pyFirmata Documentation

Release 1.0.0

Tino de Bruijn

Mar 29, 2022

CONTENTS

Module reference:

CHAPTER

ONE

PYFIRMATA

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

The Base class for any board.

add_cmd_handler(*cmd*, *func*)

Adds a command handler for a command.

auto_setup()

Automatic setup based on Firmata's "Capability Query"

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

: arg data: a bytearray of 7-bit bytes of arbitrary data

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.

exception pyfirmata.pyfirmata.**InvalidPinDefError**

exception pyfirmata.pyfirmata.**NoInputWarning**

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.

property mode

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

ping(*trigger_mode=1*, *trigger_duration=10*, *echo_timeout=65000*)

Trigger the pin and wait for a pulseIn echo.

Used with HC-SR04 ultrasonic ranging sensors (see http://www.micropik.com/PDF/HCSR04.pdf).

Note: Requires pulseIn compatible Firmata in the arduino board (see https://github.com/jgautier/arduino-1/tree/pulseIn).

Parameters

- **trigger_mode** – Uses value as a boolean, 0 to trigger LOW, 1 to trigger HIGH (default, for HC-SR04 modules).
- **trigger_duration** – Duration (us) for the trigger signal.
- **echo_timeout** – Time (us) to wait for the echo (pulseIn timeout).

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.

exception pyfirmata.pyfirmata.**PinAlreadyTakenError**

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

TWO

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

THREE

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

FOUR

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

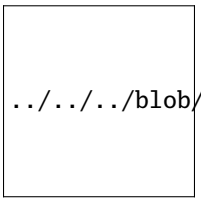
FIVE

PING SUPPORT (PULSEIN)

If you want to use ultrasonic range sensors that use a pulse to measure distance (like the very cheap and common `HC-SR04` - See *datasheet*, you will need to use a pulseIn compatible Firmata on your card.

You can download it from the pulseIn branch of the Firmata repository:

Simply connect the sensor's `Trig` and `Echo` pins to a digital pin on your board.


../../../blob/master/Examples/Figures/ping.png

And then use the ping method on the pin:

```
>>> echo_pin = board.get_pin('d:7:o')
>>> echo_pin.ping()
1204
```

You can use the `ping_time_to_distance` function to convert the result of the ping (echo time) in distance:

```
>>> from pyfirmata.util import ping_time_to_distance
>>> echo_pin = board.get_pin('d:7:o')
>>> ping_time_to_distance(echo_pin.ping())
20.4854580555607776
```

CHAPTER

SIX

NOTE

The codes will only work if you download and load the pulseIn:: code on the Arduino board! It has to be exactly the code quoted!

CHAPTER

SEVEN

CREDITS

- NeoPolus:
- Tino:

CHAPTER

EIGHT

LINKS

- Official Discord **Server_**:
- **My Discord Username:** *Aril Ogai#5646*

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX