Zentralinstitut für Seelische Gesundheit

AutonoMouse 2

Software-Dokumentation, Version 1.0

Letzte Aktualisierung: 19.7.2019

Von Bram, Michael

Kontakt:
michael.bram@ipa.fraunhofer.de

Inhaltsverzeichnis

Software-Dokumentation

Die Software von dem AutonoMouse 2 wird in Python geschrieben und benutzt das API PyQt5, das Package numpy und scipy. Die AutonoMouse2Control führt die eigentliche Jobs bzw. Trials durch. Der Schedule-Generator generiert die Schedules, die in dem Control durchgeführt werden. Die Zuordnung der Schedules zu den Mäusen erfolgt in dem Animal Window.

Schedule Generator

Das Main-Skript ist in scheduleMain.py in dem Ordner „schedule-generator(Beast)“.

Name	**scheduleMain**
Type	*Module*
Beschreibung	Das Hauptmodul des Schedule-Generators
Inhalt	**MainApp**
File	~\schedule-generator(Beast)\scheduleMain.py

Name	**MainApp**
Type	*Class*
Beschreibung	Die App des Schedule-Generators. Erbt *Qt.QMainWindow* und *mainDesign.Ui_MainWindow*
Methods	**generate()** ➔ Ruf die Method „generate_schedule“ von dem ausgewählten Class in dem Modul Schedule-Widgets wieder **select_schedule_type()** ➔ Zuordnung des ausgewählten Class im Modul Schedule-Widgets mit dem Variable „current_schedule_type“ **draw_pulse()** ➔ Zeichnet das Pulssignal für das Odour mit **save_schedule()** ➔ Speichert das Schedule, wenn ein Schedule generiert wurde
File	~\schedule-generator(Beast)\schedule-main.py

Name	**PulseInterface**

Type	*Modul*
Beschreibung	Das Modul zum Interface des Zeichnens der Pulssignale
Methods	**make_pulse(**sampling_rate, global_onset, global_offset, params_list**)** ➔ Das Interface zur Entscheidung welcher Pulse mit welchem Parameter generiert wird. Argument: sampling_rate = integer Wert; global_onset, global_offset = float (in Sekunde, einfaches Zero-Padding); params_list = eine Liste von Dictionary von Parameters; entspricht params in BeastScheduleWidget) Return: pulse_matrix = 2d-numpy Array, enthält der generierte Pulse für jedes Odour in einem Trial; t = 1d-numpy Array, enthält die Zeitachse von pulse_matrix
File	~\schedule-generator(Beast)\ SchedulePyPulse\PulseInterface.py

Name	**PulseGeneration**
Type	*Modul*
Beschreibung	Das Modul zum Zeichnen das Pulssignal
Methods	**simple_pulse(**sampling_rate, params**)** ➔ Das Method zur Erzeugund des Pulssignals Argument: sampling_rate = integer Wert; params = eine Dictionary von Parameters; entspricht ein Element in params in BeastScheduleWidget) Return: pulse = 1d-numpy Array, enthält der generierte Puls mit dem Parameter. t = 1d-numpy Array, enthält die Zeitachse von pulse In der Anwendung in AutonoMouse 2 sind alle andere Pulse noch **nicht** implementiert.
File	~\schedule-generator(Beast)\ SchedulePyPulse\PulseGeneration.py

Name	**ScheduleWidget**
Type	*Modul*
Beschreibung	Beinhaltet alle Klasse der Schedule-Typen
Inhalt	**BeastScheduleWidget** In der Anwendung in AutonoMouse 2 sind folgende Widget noch **nicht** implementiert: **PretrainWidget** **ConcGNGWidget**

	SimpleGNGWidget **SimpleCorrWidget** **CorrWidget** **ContCorrWidget** **CorrOnsetDisruptWidget** **CorrDifficultySwitchWidget** **CorrRandomisedFrequencyWidget** **CorrRandomisedFrequency2Widget** **ShatterValveWidget** **CorrDifficultySwitchCameraWidget**
File	~\schedule-generator(Beast)\ScheduleModels\ScheduleWidgets.py

