

Zentralinstitut für Seelische Gesundheit

AutonoMouse 2

Software-Dokumentation, Version 2_Ports

Letzte Aktualisierung: 24.7.2019
Von Bram, Michael

Kontakt:
michael.bram@ipa.fraunhofer.de

Inhaltsverzeichnis

Unterschied zu Version 1.0	2
Schedule Generator	2
scheduleMain	2
MainApp	2
PulseInterface	3
PulseGeneration	3
ScheduleWidget	4
Autonomouse2ScheduleWidget	4
Gen	5
AutonoMouse 2 Controller	6
main	6
MainApp	6
ExperimentControl	7
ExperimentController	8
ExperimentWorker	8
Experiment	10
Experiment	10
Mouse	10
Schedule	11
AppWindows	12

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.

Unterschied zu Version 1.0

- Es gibt 2 Leck-Ports
- Die Einstellung der Belohnung erfolgt nicht mehr im Animal-Window, sondern in Schedule-Generator. Das heißt, dass die Belohnungsparameter wird in dem Schedule bzw. Trial-Parameter gespeichert.
- Ein anderes Trial-Modell außer GNG (Go or no Go), nämlich LR (Left-Right).
- Die Belohnungsgröße und die Wahrscheinlichkeit der Belohnung kann jetzt per Schedule eingestellt werden.
- Die Daten-Klassen werden auch entsprechend geändert.
- Die randomisierte Reihenfolge der generierte Trials ist nicht mehr auf dem Reward basiert, sondern auf dem Odors.

Version 2 Ports ist nicht *backward compatible*!

Schedule Generator

Das Main-Skript ist in scheduleMain.py in dem Ordner „Autonomouse 2 Schedule“.

Name	scheduleMain
Type	<i>Module</i>
Beschreibung	Das Hauptmodul des Schedule-Generators
Inhalt	MainApp
File	~\Autonomouse 2 Schedule\scheduleMain.py

Name	MainApp
Type	<i>Class</i>
Beschreibung	Die App des Schedule-Generators. Erbt <i>Qt.QMainWindow</i> und <i>mainDesign.Ui_MainWindow</i>
Methods	generate() → Ruf die Method „generate_schedule“ von dem ausgewählten Class in dem Modul Schedule-Widgets wieder select_schedule_type()

	<p>➔ Zuordnung des ausgewählten Class im Modul Schedule-Widgets mit dem Variable „current_schedule_type“</p> <p>draw_pulse()</p> <p>➔ Zeichnet das Pulssignal für das Odour mit</p> <p>save_schedule()</p> <p>➔ Speichert das Schedule, wenn ein Schedule generiert wurde</p>
File	~\Autonomouse 2 Schedule\schedule-main.py

Name	PulseInterface
Type	<i>Modul</i>
Beschreibung	Das Modul zum Interface des Zeichnens der Pulssignale
Methods	<p>make_pulse(sampling_rate, global_onset, global_offset, params_list)</p> <p>➔ Das Interface zur Entscheidung welcher Pulse mit welchem Parameter generiert wird.</p> <p>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)</p> <p>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</p>
File	~\Autonomouse 2 Schedule\ SchedulePyPulse\PulseInterface.py

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

Name	ScheduleWidget
Type	<i>Modul</i>
Beschreibung	Beinhaltet alle Klasse der Schedule-Typen
Inhalt	<p>Autonomouse2ScheduleWidget</p> <p>In der Anwendung in AutonoMouse 2 sind folgende Widget nicht implementiert:</p> <p>PretrainWidget</p> <p>ConcGNGWidget</p> <p>SimpleGNGWidget</p> <p>SimpleCorrWidget</p> <p>CorrWidget</p> <p>ContCorrWidget</p> <p>CorrOnsetDisruptWidget</p> <p>CorrDifficultySwitchWidget</p> <p>CorrRandomisedFrequencyWidget</p> <p>CorrRandomisedFrequency2Widget</p> <p>ShatterValveWidget</p> <p>CorrDifficultySwitchCameraWidget</p>
File	~\Autonomouse 2 Schedule\ScheduleModels\ScheduleWidgets.py

