Imperial College London

Department of Bioengineering

Final Report for MSc Project

Py2Pulse: A program for the parameter optimisation of a 2-photon optogenetics laser

Author:

Alison Sanders

Supervisor:

Dr Amanda Foust

September 2020

Keywords: 2-Photon optogenetics, Ultrafast laser parameters, Chronos opsin, Python 3, Arduino, QT

Word count: 5978

Abstract

Optogenetics studies have gained significant popularity in neuroscience in recent years, enabling researchers to examine neural circuits and control their responses. Many studies have demonstrated improvements to the optical setup; but there is insufficient evidence on the optimal 2P laser parameters which evoke maximal photocurrents in opsin-transfected cells without photodamage. In order to control the deployment of ultrafast laser pulses and find the optimal photostimulation parameters, this report presents a program called Py2Pulse written in Python 3, Arduino and QT. Py2Pulse is designed to be modular. Two Hardware Abstraction Layer (HAL) modules control their respective devices independently, enabling device-interchangeability and virtual testing. There is also a data analysis module containing functions for the optimisation algorithms. The modules are integrated into the Graphical User Interface (GUI) module. The interface enables researchers to set the parameters, control the lasing and upload results for analysis without requiring coding. Currently, the project is in the testing phase, requiring additional testing of the modules, especially the laser module when the laboratory is accessible. The completion of testing and deployment into production was limited by the laboratory being inaccessible due the COVID-19 pandemic.

Contents

List of figures

List of tables

Nomenclature

1PE	1-Photon excitation	I_{peak}	Saturation current
2P	2-Photon	K_D	Opsin rate constant
2PE	2-Photon excitation	K	Kelvin
AP	Action potential	kHz	Kilohertz
CGH	Computer Generated Holography	LED	Light Emitting Diode
CHO	Chinese Hamster Ovarian	mV	Millivolts
EP	Electrophysiology	mW	Milliwatts
GPC	Generalised Phase Contrast	NA	Numerical Aperture
GUI	Graphical User Interface	τ_{off}	Time to turn off in a photocycle
HAL	Hardware Abstraction Layer	τ_{on}	Time to turn on in a photocycle
IPI	Interpulse interval	TTL	Transistor-Transistor Logic
$I_{\max}$	Maximum current evoked at power		

1. Background

Optogenetics is a biological technique which controls recombinant cells using light [1]. Cells are transfected with opsins which are light-sensitive transmembrane proteins whose ion channels open when subjected to a certain wavelength of light [2]. By transfecting a selective subpopulation of cells with opsins, researchers use light to precisely control cells' activity through light-induced hyperpolarisation or depolarisation [2].

This technique has revolutionised neuroscience over the past decade, enabling researchers to examine the neural circuitry responsible for cognition and behaviour as well as enhancing the understanding of the mechanisms underlying some neurological conditions [2,3].

Optogenetics research is particularly active: research published in the last two years makes up over 50% of all optogenetics papers published online [4]. Many studies using 2-photon (2P) optogenetics aim to gain temporally precise control over selective cells in a population, ideally *in vivo*, with high resolution in a minimally invasive manner [1,3,5–7]. Recent findings are demonstrating substantial improvements to the experimental setup which lays the foundation for successful *in vivo* applications.

Notwithstanding the significant improvements made to date, there are numerous challenges which limit the success of this technique. Like many studies, these experiments are resource-intensive, demanding a precise and tuneable optical source for 2P experiments, a highly sensitive electrophysiology recording system and skilled researchers with coding knowledge to control the hardware and analyse the data. Experiments are prone to delays due to the unpredictable nature of biological cells and the time required to transfect and culture samples. This challenge is exacerbated by the fact that there is limited evidence on the optimal 2P photostimulation parameters [8]. Therefore, the probability of thermodamage increases if the fluences used are too high and, similarly, it is more likely that reduced photocurrent signals (which are already small in magnitude) are evoked if sub-optimal parameters are chosen.

Awareness of these challenges motivated the development of Py2Pulse. This is an open-source program which aims to enable researchers to perform optogenetics experiments *in vitro.* The purpose is to identify the pulsed laser parameters, specifically the average power and repetition rate, which optimally photostimulate a monolayer of Chronos-transfected CHO cells.

The aim of this report is to explain the fundamental concepts of optogenetics experiments underpinning the project, describe the requirements, design, and implementation of Py2Pulse and finally evaluate its current state.

2. Theory

Successful optogenetics experiments rely on several practical considerations reviewed in this section. These broadly include the informed selection of the opsin to transfect the cells; the optical system used to illuminate the sample and associated stimulation parameters; and the measurement system to record the outputs of the experiments [9].

Opsins

In this project, cells are transfected with opsins to enable photostimulation. Opsins are light-sensitive proteins which can be expressed as ion channels or pumps in recombinant cells [10].

Opsins undergo a photocycle whereby the protein isomerises and changes conformation in response to light. This opens a transmembrane channel to allow ions to move down their electrochemical gradient. A current is produced when ions move across the membrane. Opsins are capable of selectively inhibiting or exciting electrical activity in subpopulations of genetically-modified cells with submillisecond precision [10]. During a typical photocycle, the opsin gets photoactivated and goes from a closed state "C" to an open state "O" , and then undergoes a period of deactivation "D" before spontaneously closing [11]. This is presented in Figure 1.

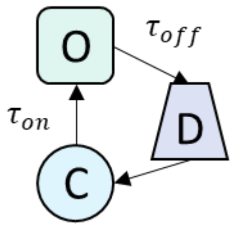


Figure 1. Typical photocycle of an opsin showing the time to turn on τ_{on} and the time to turn off τ_{off}.

The peak current can be modelled as the equilibrium between the closed state "C" to the open state "O" and is described by the Michaelis-Menten equation [12]:

$$I_{peak} = I_{\max} \times \frac{P}{P + K_D} \tag{1}$$

where K_D is the power at which I_{peak} reaches one half of $I_{\max}$. Note that I refers to the photocurrent while P refers to the average power.

The time required for each stage of the photocycle is determined by the opsin's unique kinetics. The kinetics can restrict the optical setup and frequency of action potential (AP) generation [13]. For this project, the opsin's kinetics determine the pulse repetition rate or frequency at which the cell can evoke maximal photocurrents.

Opsins vary in their spectral and temporal characteristics. For example, Chronos is known to have fast kinetics (τ_{on}≈1-2 ms and τ_{off}≈4 ms), while ReaChR or C1V1 have slow kinetics (τ_{on} ≈6–8 ms and τ_{off} ≈ 50–100 ms) [13]. Fast off-kinetics can generate a high frequency spike train associated with high temporal resolution and high pulse repetition rates [13,14]. Conversely, slower opsins might be preferable when scanning optical methods are used due to the slower temporal resolution and associated delays in photocurrent generation [14].

Finally, opsins respond optimally to certain wavelengths of light and these responses are variable. For example, Chronos's action spectrum is blue-green and upon activation, the opsin is excited and depolarises the cell by opening its cation channel [9]. Other opsin types may hyperpolarize the cell, have an inhibitory function or act as a pump [9].

Illumination techniques

The optical setup used to photostimulate the sample is an important practical consideration when designing the experiments.

2.1.1. 1PE versus 2PE

There are two main excitation mechanisms employed by optical systems: one-photon excitation (1PE) and two-photon excitation (2PE). In 1PE, a single high-energy photon is absorbed and excites a molecule into a high-energy state [15]. In contrast, 2PE involves two low-energy photons being absorbed simultaneously to reach the high-energy state [16]. During photoactivation, an opsin absorbs a photon via 1PE or 2PE and it undergoes a conformational change which opens its ion channel.

1PE is the conventional mechanism with widely reported limitations associated with poor axial resolution and limited penetration depth due to scattering [5,15]. In 1990, Denk *et al* introduced 2P laser microscopy using light of longer wavelengths. This reduces scattering resulting in intrinsically better penetration depth [16]. Furthermore, photoactivation using 2PE is proportional to the square of average power which results in improved axial confinement [15]. This relationship is shown in equation (2):

$$n_a = \frac{p_{ave}^2 \delta}{\tau_p f_p^2} \left(\frac{(NA)^2}{2\bar{h} c \lambda} \right)^2 \tag{2}$$

where n_a is the number of photons absorbed per molecule per pulse, p_{ave} is the average intensity, δ is the molecule's 2P absorption at wavelength λ, τ_p is the pulse duration, f_p is the repetition rate, NA is the numerical aperture, $\bar{h}$ is Planck's constant, c is the speed of light [16].

2.1.2. Relationship between 2PE signal, power, and repetition rate

Of interest in this project is the specific relationship between the 2PE signal (n_a), the average power (p_{ave}) and the pulse repetition rate (f_p). Equation (2) can be simplified as:

$$n_a = \frac{p_{ave}^2}{\tau_p f_p^2} \times k \tag{3}$$

where k is a constant. Furthermore, peak and average power are related by:

$$p_{ave} = p_{peak} \times \tau_p \times f_p \tag{4}$$

Also, the total number of excited molecules summed over n groups of pulse for the time Δt is [17]:

$$N_a = \sum_{n_{group}} n_a = n_a \times f_p \times \Delta t \tag{5}$$

Finally, the total number of excited molecules over time is:

$$\frac{N_a}{\Delta t} = \frac{p_0^2}{\tau_p \times f_p} \times k \tag{6}$$

According to equation (6), the 2PE signal will increase quadratically as power increases with other parameters fixed. Also, the signal will increase linearly as the repetition rate decreases with other parameters fixed.

2.1.3. Scanning versus parallel

This project employs 2PE methods which are categorised into scanning and parallel approaches. Scanning excitation involves quickly steering a laser beam over several positions of a target [1]. Parallel excitation utilises phase-modulation techniques to illuminate all targets simultaneously using low-NA Gaussian beams or using liquid crystal devices as in Computer Generated Holography (CGH) or Generalised Phase Contrast (GPC) [1,5,6].

This project selects 2P parallel excitation as it has been demonstrated to have improved temporal resolution over scanning approaches and enables whole-cell activation [1]. Specifically, low-NA Gaussian beams will be used to illuminate the sample, despite some advantages offered by CGH and GPC due to availability. It is intended to activate a 2D monolayer of cells.

Cell photodamage

This project aims to maximise the cell's transmembrane current without causing photodamage. As previously stated, this project will photostimulate the whole cell body and the current will increase quadratically as power increases until saturation when all the channels have opened. Upon photostimulation, the temperature of the cells rises linearly as the power increases [7]. A long exposure time and high repetition rates also increase the temperature but to a much lesser extent, while low repetition rates allow time for energy dissipation between pulses [7,18]. Temperature rises of 6-8 K above the baseline physiological temperature causes denaturation of intracellular proteins, leading to cell death [7]. Therefore, it is important to monitor the cells for photodamage when increasing the power. Some symptoms of photodamage may include unusual electrophysiological activity from leaking channels or damaged membranes and cell blebbing.

Electrophysiology

The transmembrane currents generated upon photostimulation, which this project aims to maximise, are to be recorded using the whole-cell voltage clamp method. This method is illustrated in Figure 2.

