PyAX-12 Documentation

Release 0.4.dev6

Jérémie Decock

September 04, 2015

Contents

A Python library to control Dynamixel AX-12 actuators.

PyAX-12 is an open source lightweight Python library to control Dynamixel AX-12+ actuators.

Contents:

CHAPTER 1

Introduction

PyAX-12 is an open source lightweight Python library to control Dynamixel AX-12+ actuators.

Note: This project is still in *beta* stage, so the API is not finalized yet.

1.1 Dependencies

- Python >= 3.0
- Python-serial

PyAX-12 is tested to work with Python 3.4 under Gnu/Linux Debian 8 and Windows 7. It should also work with Python 3.X under recent Gnu/Linux and Windows systems. It hasn't been tested (yet) on MacOSX and BSD systems.

Python-serial is required to install PyAX-12.

Note: If you use `pip` to install PyAX-12, Python-serial will be automatically downloaded and installed (see the following *install* section).

1.2 Installation

1.2.1 Gnu/Linux

You can install, upgrade, uninstall PyAX-12 with these commands (in a terminal):

```
pip install --pre pyax12
pip install --upgrade pyax12
pip uninstall pyax12
```

Or, if you have downloaded the PyAX-12 source code:

```
python3 setup.py install
```

1.2.2 Windows

Note: The following installation procedure has been tested to work with Python 3.4 under Windows 7. It should also work with recent Windows systems.

You can install, upgrade, uninstall PyAX-12 with these commands (in a command prompt):

```
py -m pip install --pre pyax12
py -m pip install --upgrade pyax12
py -m pip uninstall pyax12
```

Or, if you have downloaded the PyAX-12 source code:

```
py setup.py install
```

1.2.3 MacOSX

Note: The following installation procedure has been tested to work with Python 3.4 under MacOSX 10.6 (*Snow Leopard*). It should also work with recent MacOSX systems.

You can install, upgrade, uninstall PyAX-12 with these commands (in a terminal):

```
pip install --pre pyax12
pip install --upgrade pyax12
pip uninstall pyax12
```

Or, if you have downloaded the PyAX-12 source code:

```
python3 setup.py install
```

1.3 Example usage

In the following examples, the `dynamixel_id`, `port` and `baudrate` values should be adapted depending on your configuration:

- for **Linux** users:
  - the `port` value should be something like
    - "/dev/ttyS0", "/dev/ttyS1", ... if you use an actual serial port
    - "/dev/ttyUSB0", "/dev/ttyUSB1", ... if you use an *USB to serial* adapter (like the USB2Dynamixel adapter)
  - the `baudrate` value should be the same than the one configured in Dynamixel units

- for **Windows** users:
  - the `port` value should be something like "COM2", "COM3", ... (see the *COM port* configuration in the Windows device manager)
  - the `baudrate` value should be the same than the one configured in the Dynamixel units *and* the Windows device manager (i.e. check the *COM port* configuration in the Windows device manager)
- for **MacOSX** users:
  - the `port` value should be something like "/dev/tty.usbserial-XXX" if you use an *USB to serial* adapter like the USB2Dynamixel adapter
  - the `baudrate` value should be the same than the one configured in Dynamixel units

If you use the USB2Dynamixel device, make sure its switch is set on "TTL".

Some other examples are available in the examples directory.

1.3.1 Ping a Dynamixel

This snippet prints `True` if the specified Dynamixel unit is connected and available at the given *baudrate*; otherwise it prints `False`.

```python
from pyax12.connection import Connection

# Connect to the serial port
serial_connection = Connection(port="/dev/ttyUSB0", baudrate=57600)

dynamixel_id = 3

# Ping the third dynamixel unit
is_available = serial_connection.ping(dynamixel_id)

print(is_available)

# Close the serial connection
serial_connection.close()
```

1.3.2 Scan (search available Dynamixel units)

This snippet prints the ID list of connected and available Dynamixel units (at the given *baudrate*).

```python
from pyax12.connection import Connection

# Connect to the serial port
serial_connection = Connection(port="/dev/ttyUSB0", baudrate=57600)

# Ping the dynamixel unit(s)
ids_available = serial_connection.scan()

for dynamixel_id in ids_available:
    print(dynamixel_id)

# Close the serial connection
serial_connection.close()
```

