SoftwareX 17 (2022) 100945

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

# PlatformCommander — An open source software for an easy integration of motion platforms in research laboratories

Matthias Ertl [a,*], Carlo Prelz [b], Daniel C. Fitze [a], Gerda Wyssen [a], Fred W. Mast [a]

[a] Department of Psychology, University of Bern, Fabrikstrasse 8, 3012 Bern, Switzerland
[b] Technologieplattform Forschung, Faculty of Human Sciences, University of Bern, Fabrikstrasse 8, 3012 Bern, Switzerland

ARTICLE INFO

*Article history:*
Received 20 September 2021
Received in revised form 6 December 2021
Accepted 13 December 2021

*Keywords:*
Motion platform
Hexapod
Psychophysics
Perception research

ABSTRACT

Motion platforms are used to study human behavior and brain function during passive motions. The software for controlling motion platforms is typically developed by the respective laboratories. Such customized, closed source solutions make replications of studies or multi-center collaborations difficult. Therefore, we developed PlatformCommander, an open-source software package for interfacing two often used motion platform models (6DOF2000E., MB-E-6DOF). The software includes a server implementing the interaction with the motion platform and additional devices, and a client application, connecting to the server and controlling experiments. PlatformCommander is ideal for the synchronization of data from different sources with high temporal precision.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

## Code metadata

| | |
|---|---|
| Current code version | *0.9* |
| Permanent link to code/repository used of this code version | https://github.com/ElsevierSoftwareX/SOFTX-D-21-00176 |
| Legal Code License | *GNU General Public License 3* |
| Code Versioning system used | *git* |
| Software Code Language used | *C, Ruby, Julia* |
| Compilation requirements, Operating environments & dependencies | https://gitlab.com/KWM-PSY/platform-commander |
| If available Link to developer documentation/manual | https://gitlab.com/KWM-PSY/emulator/-/blob/master/PlatformCommander_0.9/docs/protocol_manual.pdf |
| Support email for questions | matthias.ertl@unibe.ch |

## Software metadata

| | |
|---|---|
| Current software version | *0.9* |
| Permanent link to executables of this version | https://gitlab.com/KWM-PSY/platform-commander |
| Legal Software License | *GNU General Public License 3* |
| Computing platform/ Operating System | *Linux, optimized and tested for Devuan* |
| Installation requirements & dependencies | https://gitlab.com/KWM-PSY/platform-commander |
| If available Link to user manual - if formally published include a reference to the publication in the reference list | https://gitlab.com/KWM-PSY/emulator/-/blob/master/PlatformCommander_0.9/docs/protocol_manual.pdf *or* https://zenodo.org/record/5743201 |
| Support email for questions | *matthias.ertl@unibe.ch* |

## 1. Introduction

Perception of self-motion is essential for an efficient interaction with the environment. This becomes evident in situations where the vestibular organ provides erroneous information about head motions (e.g. during a neuritis vestibularis) and thereby disabling patients to master their daily lives. Hexapods/Motion

* Corresponding author.
*E-mail addresses:* matthias.ertl@unibe.ch (Matthias Ertl), carlo.prelz@unibe.ch (Carlo Prelz), daniel.fitze@unibe.ch (Daniel C. Fitze), gerda.wyssen@unibe.ch (Gerda Wyssen), fred.mast@unibe.ch (Fred W. Mast).

https://doi.org/10.1016/j.softx.2021.100945
2352-7110/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