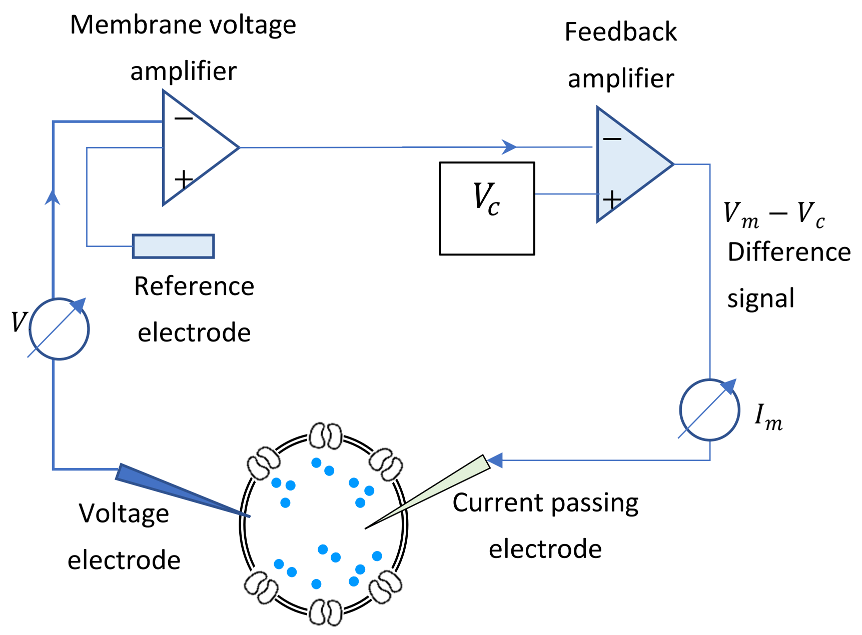

Membrane voltage amplifier
Feedback amplifier
V_c
$V_m - V_c$ Difference signal
Reference electrode
V
I_m
Voltage electrode
Current passing electrode

Figure 2. Diagram of the voltage clamp method

The voltage clamp is a negative feedback system. The voltage electrode conducts the membrane potential which is read by the membrane voltage amplifier. The output V_m is sent to the feedback amplifier which has a command voltage V_c input. The difference between the membrane potential and the command voltage generates a signal which is injected into the cell via the current passing electrode. In this manner, the membrane potential is kept constant. The membrane voltage V_m and current injected I_m are recorded by the electrophysiology data acquisition board. Current signals in the order of picoamps (pA) are commonly reported in optogenetics studies [14].

This technique is particularly effective on CHO cells which are the cells of choice for this project. Gap junctions are absent in CHO cells which restricts the contribution of other endogenous channels or neighbouring cells to a recorded electrical response [12]. This enables the voltage clamp method to record evoked photocurrents solely from optical stimulation with low background noise [19].

3. Requirements and hypothesis

This section outlines the objectives, specifies the requirements, and states the hypotheses.

Objectives

The specific objectives of the project are to:

- Develop a program, using Python and an Arduino microcontroller, to control an ultrafast pulsed ytterbium fiber laser to deploy millisecond pulses of radiation, varying pulse repetition rate and average power, given the parameters shown in Appendix A.
- Perform a power calibration to relate the actual output power measured experimentally to the command value into the system.
- Determine the optimal power and repetition rate which yield the maximum current without photodamaging the cells.

Requirements

The following functional requirements describe the tasks that the project aims to complete:

REQ 1. Control the deployment of millisecond pulses of radiation from the ultrafast ytterbium fiber laser, using the user-specified parameter values.

REQ 2. Enable the user to set the parameter values in a simple interface.

REQ 3. Enable the user to see all the command parameters utilised in the lasing, including those which are predetermined.

REQ 4. Enable the electrophysiology data results to be uploaded in the original format without pre-processing.

REQ 5. Organise the project folder such that all results can easily be referred to at any stage.

REQ 6. Analyse the electrophysiology data to calibrate the laser. The actual power measured by the picker is fitted to the input RL energy value using the sigmoid function.

REQ 7. Analyse the electrophysiology data to extract the K_D value by fitting the Michaelis-Menten equation to the photocurrent achieved versus the power density of the cell.

REQ 8. Analyse the electrophysiology data to optimise the laser parameters.

REQ 9. The program should save all relevant outputs with sensible names.

In addition, the project aims to satisfy the following non-functional requirements:

REQ 10. *Accessibility*: Users with little to no coding knowledge should be able to use the program seamlessly.

REQ 11. *Reliability*: The program should be able to run the end-to-end protocol on several cells repeatedly while maintaining its functions and without suffering from performance issues. Outputs from different cells should be clearly labelled.

REQ 12. *Performance*: Short run times should be maintained when there are multiple repetitions on different cells or very large electrophysiology datasets.

REQ 13. Usability: The interface must be clear and easy to navigate

REQ 14. Scalability: The project should be logical to allow for expansion or repurposing.

REQ 15. *Flexibility:* Enable the steps in the experimental protocol to be run independently of other steps. For example, the user must be able skip to a relevant step instead of repeating the end-to-end protocol each time.

REQ 16. Safety: The program must be safe to use, considering control over a class IV laser. For example, radiation should only be emitted if explicitly commanded by the program; the default settings should be to exist in a "safe" mode. Safety checks should be explicit.

REQ 17. Testability: The project should support testing such that finding and fixing faults is relatively straightforward.

Hypothesis

The following two hypotheses are stated for this project:

1. The optimal repetition rate which yields the largest evoked photocurrent per unit of average power will be 20 kHz, equivalent to the lowest repetition rate achievable by the laser. As explained in the Opsins section, the opsin's kinetics determine the frequency at which a cell evokes photocurrent. Chronos's two-state photocycle is reportedly 5 ± 0.4 ms [6]. Inverted, this equates to 200 Hz which is lower than the range of the laser.
2. Assuming that this problem would usually require a minimum of one FTE researcher for 6 months, it is hypothesised that Py2Pulse will enable scientists without specialised coding knowledge to carry out optogenetics experiments to optimise the pulse repetition rate and laser power in three months or less.

4. Design

This section first summarises the system's design. It then describes the modules, the HAL and the GUI in further detail.

System overview

Py2Pulse is an open-source program consisting of a collection of modules coded in Python, Arduino and QT available for download from the GitHub repository: https://github.com/AliSan123/Optogenetics_project. An overview of the Py2Pulse system is shown in Figure 3.

Beam control

Steps 1-5 in Figure 3 show the laser beam control. The *Arduino_sketch.ino* is first uploaded onto the Arduino Uno microcontroller at setup. The *Coherent.py* module then starts up the laser and sets the parameters and gated configuration via serial commands. Next, the Arduino Uno board runs a loop toggling the TTL to "HIGH" and "LOW" which is communicated to the laser via Gate 1. Finally, the laser deploys pulses when the TTL is "HIGH" for photostimulation or power calibration.

Data acquisition & analysis

Steps 6-8 in Figure 3 involve the data acquisition of the key outputs. First, beam samples are analysed by the power meter and read by the electrophysiological (EP) data acquisition board. The measured picker power is used for the power calibration. In steps 7 and 8, upon photostimulation, the cells' opsins open and induce a photocurrent measured by the voltage clamp method and read by the EP data acquisition board. The EP data is then extracted from the board and uploaded to the GUI in *.smr* format for the data analysis. A separate analytical protocol is run for each stage of the experiment (detailed in the Functions section).

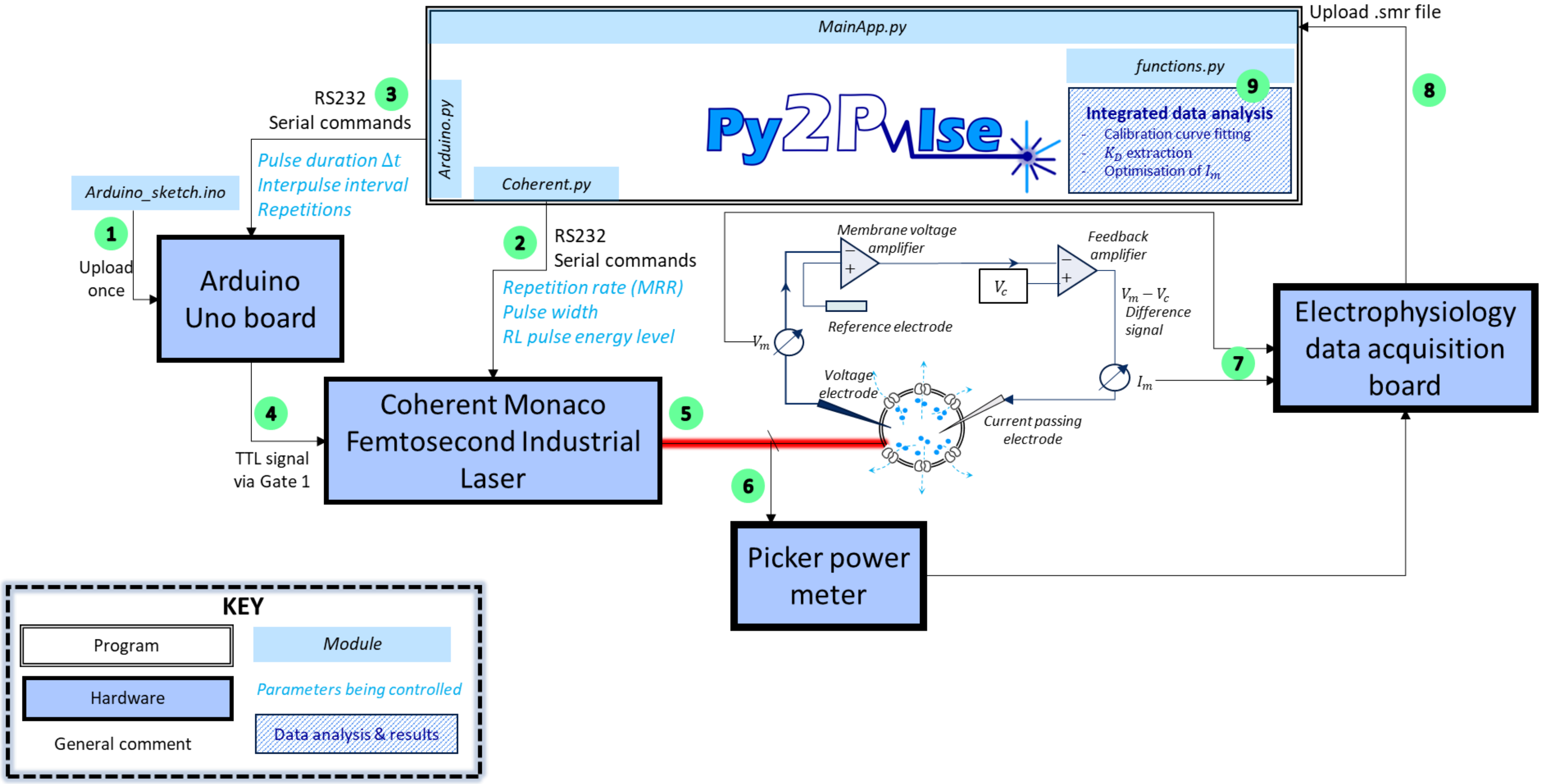


Figure 3. Diagram showing an overview of the Py2Pulse system

Modules

This text defines a module as a script of code which is run and tested independently of other system components. A modular design is advantageous to achieve several non-functional requirements: REQ 14, REQ 15 and REQ 17.

