Protocol Section 3.4: Python Syntax to program patterns of light stimulation

The following is an example python code programmed via the Mate Terminal of a Raspberry Pi 2B operating on Ubuntu Mate v16.04, freely available at https://github.com/yslo/nubuntu.

Open Ubuntu Mate terminal.

Create a new Python script with the following command;
sudo nano NameofExperiment.py

Begin your experiment from the terminal with the following command;
Sudo python NameofExperiment.py

Preprogram the number of successive iterations of the experiment using the following commands;

```
import RPi.GPIO as GPIO
import time
numTimes = 1
iterations = numTimes
```

Initialize GPIO pins leading to LEDs with the following commands;

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(First GPIO pin used, GPIO.OUT)
GPIO.setup(Second GPIO pin, GPIO.OUT)

for i in range(0, numTimes)
print “Iteration “ +str(i+1)
GPIO.output(2, True)          (Boolean value “True” refers to LEDs  OFF & “False” refers to LEDs ON.)
GPIO.output(4, True)
time.sleep(60)

GPIO.output(2, False)
GPIO.output(4, False)
time.sleep(60)

GPIO.output(2, True)
GPIO.output(4, True)
time.sleep(60)          (For this expt., LEDs were OFF for 60 s, ON for 60s, then OFF for 60s)

print “Done”
```

Clear GPIO pin commands

```
GPIO.cleanup()
```

Protocol section 4: Matlab Syntax for Tracklarva

Select movie files using the following commands

```
[dir_path]      = uigetdir;
dirs            = dir([dir_path, '/directory*']);
num_movies      = size(dirs,1);
```

Set parameters using the following commands

```
xmax            = 1885;          (max num pixels in x direction)
ymax            = 1886;          (max num pixels in y direction)
pixelSize       = 0.265;         (millimeters)
maxSpeed        = 20;            (in pixels)
maxGaps         = 3;             (in time steps)
A_const         = 20;
bodyLength      = 16.771;
bodyArea        = 70;
```

Process each movie using the following commands

```
I.num_tracks    = zeros(num_movies,1);

for imov        = 1:num_movies
   s_mov        = regexp(dirs(imov).name, '_', 'split');
   dirname      = [dir_path '/' dirs(imov).name '/filename' s_mov{end} '_'];

   times        = linspace(0, 300, 650);
   deltat       = mean(diff(times));

   dir_img      = dir([dirname '/*.tif']);
   numImages    = size(dir_img,1);
```

Calculate mean image and subtract mean border from each frame using the following commands

```
filename        = [dirname '/' dir_img(1).name];
avgframe        = im2double(imread(filename));
kk = 0;
        for i=10:10:numImages
     kk = kk+1;
     filename    = [dirname '/' dir_img(i).name];
     avgframe    = avgframe + im2double(imread(filename));
        end
   avgframe     = avgframe/kk;
```

Identify x- and y- coordinates for larvae in each frame using the following commands

```
     peaks      = cell(1,numImages);
  for i=1:numImages
```

```
    filename    = [dirname '/' dir_img(i).name];
    frame       = im2double(imread(filename));
    bwframe     = im2bw(frame - avgframe,Thwb)';
    stats       = regionprops(bwframe, 'Area', 'Orientation', 'Centroid');
    a           = cell2mat({stats.Area});
    stats(a<100)=[];
    peak        = zeros(numel(stats),9);
    coord       = cell2mat({stats.Centroid}');
    peak(:,1)   = coord(:,1);                          x-coordinates
    peak(:,2)    = coord(:,2);                         y-coordinates
    peak(:,3)   = [stats.Area]';
    peak(:,4)   = [stats.Orientation]'+ 90;
    peaks{i}    = peak;
  end
```

Build navigation trajectories from data

```
if  nargin==1
        thangle = 45;
end
if  (nargin==1) || (nargin==2)
        maxtime = 60;
end

Tracks          = [ ];
for imov        = 1:num_movies
  s_mov         = regexp(dirs(imov).name, '_', 'split');
  dirname       = [dir_path '/' dirs(imov).name '/odor' s_mov{end} '_'];
%   dirname     = [dir_path '/' dirs(imov).name '/' dirs(s_mov) '/odor' s_mov{end} '_'];
  filename      = [dirname 'tracks.mat'];
  T             = load(filename);
  Tracks        = [Tracks T.trajectories_sort];
end

num_traject     = size(Tracks,2);
```

Determine stops and runs within navigation trajectories

```
runs = zeros(num_traject,1);
for i=1:num_traject
  Tracks{i}(:,11) = abs(Tracks{i}(:,9))<thangle;
  runs(i) = 1-mean(Tracks{i}(:,11));
end
```

Once ‘runs’ are extracted from the tracks, one could use simple equations to determine ‘run’ properties such as run length, run speed, etc. Either Matlab based functions or Excel based functions could be used.