PyMata API

__init__(self, port_id='/dev/ttyACM0')

```
The constructor instantiates the entire interface. It starts the operational threads for the serial
interface as well as for the command handler.
@param port_id: Communications port specifier (COM3, /dev/ttyACM0, etc)
```

analog_mapping_query(self)

```
Send an analog mapping query message via sysex. Client retrieves the results with a
call to get_analog_mapping_request_results()
```

analog_read(self, pin)

```
Retrieve the last analog data value received for the specified pin.
@param pin: Selected pin
@return: The last value entered into the analog response table.
```

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
```

capability_query(self)

```
Send a Firmata capability query message via sysex. Client retrieves the results with a
call to get_capability_query_results()
```

close(self)

```
This method will close the transport (serial port) and exit
@return: No return value, but sys.exit(0) is called.
```

digital_read(self, pin)

```
Retrieve the last digital data value received for the specified pin.
NOTE: This command will return values for digital, pwm, etc,  pin types
@param pin: Selected pin
@return: The last value entered into the digital response table.
```

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" -
@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 all 8 bits in the "port" -
this is part of Firmata's protocol specification.
@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.

NOTE: This command is not currently part of standard arduino firmata, but is provided for legacy
support of CodeShield on an Arduino UNO.
@param pin_a: Encoder pin 1.
@param pin_b: Encoder pin 2.
@return: No return value
```

extended_analog(self, pin, data)

```
This method will send an extended data analog output command to the selected pin
@param pin: 0 - 127
@param data: 0 - 0xfffff
```

get_analog_mapping_request_results(self)
Call this method after calling analog_mapping_query() to retrieve its results
@return: raw data returned by firmata

get_analog_response_table(self)
This method returns a list of lists representing the current pin mode and associated data values for all analog pins.
All configured 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_capability_query_results(self)
Retrieve the data returned by a previous call to capability_query()
@return: Raw capability data returned by firmata

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)
Retrieve the firmware id information returned by a previous call to refresh_report_firmware()
@return: Firmata_firmware list [major, minor, file_name] or None

get_firmata_version(self)
Retrieve the firmata version information returned by a previous call to refresh_report_version()
@return: Firmata_version list [major, minor] or None

get_pin_state_query_results(self)
This method returns the results of a previous call to pin_state_query() and then resets the pin state query data to None

@return: Raw pin state query data

i2c_config(self, read_delay_time=0, pin_type=None, clk_pin=0, data_pin=0)

```
NOTE: THIS METHOD MUST BE CALLED BEFORE ANY I2C REQUEST IS MADE
This method initializes Firmata for I2c operations.
It allows setting of a read time delay amount, and to optionally track
the pins as I2C in the appropriate response table.
To track pins: Set the pin_type to ANALOG or DIGITAL and provide the pin numbers.
If using ANALOG, pin numbers use the analog number, for example A4: use 4.

@param read_delay_time: an optional parameter, default is 0
@param pin_type: ANALOG or DIGITAL to select response table type to track pin numbers
@param clk_pin: pin number (see comment above).
@param data_pin: pin number (see comment above).
@return: No Return Value
```

i2c_get_read_data(self, address)

```
This method retrieves the i2c read data as the result of an i2c_read() command.
@param address: i2c device address
@return: raw data read from device
```

i2c_read(self, address, register, number_of_bytes, read_type)

```
This method requests the read of an i2c device. Results are retrieved by a call to
i2c_get_read_data()
@param address: i2c device address
@param register: register number (can be set to zero)
@param number_of_bytes: number of bytes expected to be returned
@param read_type: I2C_READ  or I2C_READ_CONTINUOUSLY
```

i2c_stop_reading(self, address)

```
This method stops an I2C_READ_CONTINUOUSLY operation for the i2c device address specified.
@param address: address of i2c device
```

i2c_write(self, address, *args)

Write data to an i2c device.
@param address: i2c device address
@param args: a variable number of bytes to be sent to the device

pin_state_query(self, pin)
This method issues a pin state query command. Data returned is retrieved via
a call to get_pin_state_query_results()
@param pin: pin number

play_tone(self, pin, tone_command, frequency, duration)
This method will call the Tone library for the selected pin.
If the tone command is set to TONE_TONE, then the specified tone will be played.
Else, if the tone command is TONE_NO_TONE, then any currently playing tone will be disabled.
It is intended for a future release of Arduino Firmata
@param pin: Pin number
@param tone_command: Either TONE_TONE, or TONE_NO_TONE
@param frequency: Frequency of tone
@param duration: Duration of tone in milliseconds
@return: No return value

refresh_report_firmware(self)
This method will query firmata to report firmware. Retrieve the report via a
call to get_firmata_firmware_version()

refresh_report_version(self)
This method will query firmata for the report version.
Retrieve the report version via a call to get_firmata_version()

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.
NOTE: DO NOT CALL THIS METHOD FOR I2C. See i2c_config().
@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  will ignore any interval less than 10 milliseconds
@param interval: Integer value for desired sampling interval in milliseconds
@return: No return value.
```