pybpod-gui-plugin-emulator

Release 0.1.3

Nov 07, 2019

Contents

1 Contents **3**
- 1.1 Overview . . . 3
- 1.2 Installation . . . 4
- 1.3 Usage . . . 4
- 1.4 Reference . . . 10
- 1.5 Contributing . . . 10
- 1.6 Authors . . . 12
- 1.7 Changelog . . . 12
- 1.8 Indices and tables . . . 13

Index **15**

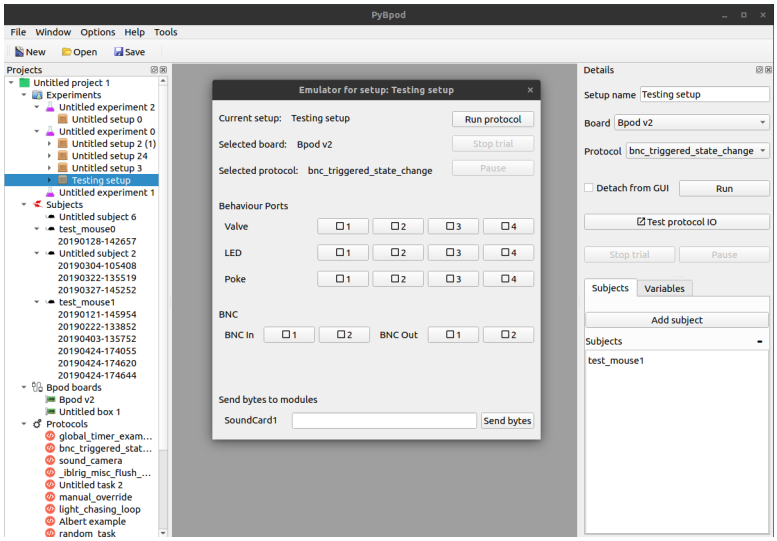

Fig. 1: PyBpod's Main window with the Emulator Window opened

This is the documentation for the Emulator plugin for PyBpod (https://pybpod.readthedocs.io/).

The Emulator plugin allows to override inputs and outputs on a running task protocol.

For further details you can see the *Overview* and the *Usage* pages. For installation instructions please see the *Installation* page.

CHAPTER 1

Contents

1.1 Overview

docs	
package	|

Emulator for PyBpod to work with the Bpod's State Machine ports.

At the moment, the Emulator for PyBpod module works by overriding inputs and outputs on a running task protocol. This will interact directly with a running State Machine in Bpod. As such, any event or state change that would occur naturally from any of those input or output changes, will occur.

- Free software: MIT license

1.1.1 Current Features

- Allows to override the Port components (i.e., LED, Poke and Valve)
- BNC In and Out value override
- Wire inputs and outputs override for Bpod 0.7
- Override Serial message for the connected modules (sends a bytes message)
- Messages are sent while the State Machine is running, triggering the events and/or state changes as if the values were coming from the real inputs/outputs.

1.1.2 Installation

Please see Installation page.

1.1.3 Documentation

https://pybpod-gui-plugin-emulator.readthedocs.io/

1.1.4 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

1.2 Installation

At the command line, in your environment:

```
pip install pybpod-gui-plugin-emulator
```

Afterwards, configure PyBpod to load the plugin:

1. On PyBpod's top menu, go to Options > Edit user settings.
2. Add 'pybpod_gui_plugin_emulator' to the end of the **GENERIC_EDITOR_PLUGINS_LIST** field:

```
GENERIC_EDITOR_PLUGINS_LIST = [ 'pybpodgui_plugin',
                    'pybpodgui_plugin_timeline',
                    ...
                    'pybpod_gui_plugin_emulator']
```

3. Restart PyBpod to load the new plugin.

1.3 Usage

After installing the plugin (please see *Installation*), a new *Test protocol IO* button will appear in each of the already configured Setups.

Warning: At the moment, it is required that a Bpod device is connected to the computer to run the module.

Note: The button will only be *active* when there is both a valid board and protocol selected in the Setup details.

When pressing the button, with a Bpod device connected, the window presented in the next figure will appear.