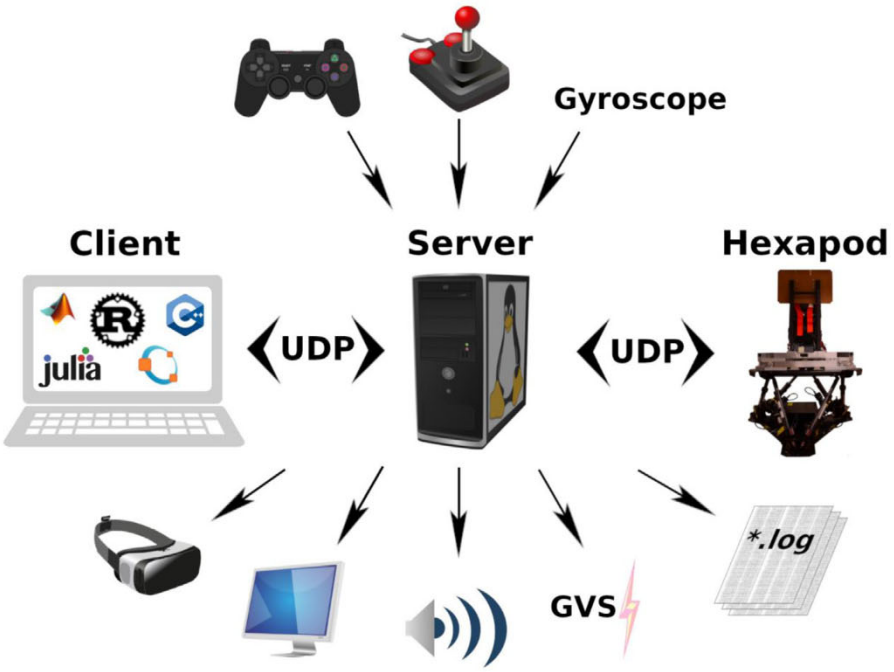


**Fig. 1.** Hardware setup. Schematic of the principle design of hardware interactions with the server. The server connects to the hexapod via UDP and also allows to control screens, audio output and a GVS-device. The server is also able to receive various input-signals, such as button presses. The client connects to the server via UDP. The experiment is controlled by a client application that sends requests to the server. The server also writes the entire communication into a log-file, that can be accessed for analysis or debugging purposes.

platforms are devices that allow to passively move participants with high temporal and spatial precision. This makes hexapods a valuable tool in clinical and basic vestibular and motion research. Compared to other stimulation techniques such as caloric vestibular stimulation or galvanic vestibular stimulation (GVS), hexapods allow for more naturalistic stimuli and the selective stimulation of all five sub-components of the peripheral vestibular organ [1]. Hexapods have been used by various research groups around the world to study vestibular thresholds [2–7], multi-sensory integration [8–12] or perceptual decision making [13–15]. Most laboratories [2,14,16–20] use hardware by the same manufacturer (MOOG) and often even the same model (6DOF2000E). However, no standard regarding a software package for interfacing the platform and controlling its motions has emerged and most labs have developed their own code for interfacing their hexapod and synchronizing its motion with other in- or output devices. These customized, closed-source solutions are in contrast to the current open-science research practice as they hinder inter-lab collaborations, comparisons, replications, or shared efforts for implementing new features. PlatformCommander is a new open-source software interface published under the GNU General Public License 3 (https://www.gnu.org/licenses/gpl-3.0.html). The software provides a tested, safe, simple and flexible interface to a MOOG-hexapod and allows to synchronize the movement with the presentation of visual (virtual reality) and auditory stimuli as well as GVS. PlatformCommander allows the implementation of different experimental paradigms ranging from uni-modal psychophysics to complex VR-based tasks. It is optimized for cross-system synchronization and high temporal accuracy and is therefore also suitable for collecting response times. The flexible architecture allows for recording a great number of behavioral data, including button presses, the position and acceleration of the head via acceleration sensors, or to synchronize the platform motions to EEG or fNIRS devices. Furthermore, behavioral parameters can influence the platform motion in real-time.

Due to the client/server architecture, PlatformCommander can be easily customized and a wide range of paradigms can be implemented. Other labs can use, copy or modify the code under the terms of the GPLv3.

## 2. Software framework

### 2.1. General setup

PlatformCommander requires at least three components to interact (Fig. 1). The first component is the hexapod (*MOOG 6DOF2000E*) which performs the motion required for the experiments. The hexapod is controlled by a server that implements the UDP communication with the hexapod according to the interface manual (MOOG 6 DOF 2000E; Models 170E122, 170E131; Nov 12, 1999). The UDP communication for a second platform model (MB-E-6DOF/12/1800KG; Dec 15, 2020) is completed and will be tested and released in the next version.

Furthermore, the server opens a UDP socket allowing a client-application to connect. An example client written in Julia [21] is part of PlatformCommander, and it is included in the repository. During the experiment the client sends request-messages to the server using a simple and flexible set of command IDs and parameters.

### 2.2. Hardware requirements

For first tests the server can be installed on hardware as thriftily as a Raspberry Pi 4 (https://www.raspberrypi.org/). We recommend using the hardware components mentioned at the gitlab repository, other hardware components may or may not work correctly in combination with PlatformCommander.