Coherent

The Coherent module comprises the *Coherent.py* script written in Python. The *Coherent()* class's functions control the Coherent Monaco Femtosecond Laser via serial communication. This is essential to meet REQ 1. The class first initialises the serial connection. Separate functions then start up the laser from standby, set the pulse parameters, turn on the diodes and pulses and, and open the shutter. After lasing, the shutter closes, and the laser returns to standby.

Arduino

This module includes *Arduino_sketch.ino* and the *Ardunio.py* script. It is required to meet REQ 1.

The Arduino sketch contains a loop which signals the TTL to be "HIGH" for pulse duration Δt and then "LOW". The sketch on the microcontroller encodes the pulse duration Δt in milliseconds using the *delay()* function. This has an accuracy of 50 ppm for 16 MHz boards (equivalent to an error of $\pm 0.25\ \mu s$ for a 5 ms pulse duration) [20]. Δt is coded as a variable in the sketch. parsed as an integer from the *Arduino.py* script.

The *Arduino()* class in *Arduino.py* initiates serial communication with the Arduino Uno and sends the pulse duration to the board. After the TTL HIGH-to-LOW loop, the program sleeps for interpulse interval controlled by Python. This has a relatively high error due to processing time. The "off" time between stimulations is not typically controlled for in experiments, but the precision of the "on" pulse duration is imperative to avoid photodamage.

Functions

The *functions.py* module is written in Python and contains functions for the data analysis. This module aims to satisfy several functional requirements: REQ 4, REQ 6, REQ 7 and REQ 8. The module contains generalised functions to load the EP block data, plot publication-quality graphs and smooth the EP data. Additionally, it contains specialised functions for algorithms detailed below.

Power Calibration

The power meter ramps up before the maximum picker voltage (mV) for an energy level is met. This is shown in Figure 4.a. The grey line indicates where the picker voltage has fully ramped up.

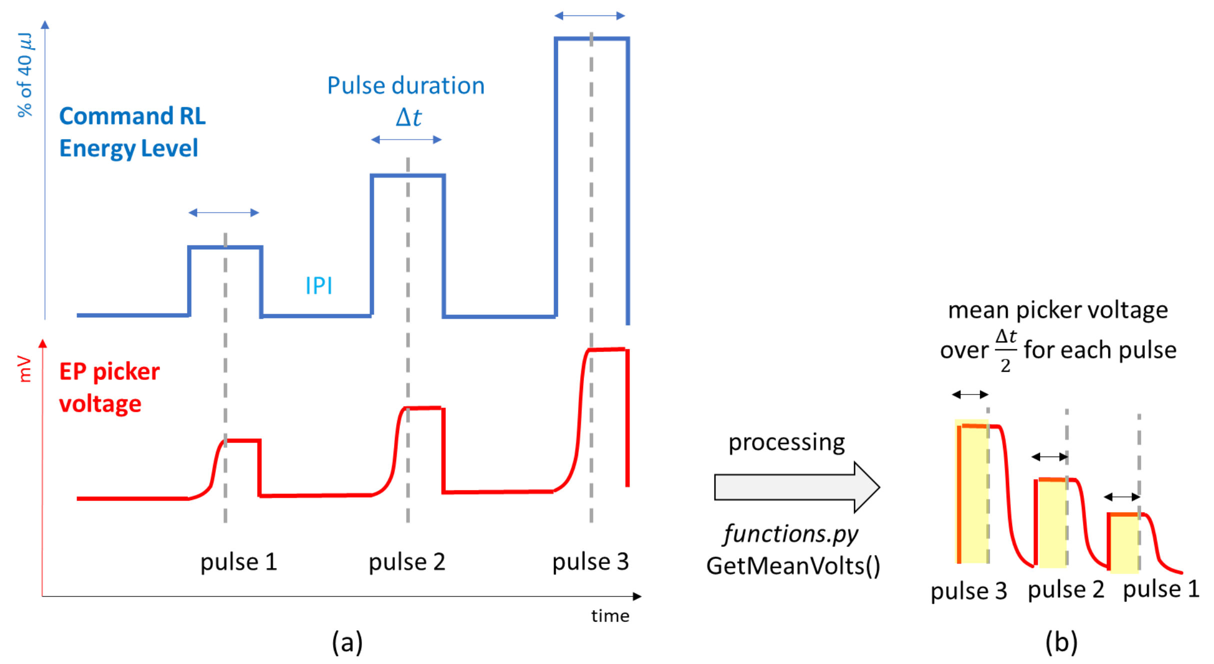


Figure 4. Diagrams of (a) the RL energy level of pulses and corresponding EP picker voltage over time; and (b) the cleaned, flipped picker voltage for analysis

The functions involved in the power calibration are *GetMeanVolts()* and *convert_V_W().* The algorithm is summarised as follows:

1. The picker voltage is smoothed using a small window.
2. The pulses are identified from the smoothed signal by extracting samples which are greater than $\bar{x} + 2\sigma$. The $\bar{x}$ and σ are calculated from the voltage over the first two seconds of EP recording where no pulses are deployed.
3. The cleaned pulses are flipped to simplify finding the start of the pulses (Figure 4.b). The picker voltage over half the pulse duration is averaged and added to a list. The result is flipped to match the original sequence.
4. The mean picker voltage is converted to power (mW) using the calibration file and constants specified in the GUI:

$$P\ (mW) = \frac{picker\ max\ measurement\ (mW)}{picker\ max\ output\ \ (mV)} \times mean\ pulse\ voltage\ (mV) \tag{7}$$

where P is the picker power.

The calibration file shows the power onto the cell versus the picker power is the directly proportional ($y = mx$), so the gradient m is:

$$m = \frac{calibration\ power\ onto\ cell\ (mW)}{calibration\ P\ (mW)} \tag{8}$$

This is used to find the actual power onto the cell:

$$actual\ power\ onto\ cell\ (mW) = m \times P(mW) \tag{9}$$

The picker power is converted to power density using the beam spot diameter set in the GUI:

$$power\ density\ (mW/\mu m^2) = \frac{actual\ power\ onto\ cell\ (mW)}{\pi \left(\frac{diameter}{2}\right)^2} \tag{10}$$

The power density is plotted against the RL energy levels.

Calculation of K_D

First, the function *GetCurrent()* returns the minimum current values for the y-axis of the Michaelis-Menten plot. The initial data processing follows the same logic as steps 1-3 described in the Power Calibration. After extracting the cleaned, flipped photocurrents, a window slides over the signal and extracts the local minimum current value for *N* pulses. The local minimum is averaged over a few points, the "buffer" window, to reduce noise.

Next the RL energy levels are converted to power density using the *sigmoid()* and *getPowerFromCalibration()* functions. The *scipy.optimize.curve_fit* function with the 'dogbox' method fits the sigmoidal relationship between the RL energy level (x) and the power density in the sample (y). The 'dogbox' method is chosen according to SciPy's documentation since this is a small problem bound by the energy levels [21].

The Michaelis-Menten plot of current density versus power density is plotted. K_D is returned from the function *getKd()* which fits the Michaelis-Menten equation to the data. The *scipy.optimize.curve_fit* is used with the Levenberg-Marquardt algorithm since it's the most efficient for small unbound problems [21].

Optimisation

Finally, the pulse repetition rate and the power density are related by modifying (6) above:

$$pulse\ repetition\ rate\ (kHz) \propto \frac{(power\ density\ (mW/\mu m^2))^2}{K_D(mW/\mu m^2)} \tag{11}$$

And rearranging yields (12):

$$power\ density\ (mW/\mu m^2) \approx \sqrt{pulse\ repetition\ rate\ (kHz) \times K_D(mW/\mu m^2)} \tag{12}$$

The pulse repetition rates are converted to RL energy levels using (12) and the function *convertPowerToEnergy()*. This fits the power calibration data (power density versus energy levels) to an inverse sigmoid function *inverse_sigmoid()* using the *scipy.optimize.curve_fit* algorithm with the 'dogbox' method [21].

Finally, a 3D optimisation curve is generated of the pulse repetition rate (x), RL energy level (y) and photocurrent (z). The parameters corresponding to the largest magnitude of photocurrent are selected as the optimum.

Hardware abstraction layer

Py2Pulse controls two hardware devices: the Coherent Monaco Femtosecond Industrial Laser and the Arduino Uno microcontroller. A challenge with programming hardware is the commands conform to a device-dependent language. To overcome this, the Hardware Abstraction Layer (HAL) is introduced.

The HAL is a software layer which interacts with the hardware at a general level, instead of at a detailed hardware level [22,23]. Py2Pulse has two HAL modules, Coherent and Arduino, which serve as an interface between the program and the hardware. Each module contains a list of functions which the hardware must implement to achieve its function. To use other device models, these functions are modified with the device's language. Additionally, the modules can be virtualised for testing the program without connecting to the hardware [23].

GUI

The Py2Pulse Graphical User Interface (GUI) enables researchers to interact with windows, view and change the laser parameters and have control over the experiments and analysis without requiring coding skills. The GUI facilitates the achievement of multiple requirements: REQ 2, REQ 3, REQ 4, REQ 5, REQ 10, REQ 13, REQ 14, REQ 15 and REQ 16.

The user interface is built in QT v5. The QT files are loaded into the *MainApp.py* Python script using PyQT5 where additional functionality is added. The interface can be easily edited and scaled in the QT Designer and reloaded instantly into *MainApp.py*.

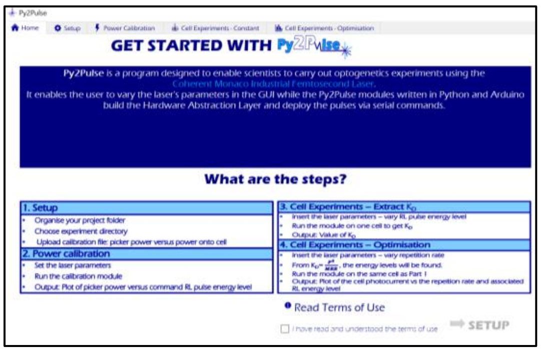
The *MainApp.py* contains six classes:

- *Ui()* initialises the main window and controls the functionality of each of the five tabs
- *TermsOfUse()* initialises the a dialog popup with the prerequisites and links to relevant documentation.
- *SafetyWindow()* initialises a dialog with safety and readiness checkboxes and controls the start of the lasing
- *UploadCalResults()* initialises a dialog enabling the upload and analysis of the calibration results.
- Similarly, *UploadPart1Results()* and *UploadPart2Results()* enable the upload and analysis of the results from parts 1 and 2 of the cell experiments.