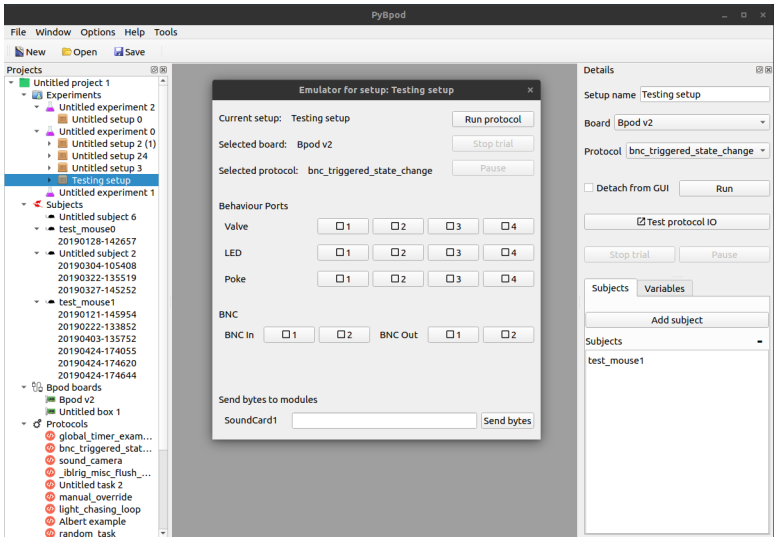

Fig. 1: PyBpod's Main window with the Emulator Window opened

At the top of the window it is possible to see the selected Setup, the selected Board and protocol. The buttons that are also available in the Setup details of PyBpod are also available in the Emulator window (i.e., Run, Stop trial and Pause).

Afterwards, a section with the Behaviour Ports is presented with three rows of buttons, each button for each available port. Each row represents the Valve, LED and the Poke.

Note: The Emulator window will **adapt automatically** depending on the Bpod device version connected. For example, when connecting a Bpod v0.7, each row for the Behaviour Ports will present 8 buttons, representing the 8 Behaviour Ports available in that model.

After the Behaviour Ports, a section with the BNC connections is displayed, with two buttons for the inputs and two for the outputs.

For Bpod v0.7 a new section with the Wire connections will appear after the BNC connections as it is possible to see in the next figure.

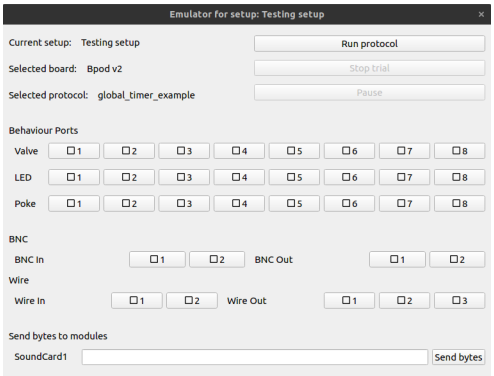

Fig. 2: Emulator Window for Bpod v0.7

When modules are connected to Bpod, they will also show up at the bottom of the window so it will be possible to send serial messages to those modules using the Emulator.

1.3.1 Interaction

To use the Emulator it is required, at the moment, that a device is connected and that a protocol is running. As such, the first step is to run the protocol using the appropriate button.

While the protocol is running, when pressing the different buttons for different actions, different events will be triggered.

As an example, if the Poke button 1 is pressed once (active state), it will trigger the 'Port1In' event. If pressed again (disabled state), it will trigger the 'Port1Out' event. As such, when running the example protocol presented below, which changes state when the 'Port1Out' event occurs, the PWM1 output channel (LED) will be turned on during the 3 seconds duration of the state 'Port3LightOn'. When pressing the Poke button 1 twice, both the 'Port1In' and 'Port1Out' events are triggered by Bpod as if there was a real interaction in the Poke of the Behaviour Port.

```
from pybpodapi.protocol import Bpod, StateMachine

my_bpod = Bpod()

sma = StateMachine(my_bpod)

sma.add_state(
    state_name='Port1LightOn',
    state_timer=1,
    state_change_conditions={Bpod.Events.Port1Out: 'Port3LightOn'},
    output_actions=[])

sma.add_state(
    state_name='Port3LightOn',
    state_timer=3,
    state_change_conditions={Bpod.Events.Tup: 'exit'},
    output_actions=[(Bpod.OutputChannels.PWM1, 255)])

my_bpod.send_state_machine(sma)

my_bpod.run_state_machine(sma)

print("Current trial info: {0}".format(my_bpod.session.current_trial))

my_bpod.close()
```

As it can be seen from this example, the protocol written can be used either with the plugin or directly, with no changes necessary to test the input and output ports and if the events are being triggered as expected.