### 2.3. Server

The core component of the presented solution is a server written in C and Ruby (https://www.ruby-lang.org). The server

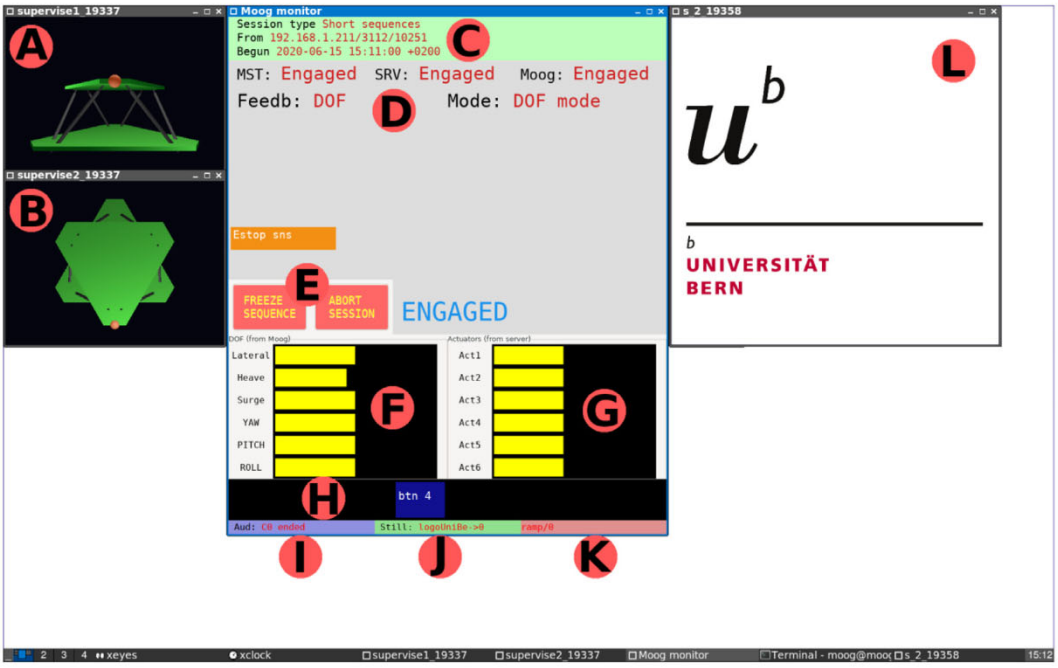


**Fig. 2.** Graphical User Interface. Screenshot of the GUI during an emulated session. **A** Front view of the simulated hexapod. **B:** Top view of the simulated hexapod. Window A and B are only present during emulated sessions. **C:** The green area contains information on the running session (session type, IP of the connected client, start date and time). **D:** This area provides information on the status of the server (SRV), the hexapod (Moog), and the expected and returned data-mode. **E:** Virtual buttons for pausing or aborting a running session. **F:** Visualization of the actuators as reported by the hexapod. Since the mode is set to DOF the bars represent lateral, heave, surge, yaw, pitch and roll. **G:** Visualization of each actuators length in meters as calculated by the simulation. **H:** This area provides feedback whenever a button press is registered via the i/o-card. **I:** The panel displays the name of the currently running audio-file. **J:** The panel displays the name of the image that is currently presented on the (VR-)screen. **K:** The panel displays the filename of the GVS-profile that is currently executed. **L:** Virtual screen that displays the image specified in (J).

(>26'000 lines of C code and >14'000 lines of Ruby code) handles the entire communication with the hexapod and connected input and output devices. The server-application is developed for GNU/Linux and optimized and tested for Devuan (https://devuan.org/). An installation manual for setting up the server is available on the GitLab page.

The server provides a graphical user interface (GUI) for the visualization of information on the current state of the server, hexapod and the connected in- and output devices (Fig. 2). The GUI displays information on the states of the server and hexapod, the running session and visualizes the current length of the six actuators. Depending on the session-type, additional information on the state of connected output devices (e.g. stimulators or screens) may be displayed. In emulated mode the motion of the platform is visualized in the GUI.

### 2.4. Clients

Client-applications to interface the server component of PlatformCommander can be written in any language allowing for network-communication. In order to connect to the server the client needs to open a UDP-socket connecting to port 1860 of the server. Once a connection is established the client requests the server to take certain actions.