Design considerations include a focus on simplicity and clarity of the protocol and values being set. It was also important to allow users to move between steps using the tabs. Safety and functional aspects are highlighted and require proactivity e.g. using checkboxes to ensure data is recorded and the environment is safe before lasing.

5. Implementation

This section presents the realisation of the design. Figure 5 references the windows which the user interacts with when using Py2Pulse, described below.



Py2Pulse

Home | Setup | Power Calibration | Cell Experiments - Constant | Cell Experiments - Optimisation

GET STARTED WITH Py2Pulse

Py2Pulse is a program designed to enable scientists to carry out optogenetics experiments using the Coherent Monaco Industrial Femtosecond Laser.
It enables the user to vary the laser's parameters in the GUI while the Py2Pulse modules written in Python and Arduino build the Hardware Abstraction Layer and deploy the pulses via serial commands.

What are the steps?

1. Setup
- Organise your project folder
- Choose experiment directory
- Upload calibration file: picker power versus power onto cell

2. Power calibration
- Set the laser parameters
- Run the calibration module
- Output: Plot of picker power versus command RL pulse energy level

3. Cell Experiments – Extract K_D
- Insert the laser parameters – vary RL pulse energy level
- Run the module on one cell to get K_D
- Output: Value of K_D

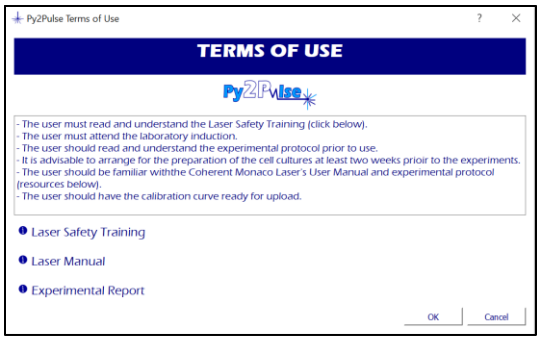
4. Cell Experiments – Optimisation
- Insert the laser parameters – vary repetition rate
- From $K_D = \frac{p^2}{MRR}$, the energy levels will be found.
- Run the module on the same cell as Part 1
- Output: Plot of the cell photocurrent vs the repetion rate and associated RL energy level

Read Terms of Use

☐ I have read and understood the terms of use

SETUP

a. *Home screen tab*



Py2Pulse Terms of Use

TERMS OF USE

Py2Pulse

- The user must read and understand the Laser Safety Training (click below).
- The user must attend the laboratory induction.
- The user should read and understand the experimental protocol prior to use.
- It is advisable to arrange for the preparation of the cell cultures at least two weeks prioir to the experiments.
- The user should be familiar withthe Coherent Monaco Laser's User Manual and experimental protocol (resources below).
- The user should have the calibration curve ready for upload.

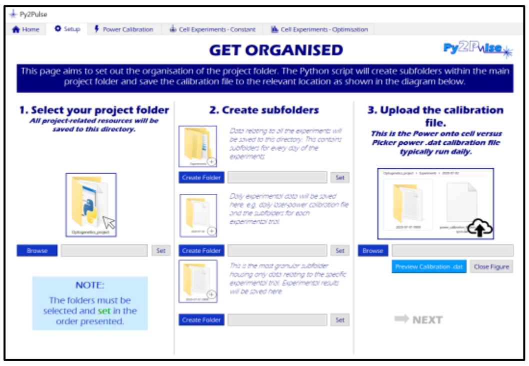
Laser Safety Training

Laser Manual

Experimental Report

OK | Cancel

b. *Terms of use dialog box*



Py2Pulse

Home | Setup | Power Calibration | Cell Experiments - Constant | Cell Experiments - Optimisation

GET ORGANISED

This page aims to set out the organisation of the project folder. The Python script will create subfolders within the main project folder and save the calibration file to the relevant location as shown in the diagram below.

1. Select your project folder
All project-related resources will be saved to this directory.

Browse | Set

NOTE:
The folders must be selected and set in the order presented.

2. Create subfolders

Data relating to all the experiments will be saved to this directory. This contains subfolders for every day of the experiments.

Create Folder | Set

Daily experimental data will be saved here. e.g. daily laser-power calibration file and the subfolders for each experimental trial.

Create Folder | Set

This is the most granular subfolder housing only data relating to the specific experimental trial. Experimental results will be saved here.

Create Folder | Set

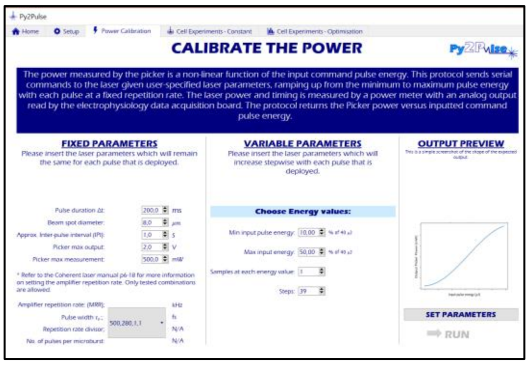
3. Upload the calibration file.
This is the Power onto cell versus Picker power .dat calibration file typically run daily.

Browse

Preview Calibration .dat | Close Figure

NEXT

c. *Setup tab*



Py2Pulse

Home | Setup | Power Calibration | Cell Experiments - Constant | Cell Experiments - Optimisation

CALIBRATE THE POWER

The power measured by the picker is a non-linear function of the input command pulse energy. This protocol sends serial commands to the laser given user-specified laser parameters, ramping up from the minimum to maximum pulse energy with each pulse at a fixed repetition rate. The laser power and timing is measured by a power meter with an analog output read by the electrophysiology data acquisition board. The protocol returns the Picker power versus inputted command pulse energy.

FIXED PARAMETERS
Please insert the laser parameters which will remain the same for each pulse that is deployed.

Pulse duration Δt: 200,0 ms
Beam spot diameter: 8,0 μm
Approx. Inter-pulse interval (IPI): 1,0 s
Picker max output: 2,0 V
Picker max measurement: 500,0 mW

* Refer to the Coherent laser manual p6-18 for more information on setting the amplifier repetition rate. Only tested combinations are allowed.

Amplifier repetition rate (MRR): kHz
Pulse width τ_p: fs
Repetition rate divisor: N/A
No. of pulses per microburst: N/A
500,280,1,1

VARIABLE PARAMETERS
Please insert the laser parameters which will increase stepwise with each pulse that is deployed.

Choose Energy values:

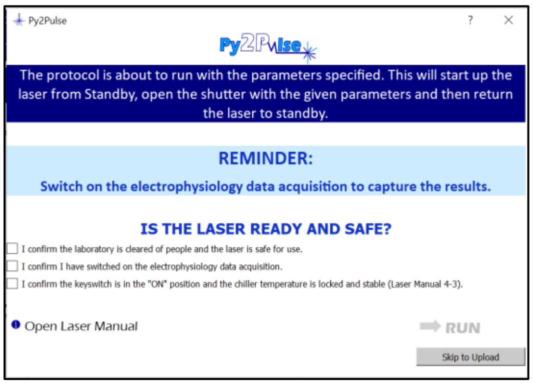
Min input pulse energy: 10,00 % of 40 μJ
Max input energy: 50,00 % of 40 μJ
Samples at each energy value: 1
Steps: 39

OUTPUT PREVIEW
This is a simple screenshot of the shape of the expected output.

SET PARAMETERS

RUN

d. *Power calibration tab*



Py2Pulse

Py2Pulse

The protocol is about to run with the parameters specified. This will start up the laser from Standby, open the shutter with the given parameters and then return the laser to standby.

REMINDER:

Switch on the electrophysiology data acquisition to capture the results.

IS THE LASER READY AND SAFE?

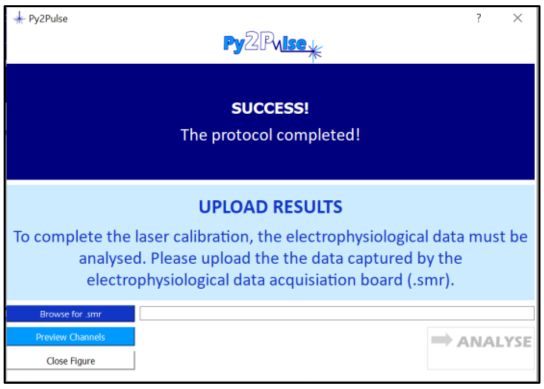
☐ I confirm the laboratory is cleared of people and the laser is safe for use.
☐ I confirm I have switched on the electrophysiology data acquisition.
☐ I confirm the keyswitch is in the "ON" position and the chiller temperature is locked and stable (Laser Manual 4-3).

Open Laser Manual

RUN

Skip to Upload

e. *Safety and readiness dialog box (before lasing)*



Py2Pulse

Py2Pulse

SUCCESS!

The protocol completed!

UPLOAD RESULTS

To complete the laser calibration, the electrophysiological data must be analysed. Please upload the the data captured by the electrophysiological data acquisition board (.smr).

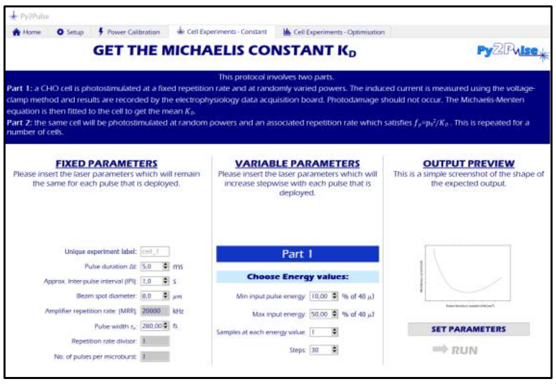
Browse for .smr

Preview Channels

Close Figure

ANALYSE

f. *Upload results dialog box (post lasing)*



Py2Pulse

Home | Setup | Power Calibration | Cell Experiments - Constant | Cell Experiments - Optimisation

GET THE MICHAELIS CONSTANT K_D

This protocol involves two parts.
Part 1: a CHO cell is photostimulated at a fixed repetition rate and at randomly varied powers. The induced current is measured using the voltage-clamp method and results are recorded by the electrophysiology data acquisition board. Photodamage should not occur. The Michaelis-Menten equation is then fitted to the cell to get the mean K_D.
Part 2: the same cell will be photostimulated at random powers and an associated repetition rate which satisfies $f_p = p_0^2/K_D$. This is repeated for a number of cells.

FIXED PARAMETERS
Please insert the laser parameters which will remain the same for each pulse that is deployed.

Unique experiment label: cell_1
Pulse duration Δt: 5,0 ms
Approx. Inter-pulse interval (IPI): 1,0 s
Beam spot diameter: 8,0 μm
Amplifier repetition rate (MRR): 20000 kHz
Pulse width τ_p: 280,00 fs
Repetition rate divisor: 1
No. of pulses per microburst: 1

VARIABLE PARAMETERS
Please insert the laser parameters which will increase stepwise with each pulse that is deployed.