Name	**BeastScheduleWidget**[1]
Type	*Class*
Beschreibung	Das Widget für die Generierung des Schedules für AutonoMouse 2 Control. Erbt *Qt.QWidgets* und *beastScheduleDesign.Ui_Form*
Methods	**flatten_value(**value**)** ➔ Tauschen value mit 0, wenn value < 0 ist. Argument: value = integer Wert Return: 0, wenn value < 0. value, wenn value > 0 **change_reward_map()** ➔ Aktuliesieren die angzeigte Reward-Map Tabelle in dem Widget **generate_schedule(**valence_map**)** ➔ Generiert ein randomisierte Schedule. Die Randomisierung wird durch das Modul „**Gen**“ Argument: valence_map = valence_map von dem parent-Fenster Return: schedule = eine Liste von „trial“ „trial“ = [rewarded, valve, valence map, lick fraction] **pulse_parameters(**trial**)** ➔ Speichern des Parameters in einem Dictionary Argument: trial = [rewarded, valve, valence map, lick fraction] Return: params = Liste von Dictionary von Parametern. Die Liste entspricht die Liste von Odours in einem Trial
File	~\schedule-generator(Beast)\ ScheduleModels\ScheduleWidgets.py

[1] Der Name wird in der nächsten Aktualisierung geändert.

Name	**Gen**
Type	*Modul*
Beschreibung	Das Modul zur Randomisierung der Sequenz der Trials in einem Schedule.
Methods	**reward_sequence(**n_trials**)** ➔ Generieren der randomisierten Sequenz. Die ersten 6 sind festgesetzt. Es wird vorgesehen, dass nicht mehr als drei gleiche Werte nacheinander sind. Argument: n_trials = integer Wert. Die Anzahl des Trials in einem Schedule Return: sequence = ein Array mit der Länge von n_trials **generate_correlation_structure(**n, rho**)** ➔ In AutonoMouse 2 **nicht** implementiert.
File	~\schedule-generator(Beast)\ Generation\Gen.py

AutonoMouse 2 Controller

Das Main-Skript ist in Main.py in dem Ordner „Autonomouse 2 Controller“. Die Sequenz für den Controller wird in einem Thread durchgeführt. Das Thread wird als Job von einem Worker-Klasse durchgeführt und die Worker-Klasse wird von von einem Controller-Klasse verwaltet. Die Klassen befinden sich in dem Modul „*ExperimentControl*“.

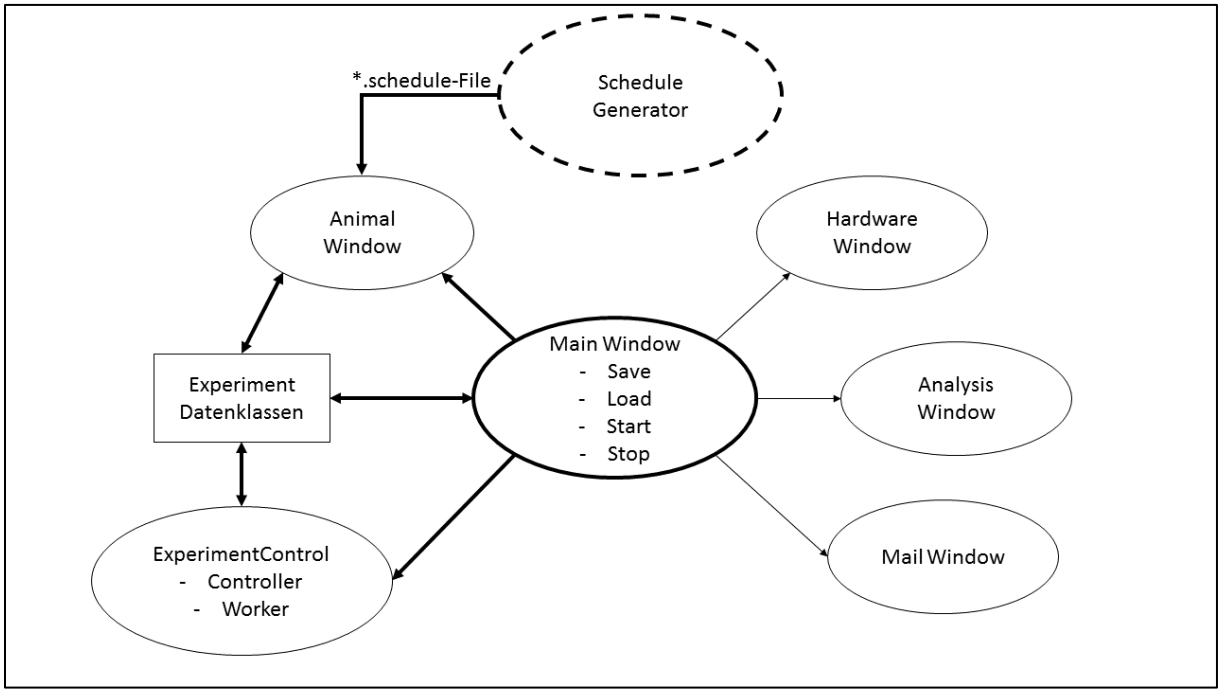

Abbildung 1: Überblick

Name	**main**
Type	*Module*
Beschreibung	Das Hauptmodul des Controls
Inhalt	**MainApp**
File	~\Autonomouse 2 Controller\main.py