1.3.3 Print the control table of the first Dynamixel unit

This snippet prints the control table of the specified Dynamixel unit (i.e. the internal state information of the Dynamixel unit).

```python
from pyax12.connection import Connection

# Connect to the serial port
serial_connection = Connection(port="/dev/ttyUSB0", baudrate=57600)

dynamixel_id = 1

# Print the control table of the specified Dynamixel unit
serial_connection.pretty_print_control_table(dynamixel_id)

# Close the serial connection
serial_connection.close()
```

This snippet should print something like that:

```
model_number................. AX-12+
firmware_version............. 24
id........................... 1
baud_rate.................... 57142.9 bps
return_delay_time............ 500 µs
cw_angle_limit............... -150.0° (0)
ccw_angle_limit.............. 150.0° (1023)
max_temperature.............. 70°C
min_voltage.................. 6.0V
max_voltage.................. 14.0V
max_torque................... 1023
status_return_level.......... 2 (respond to all instructions)
input_voltage_alarm_led...... off
angle_limit_alarm_led........ off
overheating_alarm_led........ on
range_alarm_led.............. off
checksum_alarm_led........... off
overload_alarm_led........... on
instruction_alarm_led........ off
input_voltage_alarm_shutdown. off
angle_limit_alarm_shutdown... off
overheating_alarm_shutdown... on
range_alarm_shutdown......... off
checksum_alarm_shutdown...... off
overload_alarm_shutdown...... on
instruction_alarm_shutdown... off
down_calibration............. 46
up_calibration............... 972
torque_enabled............... yes
led.......................... off
cw_compliance_margin......... 0.3° (1)
ccw_compliance_margin........ 0.3° (1)
cw_compliance_slope.......... 9.4° (32)
ccw_compliance_slope......... 9.4° (32)
goal_position................ -0.1° (511)
moving_speed................. 512
torque_limit................. 1023
present_position............. -0.1° (511)
present_speed................ 0
```

```
present_load................. 0
present_voltage.............. 12.1V
present_temperature.......... 43°C
registred_instruction........ no
moving....................... no
locked....................... no
punch........................ 32
```

1.3.4 Move the first Dynamixel unit to various position angles

This snippet moves the first Dynamixel unit to 0°, then -45°, -90°, -135°, -150° (the maximum CW angle), +150° (the maximum CCW angle), +135°, +90°, +45° and finally goes back to 0°.

```python
from pyax12.connection import Connection
import time

# Connect to the serial port
serial_connection = Connection(port="/dev/ttyUSB0", baudrate=57600)

dynamixel_id = 1

# Go to 0°
serial_connection.goto(dynamixel_id, 0, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to -45° (45° CW)
serial_connection.goto(dynamixel_id, -45, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to -90° (90° CW)
serial_connection.goto(dynamixel_id, -90, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to -135° (135° CW)
serial_connection.goto(dynamixel_id, -135, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to -150° (150° CW)
serial_connection.goto(dynamixel_id, -150, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to +150° (150° CCW)
serial_connection.goto(dynamixel_id, 150, speed=512, degrees=True)
time.sleep(2)    # Wait 2 seconds

# Go to +135° (135° CCW)
serial_connection.goto(dynamixel_id, 135, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to +90° (90° CCW)
serial_connection.goto(dynamixel_id, 90, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second

# Go to +45° (45° CCW)
serial_connection.goto(dynamixel_id, 45, speed=512, degrees=True)
time.sleep(1)    # Wait 1 second
```

```
# Go back to 0°
serial_connection.goto(dynamixel_id, 0, speed=512, degrees=True)

# Close the serial connection
serial_connection.close()
```

1.4 Related libraries

Other libraries to control Dynamixel AX-12+ actuators are referenced in the following (non comprehensive) list:

- PyPot by Inria (FLOWERS team)
- PyDynamixel by Richard Clark
- Pydyn by Fabien Benureau and Olivier Mangin (Inria FLOWER team)
- Dynamixel by Ian Danforth
- dynamixel_hr by Romain Reignier
- python_dynamixels by Jesse Merritt
- ax12 by Thiago Hersan
- Dynamixel Monitor by Christian Balkenius
- DynamixelMonitor by Slavik

CHAPTER 2

PyAX-12 API

The library provides classes which are usable by third party tools.

Note: PyAX-12 is still in *beta* stage, so the API is not finalized yet.

Modules:

2.1 Connection module

This module contain the *Connection* class communicate with Dynamixel units.

class `pyax12.connection.`**`Connection`**(*port='/dev/ttyUSB0'*, *baudrate=57600*, *timeout=0.1*, *waiting_time=0.02*)

Create a serial connection with dynamixel actuators.