### 2.5. client–server communication

The client and server communicate via simple ASCII string messages. All messages sent by the client to the server consist of at least a message ID. The message ID can be supplemented by a varying number of parameters which are separated by commas (see Table 1).

All messages sent by the server contain at least two fields, a time stamp providing the time since the session was initialized and a message ID. Depending on the exact message a varying number of parameters in comma-separated format are attached. Detailed descriptions, of all session types, commands and parameters can be found in the software manual [22]. Since a standard UDP interface is used to communicate with the server it is possible to write client applications in any programming language and it is therefore possible to integrate *PlatformCommander* with popular tools such as PsychoPy (https://www.psychopy.org/) or Psychtoolbox (http://psychtoolbox.org/).

### 2.6. Output devices

Additional to the hexapod communication, the server is able to control various optional output devices. For some outputs only certain products/models are supported, while for audio and screen output all standard hardware components should work. The server is equipped with an additional media-server that is listening on port 22110. The server allows to upload audio, image, video, 3D-model, and GVS-profile files that can be used during experiments.

#### 2.6.1. Visual stimuli subsystem (VSS)

For the presentation of visual stimuli the PlatformCommander is equipped with a powerful subsystem. The VSS provides easy access to an Open GL (https://www.opengl.org/) engine that allows to display visual stimuli on screens or VR-Headsets. Complex 3D scenes, including the positioning of generic 3D objects (sphere, cylinder, torus, cone, tetrahedron) or arbitrary objects can be defined using the widely used Polygon File Format. Free software

**Table 1**
Example of the server/client communication during one session in short-sequence mode. Further examples and a full description of all parameters can be found in the manual.

| Direction | Message | Description |
|---|---|---|
| c→s | 0, 2, 47476, D, −0.2 | Starts a session in short-sequence mode using port 47476. The platform will be controlled in DOF-mode (D) and the start position/height is −0.2 meters |
| s→c | 1016, 1, 200801.121500 | The server confirms a successful session opening (msgID: 1) at 1016 ms with the server time stamp YYMMDD. HHMMSS |
| s→c | 1020, 100, 0, 1, 0, 0, 0.0, …−0.0, −0.0, 0.0 | This is the first STATE message (msgID: 100) that the server sends to the client. It was sent 1020 ms after the session opening and indicates a master state of Idle (0), a server state of Idle (1), and a Moog status of Power up (0) |
| c→s | 3 | The client sends an ENGAGE (msgID: 3) request |
| s→c | 2000, 4 | The server confirms the engaging request was successfully sent 2000 ms after the session was initiated. At this point, the hexapod will start moving up to the height specified in the first message (−0.2 m) |
| s→c | 15225, 305 | At 15.2 s the server sent a SHORT SEQ AT HEIGHT (msgID: 305) notification, indicating that the operative height has been reached. |
| s→c | 15225, 6 | The server confirms that the engage process was completed successfully by sending an ENGAGE DONE message (msgID: 6) |
| s→c | 15235, 100, 2, 3, 3, 0.0, …0.0, −0.2, 0.0, −0.0, −0.0 | This state-message (msgID: 100) confirms that the master state is Engaged (2), the server state is Engaged (3), and the Moog state is Engaged (3). At this point the server is ready to perform an arbitrary motion as specified by the short sequence protocol. |
| c→s | 300, 0.05, −1.5.0.3.0 | The client announces that it wants to define a motion sequence (msgID: 300). The additional parameters specify the max. allowed actuator speed, and the location of the rotation axes (heave, surge, lateral) |
| c→s | 301, −0.0, −0.0, −0.15, …0.0, 0.0, 0.0, 5.0, S | The client defines a sinusoidal (S) movement in short sequence mode (msgID: 301) with a duration of 5 s to the final position: Roll = 0.0 rad, Pitch = 0.0 rad, Heave = −0.15 m, Surge = 0.0 m, Yaw = 0.0 rad, Lateral = 0.0 m. It is possible to define multiple motions that should be executed successively, without a break. |
| c→s | 302 | The client requests the execution of the motions. |
| s→c | 15319, 303, 0, 7 | The server simulates the requested motion sequence. If the motions are valid (e.g. do not violate the maximum acceleration of the platform or the maximal length of any of the actuators), it acknowledges the sequence (msgID: 303) and returns a sequence identifier (0) and the number of steps (7). Now the movement is performed. |
| s→c | 21048, 306, 0, 7 | Once the platform reached the final position the server confirms the successful execution of the motion (msgID: 306). |
| c→s | **7** | The client requests to park the platform. |
| s→c | 38581, 8 | The server acknowledges the park request. |
| s→c | 46581, 10 | The server reports that the parking process has been completed. |
| c→s | 11, 100 | The client requests to close the session and sets the temporal resolution of the log-file to 100 ms. |
| s→c | 51208, 12 | The server confirms the closing of the session. |

c = client; s = server.

