pywfom

Release 0.0.1-beta

Ryan Byrne

Apr 23, 2021

BACKGROUND:


PYWFOM

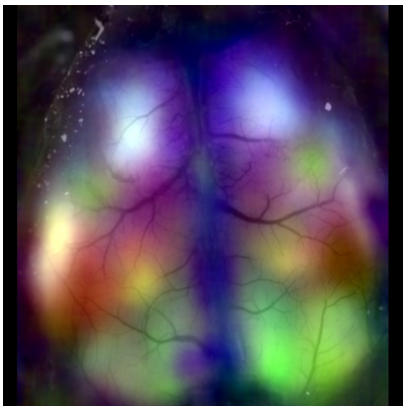
CHAPTER

ONE

WIDE-FIELD OPTICAL MAPPING (BUT PYTHON)

1.1 About WFOM

Wide-Field Optical Mapping (WFOM) is the product of the Columbia University's Laboratory for Functional Optical Imaging, run by Elizabeth Hillman



1.1.1 Technique

WFOM works by utilizing a high-speed CCD camera's exposure signal to trigger arrays of high-powered LEDs, set to specified wavelengths.

A paper describing the technique further can be found here.

1.1.2 Applications

WFOM is currently being used in a **wide** (•˘•˘) range of applications, such as:

Neurovascular Coupling

Fig. 1: Seizure caused by tumor growth imaged using WFOM

Resting State Dynamics

Fig. 2: Resting state neural and hemodynamic activity in the awake mouse brain

For a comprehensive list of WFOM applications, visit this page

1.1.3 Why PyWFOM?

`PyWFOM` combines code previously spread across numerous platforms, devices, and languages into a single, user-friendly Python Package.

Users are able to set camera parameters, stim functions, and data acquisition all in one place.

1.2 Create Virtual Machine

It is **highly recommended** you use a Virtual Environment when installing `pywfom`. This is done by running the following commands:

```
python3 -m venv myWFOM
source myWFOM/bin/activate
```

1.3 Installing PyWFOM

1.3.1 System Requirements

- Windows 10 or Linux
- Python 3.5+

1.3.2 w/ PIP

The easiest way to install `pywfom` is through the Python Package Manager, PIP

```
pip install pywfom
```

1.3.3 From Source

pywfom's Source Code is hosted on Github.

```
git clone https://github.com/ryan-byrne/pywfom.git
cd pywfom
python setup.py install
```

1.4 Arduino Setup

Running pywfom requires first setting up an Arduino to be used with the system.

1.4.1 Installing the Arduino IDE & Drivers

Download the Arduino IDE for your Operating System and follow the instructions on your screen.

Any required USB Drivers will be installed alongside the Arduino IDE.

1.4.2 Deploying to the Arduino

1. Attach the Arduino you wish to use with your pywfom system to your machine via USB.
   - **NOTE:** Arduino MEGA is suggested
2. Download the pyWFOM Arduino File
3. Start the Arduino IDE, and open the pyWFOM Arduino File
4. Verify the correct device and port are selected
5. Deploy the code to the Arduino
6. pywfom is now able to send settings to your Arduino

1.4.3 Attaching Devices

Adding devices to your pywfom system is as simple as attaching them to the pins of your Arduino.

The example below shows 3 separate BNC connectors attached to an exposure trigger from a **sCMOS camera** and **two LED drivers**.

pywfom would send this information to the Arduino using the strobing setting in config.json.

Take a look at the section on the *JSON Configuration File*.

```
{
  "strobing": {
    "leds":[
      {
        "name":"led1",
        "pin":2
      },
      {
        "name":"led2",
        "pin":5
```

(continues on next page)

