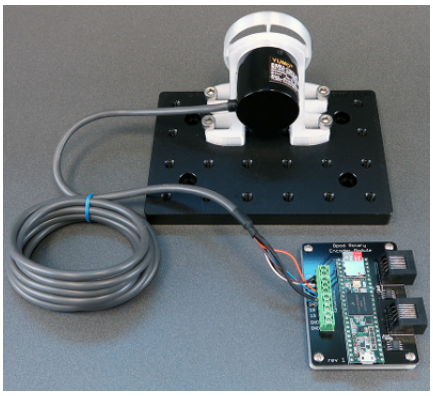
# Rotary Encoder Module Documentation

***Release v0.1.4***

**Ricardo Ribeiro**

**Nov 07, 2019**

# Getting started



CHAPTER 1

# What is the Rotary Encoder Module

The Rotary Encoder Module is a board developed by Sanworks, to record and analyze rotational movements from a motor.

On this documentation it is explained how to use the **pybpod_rotaryencoder_module** Python 3 library to control this module.

## 1.1 Getting started

### 1.1.1 Requirements

This library requires the following Python 3 packages to be installed.

- pyserial>=3.1.1
- python-dateutil
- numpy
- sip
- pyqt5
- matplotlib
- pyforms==4.901.2
- https://UmSenhorQualquer@bitbucket.org/fchampalimaud/logging-bootstrap.git

### 1.1.2 Installation

Execute the following command to install the library on an existing python3 environment.

```
pip install -U pybpod-gui-plugin-rotaryencoder
```

### 1.1.3 Use the module in the pybpod-api library

This module can be connected and disconnected from the Pybpod-api library. For this go to your user_settings.py file and add the following configuration.

```
PYBPOD_API_MODULES = [
    'pybpod_rotaryencoder_module'
]
```

**Note:** The configuration above is configured by default on the pybpod-api library. You only need to do it if you have overwritten the **PYBPOD_API_MODULES** variable in your user_settings.py.

### 1.1.4 Examples

#### 1.1.4.1 Use the rotary encoder module with the pybpod-api

```
#...

bpod = Bpod()

for m in bpod.modules:
    print( m.name, type(m) )

rotary_encoder = bpod.modules[0]
rotary_encoder.set_position_zero()

bpod.stop()
```

#### 1.1.4.2 Access the rotary encoder module directly from the USB port

```
from pybpod_rotaryencoder_module.module_api import RotaryEncoderModule

m = RotaryEncoderModule('/dev/ttyACM1')

m.enable_stream()

#print the first 100 outputs
count = 0
while count<100:
    data = m.read_stream()
    if len(data)==0:
        continue
    else:
        count += 1
        print(data)

m.disable_stream()

print('set', m.set_position(179))
m.set_zero_position()
```

(continues on next page)

(continued from previous page)

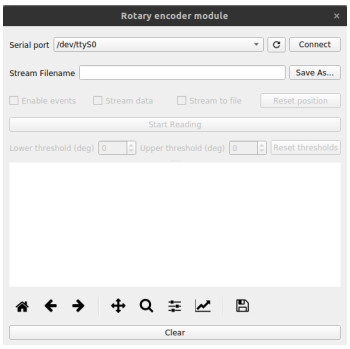
```
m.enable_thresholds([True, False, True, True, False, False, True, True])
print(m.current_position())

m.close()
```

#### 1.1.4.3 Configure the rotary encoder using the GUI

```
import pyforms
from pybpod_rotaryencoder_module.module_gui import RotaryEncoderModuleGUI


pyforms.start_app( RotaryEncoderModuleGUI, geometry=(0,0,600,500) )
```

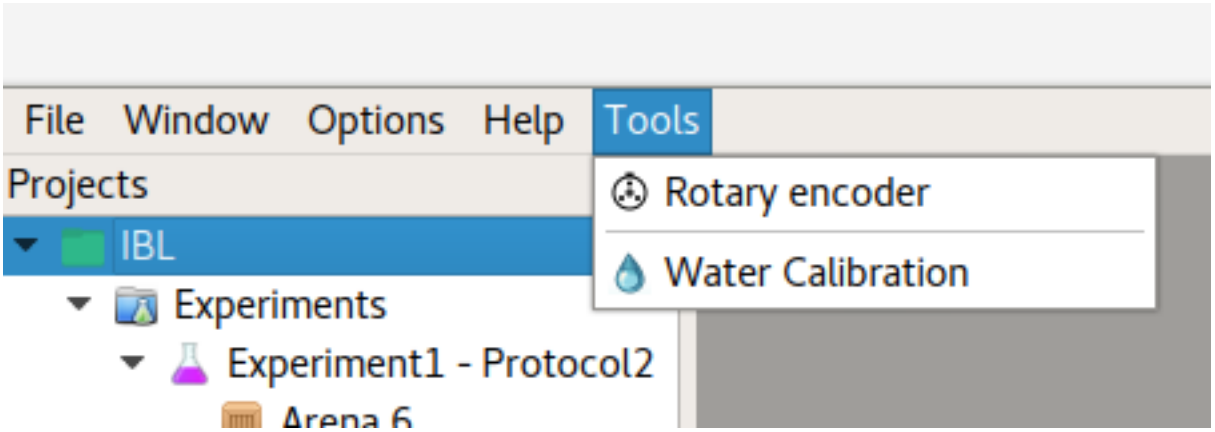



### 1.1.5 Add the Rotary Plugin to the PyBpod-GUI

In the GUI settings add the next configuration to the list of plugins

```
GENERIC_EDITOR_PLUGINS_LIST = [
    ...
    'pybpod_rotaryencoder_module',
]
```

After open the PyBpod-GUI a access the plugin in the Tools menu.