Parameters

- **`port`** (*str*) – the serial device to connect with (e.g. '/dev/ttyUSB0' for Unix users or 'COM1' for windows users).
- **`baudrate`** (*int*) – the baudrate speed (e.g. 57600).
- **`timeout`** (*float*) – the timeout value for the connection.
- **`waiting_time`** (*float*) – the waiting time (in seconds) between sending the instruction packet and the receiving the status packet.

`close`()

Close the serial connection.

`dump_control_table`(*dynamixel_id*)

Dump the *control table* of the specified Dynamixel unit.

This function can be used to backup the current configuration of the given Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

Returns the sequence of all bytes in currently the *control table*.

`flush`()

Flush the connection buffers.

`get_baud_rate`(*dynamixel_id*)

Return the communication speed (baud rate) of the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_ccw_angle_limit`(*dynamixel_id*)

Return the *counter clockwise angle limit* of the specified Dynamixel unit.

The goal position should be lower or equal than this value, otherwise the *Angle Limit Error Bit* (the second error bit of Status Packets) will be set to `1`.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_ccw_compliance_margin`(*dynamixel_id*)

Return the counter clockwise compliance margin of the specified Dynamixel unit.

The compliance feature can be utilized for absorbing shocks at the output shaft.

```
     CW

      |     goal position
      |         :\          :
      |         : \         :
      |         :  \        :
      |         :   \       :
      |         :    \      :
      |         :          :
      |         :
CCW ------------------------------ CW
      |         :     :     :
      |         :     :     :
      |         :     :     :     :\
      |         :     :     :     : \
      |         :     :     :     :  \
      |         :     :     :     :   \
      |         :     :     :     :    \
      |         :     :     :     :     :
     CCW         --- --- --- ---
                   A     B     C     D

x axis: position error
y axis: output torque

A: CCW Compliance Slope
B: CCW Compliance Margin
C: CW Compliance Slope
D: CW Compliance Margin
```

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_ccw_compliance_slope`(*dynamixel_id*)

Return the counter clockwise compliance scope of the specified Dynamixel unit.

The compliance feature can be utilized for absorbing shocks at the output shaft.

```
     CW

      |     goal position
      |         :\          :
      |         : \         :
```

```
      |         :  \        :
      |         :   \       :
      |         :    \      :
      |         :          :
      |         :
CCW ------------------------------ CW
      |         :     :     :
      |         :     :     :
      |         :     :     :     :\
      |         :     :     :     : \
      |         :     :     :     :  \
      |         :     :     :     :   \
      |         :     :     :     :    \
      |         :     :     :     :     :
     CCW         --- --- --- ---
                   A     B     C     D

x axis: position error
y axis: output torque

A: CCW Compliance Slope
B: CCW Compliance Margin
C: CW Compliance Slope
D: CW Compliance Margin
```

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_control_table_tuple`(*dynamixel_id*)

Return the *control table* of the specified Dynamixel unit in an easily human readable tuple.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_cw_angle_limit`(*dynamixel_id*)

Return the *clockwise angle limit* of the specified Dynamixel unit.

The goal position should be higher or equal than this value, otherwise the *Angle Limit Error Bit* (the second error bit of Status Packets) will be set to `1`.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_cw_compliance_margin`(*dynamixel_id*)

Return the clockwise compliance margin of the specified Dynamixel unit.

The compliance feature can be utilized for absorbing shocks at the output shaft.

```
     CW

      |     goal position
      |         :\          :
      |         : \         :
      |         :  \        :
      |         :   \       :
      |         :    \      :
      |         :          :
      |         :
CCW ------------------------------ CW
```

```
         |         :     :     :
         |         :     :     :
         |         :     :     :     :\
         |         :     :     :     : \
         |         :     :     :     :  \
         |         :     :     :     :   \
         |         :     :     :     :    \
         |         :     :     :     :     :
        CCW         --- --- --- ---
                      A     B     C     D

    x axis: position error
    y axis: output torque

    A: CCW Compliance Slope
    B: CCW Compliance Margin
    C: CW Compliance Slope
    D: CW Compliance Margin
```

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_cw_compliance_slope`(*dynamixel_id*)

Return the clockwise compliance scope of the specified Dynamixel unit.