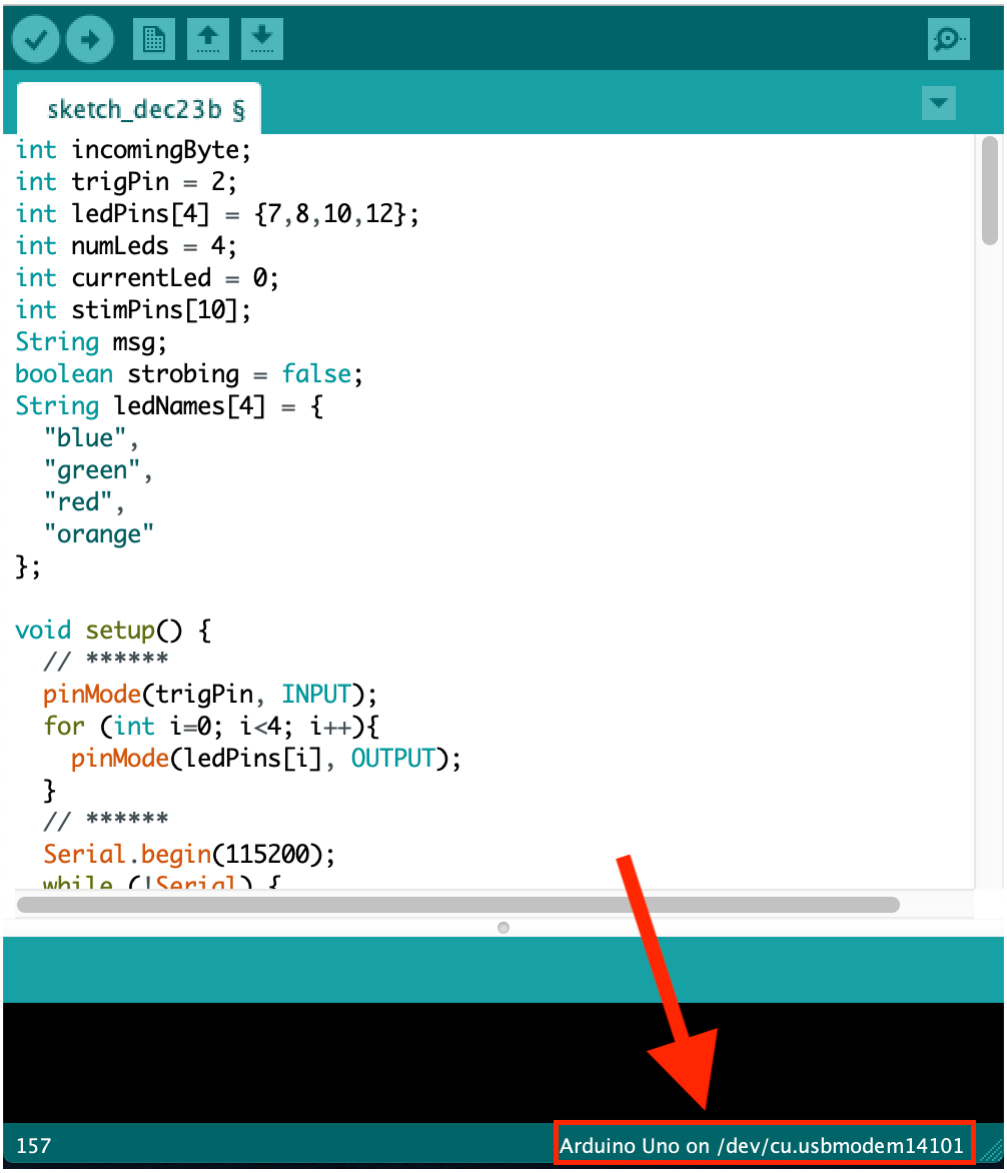


Fig. 3: These can be changed from the `Tools` Menu

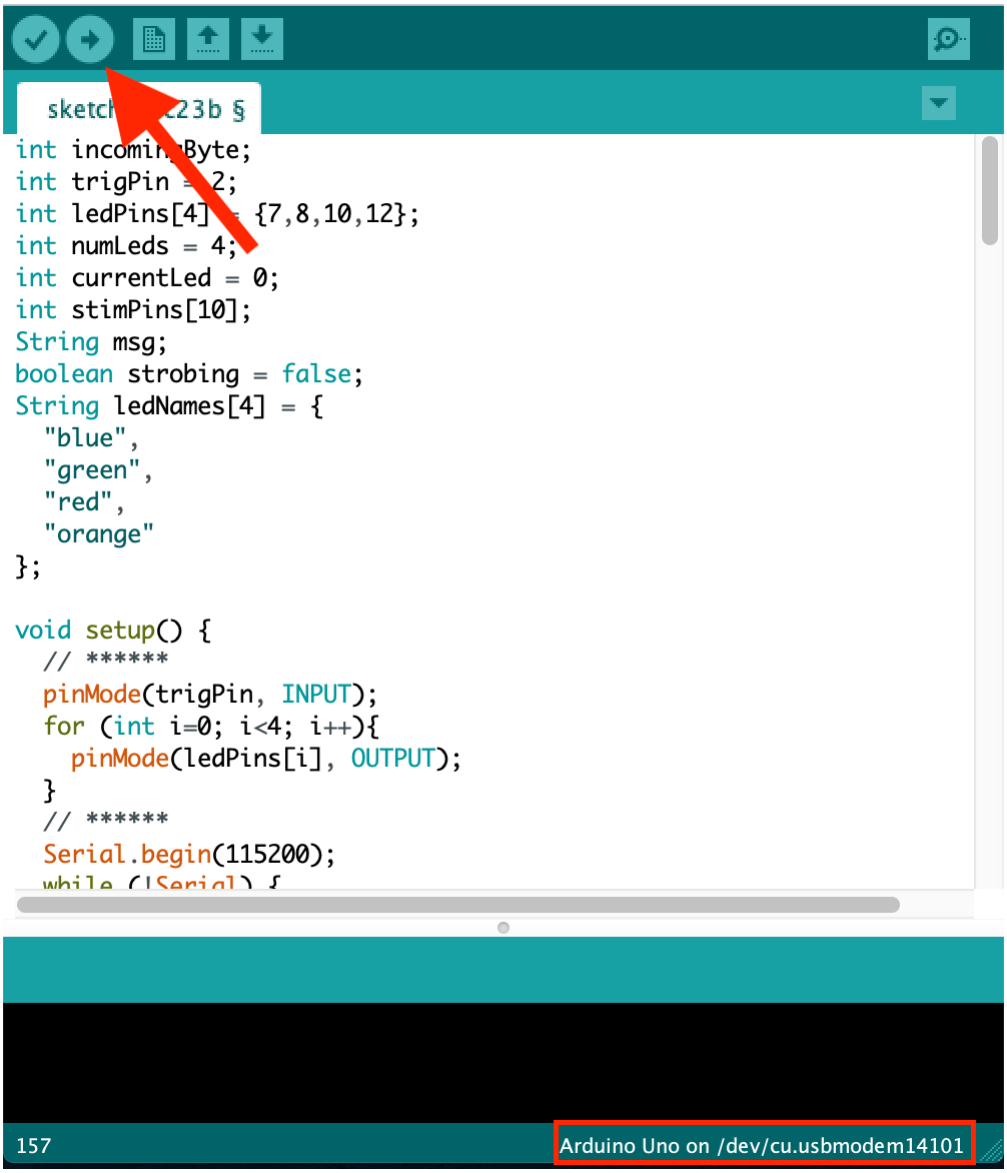


Fig. 4: Wait until the code successfully deploys

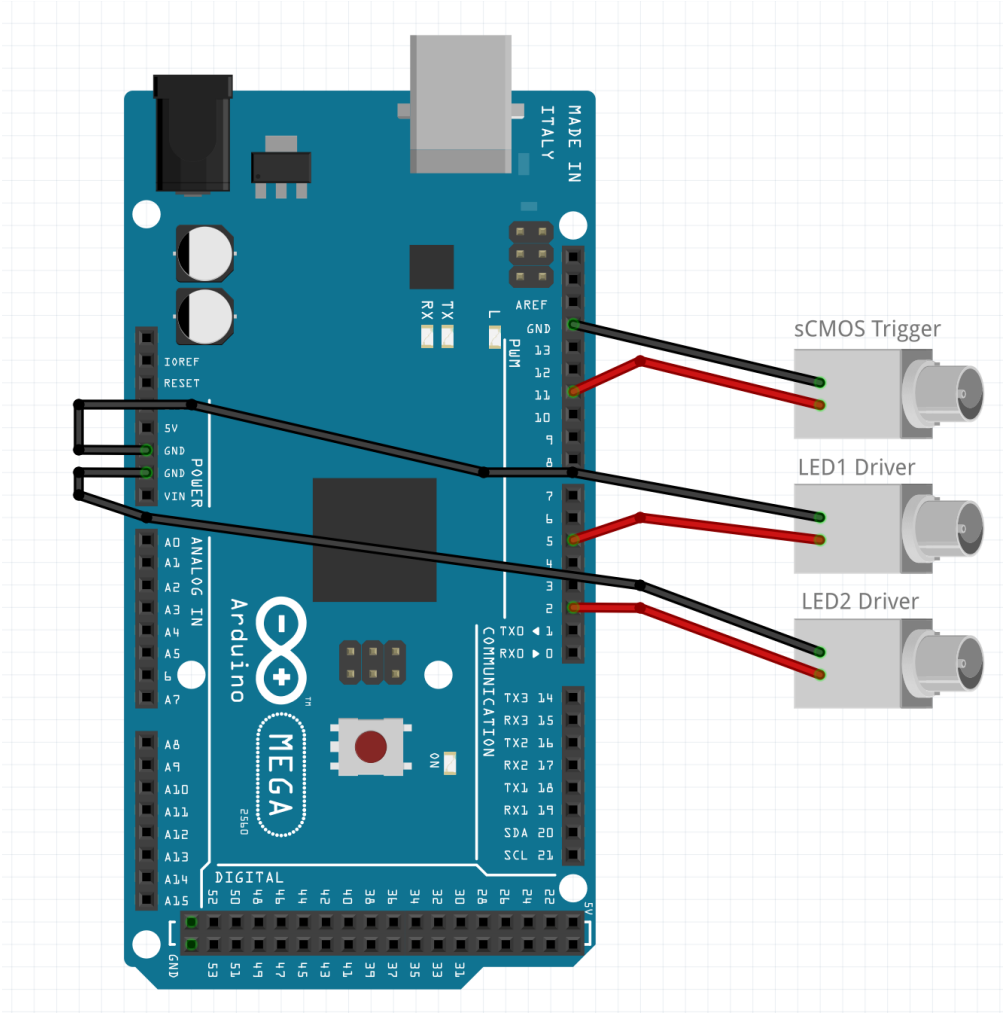


Fig. 5: Simple wiring diagram including a single sCMOS Camera and 2 LEDs

(continued from previous page)

```
        }
      ],
      "trigger":11
    ]
}
```

1.5 Quickstart

Once you've completed *Installing PyWFOM* and *Arduino Setup*, the system can start to be configured.

The quickest way to do this is running the *wfom-quickstart* command.

```
wfom-quickstart
```

1.6 API Documentation

1.6.1 System Interface

1.6.2 Camera Interface

1.6.3 Arduino Interface

1.7 Command Line Tools

The simplest way to configure, acquire, and view runs is with one of `pywfom`'s available *Command Line Tools*.

Command	Description
wfom	Creates a *System Interface* and opens the *Main Frame*
wfom-viewer	View *Acquisition Files* in the *Run Viewer*
wfom-quickstart	Quickly start an acquisition using the default settings

1.7.1 wfom

Arguments

Argument	Name	Description
-v	**–verbose**	Determines whether pyWFOM prints to the console
-t	**–test**	Runs pyWFOM in 'Test Mode'
-c	**–config**	Include a string for the location of a JSON Config File
-s	**–solis**	Runs pyWFOM in 'Solis Mode'

```
wfom -v -c path/to/config/myConfiguration.json
```

