Synquesticon user guide

# Purpose

The purpose of the Synquesticon software system is to simplify data collections for human factors and usability studies. It supports the researcher by providing a graphical editor for scripting a study flow, by managing multi-user and multi-location studies, by providing live observation functionality to the researcher, by synchronizing response data with eye tracking data, and by providing database functionality for storing and retrieving recorded data. Synquesticon can be deployed as a local installation in a lab, or on a public web server. When deployed on a web server, it can be used for remote studies, where participants conduct the tasks at their home site (e.g. training simulator).

# Installation and Architecture

The Synquesticon source code can be downloaded from github.com/synopticon/synquesticon. The GitHub repository includes the latest install instruction. Synquesticon is written in JavaScript and React. Before installing the Synquesticon repository, the user must install Node.js (nodejs.org) and MongoDB (mongodb.com). All other dependencies are installed through the repository.

Synquesticon exchanges data through MQTT, an Internet of Things protocol. An MQTT broker is automatically installed and needs to be correctly configured for Synquesticon to work (see section 9.2).

Instructions for installing Synquesticon on a web server can be found in Appendix A. An updated version of these instructions is maintained on the GitHub repository.

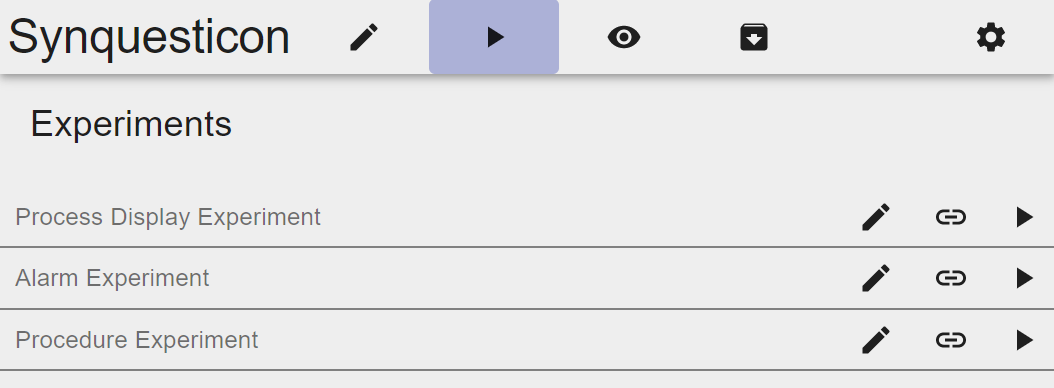
A local installation of Synquesticon is started by double-clicking the file “launcher.bat” in the Synquesticon home directory. This will launch a development server, and after a few moments, the Synquesticon home screen will open in a browser. Once started, Synquesticon can be opened in other tabs by entering the URL “localhost:3000”. A local installation of Synquesticon has to be restarted every time the computer is rebooted.

**Synquesticon Modules**

Synquesticon has four main modules, which can be accessed from the main screen (Figure 1).

* **Editor** (pencil icon). Used to author tasks and sets.
* **Player** (play icon). Used to launch a session, or to generate a web link to send to remote participants.
* **Observer** (eye icon). Used to monitor task performance in real-time.
* **Export** (download icon). Used to export data sets.

The last icon in the task bar is a settings menu, used to control device setting.



*Figure 1. Synquesticon main page with task bar at top (Edit, Play, Observe, Export, Settings) and a list of experiments.*

# Key Concepts

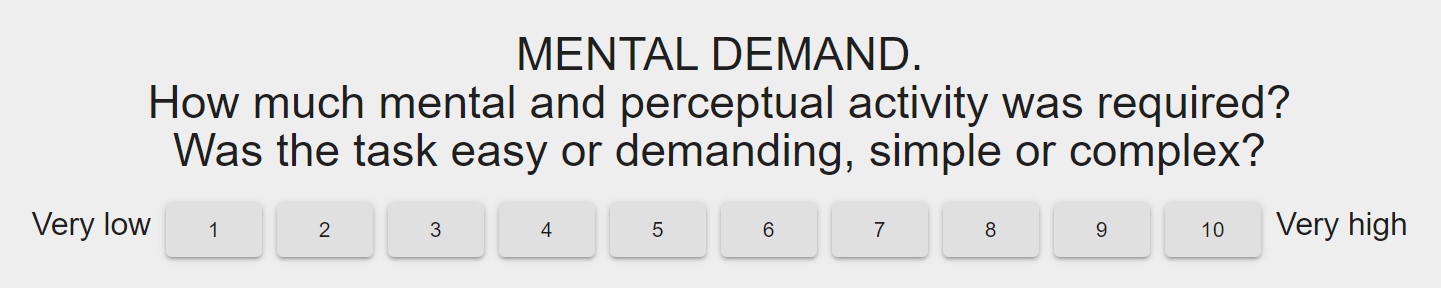
To take advantage of the full functionality of Synquesticon, a few key concepts need the be understood. These are Component, Task, Set, Event and Screen.