Part 1

Choose Energy values:

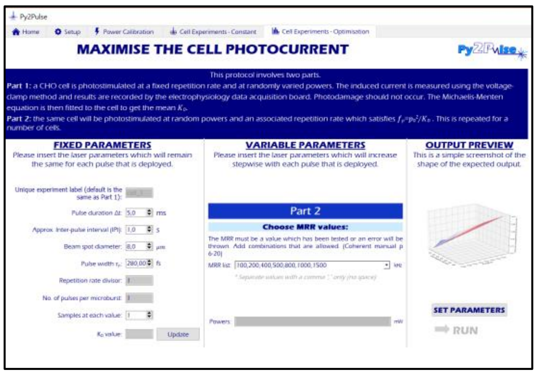
Min input pulse energy: 10,00 % of 40 μJ
Max input energy: 50,00 % of 40 μJ
Samples at each energy value: 1
Steps: 30

OUTPUT PREVIEW
This is a simple screenshot of the shape of the expected output.

SET PARAMETERS

RUN

g. *Cell experiments – find K_D tab*



Py2Pulse

Home | Setup | Power Calibration | Cell Experiments - Constant | Cell Experiments - Optimisation

MAXIMISE THE CELL PHOTOCURRENT

This protocol involves two parts.
Part 1: a CHO cell is photostimulated at a fixed repetition rate and at randomly varied powers. The induced current is measured using the voltage-clamp method and results are recorded by the electrophysiology data acquisition board. Photodamage should not occur. The Michaelis-Menten equation is then fitted to the cell to get the mean K_D.
Part 2: the same cell will be photostimulated at random powers and an associated repetition rate which satisfies $f_p = p_0^2/K_D$. This is repeated for a number of cells.

FIXED PARAMETERS
Please insert the laser parameters which will remain the same for each pulse that is deployed.

Unique experiment label (default is the same as Part 1): cell_1
Pulse duration Δt: 5,0 ms
Approx. Inter-pulse interval (IPI): 1,0 s
Beam spot diameter: 8,0 μm
Pulse width τ_p: 280,00 fs
Repetition rate divisor: 1
No. of pulses per microburst: 1
Samples at each value: 1
K_D value: | Update

VARIABLE PARAMETERS
Please insert the laser parameters which will increase stepwise with each pulse that is deployed.

Part 2

Choose MRR values:

The MRR must be a value which has been tested or an error will be thrown. Add combinations that are allowed. (Coherent manual p 6-20)

MRR list: 100,200,400,500,800,1000,1500 kHz

* Separate values with a comma ',' only (no space)

Powers: mW

OUTPUT PREVIEW
This is a simple screenshot of the shape of the expected output.

SET PARAMETERS

RUN

h. *Cell experiments – optimisation tab*

Figure 5. A summary of the program's interfaces

To start Py2Pulse, the *MainApp.py* script is executed in Python. The home screen tab appears which defines the purpose and steps involved (Figure 5.a). These steps correspond to each tab in the GUI.

Upon clicking the terms of use button, a dialog opens highlighting important safety considerations, experimental prerequisites and links to relevant documentation (Figure 5.b). Upon agreeing to the terms of use, the *Setup* button changes to green and the user is directed to the Setup tab.

The Setup tab organises the project folder (Figure 5.c). This satisfies REQ 5. The user browses and selects the project folder. The program then creates subfolders within it. The default organisation structure is shown in Appendix A. The program displays the directories for each subfolder for clarity and editing as desired. Finally, the user uploads the picker power versus the power onto the cell data for the power calibration. It is assumed to be applicable for all experiments taking place on that day. Once uploaded, the *Next* button turns green and the user is directed to the power calibration tab.

The power calibration tab sets the laser parameters for the first deployment of pulses (Figure 5.d). The page is organised in panels, distinguishing between the fixed and variable parameters. The variable parameters include the RL energy levels. The right panel displays a sample output to demonstrate the expected shape. The *Set parameters* button updates the embedded parameter values with the on-screen values. This enables values to be changed with confidence. This tab meets multiple functional and non-functional requirements: REQ 1, REQ 2, REQ 3, REQ 10 and REQ 13. Upon clicking *Run*, the Safety and readiness dialog displays, prompting the start of the EP recording and ensuring it is safe to lasing (Figure 5.e). During the lasing, the commands sent to the laser can be followed in the Python console.

On completion of the lasing, another dialog prompts the user to upload the EP data (Figure 5.f). The user may preview a plot to verify the channels. Upon clicking the *Analyse* button, the program runs the algorithm described in the Power Calibration section and displays the power calibration plot. The figure (*.png*) and data (*.csv*) are saved to the timestamped folder (see Appendix A) which satisfies requirement REQ 9.

The program advances to the cell experiments where K_D is determined (Figure 5.g). The page's layout is as before. The repetition rate is fixed at 20 MHz while the RL pulse energy is varied randomly. As above, after setting the parameters, the safety dialog appears followed by the dialog prompting for the EP results after lasing (Figure 5.e, Figure 5.f). Upon clicking the *Analyse* button, the program executes the algorithm outlined in the Calculation of $\boldsymbol{K_D}$ section. The Michealis-Menten plot of the minimum (smoothed) photocurrent versus the mean power density in the sample is displayed and saved. A text file with the K_D value and associated experiment label is also written to the folder, satisfying requirement REQ 9.

Finally, for the optimisation, the parameters are set and the calculated K_D value is shown by clicking *Update*. This is imperative to meet requirement REQ 3. The user then selects a list of repetition rates to be tested from the dropdown. The Coherent laser can only accept tested repetition rate values. Note that the current list in non-exhaustive and should be added to during testing. After lasing and uploading the results, the algorithm described in the Optimisation section is implemented. The final

3D optimisation curve of the pulse repetition rate (x), associated RL energy level (y) and corresponding evoked photocurrent (z) is shown and the corresponding data is saved.

6. Evaluation

This section assesses the testing carried out on Py2Pulse and then presents the results of the risk analysis, goals achieved and future work.

Testing

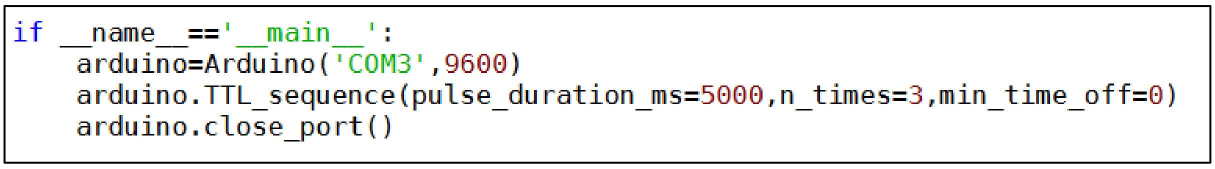
Program testing involves the execution of a component or system to verify whether the requirements have been met. Four major levels of testing are outlined.

Unit testing

Unit testing is the most granular of tests conducted. In the case of Py2Pulse, unit testing involves execution of certain modules independently.

Arduino

The *Arduino_sketch.ino* is uploaded to the Arduino Uno board, specifying the output as LED_BUILTIN for testing. The *Arduino.py* script is then executed in Python with dummy parameters parsed into the functions (see Figure 6). The Arduino's LED is programmed to light up three times for 5 seconds and is assessed visually. The LED blinked the correct number of times for five seconds, verified by Python's timer, and therefore passed the unit test. Note that timings in the seconds range are used for verification since Python is unlikely to achieve millisecond precision due to processing time. Additional verification of the pulse duration is recommended by analysing the EP data.

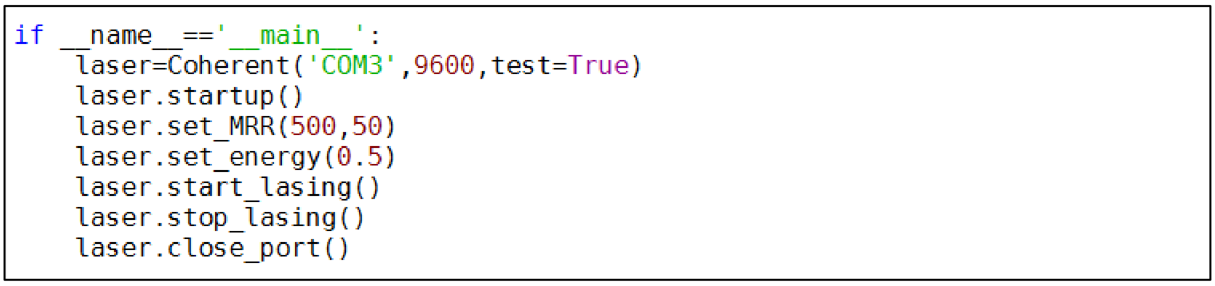


```
if __name__=='__main__':
    arduino=Arduino('COM3',9600)
    arduino.TTL_sequence(pulse_duration_ms=5000,n_times=3,min_time_off=0)
    arduino.close_port()
```

Figure 6. Code for testing the Arduino module

Coherent

An artificial laser was used to test the *Coherent.py* module. The artificial laser was made using an Arduino sketch (Available on GitHub as *Coherent_testing.ino*). The sketch simulates the responses of the Coherent Monaco laser as per the Operational Manual once it is uploaded to the Arduino Uno board. The Coherent.py script is the run with parameters initialised as shown in Figure 7. The test succeeded since the module runs without errors and prints the encoded commands to the console in the expected format. However, additional verification is required using the real laser.

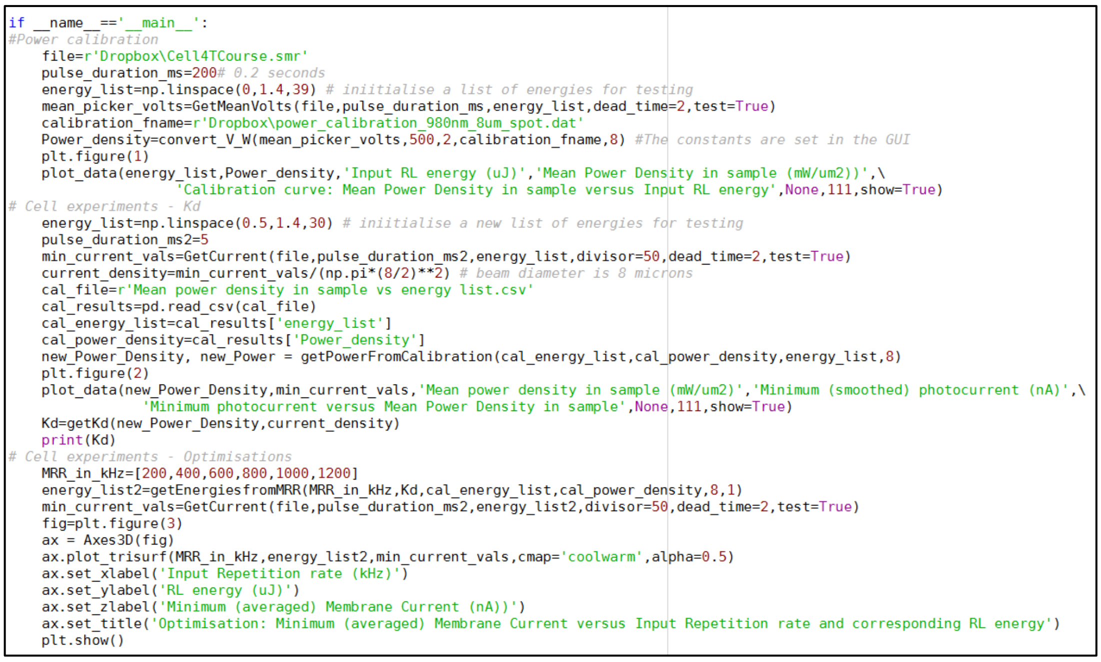


```
if __name__=='__main__':
    laser=Coherent('COM3',9600,test=True)
    laser.startup()
    laser.set_MRR(500,50)
    laser.set_energy(0.5)
    laser.start_lasing()
    laser.stop_lasing()
    laser.close_port()
```

