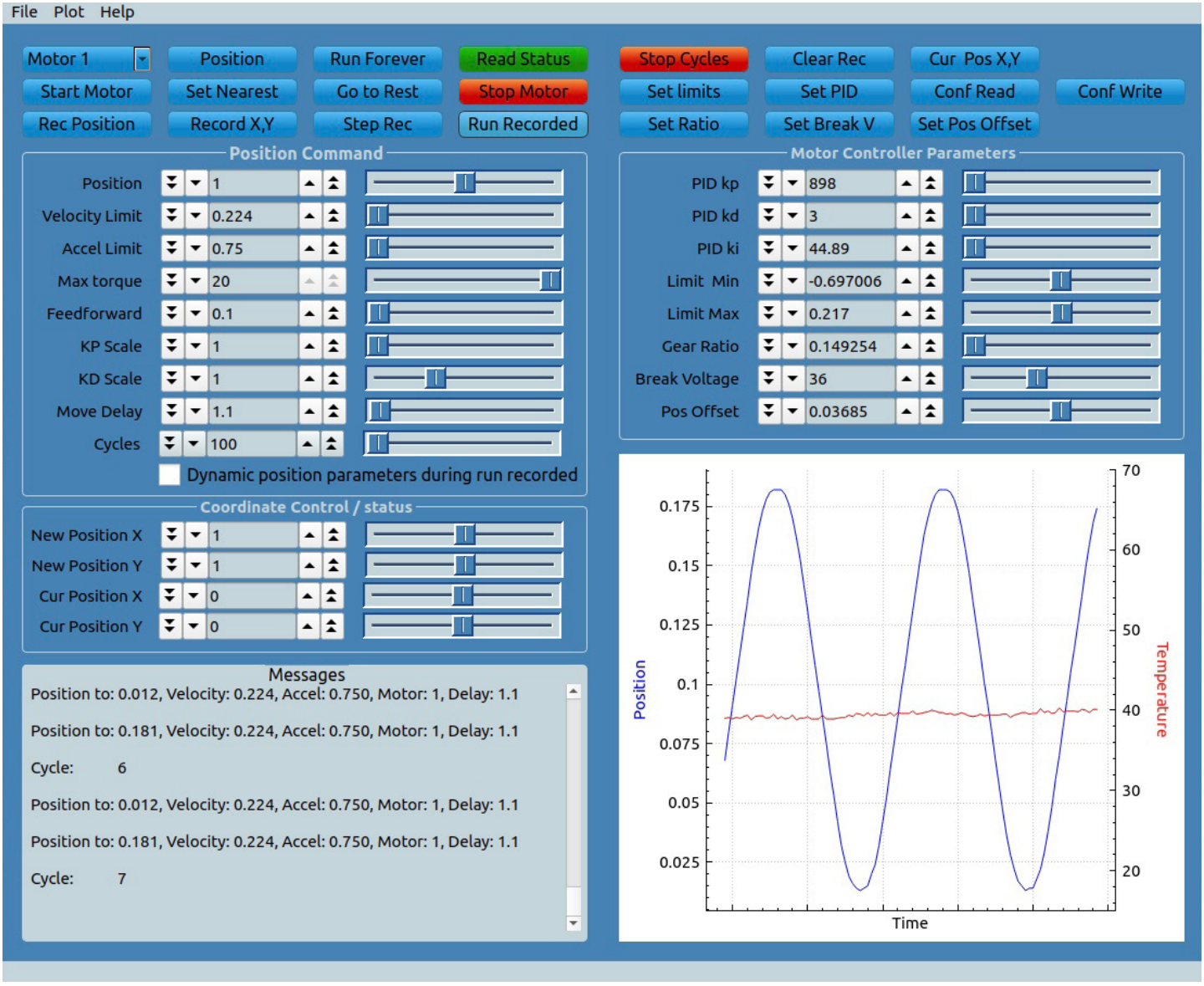
Moteus GUI Application

Instruction Manual



Contents

Introduction 3
Installation 3
Using the program 5
Buttons 5
Running a sequence of motor movements 6
Customizing the program 7
Defining motor rest positions 7
Defining Power on position 7
Defining Number_of_Motors 7
Collision detection 7

Introduction

This is a GUI Application which can be used to control individual moteus motor controllers or create a sequence of movements for multiple motors.

Parameters for each position command can be changed on the screen with direct entry or using sliders.

Individual motors may also be commanded using position, status or stop buttons.

It is also possible to position to an X,Y coordinate using Inverse Kinematics. It is also possible to display the current X,Y using forward Kinematics.