**Component**

Components are the most basic units of stimulus presentation in Synquesticon. There are currently five Component types:

* **Instruction:** Displays text to the user.
* **Image:** Displays an image to the user. Includes capability to define areas of interest in the image and record clicks on the image.
* **Button:** Can display a question text and a set of response buttons. Buttons can be defined as single or multiple choice, or as self-resetting.
* **Text:** Displays a question text and a text field where the user can enter a response.
* **Numbers:** Displays a question text and a keypad where the user can enter a numerical response.

These component types will be described in more detail in section 5.1.1 (editor) and 6.2 (player). Here is an example of a Button component that asks the user for a workload rating:



*Figure 2. Example of a Button Component.*

**Task**

A Task can contain multiple components. For instance, it may contain an Image with a set of Buttons below it; or several sets of Buttons. A Task is equivalent to a single page of content (although in multi-screen mode, content for a single Task can be distributed across multiple screens). The user proceeds to the next Task (i.e. page) by clicking the “Next” button at the bottom of each page.

**Set**

Tasks can be organized into Sets. You can think of a Set as a folder. Tasks in a Set are presented to the user in the order they appear in the set, or if desired in random order. Sets can be used to organize content. For instance, you may want to create a Set containing demographic questions, and then another for questions about process displays. You may want to present the demographic questions in order, but randomize the questions about the process displays.

Another advantage of organizing Tasks in Sets is in the ease of re-use. For instance, a Set of demographic questions can be re-used in another study simply by dragging it into that study.

**Event**

As a research tool, Synquesticon collects a lot of data while the participant works through the tasks. These include response times for answering to a Task. Each user response is an event that is logged. We distinguish between two types of events: Component events and Element events.

A Component event holds the overall response data for the Component. For instance, for a Button component, it holds the final response of the user and the time to completion.

For some studies it is necessary to go to a finer level of analysis. Element events are actions on a single element of a component, such as clicking a single button or clicking on an image. Each of these elementary events is recorded separately and are immediately shown in the observer module.

In summary, each individual action of the user is logged as an element event, and the overall Component-level data is logged as a Component event.

**Screen**

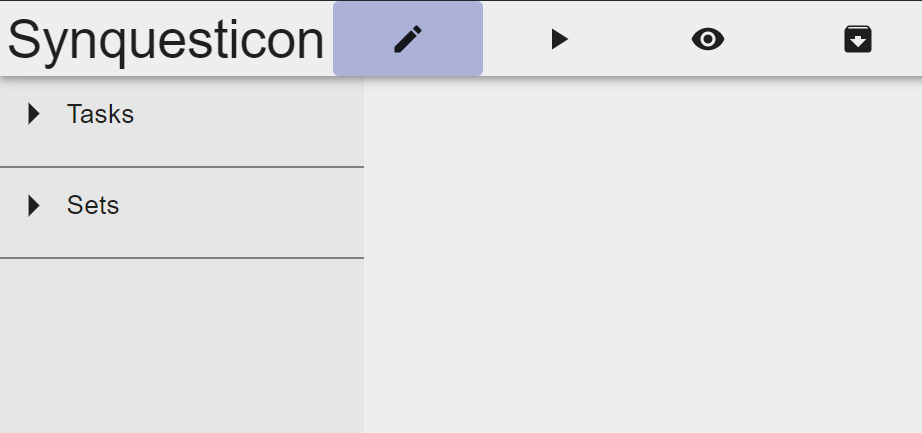
Synquesticon is a multi-screen system. That means that Components within a Task can be assigned to different screens. For instance, a Task could consist of asking “Which of these two process displays is easier to read?”, with the response buttons “Left” and “Right”. The question can be presented on a mobile device (e.g. iPad mini), while two separate monitors in front of the user will each show one process display (using the Image Component).



