LineUsPythonModule

Release 1.0.1

Robert Poll

Oct 14, 2020

CONTENTS:

Line-us is an internet connected robot drawing arm. It's small, portable and draws with a nice wobbly line using a real pen on paper. The free app lets you draw, send messages, share sketchbooks or enjoy collecting artworks from others!

Line-us was created in London by Durrell Bishop and Robert Poll. There is a lot more information on our website: https://www.line-us.com

If you have any questions please drop us an email: help@line-us.com of post on our forum: https://forum.line-us.com

CHAPTER

ONE

QUICKSTART

It's easy to get started with the Line-us module in Python, but first make sure you are running Python3 and have the module installed:

```
$ python --version
Python 3.7.5

$ pip install lineus
```

We would recommend using a virtual environment. Pipenv is my preference but there are plenty of options.

Begin by importing the LineUs class and creating a LineUs object:

```
>>> from lineus import LineUs
>>> my_line_us = LineUs()
```

The LineUs object will then immediately start listening for Line-us machines on your local network. If you only have one you can connect to it with:

```
>>> my_line_us.connect()
```

Which will return `True` if the connection was successful. Once you're connected you can start to send commands. If you want to draw you'll be sending `G01` so can do something like:

```
>>> my_line_us.g01(900, 300, 0)
>>> my_line_us.g01(900, -300, 0)
>>> my_line_us.g01(900, -300, 1000)

>>> my_line_us.g01(1200, 300, 0)
>>> my_line_us.g01(1200, -300, 0)
>>> my_line_us.g01(1200, -300, 1000)

>>> my_line_us.g01(900, 0, 0)
>>> my_line_us.g01(1200, 0, 0)
>>> my_line_us.g01(1200, 0, 1000)

>>> my_line_us.g01(1500, 150, 0)
>>> my_line_us.g01(1500, -300, 0)
>>> my_line_us.g01(1500, -300, 1000)

>>> my_line_us.g01(1500, 250, 0)
>>> my_line_us.g01(1500, 300, 0)
>>> my_line_us.g01(1500, 300, 1000)
```

It's a good idea to disconnect cleanly from your Line-us:

```
>>> my_line_us.disconnect()
```

That should cover the vast majority of what you need for most uses, but there's more information on the full API below.

Have fun!!

CHAPTER

TWO

THE LINEUS CLASS IN DETAIL

class lineus.**LineUs**

The Python module for Line-us. The module allows you to easily control your Line-us from Python using the TCP API which allow full access to all of the Line-us GCodes. Create a Line-us instance using:

```
>>> my_line_us = LineUs()
```

It is worth creating the `LineUs()` object as early as possible in your code as the search for machines begins as soon as the objet is created.

connect(*line_us_name=None*, *wait=2*, *timeout=None*)

Connect to a Line-us. If `line_us_name` is not specified then the module will connect to the first Line-us that it finds. The Bonjour search starts when the LineUs object is created and it may take some time to discover the Line-us machines so the `connect()` function allows you to set a wait time (default 2s) to allow discovery. A timeout for the TCP connection can also be set. The default is `None`, so the connection will wait forever. The simplest form of connect is:

```
>>> my_line_us.connect()
```

Returns `True` if the connection was successful.

connected()

Returns `True` if a Line-us is connected

disconnect()

Close the connection to the Line-us. Returns True.

g01(*x=None*, *y=None*, *z=None*)

Send a G01 (interpolated move), and wait for the response before returning. One or more of x, y and z must be specified. Some example commands are:

```
>>> my_line_us.g01(1000, 0, 1000)      # (x, y, z)
>>> my_line_us.g01(z=0)
>>> my_line_us.g01(1000, 500)          # (x, y)
```

Returns the reply message from Line-us, for Example:

```
ok X:1000.00 Y:0.00 Z:1000.00
```

get_hello_string()

Returns the hello string sent by Line-us wen you connect. The hello string is stored so you can run `get_hello_string()` at any time that a Line-us is connected. If no Line-us is conencted it will return `None`. The hello string has a form similar to:

```
{'VERSION': '3.2.0 Nov 22 2019 11:28:36', 'NAME': 'line-us', 'SERIAL':
↪'1575520'}
```

get_info()