like blender (https://www.blender.org/) enables creating arbitrarily complex models. Light sources can be placed at any position in the 3D scene and their intensity, the color and illumination distance can be controlled. Additionally, the VSS allows for positioning and updating the virtual camera. Images and videos, which have been uploaded through the media-server and are available on the server, can be played on virtual screens. The VSS supports flat and spherical screens of user defined dimensions which can be positioned and oriented freely in the 3D scene. It is possible to change the position or orientation of objects, including screens, during a session, or to change their visibility. Movies projected on screens can dynamically be started, paused, and resumed. Panel L of Fig. 2 shows an image presented on a flat screen positioned at an appropriate distance with a size filling the entire field of view. For a maximum of temporal control the log-file contains a timestamp for the change request as well as for the actual visualization on the screen. This makes PlatformCommander suitable for time critical applications like EEG/ERP recordings.

#### 2.6.2. Audio

Environmental sounds, particularly the noise generated by the platform, can be masked by presenting white noise when conducting experiments. For an optimal support of such efforts the PlatformCommander is able to play audio-files via connected speakers or headsets. The server is able to play any audio file uploaded through the media-server.

#### 2.6.3. Galvanic vestibular stimulation (GVS)

The PlatformCommander is able to control external GVS-devices via an analogue i/o-card. The option was developed for the interaction with a neuroConn DC-Stimulator PLUS (https:

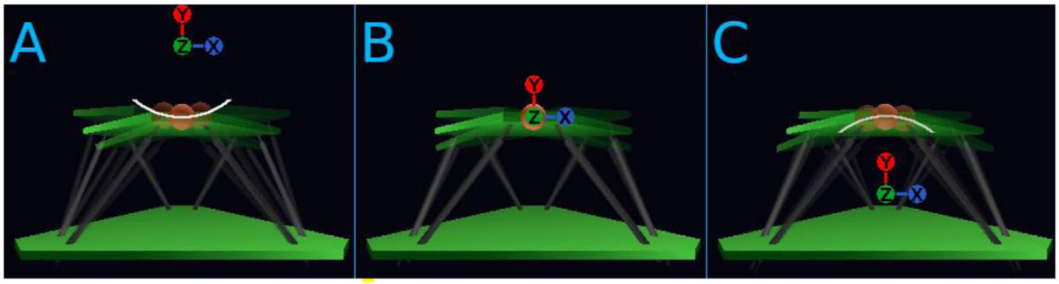


**Fig. 3.** Positioning of the rotation axes. All three panels display the front view (roll-plane) of the hexapod model used by PlatformCommander for simulations. Each of the panels shows a 10.0° rotation. However, the position of the rotation axes were placed at different points in the 3D space, as indicated by the coordinate axes (Z pointing towards the reader). In panel A the origin of the coordinate system was positioned 0.5 meters above the top of the platform, resulting in a concave trajectory for rotations about the *z*-axis. In panel B the origin was positioned at the height of the platform. In panel C the origin was 0.5 meters below the top of the platform. Comparing panel A and B one can notice the different curvatures (convex/concave) of the movement trajectory (A and C, white circle segments). The feature of positioning the coordinate origin at an arbitrary point allows to implement head-centered motions without the need of computations by the user.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 2**
Benchmark tests. Summary of the measured delay between the request of the motion and the transmission of the movement commands to the hexapod. The delay is caused by the calculation and validation of each movement step. The delays were calculated based on 20 samples.

| Motion type | Motion Duration [s] | Delay [µs] mean/SD |
|---|---|---|
| Sinusoidal | 1 | 95.65/4.60 |
| Linear | 1 | 91.3/0.47 |
| Sinusoidal | 10 | 871.45/9.05 |
| Linear | 10 | 816.55/88.63 |
| Sinusoidal | 30 | 2586.2/84.38 |
| Linear | 30 | 2517.7/15.94 |

SD = standard deviation.