*Figure 3. Conceptual representation of a multi-screen study.*

# Editor

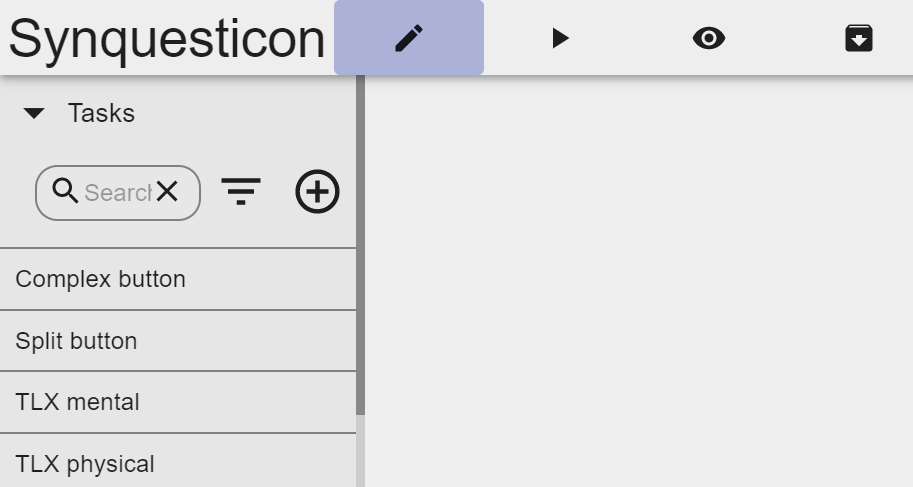
The editor has two main sections, one for Tasks and one for Sets. These can be accessed through the bar on the left side of the screen.



*Figure 4. Editor module, showing sections for Tasks and Sets on the left.*

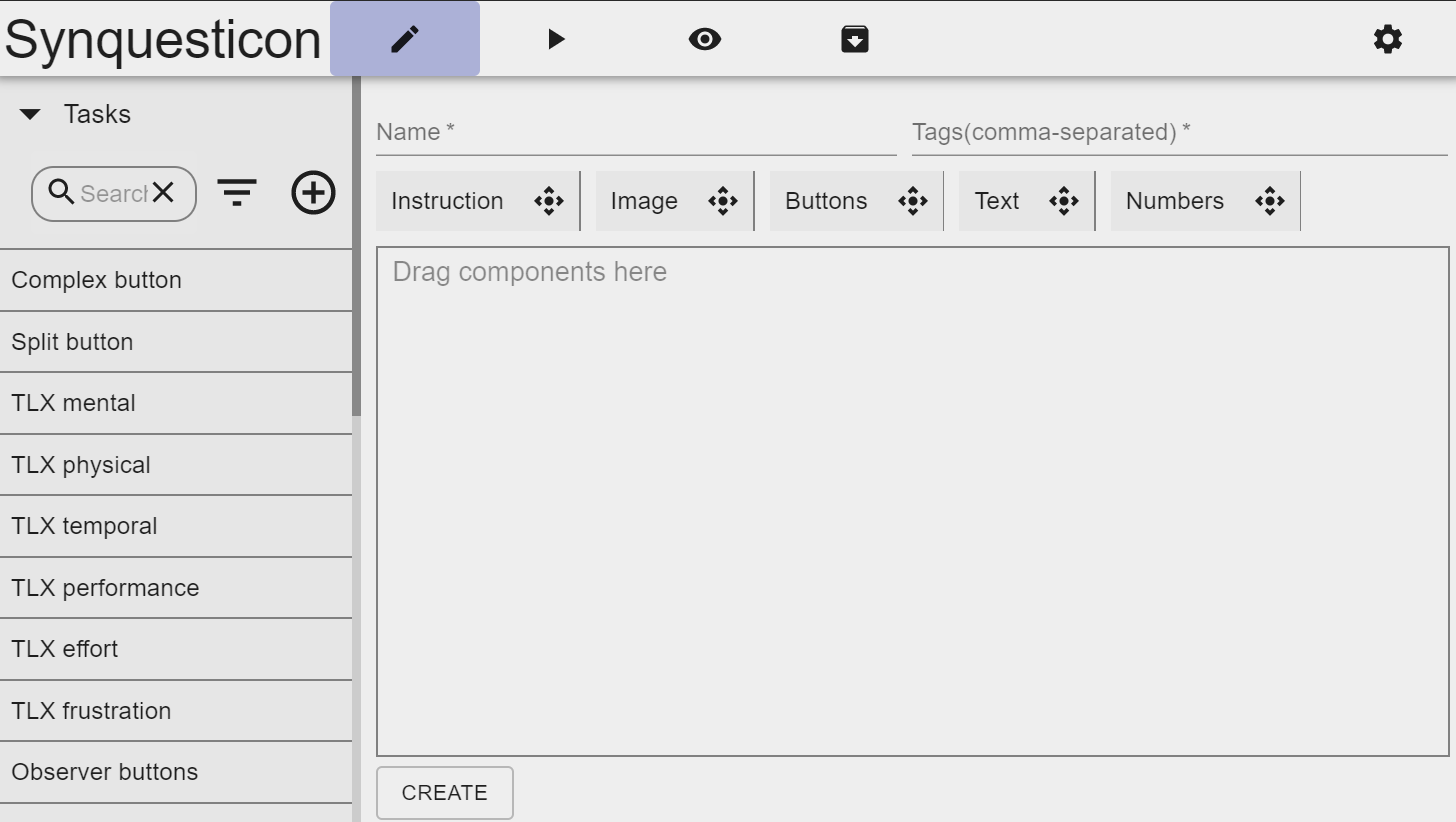
## Task Editor

Upon clicking on the Task section of the editor, a list of all previously created Tasks is presented. At the top if the list is a (+) icon for creating new Tasks. Next to it is a filter and search function, used to help manage large amounts of Tasks. Existing Tasks can be edited by clicking on them.