Returns a dictionary with infomation about the Line-us that is currently connected. Refer to the GCode spec in the Programming section of https://Line-us.com for more detail, but tan example is:

```
{'ChipID': '1575520',
 'WifiMode': '1',
 'WifiModeSet': '0',
 'WifisConfigured': '1',
 'MemDraw': '0',
 'Gestures': '0',
 'ContinuousDrawing': '0',
 'DrawingCount': '0',
 'name': 'line-us-dev',
 'mac': 'C:3A:E8:18:0A:60',
 'FlashChipID': '0x1640ef',
 'FlashChipMode': '0',
 'FlashChipSpeed': '40000000',
 'FreeHeap': '25728',
 'ResetReason': 'External System',
 'Uptime': '0d0h12m11s',
 'Time': 'Fri Nov 29 16',
 'FSUsed': '112197',
 'FSTotal': '1953282',
 'FSFree': '1841085',
 'FSPercent': '5',
 'FS': '/0000001.txt-109291;/cal-29;/key-344; ',
 'Serial': 'ikdBW+',
 'Cal': '10.79735,-1.902405,9.900639',
 'ZMap': '0;100;300;200',
 'ServoReverse': '0,0,1'}
```

get_line_us_list()

Returns an list with an element for each of the Line-us that the module has discovered. Each element in the list is a tuple of: `(name, bonjour_name, ip_address)`. You can use either the bonjour_name or the ip_address in the `connect()` function to connect, although in most circumstances ip_address is a better option as it saves a name lookup. You do not need to be connected to a Line-us to use this function:

```
>>> my_line_us = LineUs()
>>> my_line_us.get_line_us_list()
[('line-us-dev', 'line-us-dev.local.', '192.168.27.223', 1337), ('line-us-rob
↪', 'line-us-rob.local.', '192.168.27.150', 1337)]
```

get_name()

Returns the name of the Line-us that you are connected to. If you are not connected to a Line-us it will return `None`.

list_lineus_files()

This function returns a list of the files stored on your Line-us. Each element in the list is a tuple of `(file_number, size, fine_name)`. For Example:

```
>>> my_line_us.list_lineus_files()
[('1', '109291', '/0000001.txt'), ('2', '51765', '/0000002.txt')]
```

ping(*line_us_name*, *count=5*)

The `ping()` command tests the speed of the connection to a Line-us. You should be disconnected before calling it. It returns a `dict` with the following information:

```
{'mean': 3.7004387999999944,
 'min': 2.9651769999999855,
 'max': 5.283999999999955,
 'stdev': 1.025966272915769}
```

save_to_lineus(*gcode*, *position*)

Save a drawing to the Line-us internal memory. The `position` parameter is the file numebr to save to and must be between 1 and 32. The `gcode` parameter is a string with the entrie gcode to save with each GCode separated with `\n`, for example to save GCode to file number 2:

```
>>> gcode = 'G28\nG01 X1000 Y0\nG01 X1000  Y1000\n'
>>> my_line_us.save_to_lineus(gcode, 2)
```

The function returns `ok`

send_gcode(*gcode*, *parameters=''*)

Send an arbitrary GCode to Line-us. Refer to the GCode guide for details on supported commands. Some example uses are:

```
>>> my_line_us.send_gcode('G28')                        # Go to home position
>>> my_line_us.send_gcode('M550', 'Pline-us-rob')       # Re-name your Line-us
```

The function returns the response from Line-us

send_raw_gcode(*gcode*)

Send a raw GCode to Line-us. It is your responsibility to construct a vaild GCode, For example:

```
>>> my_line_us.send_raw_gcode('M550 Pline-us')
```

The function returns the response from Line-us

set_timeout(*timeout*)

This function sets the TCP timeout in seconds for the TCP connection to your Line-us. For example to set the timeout to 1.5 seconds:

```
>>> my_line_us.set_timeout(1.5)
```

Returns `` `True` `` if the timeout was successfully set.

CHAPTER

THREE

INDEX AND SEARCH

- genindex
- search

PYTHON MODULE INDEX

INDEX