1.7.2 wfom-viewer

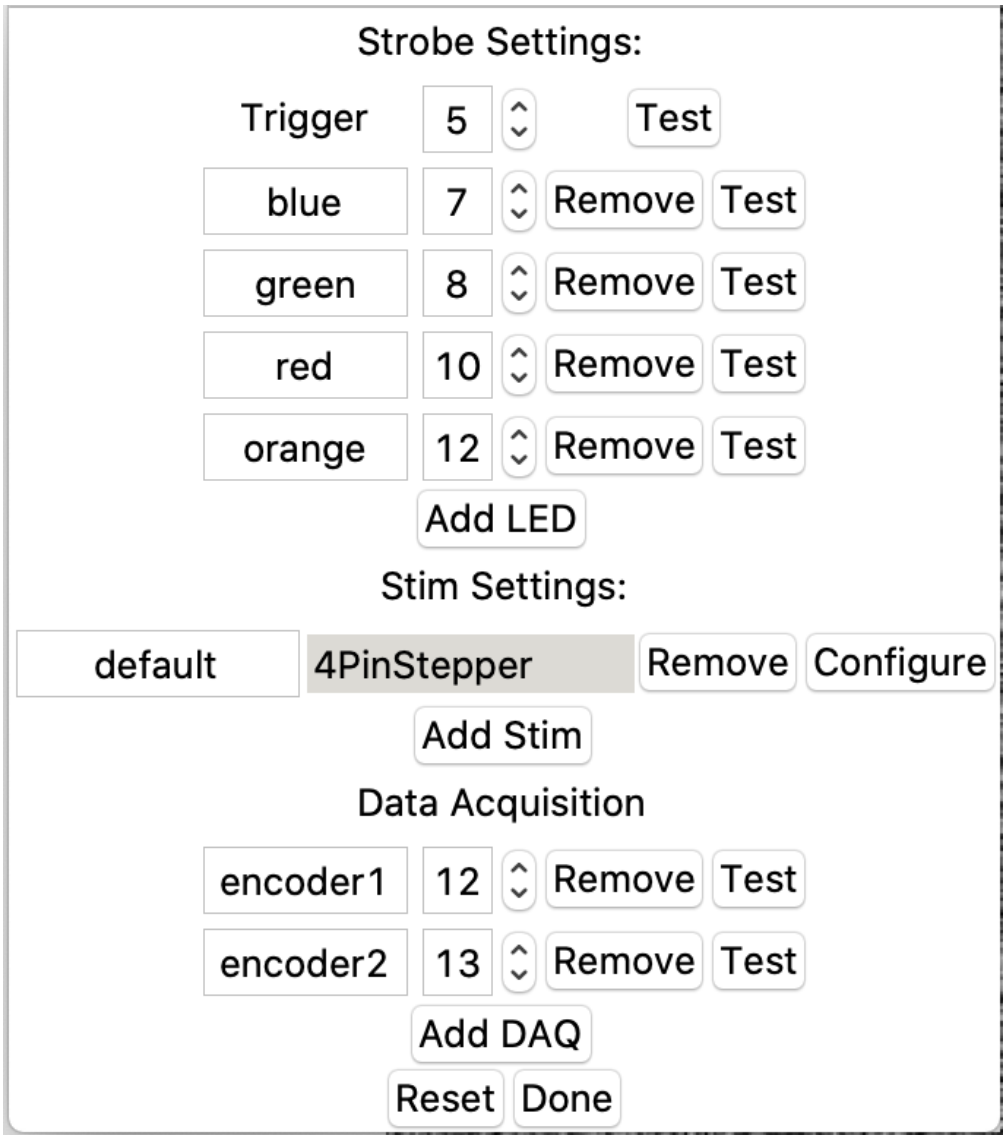
1.7.3 wfom-quickstart

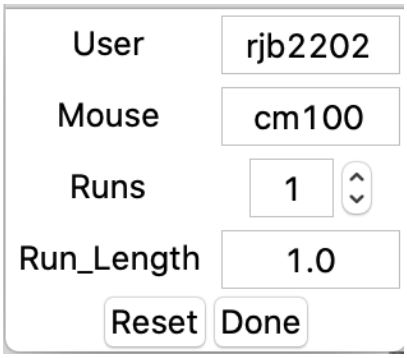
1.8 Graphical User Interfaces

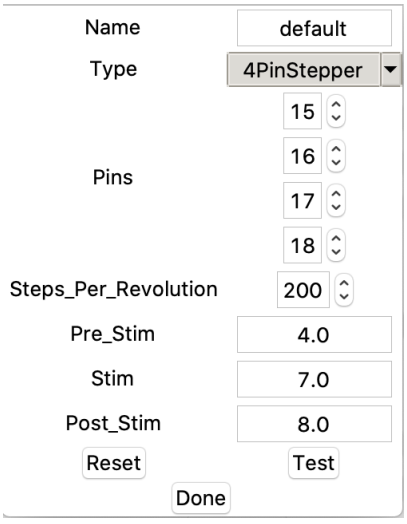
1.8.1 Main Frame

1.8.2 Run Viewer

1.8.3 Additional Windows




User
rjb2202
Mouse
cm100
Runs
1
Run_Length
1.0
Reset
Done


Name
default
Type
4PinStepper
Pins
15
16
17
18
Steps_Per_Revolution
200
Pre_Stim
4.0
Stim
7.0
Post_Stim
8.0
Reset
Test
Done

1.9 Acquisition Files

Raw Data is stored as a individual frames in a `run` directory. `frame` file are numpy array, and saved as an npz file with the following structure.

1.9.1 Structure

```
run12
├── config.json
├── frame0.npz
|    ├── cam0
|    |    └── array
|    ├── cam0
|    |    └── array
|    └──arduino
|         └── message
├── frame1.npz
.
.
.
└── frameN.npz
```

1.9.2 Numpy Frame

1.10 JSON Configuration File

`pywfom` uses a JSON file to store various metadata and settings.

Setting	Description	Type	Example
user	Name or ID of individual who ran the acquisition.	string	“rjb2202”
mouse	Name or ID of the mouse the acquisition was conducted on.	string	“cm100”
directory	Location data will be saved to	string	“C:/data”
runs	Number of runs for given acquisition	int	5
run_length	Length of each acquisition (in seconds)	float	10.0
cameras	List of camera settings	list	See *Cameras*
arduino	Dictionary of arduino settings	dict	See *Arduino*

NOTE: It is highly recommended you only alter the your *JSON Configuration File* , **do not directly edit the file itself**.