*Figure 5. Task editor with search, filter and add function at top.*

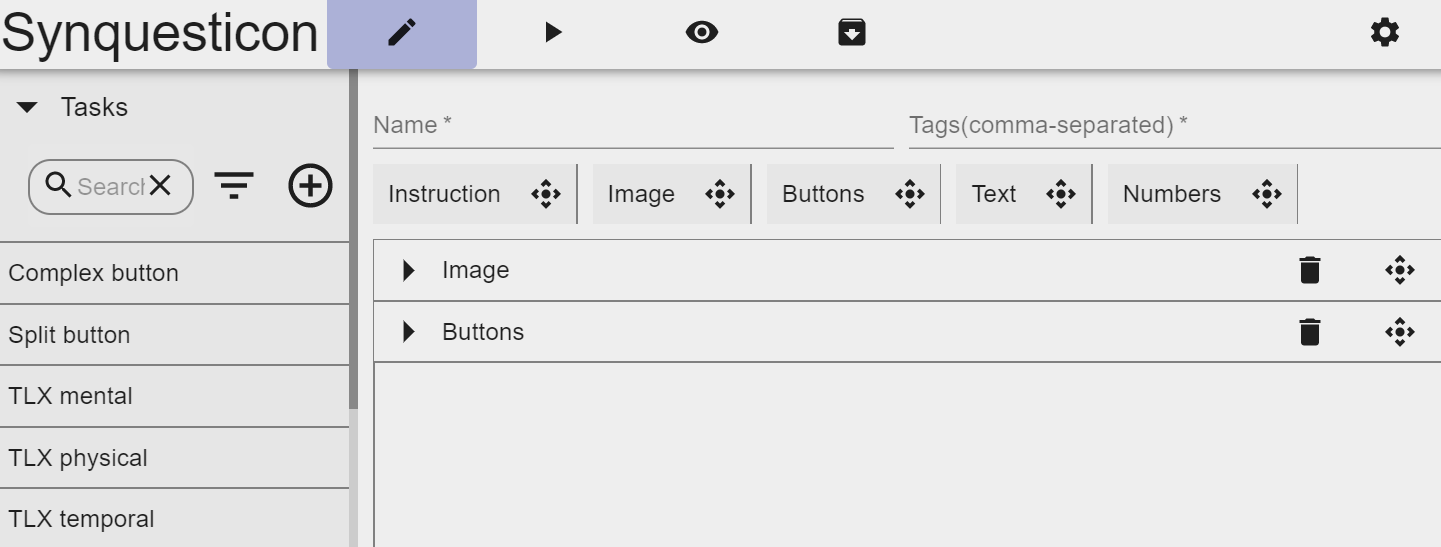
After clicking the (+) icon, an empty Task form is displayed.



*Figure 6. Component editor.*

The user can provide a name for the task as well as a comma-separated list of tags. The tags are saved in the log file, and can be used to aid in data analysis, for instance by grouping tasks by tags.

Underneath the Name / Tag fields is the Component list. A Component can be added to a Task by dragging it into the Component list.



*Figure 7. Task editor with two Components in the Component list.*

The presentation order of Components can be changed by dragging them up or down the list using the drag handle. Next to the drag handle is a “Delete” icon. Once the Task and Components have been edited, the user must click “Create” at the bottom of the screen. When editing a previously created component, instead of “Create”, there will be buttons for “Save” and “Delete”.

## Component Editor

This section explains the configuration options for each Component. The options “Hide Next” and “Screen IDs”, which are available in all Component types, will be explained in section 5.1.1.6 and 5.1.2.

### Instruction Component

The Instruction Component has an input field for the text that will be displayed to the participant.



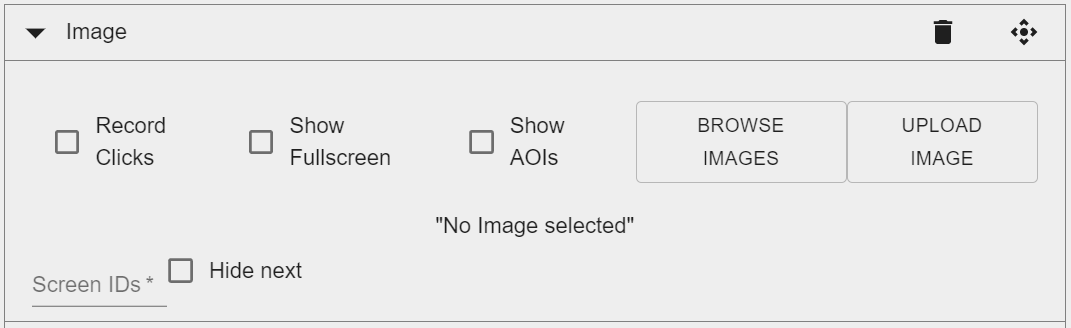
*Figure 8. Instruction editor.*

### Image Component

Editing an Image Component starts by either uploading a new image or selecting a previously uploaded one (via “Browse Images”). Once selected, the image is shown in the editor.