//www.neurocaregroup.com/dc-stimulator-plus.html). PlatformCommander is able to read a voltage-trace provided by a profile-file uploaded via the media-server and to set the output-port of the i/o-card accordingly. The stimulation start and stop can be requested by the client application and can therefore be synchronized with a hexapod movement, visual and/or auditory stimuli.

### 2.7. Input devices

In experiments participants often are instructed to press buttons to communicate a decision or to influence the platform motion. The PlatformCommander is set up to enable the use of response buttons, mouses, joysticks, or game-controllers as input devices. Additionally, data streams (e.g. from accelerometers or gyroscopes) can provide input-signals to the PlatformCommander. All input data will be logged and most of it is accessible in real-time to the client. Input devices can be connected via USB or the i/o-card.

## 3. Session types

Four standard session types (Immediate, Short sequence, Long sequence, Joystick) are currently implemented. The types are optimized for certain use-cases and differ in their behavior. The session type is selected by a one digit number included in the login-message.

### 3.1. Immediate mode

The session type "Immediate" is thought to minimize the latency between a movement request by the client and the execution of the motion by the hexapod. Nevertheless, the server confirms that (I) the next position is reachable given the platform geometry and that (II) the distance between the current and requested position does not violate the maximum velocity criteria. During an immediate session the server expects the data to be sent in degree of freedom (DOF) format by the client (see [22] for details). If the request is valid the server forwards it to the hexapod without any further processing. The immediate mode enables maximum responsiveness to user-input and allows other programs such as game engines or flight simulators to interact with the hexapod with minimal latency.

### 3.2. Short sequence mode

In this mode the server expects the client to send one or more motion sequences. Before a motion profile is defined by the user it is possible to define the location of the rotation axes (Fig. 3).

Each motion profile submitted to the server consists of a message containing eight parameters specifying the motion profile (see Table 1 for an example). The first six parameters describe the final position of the hexapod. The seventh parameter specifies the duration of the motion in seconds. The eighth parameter specifies the profile of the motion. The user can select between linear and sinusoidal interpolation. If the linear option is selected the platform moves with a constant velocity between the specified start and end point with short, steep accelerations at the start and end of the motion. This type of profile has been used by EEG-studies on hexapods [15,19]. Alternatively, a sinusoidal interpolation can be requested. PlatformCommander will then calculate the necessary steps and amplitudes to reach the end-position, while following a sinusoidal profile. Sinusoidal profiles are the most frequently used stimuli in hexapod experiments [2,4,6,13,16,23–25], but PlatformCommander enables specifying arbitrary motion profiles. Once the requested motion sequence is done the hexapod will remain in its final position and the server is waiting for new instructions. One crucial aspect to keep in mind is that in this mode the profile is calculated starting from the current position. Therefore, the same motion command will result in very different movements (directions, accelerations) when performed from different starting positions. The user has to make sure that the hexapod is at the correct starting position for the next movement. The validity checks before a movement is executed depends on the length of the motion but according to benchmark tests (Table 2) is at the order of milliseconds.

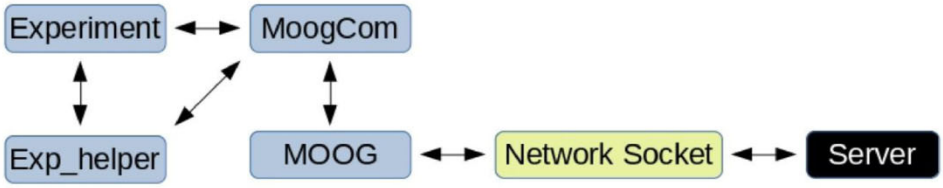


**Fig. 4.** Interaction of the client modules. A typical experiment script utilizes the Julia modules MoogCom and Exp_helper, which provide useful functions, structures, and default values. The module MOOG provides various constants, like the message IDs which are used by MoogCom. The module MOOG provides an interface to a C-library in order to interact with the server via a network socket. The network socket manages the UDP communication with the server.

### 3.3. Long sequence mode

As in short sequence mode the user has the option to specify the location of the rotation axes at the beginning of motion definitions. In long sequence mode an arbitrary number of motion profiles is submitted to the server. The specified motions constitute a library that can be used during the started session by requesting the execution of a certain motion-profile. Each motion-profile is defined by a starting position, an end position, a profile, and a motion duration. Once a motion is defined and accepted by the server it cannot be altered anymore. After finishing the definitions of the motion profiles they can be invoked in an arbitrary order and as often as wanted. All submitted motions are checked for validity before the first motion can be performed.