The compliance feature can be utilized for absorbing shocks at the output shaft.

```
        CW

         |      goal position
         |         :\          :
         |         : \         :
         |         :  \        :
         |         :   \       :
         |         :    \      :
         |         :          :
         |         :
    CCW ------------------------------ CW
         |         :     :     :
         |         :     :     :
         |         :     :     :     :\
         |         :     :     :     : \
         |         :     :     :     :  \
         |         :     :     :     :   \
         |         :     :     :     :    \
         |         :     :     :     :     :
        CCW         --- --- --- ---
                      A     B     C     D

    x axis: position error
    y axis: output torque

    A: CCW Compliance Slope
    B: CCW Compliance Margin
    C: CW Compliance Slope
    D: CW Compliance Margin
```

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_down_calibration`(*dynamixel_id*)
Return the "down calibration" value of the specified Dynamixel unit.

The calibration value is used to compensate the differences between the potentiometers used in the Dynamixel units.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_firmware_version`(*dynamixel_id*)
Return the firmware version of the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_goal_position`(*dynamixel_id*)
Return the requested goal angular position of the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_max_temperature`(*dynamixel_id*)
Return the maximum tolerated internal temperature for the specified Dynamixel unit.

If the internal temperature of the Dynamixel actuator gets higher than this value, the *Over Heating Error Bit* (the third error bit of Status Packets) will be set to `1`. The values are in degrees Celsius.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_max_torque`(*dynamixel_id*)
Return the initial maximum torque output of the specified Dynamixel unit.

This value, written in EEPROM, is copied to the *torque limit* bytes (in RAM) when the power is turned ON. Thus, *max torque* is just an initialization value for the actual *torque limit*.

If this value is equal to `0`, the Dynamixel unit is configured in *free run mode*.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_max_voltage`(*dynamixel_id*)
Return the maximum tolerated operating voltage for the specified Dynamixel unit.

If the present voltage of the Dynamixel actuator gets higher than this value, the *Voltage Range Error Bit* (the first error bit of Status Packets) will be set to `1`. The values are in Volts.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_min_voltage`(*dynamixel_id*)
Return the minimum tolerated operating voltage for the specified Dynamixel unit.

If the present voltage of the Dynamixel actuator gets lower than this value, the *Voltage Range Error Bit* (the first error bit of Status Packets) will be set to `1`. The values are in Volts.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_model_number`(*dynamixel_id*)
Return the model number of the specified Dynamixel unit.

For AX-12, this value should be 12 (0x000C).

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_moving_speed`(*dynamixel_id*)

Return the angular velocity of the specified Dynamixel unit.

This angular velocity is defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation. The maximum value (1023 or 0x3FF) corresponds to 114 RPM (provided that there is enough power supplide).

Zero is a special value meaning that the largest possible velocity is supplied for the configured voltage, e.g. no velocity control is applied.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_present_load`(*dynamixel_id*)

Return the magnitude of the load applied to the specified Dynamixel unit.

If the returned value is negative, the load is applied to the clockwise direction.

If the returned value is positive, the load is applied to the counter clockwise direction.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_present_position`(*dynamixel_id*, *degrees=False*)

Return the current angular position of the specified Dynamixel unit.

Parameters

- **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).
- **`degrees`** (*bool*) – defines the returned *position* unit. If *degrees* is `True`, *position* corresponds to the goal rotation angle *in degrees* with respect to the original position and is defined in range (-150, 150). Otherwise, *position* is a unit free angular position to the origin, defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.

`get_present_speed`(*dynamixel_id*)

Return the current angular velocity of the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_present_temperature`(*dynamixel_id*)

Return the internal temperature of the specified Dynamixel unit (in Degrees Celsius).

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_present_voltage`(*dynamixel_id*)

Return the voltage currently applied to the specified Dynamixel unit (in Volts).

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_punch`(*dynamixel_id*)

Return the minimum current supplied to the motor of the specified Dynamixel unit during operation.

The initial value is set to 0x20 and its maximum value is 0x3FF.

```
          CW

           |     goal position
```

```
      |         :\          :
      |         : \         :
      |         :  \        :
      |         :   \       :
      |         :    \      :
      |         :          :
      |         :          :        | E
      |         :
CCW ------------------------------ CW
      |         :     :     :
      |         :     :     :       | E
      |         :     :     :
      |         :     :     :     :\
      |         :     :     :     : \
      |         :     :     :     :  \
      |         :     :     :     :   \
      |         :     :     :     :    \
      |         :     :     :     :     :
     CCW         --- --- --- ---
                   A     B     C     D

x axis: position error
y axis: output torque

A: CCW Compliance Slope
B: CCW Compliance Margin
C: CW Compliance Slope
D: CW Compliance Margin
E: Punch
```

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_return_delay_time`(*dynamixel_id*)

Return the return delay time of the specified Dynamixel unit.