At the top of the Image editor, there are three options.

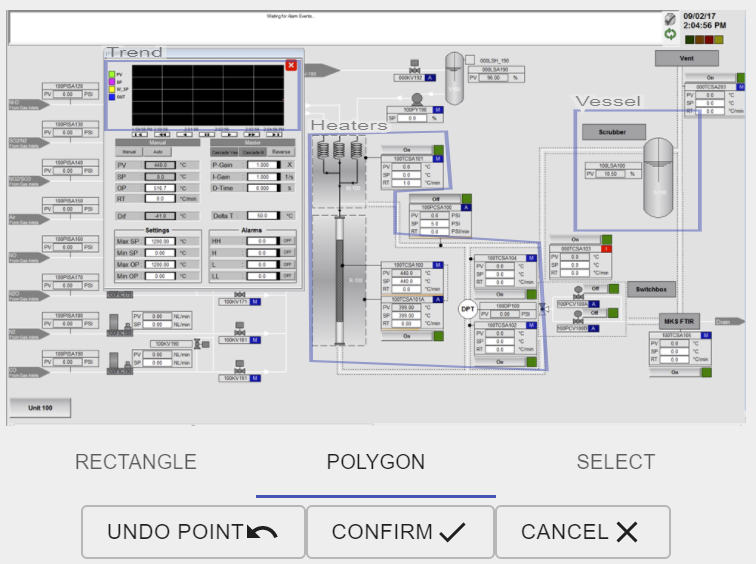
* **Record Clicks**. If checked, the player will record clicks on the image. When the participant clicks on the image, the click location will be shown as a semi-transparent red dot. This function can be used to obtain feedback from the user, e.g. when asking “Click on all areas of the display that should be improved” or “Click on all valves that are open”. Each click is logged as an event with screen location, time, and any Areas of Interest at this location.
* **Show Fullscreen.** This option stretches the image to take up the full size of the screen.
* **Show AOIs.** Areas of Interest (AOIs) can be defined on the image (see below). Normally these are hidden from the participant and only used to register clicks or visual fixations. However, in some cases it may be useful to show the AOIs to the participant, e.g. to indicate certain section in the image. When this option is checked, the AOIs are shown in the player.



*Figure 9. Image editor.*

**Area of Interest (AOI) Editor**

With the AOI editor, the user can draw and label Areas of Interest on top of the image. By default, AOIs are rectangles, but this can be changed to Polygon using the buttons below the image. Once an AOI is drawn, the user is prompted to enter a label. The “Select” option is used to rename or delete AOIs.



*Figure 10. Area of Interest editor.*

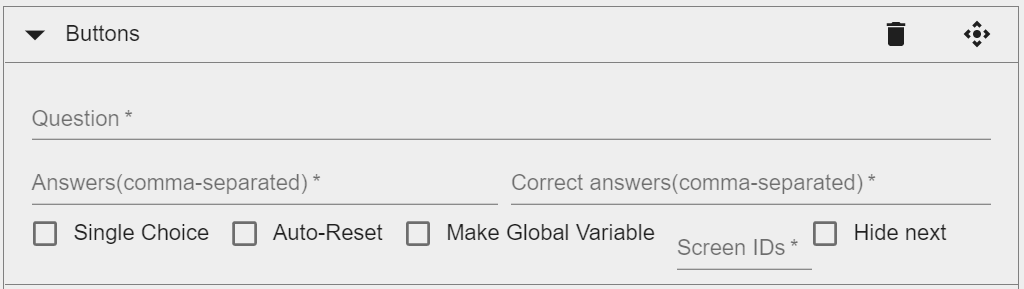
### Button Component

The Button editor consists of three main fields:

* **Question.** The question text.
* **Answer.** A comma-separated list of button labels. These will be displayed as buttons in the player.
* **Correct answers.** A comma-separated list defining the correct answer to the question. If a correct answer is defined, Synquesticon will evaluate the participant’s answer against the correct answer. If they match, the question will be logged as answered correctly.

Below these field are several options.

* **Single choice.** The user can select only one button, for instance for a Yes / No button pair.
* **Auto-Reset.** When checked, the button will be automatically unclicked after half a second. One use case for this functionality is an observation study, where buttons can be defined for each observation category.
* **Make Global Variable.** When checked, every line of data in the log file will be tagged with the response. This can be used to label experimental conditions, or to add individual / group characteristics to the data file (e.g. participant ID, crew label).