A sequence of position commands for multiple motors can be recorded and then run once or in cycles.

Movements on different motors can be safely overlapped to make things less robotic. This is accomplished through the use of movement delays which can be as short as 0. If the same motor is accessed the software waits for the motor to finish before issuing the next command.

For example, you can manually move a motor to a position and press the Record Position button. The parameters on the screen will be recorded for that motor. This can be repeated for one or multiple motors.

Then you can press run recorded and the sequence will be executed in any number of cycles.

While running; if the dynamic check box is set, the parameters on the screen will be used instead of the recorded parameters. This allows the tuning of such things as Velocity Limit or Acceleration Limit to see how it can affect the movements.

The sequence can be saved to a text file, and later reloaded. The file can be edited to alter the sequence.

A plot dynamically shows position, velocity, torque, temperature, or phase current feedback for a selected motor. Any two may be displayed at the same time.

Various moteus controller parameters are displayed and may be changed and sent to the controller. These parameters can also be permanently saved to the controller.

Installation

If the fdcanusb was not installed, Follow the instructions at: https://github.com/mjbots/fdcanusb/blob/master/70-fdcanusb.rules

This consists of doing the following;

copy file 70-fdcanusb.rules to /etc/udev/rules.d folder Then run the following from a terminal:

sudo udevadm control --reload-rules

sudo udevadm trigger --subsystem-match=tty

This program needs to be opened the first time with qt creator so it can be built for your system. After running the program once, a desktop shortcut may be created to run the program after that. The executable file is in a folder qt creates called build-MotorQT_threaded-Desktop-Debug. The executable file is called MotorQT_

threaded.

If you don’t have qt creator it can be installed as follows;

Install Qt on Ubuntu https://newbedev.com/install-qt-on-ubuntu

How to Install libqwt-qt5-dev in Ubuntu 18.04 https://www.howtoinstall.me/ubuntu/18-04/libqwt-qt5-dev/

or How to Install qt on Ubuntu 20.04 https://linuxways.net/ubuntu/how-to-install-qt-on-ubuntu-20-04/

This code uses c++23 features such as std::print, which is now available. Refer to installing c++ 14 below if it is not already installed on your system.

At the terminal enter

qtcreator

click open project

find MotorQT_threaded.pro

click open

click Configure Project

from the menu

select Build then Build All

Make sure the fdcanusb is plugged in to the motor controller and the USB.

when the build is done select from the menu,

Build Run, or hit the Play icon on the left side.

Installing g++14

If you don’t have **g++14** it can be installed as described in one of two methods.
Method 1, install using apt.
sudo add-apt-repository ppa:ubuntu-toolchain-r/test

sudo apt-get update

sudo apt install gcc-14 gcc-14-base gcc-14-doc g++-14

sudo apt install libstdc++-14-dev libstdc++-14-doc

Method 2, install from sources.
If above doesent work you may install from the sources.

This is described in the following link;
https://www.dedicatedcore.com/blog/install-gcc-compiler-ubuntu/

This may also be helpful.
https://stackoverflow.com/questions/76292505/what-am-i-supposed-to-specify-for-host-to-cross-compile-gcc-from-x86-64-for

This is procedure is summarized as follows;

```
sudo apt update
sudo apt install build-essential
sudo apt install software-properties-common
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt update
sudo apt install libmpfr-dev libgmp3-dev libmpc-dev -y
wget http://ftp.gnu.org/gnu/gcc/gcc-14.2.0/gcc-14.2.0.tar.gz
tar -xf gcc-14.2.0.tar.gz
cd gcc-14.2.0
./configure -v --build=aarch64-linux-gnu --host=aarch64-linux-gnu --target=aarch64-linux-gnu --prefix=/usr/local/gcc-14.2.0 --enable-checking=release --enable-languages=c,c++ --disable-multilib --program-suffix=-14.2.0

make -j3
sudo make install

Libraries have been installed in:
   /usr/local/gcc-14.2.0/lib/../lib64

add to .bashrc
export PATH=/usr/local/gcc-14.2.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/gcc-14.2.0/lib/gcc/aarch64-linux-gnu/14.2.0:$LD_LIBRARY_PATH
```

Using the program

Once running the Motor Control section contains Parameters which can be changed using the counter or slider.

Typing into the counter is the easiest way to precisely change a value.

Buttons

Use the buttons to command the controller.

The Motor combo box allows the selection of a motor ID

The Position button will move to the indicated position.

Read Status will return the status, Stop Motor will stop the selected motor.