1.10.1 Example JSON Configuration

```
{
  "user":"rjb2202",
  "mouse":"cm100",
  "directory":"C:/data",
  "runs": 5,
  "run_length": 2.0
  "arduino": {}
  "cameras": []
}
```

1.10.2 Arduino

```
{
  "arduino": {
    "port": "COM4",
    "data_acquisition":[
      {
        "name":"encoder",
        "pin":20
      }
    ],
    "strobing": {
      "leds":[
        {
          "name":"blue",
          "pin":7
        },
        {
          "name":"green",
          "pin":8
        }
      ],
      "trigger":2
    },
    "stim": [
      {
        "name":"default",
        "type":"2PinStepper",
        "pins":{
          "step":5,
          "dim":6
        },
        "pre_stim":4.0,
        "stim":7.0,
        "post_stim":8.0
      }
    ]
  }
}
```

1.10.3 Cameras

```
{
  "cameras": [{
    "device": "test",
    "index": 0,
    "name": "cam1",
    "height": 564,
    "width": 420,
    "offset_x": 524,
    "offset_y": 157,
    "binning": "1x1",
    "dtype": "uint16",
    "master": true,
    "framerate": 20.0
```

(continues on next page)

(continued from previous page)

```
  }, {
    "device": "test",
    "index": 0,
    "name": "cam3",
    "height": 500,
    "width": 400,
    "offset_x": 1,
    "offset_y": 50,
    "binning": "1x1",
    "dtype": "uint16",
    "master": false,
    "framerate": 10.0
  }]
}
```

1.10.4 Default Configuration

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search