The return delay time is the time it takes (in uSec) for the status packet to return after the instruction packet is sent.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_status_return_level`(*dynamixel_id*)

Say whether the specified Dynamixel unit is configured to return a *Status Packet* after receiving an *Instruction Packet*.

Returned value	Meaning
0	Do not respond to any instructions
1	Respond only to READ_DATA instructions
2	Respond to all instructions

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_torque_limit`(*dynamixel_id*)

Return the maximum torque output of the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`get_up_calibration`(*dynamixel_id*)
Return the "up calibration" value of the specified Dynamixel unit.

The calibration value is used to compensate the differences between the potentiometers used in the Dynamixel units.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`goto`(*dynamixel_id*, *position*, *speed=None*, *degrees=False*)
Set the *goal position* and *moving speed* for the specified Dynamixel unit.

Parameters

- **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFE).
- **`position`** (*int*) – the new goal position. If *degrees* is `True`, *position* corresponds to the goal rotation angle *in degrees* with respect to the original position and must be in range (-150, 150). Otherwise, *position* is a unit free rotation angle to the origin, defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.
- **`speed`** (*int*) – the new moving speed. It must be in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation. This parameter is optional; if *speed* is not specified, the *moving speed* present in the Dynamixel control table is kept and used to reach the goal position.
- **`degrees`** (*bool*) – defines the *position* unit. If *degrees* is `True`, *position* corresponds to the goal rotation angle *in degrees* with respect to the original position and must be in range (-150, 150). Otherwise, *position* is a unit free angular position, defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.

`has_angle_limit_alarm_led`(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Angle Limit Error* occurs.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`has_angle_limit_alarm_shutdown`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Angle Limit Error* occurs.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`has_checksum_alarm_led`(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Checksum Error* occurs.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`has_checksum_alarm_shutdown`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Checksum Error* occurs.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_input_voltage_alarm_led(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Input Voltage Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_input_voltage_alarm_shutdown(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Input Voltage Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_instruction_alarm_led(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Instruction Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_instruction_alarm_shutdown(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Instruction Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_overheating_alarm_led(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Overheating Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_overheating_alarm_shutdown(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Overheating Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_overload_alarm_led(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Overload Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_overload_alarm_shutdown(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Overload Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

has_range_alarm_led(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is configured to blink when an *Range Error* occurs.

Parameters **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`has_range_alarm_shutdown`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is configured to turn off its torque when an *Range Error* occurs.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`has_registred_instruction`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is currently processing a REG_WRITE command; otherwise, return `False`.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`is_led_enabled`(*dynamixel_id*)
Return `True` if the LED of the specified Dynamixel unit is ON; otherwise return `False`.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`is_locked`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is locked; return `False` otherwise.

When a Dynamixel unit is locked, only addresses 0x18 to 0x23 can be written. Once locked, it can only be unlocked by turning the power off.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`is_moving`(*dynamixel_id*)
Return `True` if the specified Dynamixel unit is moving by its own power; return `False` otherwise.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`is_torque_enable`(*dynamixel_id*)
Return `True` if the torque of the specified Dynamixel unit is enabled; otherwise return `False`.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`ping`(*dynamixel_id*)
Ping the specified Dynamixel unit.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

Returns `True` if the specified unit is available, `False` otherwise.

`pretty_print_control_table`(*dynamixel_id*)
Print the *control table* of the specified Dynamixel unit in an easily human readable format.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

`print_control_table`(*dynamixel_id*)
Print the *control table* of the specified Dynamixel unit in an "raw" format.

To get the same output in a more easily human readable format, use the *pretty_print_control_table* function.

Parameters **`dynamixel_id`** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).

read_data (*dynamixel_id*, *address*, *length*)
Read bytes form the control table of the specified Dynamixel unit.

Parameters

- **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).
- **address** (*int*) – the starting address of the location where the data is to be read.
- **length** (*int*) – the length of the data to be read.

scan (*dynamixel_id_bytes=None*)
Return the ID sequence of available Dynamixel units.

Parameters **dynamixel_id_bytes** (*bytes*) – a sequence of unique ID of the Dynamixel units to be pinged.

send (*instruction_packet*)
Send an instruction packet.

Parameters **instruction_packet** – can be either a *Packet* instance or a "bytes" string containing the full instruction packet to be sent to Dynamixel units.

set_ccw_angle_limit (*dynamixel_id*, *angle_limit*, *degrees=False*)
Set the *counter clockwise angle limit* of the specified Dynamixel unit to the specified *angle_limit*.

