API Documentation

API Documentation

December 30, 2015

Contents

1 Module liveplot

This module provides the class LivePlot which can plot data from a serial device live.

Tested on Ubuntu 15.10 with Arduino Uno.

Author: Sverre

1.1 Functions

example()

Example use of LivePlot.

The serial device is an Arduino Uno connected on /dev/ttyACM0. The Arduino is connected to a TMP36 sensor with +V_S connected to +5.0 VCC and V_OUT connected to A0. The Arduino is connected on the port /dev/ttyACM0 and loaded with the following sketch:

```
const int sensor = 0;

void setup(){
    Serial.begin(9600);
}

void loop(){
    delay(1000);
    int signal = analogRead(sensor);
    Serial.println(signal);
}
```

1.2 Variables

Name	Description
__package__	**Value:** `None`

1.3 Class LivePlot

object

multiprocessing.process.Process

liveplot.LivePlot

This class handles extends multiprocessing.Process so that the computation and plotting happens in a new thread.

To start the plotting, call the method `start()`. The thread will close if the main thread closes, so if no work is done, nothing will be plotted. This can be avoided by calling `raw_input()`.

The thread can be closed with the method `join()`.

1.3.1 Methods

___init___(*self*, *ser*, *comp*, *dec*=`None`, *prop*=`None`, *save*=`None`, *clean*=`None`, *cb*=`None`, *verb*=`False`)

x.___init___(...) initializes x; see help(type(x)) for signature

Parameters

- `ser`: The serial device to communicate with.

  (type=Serial)
- `comp`: The function to be used on the input from the serial device to convert it to something plotable.

  (type=`callable`)
- `dec`: String with decorations for the plot such as symbol for data points.

  (type=`str`)
- `prop`: Properties of the plot such as alpha and color.

  (type=`tuple`)
- `save`: File to save the processed data to.

  (type=`str`)
- `cb`: Not implemented.
- `clean`: Not implemented.
- `verb`: Boolean indicating if the data is to be printed to the terminal.

  (type=`bool`)

Overrides: object.___init___

To Do:

- Implement callback possibilities.
- Implement the possibility to only read new serial data.

run(*self*)

Read and process the data from the serial device, plot it and potentially save it to a file. The thread runs continuously until `join()` is called or the script terminates.

Overrides: multiprocessing.process.Process.run

join(*self*)

Stops the thread safely.

Overrides: multiprocessing.process.Process.join

plot(*self*)

Plots the data.

save_data(*self*, *data*)

Saves the data to a file.

Parameters

`data:` The data to be saved.

Inherited from multiprocessing.process.Process

__repr__(), is_alive(), start(), terminate()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.3.2 Properties

Name	Description
Inherited from multiprocessing.process.Process	
authkey, daemon, exitcode, ident, name, pid	
Inherited from object	
__class__	

Index