Name	Autonomouse2ScheduleWidget
Type	<i>Class</i>
Beschreibung	<p>Das Widget für die Generierung des Schedules für AutonoMouse 2 Control.</p> <p>Erbt <i>Qt.QWidgets</i> und <i>beastScheduleDesign.Ui_Form</i></p>
Methods	<p>flatten_value(value)</p> <p>→ Tauschen value mit 0, wenn value < 0 ist. Argument: value = integer Wert Return: 0, wenn value < 0. value, wenn value > 0</p> <p>change_reward_map()</p> <p>→ Aktualisieren die angezeigte Reward-Map Tabelle in dem Widget</p> <p>generate_schedule(valence_map)</p> <p>→ Generiert ein randomisierte Schedule. Die Randomisierung wird durch das Modul „Gen“</p>

	<p>Argument: valence_map = valence_map von dem parent-Fenster</p> <p>Return: schedule = eine Liste von „trial“</p> <p>„trial“ = [rewarded, valve, valence map, lick fraction]</p> <p>pulse_parameters(trial)</p> <p>➔ Speichern des Parameters in einem Dictionary</p> <p>Argument: trial = [rewarded, valve, valence map, lick fraction]</p> <p>Return: params = Liste von Dictionary von Parametern. Die Liste entspricht die Liste von Odours in einem Trial</p>
File	~\Autonomouse 2 Schedule\ ScheduleModels\ScheduleWidgets.py

Name	Gen
Type	<i>Modul</i>
Beschreibung	Das Modul zur Randomisierung der Sequenz der Trials in einem Schedule.
Methods	<p>odor_sequence(odour_choice,n_trials)</p> <p>➔ Generieren der randomisierten Sequenz der Trials.</p> <p>Argument: odour_choice = die Wahl des Odours, die in dem Autonomouse2ScheduleWidget eingestellt wurde. n_trials = die Anzahl des Trials, die generiert werden sollen.</p> <p>reward_sequence(n_trials)</p> <p>➔ Generieren der randomisierten Sequenz. Die ersten 6 sind festgesetzt. Es wird vorgesehen, dass nicht mehr als drei gleiche Werte nacheinander sind.</p> <p>Argument: n_trials = integer Wert.</p> <p>Die Anzahl des Trials in einem Schedule</p> <p>Return: sequence = ein Array mit der Länge von n_trials</p> <p>generate_correlation_structure(n, rho)</p> <p>➔ In AutonoMouse 2 <u>nicht</u> implementiert.</p>
File	~\Autonomouse 2 Schedule\ Generation\Gen.py

Autonmouse 2 Controller

Das Main-Skript ist in Main.py in dem Ordner „Autonmouse 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*“.