Figure 7. Code for testing the Coherent module

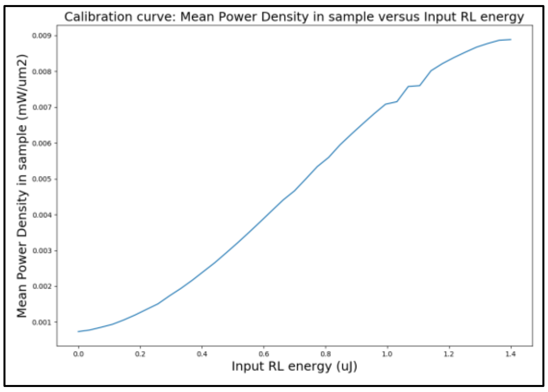
Data analysis

The data analysis module is evaluated using a fictitious electrophysiology dataset from historic experiments. The test code is shown in Figure 8 and the resultant plots are shown in Figure 9. Note the energy values and repetition rates have been initialised as guesses in the testing code and so the results are not expected to be accurate. Instead, the shapes of the curves are important. These plots demonstrate the shapes of the expected outputs and therefore pass the module test. This verification can be improved by further testing using the real EP data from the experiments.

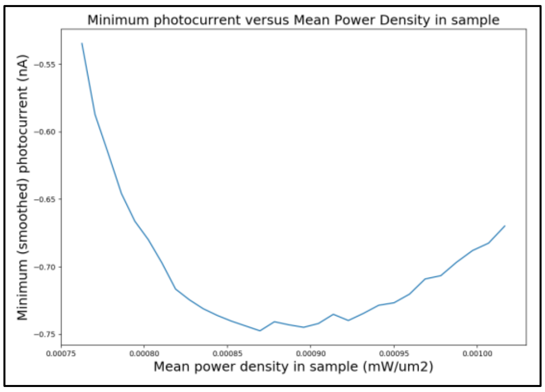


```
if __name__=='__main__':
#Power calibration
    file=r'Dropbox\Cell4TCourse.smr'
    pulse_duration_ms=200# 0.2 seconds
    energy_list=np.linspace(0,1.4,39) # iniitialise a list of energies for testing
    mean_picker_volts=GetMeanVolts(file,pulse_duration_ms,energy_list,dead_time=2,test=True)
    calibration_fname=r'Dropbox\power_calibration_980nm_8um_spot.dat'
    Power_density=convert_V_W(mean_picker_volts,500,2,calibration_fname,8) #The constants are set in the GUI
    plt.figure(1)
    plot_data(energy_list,Power_density,'Input RL energy (uJ)','Mean Power Density in sample (mW/um2))',\
                   'Calibration curve: Mean Power Density in sample versus Input RL energy',None,111,show=True)
# Cell experiments - Kd
    energy_list=np.linspace(0.5,1.4,30) # iniitialise a new list of energies for testing
    pulse_duration_ms2=5
    min_current_vals=GetCurrent(file,pulse_duration_ms2,energy_list,divisor=50,dead_time=2,test=True)
    current_density=min_current_vals/(np.pi*(8/2)**2) # beam diameter is 8 microns
    cal_file=r'Mean power density in sample vs energy list.csv'
    cal_results=pd.read_csv(cal_file)
    cal_energy_list=cal_results['energy_list']
    cal_power_density=cal_results['Power_density']
    new_Power_Density, new_Power = getPowerFromCalibration(cal_energy_list,cal_power_density,energy_list,8)
    plt.figure(2)
    plot_data(new_Power_Density,min_current_vals,'Mean power density in sample (mW/um2)','Minimum (smoothed) photocurrent (nA)',\
                'Minimum photocurrent versus Mean Power Density in sample',None,111,show=True)
    Kd=getKd(new_Power_Density,current_density)
    print(Kd)
# Cell experiments - Optimisations
    MRR_in_kHz=[200,400,600,800,1000,1200]
    energy_list2=getEnergiesfromMRR(MRR_in_kHz,Kd,cal_energy_list,cal_power_density,8,1)
    min_current_vals=GetCurrent(file,pulse_duration_ms2,energy_list2,divisor=50,dead_time=2,test=True)
    fig=plt.figure(3)
    ax = Axes3D(fig)
    ax.plot_trisurf(MRR_in_kHz,energy_list2,min_current_vals,cmap='coolwarm',alpha=0.5)
    ax.set_xlabel('Input Repetition rate (kHz)')
    ax.set_ylabel('RL energy (uJ)')
    ax.set_zlabel('Minimum (averaged) Membrane Current (nA))')
    ax.set_title('Optimisation: Minimum (averaged) Membrane Current versus Input Repetition rate and corresponding RL energy')
    plt.show()
```

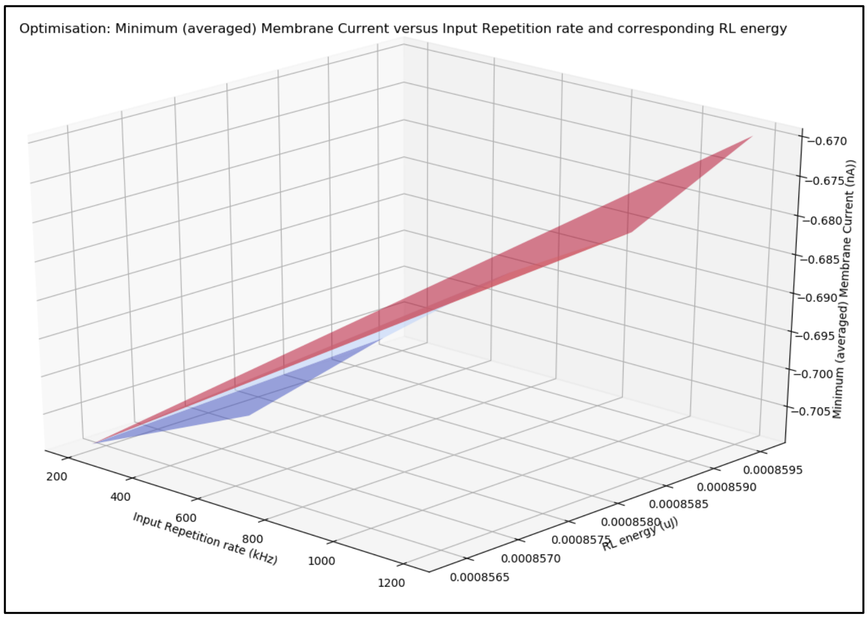
Figure 8. Code for testing the Data analysis module



a. Power calibration plot



b. Michaelis-Menten plot



c. Optimisation curve

Figure 9. Plots resulting from running the testing code on fictitious data

Integration testing

The modules are consolidated in the *MainApp.py*. Integration testing therefore involves executing this script which incorporates all the modules into a single program. In the testing phase, the laser module simply writes commands to a file for the tester to log the commands sent. The lasing is simulated by the blinking LED indicator on the Arduino Uno microcontroller. The data analysis is performed by integrating the user-inputted parameters with a testing dataset.

The results of executing the *MainApp.py* show no errors thrown from the start to the finish of the protocol. The LED blinks the correct number of times; however, the pulse duration cannot be verified at this stage. Finally, the plots resemble the same shapes as shown in Figure 9.

System and acceptance testing

The COVID-19 pandemic restricted access to the laboratory, and therefore system and acceptance testing could not take place.

Risk analysis

This section summarises the risk analysis conducted on the project in its current testing phase. The aim is to identify and manage the risks to support decision-making and actions to achieve the overall objectives. It also serves to increase confidence and predictability of the system.

The risk analysis is conducted using a standard risk management model shown in Figure 10 [24]. Note that it is an iterative process. Risks should be continually monitored and managed in subsequent phases.

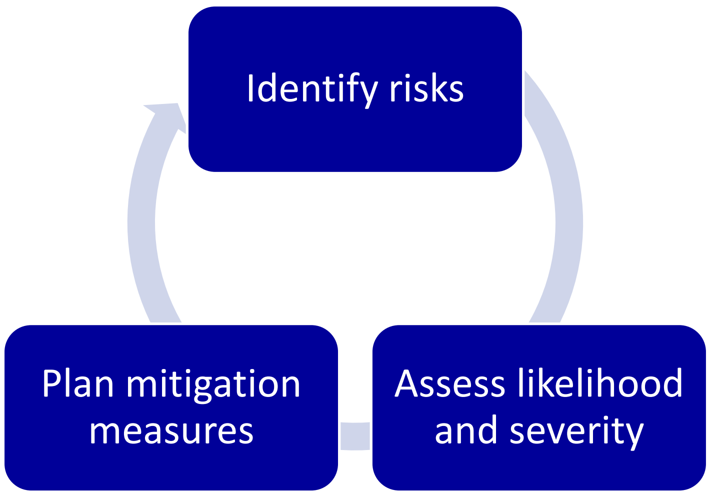


Figure 10. Summary of the risk management process (Adapted from DSDM Consortium (2010))

A system of categorisation facilitated the identification of risks. First, four main categories are identified:

1. Functional: relating to the functional requirements (see Requirements).
2. Non-functional: relating to the non-functional requirements (see Requirements).
3. Operational: associated with maintenance and standard working tasks.
4. Commercial: relating to costs versus benefits of the system.

Within these, four sub-categories of risk are identified:

1. Process: associated with the complexity of the process and the probability of process failure.
2. People: relating to the capability and skills required for effective operations and maintenance.
3. Technical: associated with system failure due to program, hardware, communication, or other technical issues.
4. Data: associated with erroneous data inputs and/or outputs.

Each risk has a score which determines the response strategy. The score is calculated as the product of the likelihood and severity of the risk according to the matrix in Figure 11.