The goal position should be lower or equal than this value, otherwise the *Angle Limit Error Bit* (the second error bit of Status Packets) will be set to `1`.

Parameters

- **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).
- **angle_limit** (*int*) – the *counter clockwise angle limit* to be set for the specified Dynamixel unit. If *degrees* is `True`, this value is defined in degrees and must be in range (-150, 150); otherwise, it is an unit free angle and must be in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.
- **degrees** (*bool*) – defines the *angle_limit* unit. If *degrees* is `True`, *angle_limit* is defined *in degrees* and must be in range (-150, 150). Otherwise, *angle_limit* is a unit free angular limit, defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.

set_cw_angle_limit (*dynamixel_id*, *angle_limit*, *degrees=False*)
Set the *clockwise angle limit* of the specified Dynamixel unit to the specified *angle_limit*.

The *goal position* should be higher or equal than this value, otherwise the *Angle Limit Error Bit* (the second error bit of Status Packets) will be set to `1`.

Parameters

- **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFD).
- **angle_limit** (*int*) – the *clockwise angle limit* to be set for the specified Dynamixel unit. If *degrees* is `True`, this value is defined in degrees and must be in range (-150, 150); otherwise, it is an unit free angle and must be in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.
- **degrees** (*bool*) – defines the *angle_limit* unit. If *degrees* is `True`, *angle_limit* is defined *in degrees* and must be in range (-150, 150). Otherwise, *angle_limit* is a unit free angular limit, defined in range (0, 1023) i.e. (0, 0x3FF) in hexadecimal notation.

write_data (*dynamixel_id*, *address*, *data*)
Write bytes to the control table of the specified Dynamixel unit.

Parameters

- **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit. It must be in range (0, 0xFE).
- **address** (*int*) – the starting address of the location where the data is to be written.
- **data** (*bytes*) – the bytes of the data to be written (it can be an integer, a sequence of integer, a bytes or a bytearray).

2.2 InstructionPacket module

This module contain the *InstructionPacket* class which implements "instruction packets" (the packets sent by the controller to the Dynamixel actuators to send commands).

class `pyax12.instruction_packet.`**`InstructionPacket`**(*dynamixel_id*, *instruction*, *parameters=None*)

The "instruction packet" is the packet sent by the main controller to the Dynamixel units to send commands.

The structure of the instruction packet is as the following:

0XFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

Parameters

- **dynamixel_id** (*int*) – the the unique ID of the Dynamixel unit which have to execute this instruction packet.
- **instruction** (*int*) – the instruction for the Dynamixel actuator to perform.
- **parameters** (*bytes*) – a sequence of bytes used if there is additional information needed to be sent other than the instruction itself.

`instruction`

The instruction for the Dynamixel actuator to perform.

This member is a read-only property.

2.3 Packet module

This module contains the general *Packet* class which implements the either "instruction packet" (the packets sent by the controller to the Dynamixel actuators to send commands) or "status packet" (the response packets from the Dynamixel units to the main controller after receiving an instruction packet).

class `pyax12.packet.`**`Packet`**(*dynamixel_id*, *data*)

The general raw *Packet* class.

It implements the either "instruction packet" (the packets sent by the controller to the Dynamixel actuators to send commands) or "status packet" (the response packets from the Dynamixel units to the main controller after receiving an instruction packet).

The structure of a general *Packet* is as the following:

0xFF	0xFF	ID	LENGTH	DATA...	CHECK SUM

This class has been made for debugging purpose and is not intended to be used widely to create packets. Instead, it is recommanded to use *InstructionPacket* or *StatusPacket* classes to build *Packet* instances.

Parameters

- **dynamixel_id** (*int*) – the unique ID of a Dynamixel unit (from 0x00 to 0xFD), 0xFE is a broadcasting ID.
- **data** (*bytes*) – a sequence of byte containing the packet's data: the instruction to perform or the status of the Dynamixel actuator. This *data* argument contains the fifth to the penultimate byte of the full built packet.

checksum

The packet checksum, used to prevent packet transmission error.

This member is a read-only property.

data

A sequence of byte containing the packet's data, i.e. from the fifth to the penultimate byte of the "full packet".

It contains either:

- the instruction to perform and its parameters if the packet is an "instruction packet";
- or the status of the Dynamixel actuator (the "error" and "parameters" fields) if the packet is a "status packet".

This member is a read-only property.

dynamixel_id

The unique ID of a Dynamixel unit concerned with this packet.