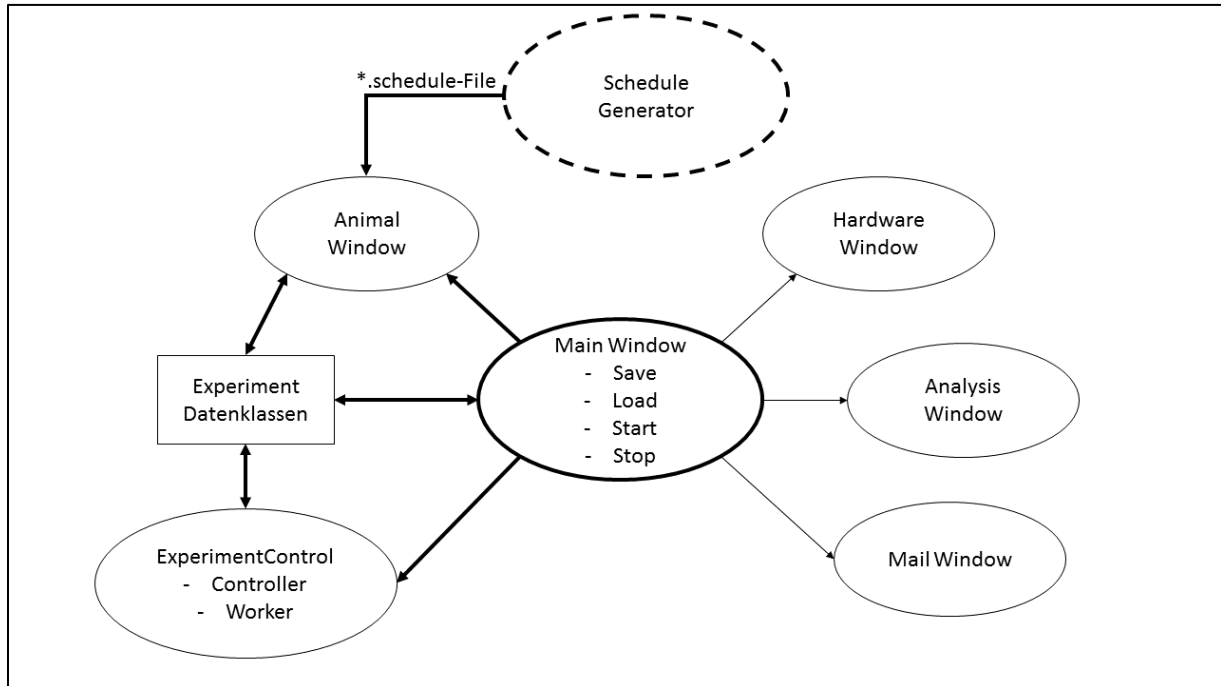


Abbildung 1: Überblick

Name	main
Type	<i>Module</i>
Beschreibung	Das Hauptmodul des Controls
Inhalt	MainApp
File	~\Autonmouse 2 Controller\main.py

Name	MainApp
Type	<i>Class</i>
Beschreibung	Die App des Controls. Main Window. Erbt <i>Qt.QMainWindow</i> und <i>mainDesign.Ui_MainWindow</i>
Methods	setup_experiment_bindings(experiment)

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

Name	ExperimentControl
Type	<i>Module</i>
Beschreibung	Das Modul mit Worker-Controller Klassen
Inhalt	ExperimentWorker, ExperimentController

File	~\Autonomous 2 Controller\Controllers\ExperimentControl.py
------	--

Name	ExperimentController
Type	<i>Class</i>
Beschreibung	Verwaltet das Thread. Start- und Stop-Button werden hier implementiert. Erbt <i>MainApp</i>
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	~\Autonomous 2 Controller\Controllers\ExperimentControl.py

Name	ExperimentWorker
Type	<i>Class</i>
Beschreibung	Beinhaltet den Job, der im Thread durchgeführt wird (siehe die Methode trial()). Erbt <i>ExperimentController</i>
Methods	trial() → Der Job, der im Thread durchgeführt wird. Die Signale <i>trial_end</i> und <i>finished</i> werden am Ende eines Trials ausgegeben bzw. wenn das Stop-Button gedrückt wird. check_status() → Überprüft die Zeit, ob ein <i>Deadman</i> E-Mail gesendet werden soll. check_licks()

	<p>→ 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.</p> <p>animal_present()</p> <p>→ Überprüft, ob die Lichtschranke gebrochen ist.</p> <p>get_present_animal()</p> <p>→ Liest der Id-Tag der Maus, die die Lichtschranke gebrochen ist.</p> <p>reward_animal(animal)</p> <p>→ Gibt das Wasser. Das Ventil der Wasserabgabe wird so lange geöffnet, wie das Reward eingestellt ist. Näheres zum Reward-Einstellung, siehe Animal-Window-Klasse.</p> <p>Argument: eine Klasse von dem Experiment-Modul. Näheres zu der animal-Klasse, siehe Experiment-Modul</p> <p>timeout()</p> <p>→ Softwaretiming für das Timeout. Der Rechner schläft für gegebener Zeit.</p> <p>save_data(argument)</p> <p>→ Wandelt die zu speichernde Daten in geeignete Format und speichert die Daten in csv-Datei.</p> <p>Argument: Daten, die gespeichert werden. Diese sind</p> <ul style="list-style-type: none"> - animal_id: der Id der Maus - timestamp: die Zeit, wann ein Trial beendet wird - rewarded: Boolean Variabel, ob das Trial belohnt werden 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 Abgabe des Odours geleckt hat.
File	~\Autonmouse 2 Controller\Controllers\ExperimentControl.py