Contrary to the short sequence mode any motion-profile in long sequence mode has a defined, fixed starting point. The starting point is defined by the first position array of the motion-profile. This ensures that the specified motion does only need to be tested for validity once and can at run-time (during the experiment) be executed without any computational overhead. Though, the invocation of a sequence is a two-step process. At first the hexapod calculates an adequate movement to the starting position and then performs the corresponding movement towards it. Once the start position is reached the movement sequence will be executed as soon as it is requested by the client.

### 3.4. Joystick mode

The joystick session type requires a connected joystick and will configure the server so that it aims to match the joysticks position with the platform.

### 3.5. Data logging

Using the server for controlling all relevant stimuli (e.g. motion, images, audio) of an experiment offers the opportunity to get consistent time stamps for each event. Using only one computer with one clock overcomes synchronization issues and allows for an easy reconstruction of the event sequence during the analysis. During a session PlatformCommander creates log-files containing information on the state of the session at any given time-point. Once a session is closed the log-file can be transformed into a standardized .csv-file with an equidistant sampling rate. It might be convenient to generate customized log-files, streamlined for the planned analysis on the client side, rather than using the exhaustive files generated by the server.

## 4. Client-applications

The client/server design allows the implementation of client-applications in almost any programming language. This enables researchers to continue using their preferred programming language. As part of PlatformCommander we also publish the client library used in our lab. The client is written in the Julia programming language (https://julialang.org/) utilizing C-functions for fast and reliable network communication. The implementation of the Julia client consists of multiple interacting modules which relations are visualized in Fig. 4. The application provides default parameters and functions ensuring the correctness of input before sending messages to the server.

## 5. Conclusion

To our knowledge PlatformCommander is the first open source software for controlling motion platforms with a focus on research applications. PlatformCommander provides fine-grained movement control and interaction with a great number of recent input and output devices.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

We thank Andreas Szukics and Noel Strahm for their code contributions and for testing the software.

## References

[1] Ertl M, Boegle R. Investigating the vestibular system using modern imaging techniques—A review on the available stimulation and imaging methods. J Neurosci Methods 2019;326(July). http://dx.doi.org/10.1016/j.jneumeth.2019.108363.

[2] Grabherr L, Nicoucar K, Mast FW, Merfeld DM. Vestibular thresholds for yaw rotation about an earth-vertical axis as a function of frequency. Exp Brain Res 2008;186(4):677–81. http://dx.doi.org/10.1007/s00221-008-1350-8.

[3] Lim K, Merfeld DM. Signal detection theory and vestibular perception: II. Fitting perceptual thresholds as a function of frequency. Exp Brain Res 2012;222(3):303–20. http://dx.doi.org/10.1007/s00221-012-3217-2.

[4] Bremova T, Caushaj A, Ertl M, Strobl R, Böttcher N, Strupp M, et al. Comparison of linear motion perception thresholds in vestibular migraine and Menière's disease. Eur Arch Otrhinolaryngol 2016;273(10):2931–9. http://dx.doi.org/10.1007/s00405-015-3835-y.

[5] Karmali F, Chaudhuri SE, Yi Y, Merfeld DM. Determining thresholds using adaptive procedures and psychometric fits: Evaluating efficiency using theory, simulations, and human experiments. Exp Brain Res 2016;234(3):773–89. http://dx.doi.org/10.1007/s00221-015-4501-8.

[6] Klaus MP, Schöne CG, Hartmann M, Merfeld DM, Schubert MC, Mast FW. Roll tilt self-motion direction discrimination training: First evidence for perceptual learning. Atten Percept Psychophysics 2020. http://dx.doi.org/10.3758/s13414-019-01967-2.

[7] Kobel MJ, Wagner AR, Merfeld DM, Mattingly JK. Vestibular thresholds: A review of advances and challenges in clinical applications. Front Neurol 2021;12:643634. http://dx.doi.org/10.3389/fneur.2021.643634.

[8] Garzorz IT, MacNeilage PR. Visual-vestibular conflict detection depends on fixation. Curr Biol 2017;27(18):2856–61.e4. http://dx.doi.org/10.1016/j.cub.2017.08.011.