Name	**MainApp**
Type	*Class*
Beschreibung	Die App des Controls. Erbt *Qt.QMainWindow* und *mainDesign.Ui_MainWindow*
Methods	**setup_experiment_bindings(**experiment**)** ➔ Bindet die Klasse „Experiment“ im MainApp ein. Wird beim Speichern und Laden aufgerufen.

	Argument: experiment = Klasse aus dem Modul „Experiment“ **load_config_data()** ➔ Ladet die Hardware-Einstellung ein **save_experiment()** ➔ Speichern des gebundenen Experiments mit Pickle. Bevor das Thread gestartet wird, muss eine Experiment-Datei gespeichert werden. **load_experiment()** ➔ Laden des gebundenen Experiments Folgende Methods sind für die Qt-Signalen wichtig: **thread_control()** **windows_control()** **experiment_saved()** **status_changed()** **open_animal_window()** **open_hardware_window()** **open_control_window()** **open_mail_window()** **open_analysis_window()** **update_trial_view()** **update_data_view()** **update_graphics_view()** **on_trial_selected()** **update_experiment_info()**
File	~\Autonomouse 2 Controller\main.py

Name	**ExperimentControl**
Type	*Module*
Beschreibung	Das Modul mit Worker-Controller Klassen
Inhalt	**ExperimentWorker, ExperimentController**
File	~\Autonomouse 2 Controller\Controllers\ExperimentControl.py

Name	**ExperimentController**
Type	*Class*
Beschreibung	Verwaltet das Thread. Start- und Stop-Button werden hier implementiert. Erbt *MainApp*
Methods	**update_pref(**new_pref**)** ➔ Aktualisiert Änderungen vom Hardware-Window, dass die Änderungen sofort im Thread angenommen werden. Argument: new_pref ist ein Qt-Signal, das von dem Hardware-Window bei Änderungen der Hardwareeinstellungen ausgegeben wird. **start()** ➔ Startet das Thread und speichert die Startzeit **stop()** ➔ Terminiert das Thread
File	~\Autonomouse 2 Controller\Controllers\ExperimentControl.py

Name	**ExperimentWorker**
Type	*Class*
Beschreibung	Beinhaltet den Job, der im Thread durchgeführt wird (siehe die Methode trial()). Erbt *ExperimentController*
Methods	**trial()** ➔ Der Job, der im Thread durchgeführt wird. Die Signale *trial_end* und *finished* werden am Ende eines Trials ausgegeben bzw. wenn das Stop-Button gedrückt wird. **check_status()** ➔ Überprüft die Zeit, ob ein *Deadman* E-Mail gesendet werden soll. **check_licks()** ➔ Überprüft die Anzahl des Licks einer Maus in einem gegebenen Zeitfenster. Wenn die Anzahl zu niedrig ist, eine Warnung-E-

	Mail wird gesendet. Die Daten zur Überprüfung sind die gespeicherte Lick-Liste im Mouse-Objekt. **animal_present()** ➔ Überprüft, ob die Lichtschranke gebrochen ist. **get_present_animal()** ➔ Liest der Id-Tag der Maus, die die Lichtschranke gebrochen ist. **reward_animal(**animal**)** ➔ Gibt das Wasser. Das Ventil der Wasserabgabe wird so lange geöffnet, wie das Reward eingestellt ist. Näheres zum Reward-Einstellung, siehe Animal-Window-Klasse. Argument: eine Klasse von dem Experiment-Modul. Nähres zu der animal-Klasse, siehe Experiment-Modul **timeout()** ➔ Softwaretiming für das Timeout. Der Rechner schläft für gegebener Zeit. **save_data(**argument) ➔ Wandelt die zu speichernde Daten in geeignete Format und speichert die Daten in csv-Datei. Argument: Daten, die gespeichert werden. Diese sind - animal_id: der Id der Maus - timestamp: die Zeit, wann ein Trial beendet wird - rewarded: Boolean Variabel, ob das Trial belohnt warden soll - wait_response: die Anzahl des Licks vor dem Abgabe des Odours - response: Boolean Variabel, ob die Maus in diesem Trial geleckt wird. - correct: Boolean Variabel, ob die Maus des Trial richtig durchgeführt hat. - timeout: Boolean Variabel, ob ein Timeout gegeben wurde - pulses: Das digitale Pulssignal von dem Lecken. Ist obsolet und nicht in csv-Datei gespeichert - time_axis: Die Zeitachse des Pulssignals. Ist obsolet und nicht in csv-Datei gespeichert. - file_timestamp: die Startzeit. Wird für die Namengebung der csv-Datei weitergegeben. - lick_time: Die Zeiten, wann eine Maus nach der Abgabe des Odours geleckt hat. - licks: Die Anzahl des Licks nach der Abgabe des Odours - wait_time: Die Zeiten, wann eine Maus vor der Abagabe des Odours geleckt hat.

