pyFirmata Documentation

Release 0.9.5

Tino de Bruijn

November 18, 2014

Contents

Module reference:

CHAPTER 1

pyFirmata

class pyfirmata.pyfirmata.**Board**(*port*, *layout*, *baudrate=57600*, *name=None*)

The Base class for any board.

add_cmd_handler(*cmd*, *func*)

Adds a command handler for a command.

exit()

Call this to exit cleanly.

get_firmata_version()

Returns a version tuple (major, minor) for the firmata firmware on the board.

get_pin(*pin_def*)

Returns the activated pin given by the pin definition. May raise an `InvalidPinDefError` or a `PinAlreadyTakenError`.

Parameters **pin_def** – Pin definition as described below, but without the arduino name. So for example `a:1:i`.

'a' analog pin Pin number 'i' for input 'd' digital pin Pin number 'o' for output

'p' for pwm (Pulse-width modulation)

All seperated by `:`.

iterate()

Reads and handles data from the microcontroller over the serial port. This method should be called in a main loop or in an `Iterator` instance to keep this boards pin values up to date.

pass_time(*t*)

Non-blocking time-out for `t` seconds.

send_sysex(*sysex_cmd*, *data=[]*)

Sends a SysEx msg.

Parameters

- **sysex_cmd** – A sysex command byte
- **data** – A list of 7-bit bytes of arbitrary data (bytes may be already converted to chr's)

servo_config(*pin*, *min_pulse=544*, *max_pulse=2400*, *angle=0*)

Configure a pin as servo with min_pulse, max_pulse and first angle. `min_pulse` and `max_pulse` default to the arduino defaults.

setup_layout(*board_layout*)
Setup the Pin instances based on the given board layout. Maybe it will be possible to do this automatically in the future, by polling the board for its type.

class pyfirmata.pyfirmata.**Pin**(*board*, *pin_number*, *type=2*, *port=None*)
A Pin representation

disable_reporting()
Disable the reporting of an input pin.

enable_reporting()
Set an input pin to report values.

mode
Mode of operation for the pin. Can be one of the pin modes: INPUT, OUTPUT, ANALOG, PWM. or SERVO (or UNAVAILABLE).

read()
Returns the output value of the pin. This value is updated by the boards Board.iterate() method. Value is always in the range from 0.0 to 1.0.

write(*value*)
Output a voltage from the pin

Parameters **value** – Uses value as a boolean if the pin is in output mode, or expects a float from 0 to 1 if the pin is in PWM mode. If the pin is in SERVO the value should be in degrees.

class pyfirmata.pyfirmata.**Port**(*board*, *port_number*, *num_pins=8*)
An 8-bit port on the board.

disable_reporting()
Disable the reporting of the port.

enable_reporting()
Enable reporting of values for the whole port.

write()
Set the output pins of the port to the correct state.

CHAPTER 2

Installation

The preferred way to install is with pip:

```
pip install pyfirmata
```

If you install from source with `python setup.py install`, don't forget to install `pyserial` as well.

CHAPTER 3

Usage

Basic usage:

```
>>> from pyfirmata import Arduino, util
>>> board = Arduino('/dev/tty.usbserial-A6008rIF')
>>> board.digital[13].write(1)
```

To use analog ports, it is probably handy to start an iterator thread. Otherwise the board will keep sending data to your serial, until it overflows:

```
>>> it = util.Iterator(board)
>>> it.start()
>>> board.analog[0].enable_reporting()
>>> board.analog[0].read()
0.661440304938
```

If you use a pin more often, it can be worth it to use the `get_pin` method of the board. It let's you specify what pin you need by a string, composed of 'a' or 'd' (depending on wether you need an analog or digital pin), the pin number, and the mode ('i' for input, 'o' for output, 'p' for pwm). All seperated by `:`. Eg. `a:0:i` for analog 0 as input, or `d:3:p` for digital pin 3 as pwm.:

```
>>> analog_0 = board.get_pin('a:0:i')
>>> analog_0.read()
0.661440304938
>>> pin3 = board.get_pin('d:3:p')
>>> pin3.write(0.6)
```

CHAPTER 4

Board layout

If you want to use a board with a different layout than the standard Arduino, or the Arduino Mega (for wich there exist the shortcut classes `pyfirmata.Arduino` and `pyfirmata.ArduinoMega`), instantiate the Board class with a dictionary as the `layout` argument. This is the layout dict for the Mega for example:

```
>>> mega = {
...           'digital' : tuple(x for x in range(54)),
...           'analog' : tuple(x for x in range(16)),
...           'pwm' : tuple(x for x in range(2,14)),
...           'use_ports' : True,
...           'disabled' : (0, 1, 14, 15) # Rx, Tx, Crystal
...           }
```

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

p

A

B

D

E

G

I

M

P

R

S

W