This byte either:

- has a value between 0x00 and 0xFD to affect the corresponding Dynamixel unit
- or has the value 0xFE to affect all connected units (0xFE is the "broadcasting" ID).

This member is a read-only property.

header

The header of the packet.

This pair of byte should always be equals to `b'\xff\xff'`.

This member is a read-only property.

length

The so called "length" of the packet.

This is not the actual length of the full packet (*self._bytes*) but its number of bytes after its fourth byte, i.e.:

```
len(self._bytes[4:])
```

or in other words:

```
len(self._bytes) - 4
```

This value (so called "LENGTH") defines the fourth byte of each packet.

This member is a read-only property.

parameters

A sequence of byte used if there is additional information needed to be read (other than the error itself).

This member is a read-only property.

to_byte_array()

Return the packet as a bytearray (a mutable sequence of bytes).

This function returns something like:

```
bytearray(b'\xff\xff\xfe\x04\x03\x03\x01\xf6')
```

to_bytes()

Return the packet as a bytes string (an immutable sequence of bytes).

This function returns something like:

```
b'\xff\xff\xfe\x04\x03\x03\x01\xf6'
```

to_integer_tuple()

Return the packet as a tuple of integers.

This function returns something like:

```
(255, 255, 254, 4, 3, 3, 1, 246)
```

to_printable_string()

Return the packet as a string of hexadecimal values.

This function returns something like:

```
ff ff fe 04 03 03 01 f6
```

pyax12.packet.**compute_checksum**(*byte_seq*)

Compute and return the checksum of the *byte_seq* packet.

The checksum is the value of the last byte of each packet. It is used to prevent transmission errors of packets. Checksums are computed as follow:

```
Checksum = ~(dynamixel_id + length + data1 + ... + dataN)
```

where ~ represent the NOT logic operation.

If the computed value is larger than 255, then only its lower byte is defined as the checksum value.

Parameters **byte_seq** (*bytes*) – a byte sequence containing the packet's bytes involved in the computation of the checksum (i.e. from the third to the penultimate byte of the "full packet" considered).

2.4 StatusPacket module

This module contain the *StatusPacket* class which implements "status packets" (the response packets from the Dynamixel units to the main controller after receiving an instruction packet).

class pyax12.status_packet.**StatusPacket**(*packet*)

The "status packet" is the response packet from the Dynamixel units to the main controller after receiving an "instruction packet".

The structure of the status packet is as the following:

0XFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	...	PARAMETER N	CHECK SUM

StatusPacket is not intended to be instancied by users (except maybe for testing and debugging prupose). Under normal conditions of use, *StatusPacket*'s instances are automatically created by the *Connection* class.

Parameters **packet** (*bytes*) – a sequence of bytes containing the full status packet returned by Dynamixel units. It must be compatible with the "bytes" type.

angle_limit_error

A boolean which is set to True if the goal position is set outside of the range between "CW Angle Limit" and "CCW Angle Limit".

This member is a read-only property.

`checksum_error`
A boolean which is set to True if the checksum of the instruction packet is incorrect.

This member is a read-only property.

`error`
The byte representing errors sent from the Dynamixel unit.

This member is a read-only property.

`input_voltage_error`
A boolean which is set to True if the voltage is out of the operating voltage range as defined in the control table.

This member is a read-only property.

`instruction_error`
A boolean which is set to True if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.

This member is a read-only property.

`overheating_error`
A boolean which is set to True if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.

This member is a read-only property.

`overload_error`
A boolean which is set to True if the specified maximum torque can't control the applied load.

This member is a read-only property.

`range_error`
A boolean which is set to True if the instruction sent is out of the defined range.

This member is a read-only property.

2.5 Utils module

This module contains some general purpose utility functions.

`pyax12.utils.`**`int_to_little_endian_bytes`**(*integer*)
Converts a two-bytes integer into a pair of one-byte integers using the little-endian notation (i.e. the less significant byte first).

The *integer* input must be a 2 bytes integer, i.e. *integer* must be greater or equal to 0 and less or equal to 65535 (0xffff in hexadecimal notation).

For instance, with the input decimal value `integer=700` (0x02bc in hexadecimal notation) this function will return the tuple `(0xbc, 0x02)`.

Parameters **`integer`** (*int*) – the 2 bytes integer to be converted. It must be in range (0, 0xffff).

`pyax12.utils.`**`little_endian_bytes_to_int`**(*little_endian_byte_seq*)
Converts a pair of bytes into an integer.