## 1.2 Changelog

### 1.2.1 0.1.4 (2019-11-07)

- Add support for enabling/disabling moduleOutputStream in the GUI

### 1.2.2 0.1.3 (2019-07-23)

- Minor documentation fixes
- Fixed ReadTheDocs requirements

### 1.2.3 0.1.2 (2019-07-19)

- Added ComboBox and a refresh button for the serial port selection for the Rotary Encoder

## 1.3 USB

**class** pybpod_rotaryencoder_module.module_api.**RotaryEncoderModule**(*serialport=None*)

Constructor of the RotaryEncoderModule object A serial connection to the Rotary Encoder board is opened at the construction of the object.

**Variables** **serialport** (*str*) – PC serial port where the module is connected

**open**(*serialport*)

Opens a serial connection to the Rotary Encoder board.

**Variables** **serialport** (*str*) – PC serial port where the module is connected

**close**()

Closes the serial connection to the Rotary Encoder board.

**enable_evt_transmission**()

Enables the transmission of threshold crossing events to the Bpod state machine.

**disable_evt_transmission**()

Disables the transmission of events.

**enable_module_outputstream**()

Enables the streaming of current position data directly to another Bpod module (e.g. DDS, AnalogOutput).

**disable_module_outputstream**()

Disables the streaming of current position data directly to another Bpod module.

**enable_stream**()

Enables the streaming of the position and the time measurements to the USB port.

**disable_stream**()

Disables the streaming of the position and the time measurements to the USB port.

**read_stream**()

Reads the data being streamed through the USB port.

**current_position**()

Retrieves the current position.

**set_zero_position**()
Sets current rotary encoder position to zero.

**set_position**(*degrees*)
Sets the current position in degrees.

**Variables** **degrees** (*int*) – current position in degrees.

**enable_thresholds**(*thresholds*)
Enables the thresholds.

**Variables** **thresholds** (*list(boolean)*) – list of 6 booleans indicating which thresholds are active to trigger events.

**enable_logging**()
Enables the logging to the SD Card.

**disable_logging**()
Disables the logging to the SD Card.

**get_logged_data**()
Retrieves the logged data in the SD Card.

**set_prefix**(*prefix*)
Sets 1-character prefix for module output stream.

**Variables** **prefix** (*char*) – One character to be used as prefix.

**set_thresholds**(*thresholds*)
Sets the thresholds values to trigger the events.

**Variables** **thresholds** (*list(int)*) – List, in maximum, of 6 thresholds to trigger events.

**set_wrappoint**(*wrap_point*)
Sets wrap point (number of tics in a half-rotation)

**Variables** **wrap_point** (*int*) – number of tics in a half-rotation.

### 1.3.1 Usage example

```
from pybpod_rotaryencoder_module.module_api import RotaryEncoderModule

m = RotaryEncoderModule('/dev/ttyACM1')

m.start_logging()

m.enable_stream()

#print the first 100 outputs
count = 0
while count<100:
    data = m.read_stream()
    if len(data)==0:
        continue
    else:
        count += 1
        print(data)

m.disable_stream()
```

(continues on next page)

(continued from previous page)

```
m.stop_logging()

m.set_zero_position()

m.close()
```

## 1.4 pybpodapi

**class** pybpod_rotaryencoder_module.module.**RotaryEncoder**(*connected=False*, *module_name=''*, *firmware_version=0*, *events_names=[]*, *n_serial_events=0*, *serial_port=None*)

Bases: pybpodapi.bpod_modules.bpod_module.BpodModule

**Note:** This API was based on the Rotary Encoder board documentation.

**create_resetpositions_trigger**()
Create a trigger to reset the positions and the threshods :return: trigger_id

**activate_outputstream**()
Activate module output stream.

**deactivate_outputstream**()
Deactivate module output stream.

**stop_streaming_and_logging**()
Stops streaming + logging.

**enable_positions_threshold**()
Enable all position thresholds.

**set_position_zero**()
Set current rotary encoder position to zero.

**starts_logging**()
Start logging position+time data to the microSD card.

**stops_logging**()
Finish logging position+time data to the microSD card.

**timestamp_byte**(*value*)
value as to be a byte

### 1.4.1 Usage example

```
#...

bpod = Bpod()

# get the module connected to the first bpod serial port.
rotary_encoder = bpod.modules[0]
```

(continues on next page)

(continued from previous page)

```
# call a function of the module
rotary_encoder.activate_outputstream()

bpod.stop()
```

# Python Module Index

## p

# Index

## T