pymata

Modules

sys threading time

Classes

PyMata

class **PyMata**

```
    This class contains the API for the PyMata library

    This code is designed to be used with StandardFirmata,
    If you require support for CodeShield on an Arduino UNO,
    please use the included custom Arduino sketch called NotSoStandardFirmata.
```

Methods defined here:

__init__(self, port_id='/dev/ttyACM0', number_digital_pins=20, number_analog_pins=6)

```
    The constructor builds the entire interface, connects to Firmata and awaits user commands
    @param port_id: Communications port specifier (COM3, /dev/ttyACM0, etc)
    @param number_digital_pins: Number of digital pins that the _arduino contains
    @param number_digital_pins: Number of analog pins that the _arduino contains
    @return: This function does not return
```

analog_read(self, pin)

```
    Read the value for the specified pin.
    @param pin: Selected pin
    @return: The last value entered in the response table if returned.
```

analog_write(self, pin, value)

```
    Set the specified pin to the specified value
    @param pin: Pin number
    @param value: Pin value
    @return: No return value
```

close(self)
```
    This method will close the transport (serial port) and exit
    @return: No return value.
```

digital_read(self, pin)
```
    Read the value for the specified pin.
    NOTE: This command will return values for digital, pwm, and encoder pin types
    @param pin: Selected pin
    @return: The last value entered in the response table if returned.
```

digital_write(self, pin, value)
```
    Set the specified pin to the specified value
    @param pin: pin number
    @param value: pin value
    @return: No return value
```

disable_analog_reporting(self, pin)
```
    Disables analog reporting for a single analog pin.
    @param pin: Analog pin number. For example for A0, the number is 0.
    @return: No return value
```

disable_digital_reporting(self, pin)
```
    Disables digital reporting. By turning reporting off for this pin, reporting
    is disabled for all 8 bits in the "port" - this is part of Firmata's design
    @param pin: Pin and all pins for this port
    @return: No return value
```

enable_analog_reporting(self, pin)
```
    Enables analog reporting. By turning reporting on for a single pin,
    @param pin: Analog pin number. For example for A0, the number is 0.
    @return: No return value
```

enable_digital_reporting(self, pin)
```
    Enables digital reporting. By turning reporting on for a pin, reporting
    is enabled for all 8 bits in the "port" - this is part of Firmata's design
    @param pin: Pin and all pins for this port
    @return: No return value
```

encoder_config(self, pin_a, pin_b)
```
    This command enables the rotary encoder (2 pin + ground) and will
```

```
    enable encoder reporting.
    It is intended to be used with NotSoStandardFirmata.
    @param pin_a: Encoder pin 1.
    @param pin_b: Encoder pin 2.
    @return: No return value
```

get_analog_response_table(self)

```
    This method returns a list of lists representing the current pin mode and associated data for all
    analog pins.
    All pin types, both input and output will be listed. Output pin data will contain zero.
    @return: The last update of the digital response table
```

get_digital_response_table(self)

```
    This method returns a list of lists representing the current pin mode and associated data for all
    digital pins.
    All pin types, both input and output will be listed. Output pin data will contain zero.
    @return: The last update of the digital response table
```

get_firmata_firmware_version(self)

```
    Get the firmware id information
    NOTE: For Leonardo Boards it will return None
    @return: Firmata_firmware  list [major, minor, file_name] or None
```

get_firmata_version(self)

```
    Get the firmata version information
    NOTE: For Leonardo Boards it will return None
    @return: Firmata_version list [major, minor] or None
```

is_firmata_ready(self, timeout=0)

```
    This method checks to see if Firmata is ready to accept commands
    @param timeout: If zero, will check for version string, else
                    will wait for the timeout in seconds
    @return: True if ready, False if not ready
```

play_tone(self, pin, frequency, duration)

```
    This method will call the Tone library for the selected pin.
    It is intended to be used with NotSoStandardFirmata.
    @param pin: Pin number
    @param frequency: Frequency of tone
    @param duration: Duration of tone in milliseconds
    @return: No return value
```

reset(self)

```
    This command sends a reset message to the Arduino. The response tables will be reinitialized
```

```
        @return: No return value.
```

servo_config(self, pin, min_pulse=544, max_pulse=2400)

```
        Configure a pin as a servo pin. Set pulse min, max in ms.
        @param pin: Servo Pin.
        @param min_pulse: Min pulse width in ms.
        @param max_pulse: Max pulse width in ms.
        @return: No return value
```

set_pin_mode(self, pin, mode, pin_type)

```
        This method sets a pin to the desired pin mode for the pin_type.
        It automatically enables data reporting..
        @param pin: Pin number (for analog use the analog number, for example A4: use 4
        @param mode: INPUT, OUTPUT, PWM, SERVO, ENCODER or TONE
        @param pin_type: ANALOG or DIGITAL
        @return: No return value
```

set_sampling_interval(self, interval)

```
        This method sends the desired sampling interval to Firmata.
        Note: Standard Firmata (and NotSoStandardFirmata) will ignore any interval less than 10 milliseconds
        @param interval: integer value for desired sampling interval in milliseconds
        @return: No return value.
```

Data and other attributes defined here:

ANALOG = 2

DIGITAL = 32

ENCODER = 7

HIGH = 1

INPUT = 0

LOW = 0

OUTPUT = 1

PWM = 3

REPORTING_DISABLE = 0

REPORTING_ENABLE = 1

SERVO = 4

TONE = 8

Data

__author__ = 'Copyright (c) 2013 Alan Yorinks All rights reserved.'

Author