The *little_endian_byte_seq* input must be a 2 bytes sequence defined according to the little-endian notation (i.e. the less significant byte first).

For instance, if the *little_endian_byte_seq* input is equals to `(0xbc, 0x02)` this function returns the decimal value `700` (0x02bc in hexadecimal notation).

Parameters **`little_endian_byte_seq`** (*bytes*) – the 2 bytes sequence to be converted. It must be compatible with the "bytes" type and defined according to the little-endian notation.

`pyax12.utils.`**`pretty_hex_str`**(*byte_seq*, *separator=', '*)

Converts a squence of bytes to a string of hexadecimal numbers.

For instance, with the input tuple `(255, 0, 10)` this function will return the string `"ff,00,0a"`.

Parameters

- **`byte_seq`** (*bytes*) – a sequence of bytes to process. It must be compatible with the "bytes" type.
- **`separator`** (*str*) – the string to be used to separate each byte in the returned string (default ",").

`pyax12.utils.`**`dxl_angle_to_degrees`**(*dxl_angle*)

Normalize the given angle.

PxAX-12 uses the position angle (-150.0°, +150.0°) range instead of the (0°, +300.0°) range defined in the Dynamixel official documentation because the former is easier to use (especially to make remarkable angles like right angles or 45° and 135° angles).

Parameters **`dxl_angle`** (*int*) – an angle defined according to the Dynamixel internal notation, i.e. in the range (0, 1023) where:

- 0 is a 150° clockwise angle;
- 1023 is a 150° counter clockwise angle.

Returns

an angle defined in degrees in the range (-150.0°, +150.0°) where:

- -150.0 is a 150° clockwise angle;
- +150.0 is a 150° counter clockwise angle.

Return type float.

`pyax12.utils.`**`degrees_to_dxl_angle`**(*angle_degrees*)

Normalize the given angle.

PxAX-12 uses the position angle (-150.0°, +150.0°) range instead of the (0°, +300.0°) range defined in the Dynamixel official documentation because the former is easier to use (especially to make remarkable angles like right angles or 45° and 135° angles).

Parameters **`angle_degrees`** (*float*) – an angle defined in degrees the range (-150.0°, +150.0°) where:

- -150.0 is a 150° clockwise angle;
- +150.0 is a 150° counter clockwise angle.

Returns

an angle defined according to the Dynamixel internal notation, i.e. in the range (0, 1023) where:

- 0 is a 150° clockwise angle;
- 1023 is a 150° counter clockwise angle.

Return type int.

CHAPTER 3

Developer's notes

3.1 Source code

The source code is currently available on GitHub under the terms and conditions of the *MIT license*. Fork away!

3.2 Bug reports

To search for bugs or report them, please use the PyAX-12 Bug Tracker at:

https://github.com/jeremiedecock/pyax12/issues

3.3 Contribute

PyAX-12 is written for Python 3.x. Python 2.x is *not* supported.

The TODO.md file contains the TODO list.

All contributions should at least comply with the following PEPs:

- PEP8 "Python's good practices"
- PEP257 "Docstring Conventions"
- PEP287 "reStructuredText Docstring Format"

All contribution should be properly documented and tested with unittest and/or doctest.

pylint, pep8 and pyflakes should also be used to check the quality of each module.

3.4 Changes

3.4.1 0.1 (2010-08-15)

Initial version of the four main modules:

- *connection*
- *packet*
- *instruction_packet*

- *status_packet*

Packaging:

- write the *setup.py* file
- write a script to automatically build the Debian package

3.4.2 0.2 (2015-06-10)

Changes:

- switch to Python 3
- update the docstrings
- update the *README* file
- rename the *COPYING* file
- add an example (*example/move.py*)
- improve the *Connection.send* function
- miscellaneous updates and bug fixes

3.4.3 0.3 (2015-06-12)

Changes:

- change the package name (from *pydynamixel* to *pyax12*), as the project now focus on Dynamixel AX-12 actuators
- miscellaneous updates and bug fixes
- Web site: http://www.jdhp.org/projects_en.html#pyax12
- Online documentation: http://pyax-12.readthedocs.org
- Source code: https://github.com/jeremiedecock/pyax12
- Issue tracker: https://github.com/jeremiedecock/pyax12/issues
- PyAX-12 on PyPI: https://pypi.python.org/pypi/pyax12

In collaboration with http://www.vorobotics.com

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Credits

Created by Jérémie Decock.

CHAPTER 6

License

The PyAX-12 library is provided under the terms and conditions of the MIT license:

Index