The available input and output channel names, as well as the event names, for both the Bpod v0.7 and Bpod v2 are presented in the next sections.

Note: For either case of the Bpod's hardware version, it is assumed that firmware version **22** is installed.

1.3.2 Input channel names

Input channel	Bpod v0.7	Bpod v2
Serial1	✓	✓
Serial2	✓	✓

Continued on next page

Table 1 – continued from previous page

Input channel	Bpod v0.7	Bpod v2
Serial3	✓	✓
Serial4		✓
Serial5		✓
USB1	✓	✓
BNC1	✓	✓
BNC2	✓	✓
Wire1	✓	
Wire2	✓	
Port1	✓	✓
Port2	✓	✓
Port3	✓	✓
Port4	✓	✓
Port5	✓	
Port6	✓	
Port7	✓	
Port8	✓	
GlobalTimer1	✓	✓
GlobalTimer2	✓	✓
GlobalTimer3	✓	✓
GlobalTimer4	✓	✓
GlobalTimer5	✓	✓
GlobalTimer6		✓
GlobalTimer7		✓
GlobalTimer8		✓
GlobalTimer9		✓
GlobalTimer10		✓
GlobalTimer11		✓
GlobalTimer12		✓
GlobalTimer13		✓
GlobalTimer14		✓
GlobalTimer15		✓
GlobalTimer16		✓

1.3.3 Output channel names

Output channel	Bpod v0.7	Bpod v2
Serial1	✓	✓
Serial2	✓	✓
Serial3	✓	✓
Serial4		✓
Serial5		✓
SoftCode	✓	✓
BNC1	✓	✓
BNC2	✓	✓
Wire1	✓	
Wire2	✓	
Wire3	✓	
PWM1	✓	✓

Continued on next page

Table 2 – continued from previous page

Output channel	Bpod v0.7	Bpod v2
PWM2	✓	✓
PWM3	✓	✓
PWM4	✓	✓
PWM5	✓	
PWM6	✓	
PWM7	✓	
PWM8	✓	
Valve1	✓	✓
Valve2	✓	✓
Valve3	✓	✓
Valve4	✓	✓
Valve5	✓	
Valve6	✓	
Valve7	✓	
Valve8	✓	
GlobalTimerTrig	✓	✓
GlobalTimerCancel	✓	✓
GlobalCounterReset	✓	✓

1.3.4 Event names

Note: In the following table, to reduce the size of the table, a convention was defined to aggregate several names of the events. For example, where it can be read Serial1_[1-15], it means that we can have Serial1_1, Serial1_2, until Serial1_15.

Event names	Bpod v0.7	Bpod v2
Serial1_[1-15]	✓	✓
Serial2_[1-15]	✓	✓
Serial3_[1-15]	✓	✓
Serial4_[1-15]		✓
Serial5_[1-15]		✓
SoftCode[1-15]	✓	✓
BNC1High	✓	✓
BNC1Low	✓	✓
BNC2High	✓	✓
BNC2Low	✓	✓
Port1In	✓	✓
Port1Out	✓	✓
Port2In	✓	✓
Port2Out	✓	✓
Port3In	✓	✓
Port3Out	✓	✓
Port4In	✓	✓
Port4Out	✓	✓
Port5In	✓	
Port5Out	✓	
Port6In	✓	