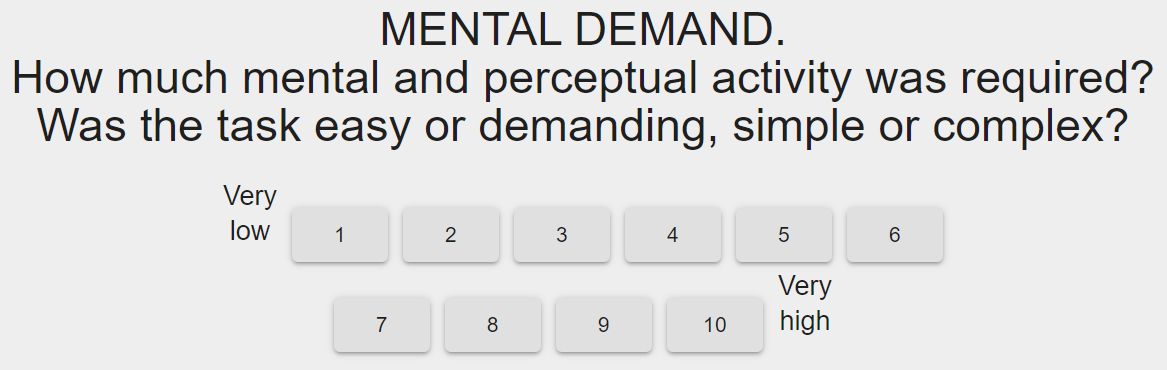


*Figure 11. Button editor*

**Formatting**

By default, the comma-separated list of labels in the Answer section will be rendered as buttons in the player. However, sometimes more control over the layout is desirable. The following control characters can be used to control the layout:

* **// (double forward slash).** When text in the Answer section is preceded by //, the following text is rendered in the player as normal text and not as a button. This can be useful for showing anchors to the left and right of a numerical scale.
* **\n (new line character).** The new line character can be used to insert a line break in a list of buttons. This can be useful for separating groups of buttons within a single component. The character can also be inserted into a text label to split the label.



*Figure 12. A Button Component with anchors. The button list is split after the "6" button. The anchors are also split. This button set is defined in the editor as “//Very\nlow,1,2,3,4,5,6,\n,7,8,9,10,//Very\nhigh”.*

**Command buttons**

You can send commands via buttons. These commands are passed up from ButtonElement.js to /Play/index.js, and handled there in the function commandCallback.

In the editor, you define a command by adding the characters "??" to the button text. For example, to start motion recording you define a button as:

Start recording??recordMotion

Multiple commands can be issued by the same button. These are separated by the characters "&&".

Start recording??recordMotion&&tag=waiting

The following commands are currently available:

**recordMotion** - Adds the event listener "devicemotion" when the button is clicked and removes it when the button is unclicked. The listener is also removed when the page /Play/index.js is unloaded.

**tag** - Adds a tag to the motion object. This can be used for ground truth. Currently you can supply only a single tag. There are two options for using this: If "tag" is used without a parameter, it takes the name of the button.

Walking??tag, BikeRiding??tag, StairClimbing??tag

You can also supply the tag value as a parameter:

Walking??tag=walk, BikeRiding??tag=ride,

StairClimbing??tag=climb

**mqtt** - Sends a mqtt message. The @ character is used to separate topic and message.

Send MQTT Message??message@/messageTopic/subtopic

The ";;" characters can be used to define multiple messages.

Send MQTT Message??Hello world@worldTopic;;

Hello IFE@IFEtopic

### Text Component

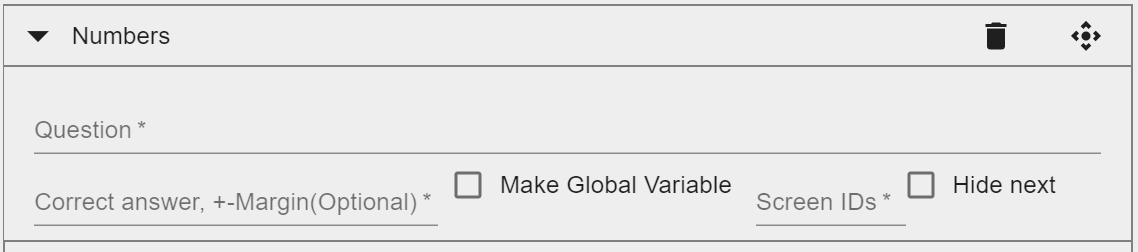
The text component displays a question and a text field to the participant. A question text and correct answer can be defined in the editor. If “Make Global Variable” is checked, every line of data in the log file will be labeled with the response.