[9] Keshavarz B, Ramkhalawansingh R, Haycock B, Shahab S, Campos JL. Comparing simulator sickness in younger and older adults during simulated driving under different multisensory conditions. Transp Res Traffic Psychol Behav 2018;54:47–62. http://dx.doi.org/10.1016/j.trf.2018.01.007.

[10] Shao M, DeAngelis GC, Angelaki DE, Chen A. Clustering of heading selectivity and perception-related activity in the ventral intraparietal area. J Neurophysiol 2018;119(3):1113–26. http://dx.doi.org/10.1152/jn.00556.2017.
[11] Chen X, DeAngelis GC, Angelaki DE. Flexible egocentric and allocentric representations of heading signals in parietal cortex. Proc Natl Acad Sci USA 2018;115(14):E3305–12. http://dx.doi.org/10.1073/pnas.1715625115.
[12] Yakubovich S, Israeli-Korn S, Halperin O, Yahalom G, Hassin-Baer S, Zaidel A. Visual self-motion cues are impaired yet overweighted during visual–vestibular integration in Parkinson's disease. Brain Commun 2(1):fcaa035. http://dx.doi.org/10.1093/braincomms/fcaa035.
[13] Ellis AW, Mast FW. Toward a dynamic probabilistic model for vestibular cognition. Front Psychol 2017;8:1–7. http://dx.doi.org/10.3389/fpsyg.2017.00138.
[14] Hou H, Zheng Q, Zhao Y, Pouget A, Gu Y. Neural correlates of optimal multisensory decision making under time-varying reliabilities with an invariant linear probabilistic population code. Neuron 2019;104(5):1010–21.e10. http://dx.doi.org/10.1016/j.neuron.2019.08.038.
[15] Ertl M, Klaus MP, Mast FW, Brandt T, Di-eterich M. Spectral fingerprints of correct vestibular discrimination of the intensity of body accelerations. NeuroImage 2020. http://dx.doi.org/10.1016/j.neuroimage.2020.117015.
[16] MacNeilage PR, Banks MS, DeAngelis GC, Angelaki DE. Vestibular heading discrimination and sensitivity to linear acceleration in head and world coordinates. J Neurosci: The Off J the Soc Neurosci 2010;30(27):9084–94. http://dx.doi.org/10.1523/JNEUROSCI.1304-10.2010.
[17] Crane BT. Roll aftereffects: Influence of tilt and inter-stimulus interval. Exp Brain Res 2012;223(1):89–98. http://dx.doi.org/10.1007/s00221-012-3243-0.
[18] Kolev OI, Nicoucar K. Flash induced afterimage versus single spot visual object influence on visual-vestibular interaction in detection threshold for self-motion perception. Neurosci Lett 2014;564:43–7. http://dx.doi.org/10.1016/j.neulet.2014.02.002.
[19] Ertl M, Moser M, Boegle R, Conrad J, Eulenburg Pzu, Dieterich M. The cortical spatiotemporal correlate of otolith stimulation: Vestibular evoked potentials by body translations. NeuroImage 2017;155(2016):50–9. http://dx.doi.org/10.1016/j.neuroimage.2017.02.044.
[20] Sasaki R, Angelaki DE, DeAngelis GC. Processing of object motion and self-motion in the lateral sub-division of the medial superior temporal area in macaques. J Neurophysiol 2019;121(4):1207–21. http://dx.doi.org/10.1152/jn.00497.2018.
[21] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. SIAM Rev 2017;59(1):65–98. http://dx.doi.org/10.1137/141000671.
[22] Ertl M, Prelz C, Fitze D, Wyssen G, Mast F. Manual PlatformCommander Version 0.9. 2021, https://zenodo.org/record/5743201.
[23] Cuturi LF, MacNeilage PR. Systematic biases in human heading estimation. PLoS One 2013;8(2):e56862. http://dx.doi.org/10.1371/journal.pone.0056862.
[24] Keywan A, Jahn K, Wuehr M. Noisy galvanic vestibular stimulation primarily affects otolith-mediated motion perception. Neuroscience 2019;399:161–6. http://dx.doi.org/10.1016/j.neuroscience.2018.12.031.
[25] Ertl M, Zu Eulenburg P, Woller M, Dieterich M. The role of delta and theta oscillations during ego-motion in healthy adult volunteers. Exp Brain Res 2021. http://dx.doi.org/10.1007/s00221-020-06030-3.