Continued on next page

Table 3 – continued from previous page

Event names	Bpod v0.7	Bpod v2
Port6Out	✓	
Port7In	✓	
Port7Out	✓	
Port8In	✓	
Port8Out	✓	
GlobalTimer1_Start	✓	✓
GlobalTimer2_Start	✓	✓
GlobalTimer3_Start	✓	✓
GlobalTimer4_Start	✓	✓
GlobalTimer5_Start	✓	✓
GlobalTimer6_Start		✓
GlobalTimer7_Start		✓
GlobalTimer8_Start		✓
GlobalTimer9_Start		✓
GlobalTimer10_Start		✓
GlobalTimer11_Start		✓
GlobalTimer12_Start		✓
GlobalTimer13_Start		✓
GlobalTimer14_Start		✓
GlobalTimer15_Start		✓
GlobalTimer16_Start		✓
GlobalTimer1_End	✓	✓
GlobalTimer2_End	✓	✓
GlobalTimer3_End	✓	✓
GlobalTimer4_End	✓	✓
GlobalTimer5_End	✓	✓
GlobalTimer6_End		✓
GlobalTimer7_End		✓
GlobalTimer8_End		✓
GlobalTimer9_End		✓
GlobalTimer10_End		✓
GlobalTimer11_End		✓
GlobalTimer12_End		✓
GlobalTimer13_End		✓
GlobalTimer14_End		✓
GlobalTimer15_End		✓
GlobalTimer16_End		✓
GlobalCounter1_End	✓	✓
GlobalCounter2_End	✓	✓
GlobalCounter3_End	✓	✓
GlobalCounter4_End	✓	✓
GlobalCounter5_End	✓	✓
GlobalCounter6_End		✓
GlobalCounter7_End		✓
GlobalCounter8_End		✓
Condition1	✓	✓
Condition2	✓	✓
Condition3	✓	✓
Condition4	✓	✓

Continued on next page

Table 3 – continued from previous page

Event names	Bpod v0.7	Bpod v2
Condition5	✓	✓
Condition6		✓
Condition7		✓
Condition8		✓
Condition9		✓
Condition10		✓
Condition11		✓
Condition12		✓
Condition13		✓
Condition14		✓
Condition15		✓
Condition16		✓
Tup	✓	✓

1.4 Reference

1.4.1 pybpod_gui_plugin_emulator

class pybpod_gui_plugin_emulator.**EmulatorGUI**(*parent_win=None*)

Bases: pyforms_gui.basewidget.BaseWidget

Main GUI for the Emulator module. This GUI window adapts automatically to the different Bpod versions that are connected to the computer to present correctly the number of Ports available as well as the connected modules to the Bpod modules ports.

Parameters **parent_win** – The Setup object reference that this Emulator will be associated.

show()

Overrides the BaseWidget implementation of the show method in order to update the textual information of the board and protocol used, in case of being updated in the main window after creation of this EmulatorGUI window. :return:

update_board(*board*)

Method to update the board name :param board: The Board to be used to update the information in the UI, if available. :return:

update_task(*task*)

Method to update the task name :param task: The Task to be used to update the information in the UI, if available. :return:

1.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

1.5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.

- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

1.5.2 Documentation improvements

pybpod-gui-plugin-emulator could always use more documentation, whether as part of the official pybpod-gui-plugin-emulator docs, in docstrings, or even on the web in blog posts, articles, and such.

1.5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://bitbucket.org/fchampalimaud/pybpod-gui-plugin-emulator/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

1.5.4 Development

To set up *pybpod-gui-plugin-emulator* for local development:

1. Fork pybpod-gui-plugin-emulator (look for the "Fork" button).
2. Clone your fork locally:

```
git clone https://your_name_here@bitbucket.com:fchampalimaud/pybpod-gui-plugin-
↪emulator/pybpod-gui-plugin-emulator.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

1.6 Authors

- Luís Teixeira

1.7 Changelog

1.7.1 0.1.3 (2019-05-14)

- Fixed override messages not being sent properly on Windows
- Fix for pause not working

1.7.2 0.1.2 (2019-05-13)

- Fix for README to comply with PyPI support for reStructuredText

1.7.3 0.1.1 (2019-05-13)

- Fix for disappearing board on Setup window

1.7.4 0.1.0 (2019-05-03)

- First release on PyPI.
- Added support for Bpod version detection and automatic UI adaptation to the different input/output ports and connected modules
- Ports components can be overriden (i.e., LED, Poke and Valve)

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

- BNC In and Out value override
- Wire inputs and outputs override for Bpod 0.7
- Override Serial message for the connected modules (bytes message)

1.8 Indices and tables

- genindex
- modindex
- search

Index

E

EmulatorGUI (*class in pybpod_gui_plugin_emulator*), 10

S

show() (*pybpod_gui_plugin_emulator.EmulatorGUI method*), 10

U

update_board() (*pybpod_gui_plugin_emulator.EmulatorGUI method*), 10

update_task() (*pybpod_gui_plugin_emulator.EmulatorGUI method*), 10