*Figure 13. Text editor.*

### Numbers Component

The Numbers component displays a question text and a numeric keypad to the participant. In addition to providing a correct answer, a response margin can be provided. An example for when this is useful are tasks where the participant reads an analogue gauge. It is unlikely that the participant will enter the exact correct number. For instance, by entering “64.7, 2”, answers between 62.7 and 66.7 are scored as correct.



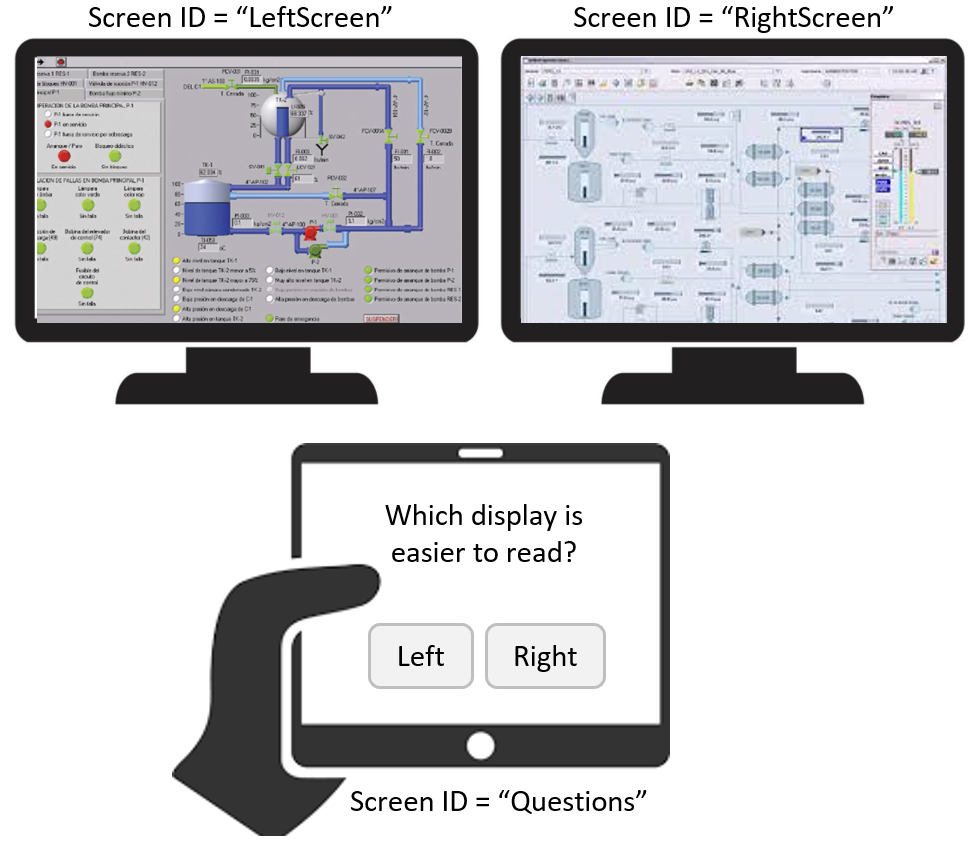
*Figure 14. Numbers editor.*

## “Hide Next” Option

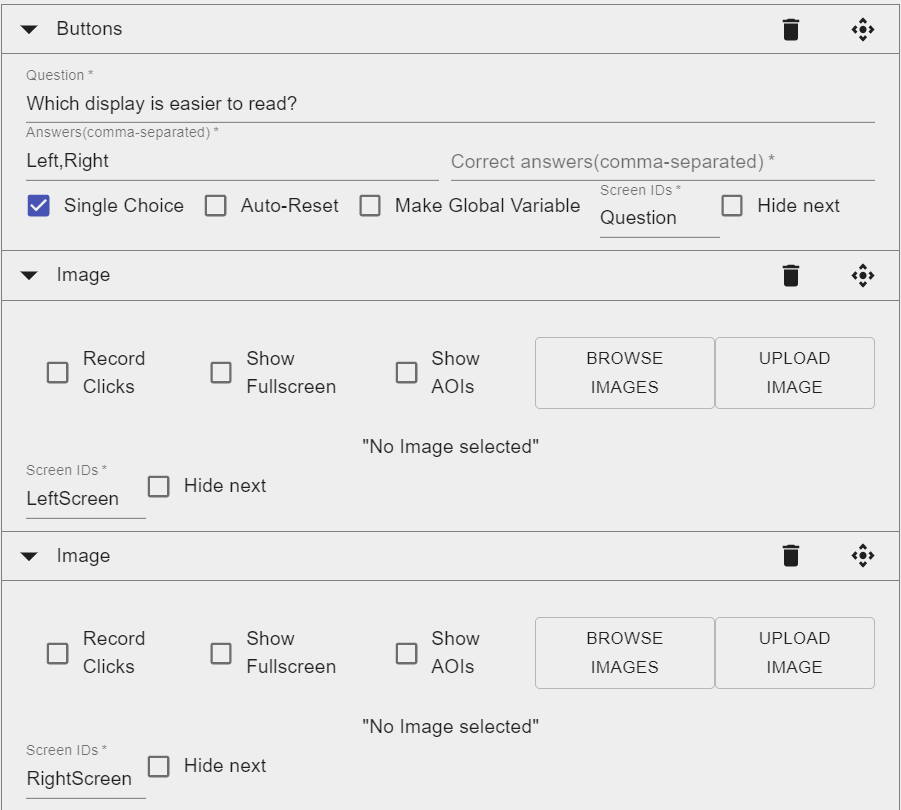
In the editor, there is a “Hide Next” option for every component. By default, when an experiment is started in the player there is a “Next” button in the lower right corner of every page. There are some cases where you may want to hide this button. This applies mainly to multi-screen studies where you show questions on one screen and images on other screens. In that case, the Next button can be hidden on the image screens and only shown on the question screen. Keep in mind that a Next button is needed on at least one screen, otherwise the participant cannot proceed to the next task.