		Severity					
		0	1	2	3	4	5
Likelihood	0	0	0	0	0	0	0
	1	0	1	2	3	4	5
	2	0	2	4	6	8	10
	3	0	3	6	9	12	15
	4	0	4	8	12	16	20
	5	0	5	10	15	20	25

Likelihood		Severity	
0	Impossible	0	No consequence
1	Highly unlikely	1	No harm; small impact on objectives
2	Unlikely	2	No harm; moderate impact on objectives
3	Feasible	3	No harm; fairly high impact on objectives
4	Likely	4	Minor harm and/or serious impact on objectives
5	Highly likely	5	Serious harm

Figure 11. Risk matrix showing the colour coding and definitions of likelihood and severity

The results of the full risk analysis are detailed in the Risk analysis section. In summary, the largest risks are:

1. The Coherent module does not work as intended due to human error or discrepancies between the operational manual and realisation. There is a high likelihood of error (=4) and a very high severity (=5) since the laser may not deploy pulses or the pulses may not deploy as intended which poses a safety risk. This would fail REQ 1 and REQ 16. The mitigation plan is to perform unit testing on the laser, followed by integration testing to validate the powers. A code review may be required.
2. The format of the electrophysiology data varies from the testing dataset used. There is a high likelihood (=4) of disparities which would break the data analysis module, so the severity is high (=4). Requirements REQ 6, REQ 7, REQ 8 and REQ 9 would fail. The mitigation plan is as above: to test the data analysis module on results generated experimentally followed by integration and system testing. The researcher should make use of the "preview channels" feature in the GUI to validate the channels' data look as expected.
3. The curve-fitting algorithms may not converge, so no/ incorrect results are generated. There is a high likelihood (=4) since the algorithms were assessed on a testing dataset. The consequences are severe (=4), failing REQ 6, REQ 7, REQ 8 and REQ 9. The mitigations plan is to perform integration and system testing on several datasets generated from the experiments and verify the convergence and results. As a response, the *scipy.interpolate.interp1d* library can be used to estimate the results.
4. From a commercial perspective, it is likely (=4) that there are many unforeseen complexities which arise when productionising the program in the laboratories. Being deprecated in the testing phase would mean the Objectives fail (severity=4). Mitigation measures include

expanding the risk analysis to increase confidence in the program and improve predictability. The author has also agreed to continue development into production on a flexible basis. As a response, the modules can be repurposed for other projects.

Goals achieved

Py2Pulse partially fulfils the objectives of the project. Specifically, 14 out of the 17 requirements are met with an acceptable level of risk as referenced in the Design and Implementation sections.

This project faced a major limitation of not being able to access the laser in the laboratory due to the COVID-19 pandemic. The requirements which remain unfulfilled are the testability and reliability of the program, and the functionality of deploying laser pulses as these require access to the laser (REQ 1, REQ 11, REQ 17).

Further work

To fulfil the outstanding requirements, the laser module must be tested on the laser. All combinations of the allowable repetition rates from the Laser's GUI should be added to Py2Pulse. All levels of testing outlined must be carried out in order. To mitigate multiple risks identified in Appendix C, a user manual should be included and training on the EP board should be provided. A detailed business proposal estimating resources required for deployment is necessary to advance the program to production.

In addition to meeting the outstanding requirements, the following enhancements to the program are recommended:

- Exception handling should be added to each of the modules. This would improve the testability of the program.
- Improve the file storage such that files are saved to the Cloud instead of locally.
- Add restrictions to the GUI to confine parameters to reasonable ranges.

7. Conclusion

Py2Pulse was developed to overcome some of the pertinent challenges of conducting optogenetics experiments. It enables researchers to deploy millisecond pulses of radiation onto a monolayer of recombinant cells and optimise the laser parameters for maximal photocurrent generation. It features an interactive user interface which allows parameters to be viewed and manipulated, automated folder organisation, integrated data analysis and a guided user experience through the experimental protocol. Py2Pulse is in its testing phase, requiring access to the laser to progress through testing before being deployed.

8. Acknowledgements

I would like to thank my Supervisor, Dr Amanda Foust, for her continuous guidance and support in the development of the program. As an expert in the field of Neuroscience, Dr Foust initiated the project topic and provided insightful feedback from inception to its current state. It is with thanks to her clear direction for the project that the COVID-19 pandemic had minor implications on the project's development.

I would also like to give special thanks to Dr Peter Quicke for providing the historical electrophysiology data used for testing the data analysis. In addition to this, I am very grateful for all the time taken out to work through my code, suggest improvements and explain complicated concepts in a clear and concise manner. I owe my new-found knowledge of Object-Oriented Programming and Hardware Abstraction Layers to you.

9. References

1. Chaigneau E, Ronzitti E, Gajowa MA, Soler-Llavina GJ, Tanese D, Brureau AYB, et al. Two-photon holographic stimulation of ReaChR. Front Cell Neurosci. 2016;10(OCT2016).
2. Barnett SC, Perry BAL. Optogenetic stimulation : Understanding memory and treating deficits. 2018;(April):457–70.
3. Chen XI, Ronzitti E, Lee BR, Daigle TL, Dalkara D, Zeng X, et al. In Vivo Submillisecond Two-Photon Optogenetics with Temporally Focused Patterned Light. J Neurosci. 2020;39(18):3484–97.
4. Google Scholar [Internet]. 2020 [cited 2020 Sep 7]. Available from: https://scholar.google.com/scholar?q=optogenetics&hl=en&as_sdt=0%2C5&as_ylo=2018&as_yhi=2020
5. Oron D, Papagiakoumou E, Anselmi F, Emiliani V. Two-photon optogenetics [Internet]. 1st ed. Vol. 196, Progress in Brain Research. Elsevier B.V.; 2012. 119–143 p. Available from: http://dx.doi.org/10.1016/B978-0-444-59426-6.00007-0
6. Papagiakoumou E, Ronzitti E, Chen I, Gajowa M, Picot A, Emiliani V. Chapter 10. Two-Photon Optogenetics by Computer-Generated Holography. Neuromethods, Optogenetics A Roadmap. 2018;133:175–97.
7. Picot A, Dominguez S, Liu C, Chen IW, Tanese D, Ronzitti E, et al. Temperature Rise under Two-Photon Optogenetic Brain Stimulation. Cell Rep. 2018;24(5):1243-1253.e5.
8. Soor NS, Quicke P, Howe CL, Pang KT, Neil MAA, Schultz SR, et al. All-optical crosstalk-free manipulation and readout of Chronos-expressing neurons. J Phys D Appl Phys. 2019;52(10).
9. Quinn TA. Principles of Optogenetic Methods and Their Application to Cardiac Experimental Systems. 2019;10(September).
10. Klapoetke NC, Murata Y, Kim SS, Pulver SR, Birdsey-Benson A, Cho YK, et al. Independent optical excitation of distinct neural populations. Nat Methods. 2014;11(3):338–46.
11. Saran S, Gupta N, Royb S. Theoretical analysis of low-power fast optogenetic control of firing of Chronos-expressing neurons. Neurophotonics. 2018;5(2).
12. Conti R, Assayag O, Sars V De, Guillon M, Emiliani V. Computer Generated Holography with Intensity-Graded Patterns. 2016;10(October):1–11.

13. Chen IW, Papagiakoumou E, Emiliani V. Towards circuit optogenetics. Curr Opin Neurobiol [Internet]. 2018;50:179–89. Available from: https://doi.org/10.1016/j.conb.2018.03.008

14. Ronzitti E, Conti R, Zampini V, Tanese D, Foust AJ, Klapoetke N, et al. Submillisecond optogenetic control of neuronal firing with two-photon holographic photoactivation of chronos. J Neurosci. 2017;37(44):10679–89.

15. Diaspro A, Chirico G. Two-photon excitation microscopy. Adv Imaging Electron Phys. 2003;126:399–429.

16. Denk W, Strickler J, Webb W. Two-Photon Laser Scanning Fluorescence Microscopy. Science (80-). 1990;248(4951):73–6.

17. Karpf S, Eibl M, Sauer B, Reinholz F, Hüttmann G, Huber R. Two-photon microscopy using fiber-based nanosecond excitation. Biomed Opt Express. 2016;7(7):2432–2440.

18. Macias-Romero C, Zubkovs V, Wang S, Roke S. Wide-field medium-repetition-rate multiphoton microscopy reduces photodamage of living cells. Biomed Opt Express. 2016;7(4):1458.

19. Gamper N, Stockand JD, Shapiro MS. The use of Chinese hamster ovary (CHO) cells in the study of ion channels. 2005;51:177–85.

20. Arduino. Arduino Forum: Arduino clock accuracy [Internet]. 2009. Available from: https://forum.arduino.cc/index.php?topic=13289.0#:~:text=The accuracy of the millis,right for a cheap crystal).

21. The SciPy Community. scipy.optimize.curve_fit [Internet]. 2020 [cited 2020 Sep 6]. Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

22. Huang, Dijiang; Wu H. Chapter 2 - Virtualization. In: Mobile Cloud Computing: Foundations and Service Models. Elsevier; 2018. p. 31–64.

23. Binder JM, Stark A, Tomek N, Scheuer J, Frank F, Jahnke KD, et al. Qudi: A modular python suite for experiment control and data processing. SoftwareX [Internet]. 2017;6:85–90. Available from: http://dx.doi.org/10.1016/j.softx.2017.02.001

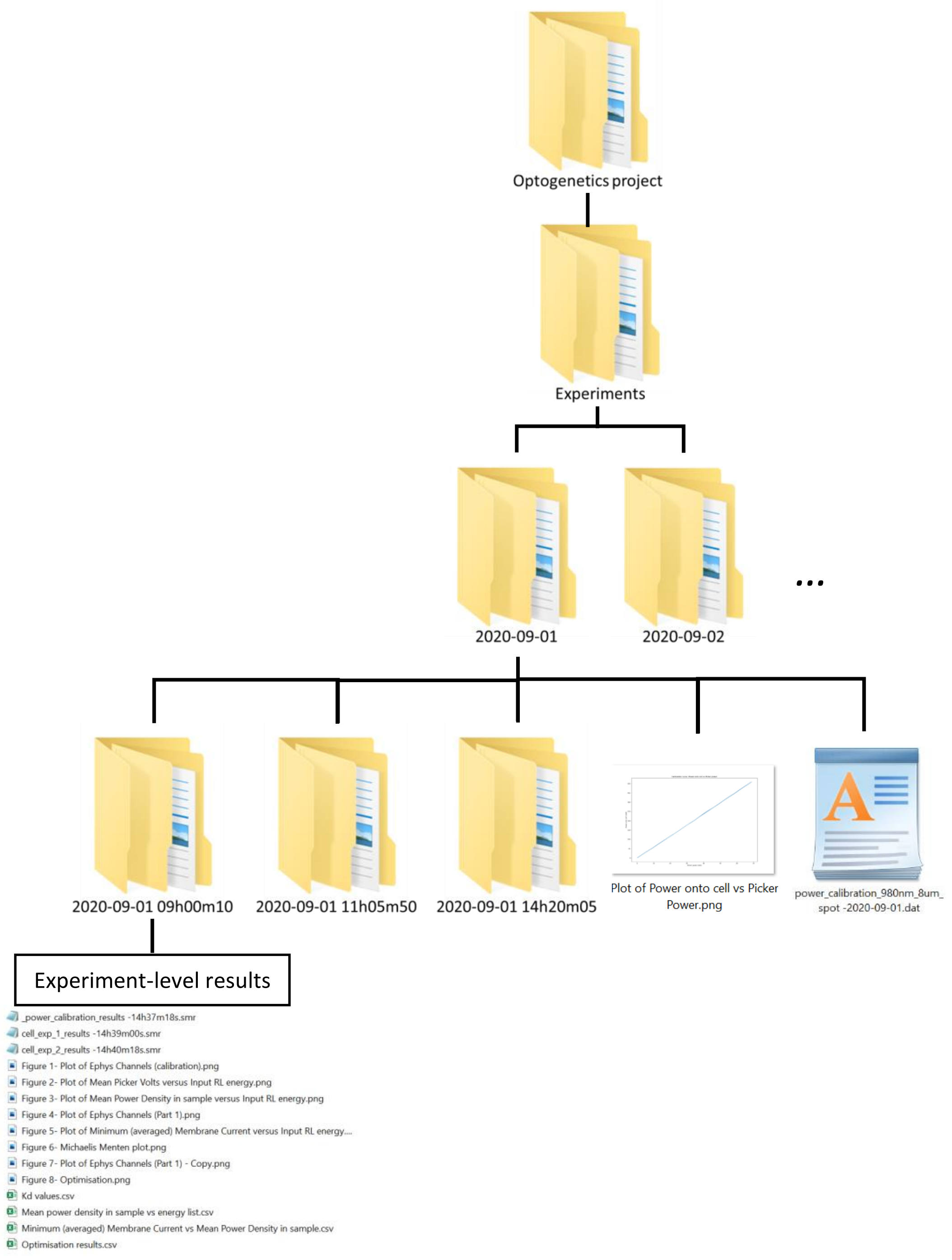
24. DSDM Consortium. Agile Project Management Handbook v2.0. Agile Business Consortium; 2010. 177–180 p.

