README

Perry Kundert

December 22, 2014

Contents

1 SMC Actuator Position Control via Modbus/RTU

The `cpppo_positioner` module allows control of the position of a set of actuators by initiating a communication channel and issuing new position directives via each actuator controller. The current state is polled as necessary via Modbus/RTU reads, and data updates and state changes are performed via Modbus/RTU writes.

1.1 Installing

Clone the repository, and run the setup.py installer:

```
$ git clone git@github.com:pjkundert/cpppo_positioner.git
$ cd cpppo_positioner
$ python setup.py install
$ python
Python 2.7.6 (default, Sep  9 2014, 15:04:36)
```

```
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cpppo_positioner
>>>
```

1.2 Positioning

A Python API is provided to implement positioning control.

1.2.1 `smc.smc_modbus`

This class is the gateway for accessing multiple SMC positioning actuators connected via RS-485 serial. The serial port parameters are `/dev/ttyS1`, 38400 Baud, 8 bits, 1 stop, no parity, and a .25s poll rate. These can all be specified as keyword arguments. See `cpppo_positioner/smc.py` for details.

```
from cpppo_positioner import smc
gateway          = smc.smc_modbus()
```

1.2.2 `.position` – Complete operation, Initiate new position

The .position method checks that any current position operation is complete, and then sends any new position data, starting the new position operation. If no new data is provided (eg. only `actuator` and/or `timeout` provided), then only the operation completion is checked; no new positioning operation is initiated.

```
gateway.position( actuator=1, timeout=10.0, position=12345, speed=100, ... )
```

The full set of positioning parameters defined by the SMC actuator is:

keyword	units	description
movement_mode		1: absolute, 2: relative
speed	mm/s	1-65535
position	.01 mm	+/-2147483647
acceleration	mm/s^2	1-65535
deceleration	mm/s^2	1-65535
pushing_force	%	0-100
trigger_level	%	0-100
pushing_speed	mm/s	1-65535
moving_force	%	0-300
area_1	.01 mm	+/-2147483647
area_2	.01 mm	+/-2147483647
in_position	.01 mm	1-2147483647

It is recommended to specify all the values at least for the initial positioning; any values not specified in subsequent position calls will not be changed.

1.2.3 `.status` – Return full status and position data

Returns the current complete set of status and data values for the actuator. If any value has not yet been polled, it will be `None`. Here is an example (formatted for readability)

```
>>> gateway.status( actuator=1 )
{
    "X40_OUT0": false,
    "X41_OUT1": false,
    "X42_OUT2": false,
    "X43_OUT3": false,
    "X44_OUT4": false,
    "X45_OUT5": false,
    "X48_BUSY": false,
    "X49_SVRE": false,
    "X4A_SETON": false,
    "X4B_INP": false,
    "X4C_AREA": false,
    "X4D_WAREA": false,
    "X4E_ESTOP": false,
    "X4F_ALARM": false,
    "Y10_IN0": false,
    "Y11_IN1": false,
    "Y12_IN2": false,
    "Y13_IN3": false,
    "Y14_IN4": false,
    "Y15_IN5": false,
    "Y18_HOLD": false,
    "Y19_SVON": false,
    "Y1A_DRIVE": false,
    "Y1B_RESET": false,
    "Y1C_SETUP": false,
    "Y1D_JOG_MINUS": false,
    "Y1E_JOG_PLUS": false,
    "Y30_INPUT_INVALID": false,
    "acceleration": 0,
    "area_1": 0,
    "area_2": 0,
    "current_position": 0,
    "current_speed": 0,
    "current_thrust": 0,
    "deceleration": 0,
    "driving_data_no": 0,
    "in_position": 0,
    "movement_mode": 0,
    "moving_force": 0,
    "operation_start": 0,
    "position": 0,
    "pushing_force": 0,
    "pushing_speed": 0,
    "speed": 0,
    "target_position": 0,
    "trigger_level": 0
}
```

1.2.4 Command- or Pipe-line usage

An executable module entry point (`python -m cpppo_positioner`), and a convenience executable script (`cpppo_positioner`) are supplied.

If your application generates a stream of actuator position data, or if you have some manual positions you wish to move to, you can use the command-line interface. You may supply one or more actuator positions in blobs of JSON data (an actual position would have more entries, such as `acceleration`, `deceleration`, `timeout`, …):

```
$ position='{ "actuator": 0, "position": 12345, "speed": 100 }'
```

These positions may be supplied either as single parameters on the command line, or as separate lines of input (if standard input is selected, by supplying a '-' option):

```
$ python -m cpppo_positioner --address gateway -v "$position"
$ echo "$position" | cpppo_positioner --address gateway -v -
```

- Quoting double-quotes on Windows Powershell
  Note that on Windows Cmd or Powershell, it is very difficult to quote

double-quote characters in strings. In Powershell, you need to use the bash-slash + back-tick before each double-quote. Unexpectedly, using a single-quoted string does **not** allow you to contain double-quotes.

You can get double quotes into a string:

```
PS > $position = ’{ "actuator": 0, "position": 12345, "speed": 100 }’
PS > $position
’{ "actuator": 0, "position": 12345, "speed": 100 }’
PS >
```

However, when you try to use them, they are re-interpreted on inclusion in a command:

```
PS > python -m cpppo_positioner --address gateway -v "$position"
PS > python -m cpppo_positioner -v "$position"
... Invalid position data: { actuator: 0, position: 12345, speed: 100 };
    Expecting property name: line 1 column 3 (char 2)
```

So, the only way to do this is to use the strange back-slash + back-tick double-escape, directly as a command-line argument:

```
PS > python -m cpppo_positioner --address gateway -v ’{ \‘"actuator\‘": 0,
```

Recommendation: use Linux or Mac, or install Cygwin and use bash on Windows. Trust me; this is just the tip of the iceberg…

1.3 SMC Gateway Simulator

A basic simulator of some of the Modbus/RTU I/O behaviour of an SMC actuator is implemented for testing purposes.

Ensure that either you have installed the $\text{cpppo}_{\text{positioner}}$, **or** are in the directory containing the cloned $\text{cpppo}_{\text{positioner}}$ repository): To simulate an SMC positioning actuator 1 on /dev/ttyS0:

```
$ python -m cpppo_positioner.simulator -v /dev/ttyS0 1
```