Name	Experiment
Type	<i>Module</i>
Beschreibung	Das Modul mit den Daten-Klassen der Experiment
Inhalt	Experiment, Mouse, Schedule, Trial
File	~\Autonmouse 2 Controller\Controllers\ExperimentControl.py

Name	Experiment
Type	<i>Class</i>
Beschreibung	Beinhalte die Liste von dem Mouse-Objekt und die Liste von allem durchgeführten Trials.
Methods	add_mouse(id) ➔ addiert ein Mouse-Objekt mit dem Parameter <i>id</i> in der Liste Argument: <i>id</i> = Maus-RFID add_trial(argument) ➔ Append der Trial-Daten in einer Liste. Diese Liste wird in einer Tabelle im Main Window dargestellt. save() ➔ Pickle der Experiment-Daten
File	~\Autonmouse 2 Controller\Modells\Experiment.py

Name	Mouse
Type	<i>Class</i>
Beschreibung	Beinhalte die Liste von dem Schedule-Objekt, die die Maus durchführen soll und die Liste Lick-Daten einer Maus.

Methods	<p>update_licks(timestamp, rewarded, licks_before, licks_after, save_path, water_given, correct_timeout)</p> <p>➔ Addiert die Licks-Daten in der Liste und speichert diese.</p> <p>Argument: Licksdaten. Diese Daten werden bei dem Method <i>check_licks</i> in dem <i>ExperimentWorker</i>-Klasse wiederverwendet.</p> <p>add_schedule(argument)</p> <p>➔ Addiert ein Schedule-Objekt in der Schedule-Liste. Das Schedule-Objekt wird mit dem Parameter aus den gespeicherten Schedules konstruiert. Diese gespeicherten Schedules wurden von dem Schedule-Generator generiert.</p> <p>current_trial()</p> <p>➔ Gibt die aktuelle Trial-Parameter zurück</p> <p>current_trial_pulse()</p> <p>➔ Gibt die aktuelle Odoursignal-Parameter von einem Trial zurück.</p> <p>current_trial_idx()</p> <p>➔ Gibt der Index des aktuellen Trials zurück.</p> <p>advance_trial()</p> <p>➔ Ändert der Index zu dem Index des nächsten Trials.</p>
File	~\Autonomous 2 Controller\Modells\Experiment.py

Name	Schedule
Type	<i>Class</i>
Beschreibung	Beinhalte die Liste von dem Trial-Objekt, die die Maus durchgeführt hat und die Liste der Trial-Parameter aus der gespeicherten Schedule.
Methods	<p>add_trial_data(timestamp, wait_response, correct, timeout, licks, rewarded)</p> <p>➔ Addiert die Trial-Daten in der Liste der Trial-Objekt</p> <p>Argument: Trial-Daten. Diese Daten werden bei der Analyse und <i>deadman</i> E-Mail wiederverwendet.</p> <p>n_trials()</p> <p>➔ Gibt die Anzahl des Schedule-Trials zurück</p> <p>trial_left()</p>

	➔ Gibt die Anzahl der durchzuführende Trials in der gespeicherten Schedule zurück
File	~\Autonmouse 2 Controller\Modells\Experiment.py

Name	AppWindows
Type	<i>Module</i>
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 Performance der Mäuse
File	~\Autonmouse 2 Controller\Windows\AppWindows.py