File	~\Autonomouse 2 Controller\Controllers\ExperimentControl.py

Name	**Experiment**
Type	*Module*
Beschreibung	Das Modul mit den Daten-Klassen der Experiment
Inhalt	**Experiment, Mouse, Schedule, Trial**
File	~\Autonomouse 2 Controller\Controllers\ExperimentControl.py

Name	**Experiment**
Type	*Class*
Beschreibung	Beinhalte die Liste von dem Mouse-Objekt und die Liste von allem durchgeführten Trials.
Methods	**add_mouse(**id,water**)** ➔ addiert ein Mouse-Objekt mit dem Parameter *id* und *water* in der Liste Argument: *id* = Maus-RFID, *water* = die Zeitfenster, wie lange der Ventil für die Belohnungsabgabe geöffnet werden soll. **add_trial()** ➔ Append der Trial-Daten in einer Liste. Diese Liste wird in einer Tabelle im Main Window dargestellt. **save()** ➔ Pickle der Experiment-Daten
File	~\Autonomouse 2 Controller\Modells\Experiment.py

Name	**Mouse**
Type	*Class*
Beschreibung	Beinhalte die Liste von dem Schedule-Objekt, die die Maus durchführen soll und die Liste Lick-Daten einer Maus.
Methods	**update_licks(**timestamp, rewarded, licks_before, licks_after, total_licks**)**

	➔ Addiert die Licks-Daten in der Liste und speichert diese. Argument: Licksdaten. Diese Daten werden bei dem Method *check_licks* in dem *ExperimentWorker*-Klasse wiederverwendet. **add_schedule()** ➔ Addiert ein Schedule-Objekt in der Schedule-Liste. Das Schedule-Objeckt wird mit dem Parameter aus den gespeicherten Schedules konstruiert. Diese gespeicherten Schedules wurden von dem Schedule-Generator generiert. **current_trial()** ➔ Gibt die aktuelle Trial-Parameter zurück **current_trial_pulse()** ➔ Gibt die aktuelle Odoursignal-Parameter von einem Trial zurück. **current_trial_idx()** ➔ Gibt der Index des aktuellen Trials zurück. **advance_trial()** ➔ Ändert der Index zu dem Index des nächsten Trials.
File	~\Autonomouse 2 Controller\Modells\Experiment.py

Name	**Schedule**
Type	*Class*
Beschreibung	Beinhalte die Liste von dem Trial-Objekt, die die Maus durchgeführt hat und die Liste der Trial-Parameter aus der gespeicherten Schedule.
Methods	**add_trial_data(**timestamp, wait_response, correct, timeout, rewarded**)** ➔ Addiert die Trial-Daten in der Liste der Trial-Objekt Argument: Trial-Daten. Diese Daten werden bei der Analyse und *deadman* E-Mail wiederverwendet. **n_trials()** ➔ Gibt die Anzahl des Schedule-Trials zurück **trial_left()** ➔ Gibt die Anzahl der durchzuführende Trials in der gespeicherten Schedule zurück
File	~\Autonomouse 2 Controller\Modells\Experiment.py

Name	**AppWindows**
Type	*Module*
Beschreibung	Das Modul mit den Sub-Window-Klassen
Inhalt	**ControlWindow**: Der Window zur Fernüberwachung mit Kamera**,** **MailWindow**: Der Window zur Aktualisierung der Mailing-Liste**,** **Animal Window**: Der Window zur Aktualisierung der Animal-Liste in dem Experiment-Klasse**,** **HardwareWindow**: Der Window zur Aktualisierung der Hardware-Preferences**,** **AnalysisWindow**: Der Window zum Anzeigen der Peformance der Mäuse
File	~\Autonomouse 2 Controller\Windows\AppWindows.py