Run Forever will run until a motor position limit is reached, or forever.

Limit Max, Limit Min, PID kp. PID kd, and PID ki are parameters on the GUI screen which display moteus controller parameters for the selected motor. These can be edited and sent to the controller with the Write Limits and Write PID buttons.

Write limits will send the Limit Min and Limit Max from the GUI to the moteus controller for the selected motor.

Write PID will send the PID Parameters from the GUI to the moteus controller for the selected motor.

Conf Write will permanently save any parameters sent to the moteus controller for the selected motor. Note the selected motor should be stopped prior to sending this command.

Start Motor will will start the motor at its current position.

Set nearest causes the servo to select a whole number of internal motor rotations so that the final position is as close to 0 as possible.

Running a sequence of motor movements

Stop motors so they can be manually moved.

Select a motor ID, and set the screen parameters as desired.

Move a motor manually to the position desired.

Make sure the Move Delay is set large enough for desired the move to complete. If the delay is less the movement, the software will wait for the movement to complete the next time the motor is used.

Movements on different motors may be overlapped by setting the movement delay to zero or a short delay on intermediate steps.

Click the ‘Rec Position’ button to add the position and all parameters to the list.

Repeat this procedure for as many steps as desired.

When done click the ‘Run Recorded’ button and see if it acts as expected.

The ‘Step Rec’ button may be used instead of Run Recorded to step through the recorded positions one at a time.

Use Clear Rec if desired to erase the recorded list.

Click Stop Cycles to stop cycling through the recorded positions.

The Dynamic check box allows the screen parameters to override some recorded parameters for the selected motor. This is useful when tunning movements with different velocities, accelerations for example. Velocity Limit, Accel Limit, Max torque, Feedforward, KP Scale, and KD Scale can be dynamic.

If desired you may save the Recorded list to a text file using the File Save in the menu.

You can use File Open to read in a previously saved recorded list.

You can edit the text file to change parameters or delete an entire move.

Be careful not to remove part of a sequence. It is OK to remove the whole sequence including the sequence number and all parameters. The sequence number value is ignored but they must be there. The sequence is executed from top to bottom.

The Record X Y button may be used instead of the Record Position button. In this case the New Position X parameter and the New Position Y parameter will be used to calculate moves for motor 1 and motor 2, and add these positions to the record list. Note that this must be enabled with the Collision_Check_enable as discussed in the Collision detection section below.

The currentXY button may be used to determine the X, Y for the current motor 1 and motor 2 positions. These values will be displayed on the screen in Cur Position X and Cur Position Y. Note that this must be enabled with the Collision_Check_enable as discused in the Collision detection section below.

Customizing the program

Defining motor rest positions

mainwindow.h contains an array called Motor_rest_position. The first entry in the array is motor 1.

This array contains motor positions to be used when the Go to Rest button is pressed.

Defining Power on position

mainwindow.h contains an array called nearest_offset. The first entry in the array is motor 1.

This contains motor positions where the motors will be when powering up. During power on the Moteus controller calculates the encoder position to the nearest half turn. This offset is used by a nearest command to correctly find the position at Power on.

The offset should be relative to where the 0 which was set with the moteus command “moteus.moteus_tool --target M --zero-offset” , where M refers to the motor number.

Defining Number_of_Motors

mainwindow.h contains a parameter called Number_of_Motors, which should be set to the number of motors you have.

Collision detection

motorworker.h contains parameters used for collision detection for a two segment arm system.

The program will avoid position moves which are restricted in an x,y coordinate system, or by rotation limitations set in the controller. Edit the parameters to suit your needs.

Collision_Check_enable must be set to true to enable collision detection. The function is disabled by default because the program assumes two motors are setup as follows;

Motor 1 is the origin connected to arm 1.

Motor 2 is connected between arm 1 and arm 2.

The end of arm 2 farthest from the motor is the X and Y coordinate.

(origin) m1-------M2--------X,Y

bool Collision_Check_enable = false; // enables forward and reverse kinamatics checks

L1 = 4.7; // Length of arm 1 between motor 1 (origin) and motor 2

L2 = 8.0; // Length of arm 2 between motor 2 and end position

min_Y = -8.2; // This is the minimum y position (relative to the origin)

min_Pos_X = 1.5; // This is the minimum positive X closest to the origin

min_Neg_X = -1.5; // This is the minimum negative X closest to the origin

inner_radius = L1 - L2; // This is the unreachable inner radius around the origin of X,Y

Motor2_rotation_limit = 0.349943; // This is + or - limit arm 2 can rotate without hitting something.