Appendix A. Laser parameters

Table 1. Input parameters for the ultrafast pulse ytterbium fiber laser

Parameter (Symbol)	**Value**
Central wavelength, λ	1035 nm
Pulse duration, τ_p	280 fs
Time, Δt	5 ms
Repetition rate, f_p	Range between 20 kHz – 20 MHz.
Average power, p_{ave}	To be determined experimentally. Starting value is 40 mW (100%).

Appendix B. Folder organisation



```
_power_calibration_results -14h37m18s.smr
cell_exp_1_results -14h39m00s.smr
cell_exp_2_results -14h40m18s.smr
Figure 1- Plot of Ephys Channels (calibration).png
Figure 2- Plot of Mean Picker Volts versus Input RL energy.png
Figure 3- Plot of Mean Power Density in sample versus Input RL energy.png
Figure 4- Plot of Ephys Channels (Part 1).png
Figure 5- Plot of Minimum (averaged) Membrane Current versus Input RL energy....
Figure 6- Michaelis Menten plot.png
Figure 7- Plot of Ephys Channels (Part 1) - Copy.png
Figure 8- Optimisation.png
Kd values.csv
Mean power density in sample vs energy list.csv
Minimum (averaged) Membrane Current vs Mean Power Density in sample.csv
Optimisation results.csv
```

Figure 12. Overview of the default folder organisation structure created by Py2Pulse

Appendix C. Risk analysis

Table 2. Detailed risk analysis showing the severity, likelihood and mitigation strategy for each identified risk

Category	Sub-category	Description	Consequence	Likelihood (0-5)	Severity (0-5)	Risk score	Mitigation plan	Response plan (if the risk materialises)
Functional	Process	The Coherent module was built using the Operational Manual. The serial communication commands were developed from multiple sections in the manual. Due to human error, incorrect commands may have been chosen/ commands may have been missed.	Laser will not deploy pulses; or laser will deploy pulses with the incorrect MRR, energy level or pulse width, or will not start up or shut down correctly.	4	5	20	1) Unit testing - check that pulses are deployed by running the Coherent module independently on the laser. 2) Integration testing - run the Py2Pulse power calibration and check that the calibration curve has the expected sigmoid shape.	Review the Coherent module and fix the bugs.
Functional	Data	The format of the production electrophysiology data is different from the testing data format.	The data analysis module in Py2Pulse will fail, so there will be no results.	4	4	16	1) Unit testing: test the data analysis module independently on results generated from each step of the experimental protocol. 2) Integration testing: run the Py2Pulse program and preview the channels data after uploading the file each time to verify the data looks as expected.	Review the data analysis module and fix the bugs.
Non-functional	Data	Reliability: The curve-fitting algorithms (scipy.optimize.curve_fit) may fail to converge.	The data analysis module will fail, so no results will be generated.	4	4	16	1) Integration testing - run the data analysis module on several datasets generated from the experiments and verify the outputs and convergence 2) System and	Use scipy.interpolate.interp1d library to interpolate values from the curve. Note that these

							acceptance testing - run the protocol end-to-end on real cells and verify the models converge	will not be as accurate.
Commercial	Process	There could be a lot of unforeseen complexities that arise when completing the testing of the program (time consuming and costly)	The program remains stale in testing phase and is deprecated.	4	4	16	1) This report clarifies the riskiest elements and details the design to enable debugging and improve confidence in the program; 2) the author has agreed to continue the development of the program into production on a flexible basis	The modules can be repurposed for other projects
Commercial	Data	It may be that the results of this study are inconclusive.	The cost of building and carrying out the experiments would outweigh the benefits.	4	4	16	1) A user manual should be included in the production version to ensure the researcher is meticulous in following the experimental protocol. This would ensure valid results are collected with each run.	N/A
Functional	People	The user does not know how to use the electrophysiology data acquisition board.	No results will be captured.	4	3	12	1) Add this as a prerequisite to the GUI Terms of Use. 2) Include training as part of induction.	Ask for assistance from a senior researcher
Functional	People	The user is not informed of reasonable ranges of parameters to input into the GUI, and inputs invalid ranges	Either the pulses will be deployed which may damage the cells; or an exception will be raised by the laser and pulses will not be deployed.	3	4	12	1) Limits have been applied to certain parameters on the GUI already, but this should be extended all parameters to only allow an acceptable range. 2) Starting parameters are provided in Appendix A.	Provide an induction course which covers the allowable ranges and repercussions of invalid values.

Non-functional	Data	The outputs of the experiments run on multiple cells may overwrite each time if the researcher fails to update the timestamped directory.	Loss of some results	4	3	12	1) The code should be improved to include the cell label with each set of results generated. 2) A user manual should accompany the productionised version of the program, instructing the researcher to begin each new experiment from the organisation tab.	N/A
Operational	Data	Photobleaching may occur in the cells if photostimulated too often in a period of time.	The evoked photocurrents will be lower than expected, leading to inaccurate results.	3	4	12	1) Ensure that there is sufficient time between stimulations of the same cell 2) Change cells after the completion of the protocol.	Repeat the experiment on the same cell on different days and calculate the mean and standard deviations of the evoked photocurrents.
Commercial	People	The project requires at least one skilled researcher with coding skills to complete the deployment; as well as one scientist to conduct the experiments which may be costly.	The program either remains in testing phase or is deployed and no experiments are carried out.	3	4	12	1) A business case proposal should be prepared detailing the estimated resources and costs required to achieve the objectives to aid decision-making.	N/A

Commercial	Technical	Technical faults may arise. These may be hardware faults or software bugs.	Hardware repairs are costly; and software repairs may take time and skilled resources.	3	4	12	1) The GUI has included a link to the laser manual which details the requirements for normal operation 2) This report has detailed the design of the software as well as highlighted the riskiest elements	N/A
Operational	Technical	The chiller may require maintenance or a refill of the coolant.	The laser may operate outside the recommended operating conditions; may overheat.	2	5	10	1) The GUI has included a link to the laser manual which details the requirements for the chiller 2) A log of the maintenance plan for the chiller should be kept updated in the laboratory.	Request for an expert to perform a maintenance check on the coolant system.
Functional	Technical	The voltage clamp setup is faulty.	No/ incorrect photocurrent data will be captured.	3	3	9	1) A log of the history of usage and any issues encountered should be kept in the laboratory and checked prior to the experiment.	The device should be repaired or replaced (Note replacement is costly).

Operational	Process	The program creates copies of files and saves all results in .csv and .png formats so the local computer may run out of storage space.	The program will fail to save the results when it reaches capacity.	3	3	9	1) Back up results to the cloud and clear directories on a regular basis (daily/ weekly depending on throughput) 2) The code can be improved to move instead of copy uploaded files 3) The program can be enhanced to push results to an online directory instead of locally.	Monitor the computer space and move files to a separate hard drive regularly.
Non-functional	People	The researcher must have had the induction to enter the laboratory.	The researcher is unable to access the laser to conduct the experiments.	2	4	8	1) The Terms of Use includes the laboratory induction as a prerequisite to inform the researcher of this requirement.	Allow the researcher to familiarise themselves with the program in the testing phase without needing the hardware.
Non-functional	Technical	The system may suffer performance issues when running for long periods or analysing large datasets.	The program crashes or is very slow. Minor losses to results (if any).	4	2	8	1) System testing: Test the protocol on several cells and log the performance of each run and each upload of the electrophysiology data. 2) Keep a log of the time taken to conduct the experiments in testing phases	Modify the protocol to restart after each cell experiment.
Functional	Technical	The 3D optimisation plot might be required as a .png image and currently this is not available.	The researcher is unable to use the .png optimisation plot at a later stage.	4	2	8	Upgrade the code to utilise the pyqtgraph library instead of the mpl_toolkits Axes3D library. The former enables exporting.	The results are saved as a text file and can be plotted at any time and screenshotted manually.

Functional	Technical	The electrophysiology data acquisition board is faulty.	No results or inaccurate results will be captured.	2	3	6	1) A log of the history of usage and any issues encountered should be kept in the laboratory and checked prior to the experiment.	The device should be repaired or replaced (Note replacement is costly).
Functional	Technical	The power meter is faulty.	No results or inaccurate results will be captured.	2	3	6	1) A log of the history of usage and any issues encountered should be kept in the laboratory and checked prior to the experiment.	The device should be repaired or replaced (Note replacement is costly).
Operational	People	The program relies on the availability of a researcher to conduct the experiments. It may be that there is little to no capacity for some time.	There will be delays to achieving the objective of parameter optimisation.	3	2	6	1) Detail the expected resources required for completion to action a plan to increase capacity.	N/A
Functional	Technical	The laser system is faulty.	The laser could overheat, deploy high-energy pulses continuously or have an electrical fault which could start a fire.	1	5	5	1) The laser should be tested and approved by the manufacturer prior to use. 2) The laser safety training has been added as a prerequisite into the Terms of Use. PPE and fire safety is detailed in the training. 3) The researcher must not be inside the laboratory while lasing is taking place.	The device should be repaired or replaced (Note replacement is costly).

Non-functional	Process	The program is too complicated to be used by a researcher without coding knowledge.	The experimental protocol cannot take place.	1	4	4	1) Design decisions such as panelled layouts and colour coding have been implemented to simplify the user experience. 2) Include a detailed user manual when the program goes live	Redesign the GUI to enhance the user experience and simplify the program.