Timeflux BITalino

Release 0.3.0

Pierre Clisson

Oct 03, 2021

CONTENTS

1 Installation **3**
1.1 API Reference 3

Python Module Index **7**

Index **9**

This plugin provides a driver to connect to BITalino devices.

When connecting a BITalino device for the first time, the default pairing code is: *1234*.

CHAPTER

ONE

INSTALLATION

First, make sure that Timeflux is installed.

You can then install this plugin in the *timeflux* environment:

```
$ conda activate timeflux
$ pip install timeflux_bitalino
```

1.1 API Reference

This page contains auto-generated API reference documentation.

timeflux_bitalino

1.1.1 timeflux_bitalino

timeflux_bitalino.helpers

helpers

timeflux_bitalino.helpers.transfer

Transfer functions

transfer

timeflux_bitalino.helpers.transfer.**VCC = 3.3**

timeflux_bitalino.helpers.transfer.**ECG**(*signal*, *resolution=10*)
ECG value in millivolt [-1.5, 1.5]

timeflux_bitalino.helpers.transfer.**EMG**(*signal*, *resolution=10*)
EMG value in millivolt [-1.64, 1.64]

timeflux_bitalino.helpers.transfer.**EDA**(*signal*, *resolution=10*)
EDA value in microsiemens [0, 25]

timeflux_bitalino.helpers.transfer.**EEG**(*signal*, *resolution=10*)
EEG value in microvolts [-39.49, 39.49]

timeflux_bitalino.helpers.transfer.**PZT**(*signal*, *resolution=10*)
EEG value in percents [-50%, 50%]

timeflux_bitalino.nodes

---

nodes

timeflux_bitalino.nodes.driver

---

driver

timeflux_bitalino.nodes.driver.**TRANSFER**

class timeflux_bitalino.nodes.driver.**Bitalino**(*port*, *rate=1000*, *channels=('A1', 'A2', 'A3', 'A4', 'A5', 'A6')*, *sensors=None*)

Bases: timeflux.core.node.Node

BITalino driver.

This node connects to a BITalino device and streams data at a provided rate. It is based on the original BITalino Python library, with some performance improvements and careful timestamping.

Two output streams are provided. The default output is the data read from the analog and digital channels. The `o_offsets` output provides continuous offsets between the local time and the estimated device time. This enables drift correction to be performed during post-processing, although no significant drift has been observed during testing.

Variables

- **o** (*Port*) – BITalino data, provides DataFrame.
- **o_offsets** (*Port*) – Time offsets, provide DataFrame.

Parameters

- **port** (*string*) – The serial port. e.g. `COM3` on Windows; `/dev/tty.bitalino-DevB` on MacOS; `/dev/ttyUSB0` on GNU/Linux.
- **rate** (*int*) – The device rate in Hz. Possible values: `1`, `10`, `100`, `1000`. Default: `1000`.
- **channels** (*tupple*) – The analog channels to read from. Default: `('A1', 'A2', 'A3', 'A4', 'A5', 'A6')`.
- **sensors** (*dict*) – The map of attached sensors. If set, transfer functions will be applied. e.g. `{"A1": "ECG", "A3": "EMG"}`. Default: `None`.

Example

```yaml
graphs:

  - id: acquisition
    nodes:
    - id: bitalino
      module: timeflux_bitalino.nodes.driver
      class: Bitalino
      params:
        port: /dev/tty.BITalino-02-44-DevB
        rate: 1000
        sensors:
          A1: ECG
          A2: EEG
          A3: EDA
    - id: pub_bitalino
      module: timeflux.nodes.zmq
      class: Pub
      params:
        topic: bitalino
    - id: pub_offsets
      module: timeflux.nodes.zmq
      class: Pub
      params:
        topic: offsets
    edges:
    - source: bitalino
      target: pub_bitalino
    - source: bitalino:offsets
      target: pub_offsets
    rate: 30

  - id: display
    nodes:
    - id: subscribe
      module: timeflux.nodes.zmq
      class: Sub
      params:
        topics: [ bitalino, offsets ]
    - id: ui
      module: timeflux_ui.nodes.ui
      class: UI
    - id: debug
      module: timeflux.nodes.debug
      class: Display
    edges:
      - source: subscribe:bitalino
        target: ui:bitalino
      - source: subscribe:bitalino
        target: debug
    rate: 10
```

(continues on next page)

(continued from previous page)

```
  # - id: record
  #   nodes:
  #   - id: sub
  #     module: timeflux.nodes.zmq
  #     class: Sub
  #     params:
  #       topics: [ bitalino, offsets ]
  #   - id: save
  #     module: timeflux.nodes.hdf5
  #     class: Save
  #   edges:
  #     - source: sub:bitalino
  #       target: save:bitalino
  #     - source: sub:offsets
  #       target: save:offsets
  #   rate: 1

  - id: broker
    nodes:
    - id: broker
      module: timeflux.nodes.zmq
      class: Broker
```

Notes:

Attention: Make sure to set your graph rate to an high-enough value, otherwise the device internal buffer may saturate, and data may be lost. A 30Hz graph rate is recommended for a 1000Hz device rate.

Instantiate the node.

update(*self*)
Update the input and output ports.

terminate(*self*)
Perform cleanup upon termination.

PYTHON MODULE INDEX

t

timeflux_bitalino, 3
timeflux_bitalino.helpers, 3
timeflux_bitalino.helpers.transfer, 3
timeflux_bitalino.nodes, 4
timeflux_bitalino.nodes.driver, 4

INDEX

B

Bitalino (*class in timeflux_bitalino.nodes.driver*), 4

E

ECG() (*in module timeflux_bitalino.helpers.transfer*), 3
EDA() (*in module timeflux_bitalino.helpers.transfer*), 3
EEG() (*in module timeflux_bitalino.helpers.transfer*), 3
EMG() (*in module timeflux_bitalino.helpers.transfer*), 3

M

module
    timeflux_bitalino, 3
    timeflux_bitalino.helpers, 3
    timeflux_bitalino.helpers.transfer, 3
    timeflux_bitalino.nodes, 4
    timeflux_bitalino.nodes.driver, 4

P

PZT() (*in module timeflux_bitalino.helpers.transfer*), 4

T

terminate() (*timeflux_bitalino.nodes.driver.Bitalino method*), 6
timeflux_bitalino
    module, 3
timeflux_bitalino.helpers
    module, 3
timeflux_bitalino.helpers.transfer
    module, 3
timeflux_bitalino.nodes
    module, 4
timeflux_bitalino.nodes.driver
    module, 4
TRANSFER (*in module timeflux_bitalino.nodes.driver*), 4

U

update() (*timeflux_bitalino.nodes.driver.Bitalino method*), 6

V

VCC (*in module timeflux_bitalino.helpers.transfer*), 3