## Multi-Screen Functionality

As previously mentioned, Synquesticon supports multi-screen studies. To assign a component to a specific screen, the editor provides a “Screen ID” field for every component. Here the user can enter a label. When running an experiment, a screen ID can be set via the Synquesticon settings menu. In that case, the component is only shown if the screen IDs match. When the user clicks “Next” on any screen, all screens will be updated to the next Task.



*Figure 15. A multi-screen study with Screen IDs.*



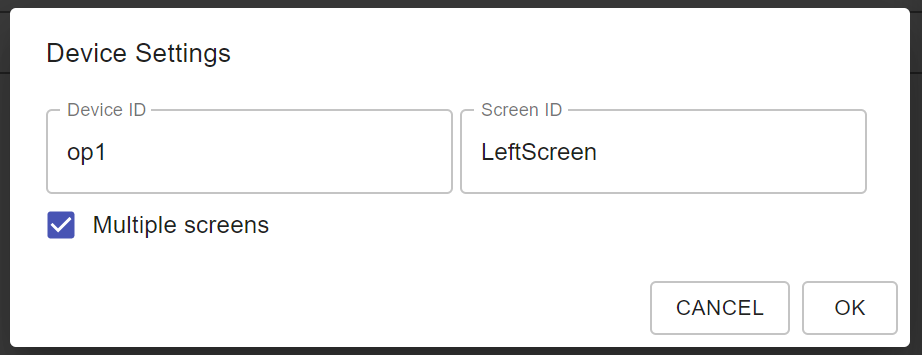
*Figure 16. Definition of screen IDs in the editor.*

When more than one Component with the same Screen ID is defined, these components will be shown on the same screen in the order they appear in the editor.

By entering a comma-separated list of labels in the “Screen ID” field, the component can be shown on several screens. For instance, to display the text “Wait for instructions” on all screen, an instruction component with that text can be defined, and its Screen ID can be set to “Questions, RightScreen, LeftScreen”.

***Setting Screen ID via Settings***

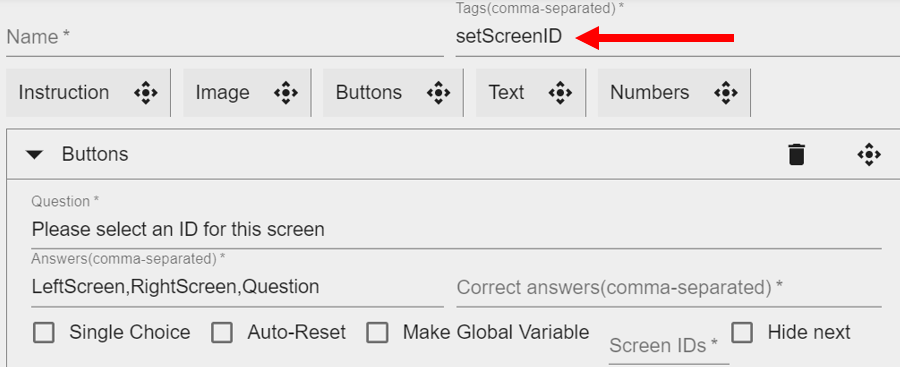
The Screen ID for a monitor can be assigned via the Settings icon at the top-right of the Synquesticon menu bar, under the heading “Device Settings”. Here, the option “Multiple Screens” must be checked and a Screen ID can be entered. If a comma-separated list of IDs is entered, then the screen will show any component with any of these IDs.



*Figure 17. Defining Screen ID via Device Settings.*

***Setting Screen ID via Buttons***

The easier way to define the Screen ID is through the button component. To tell the system that the response to the buttons should be set as a screen ID, the tag “setScreenID” has to be added to the task.

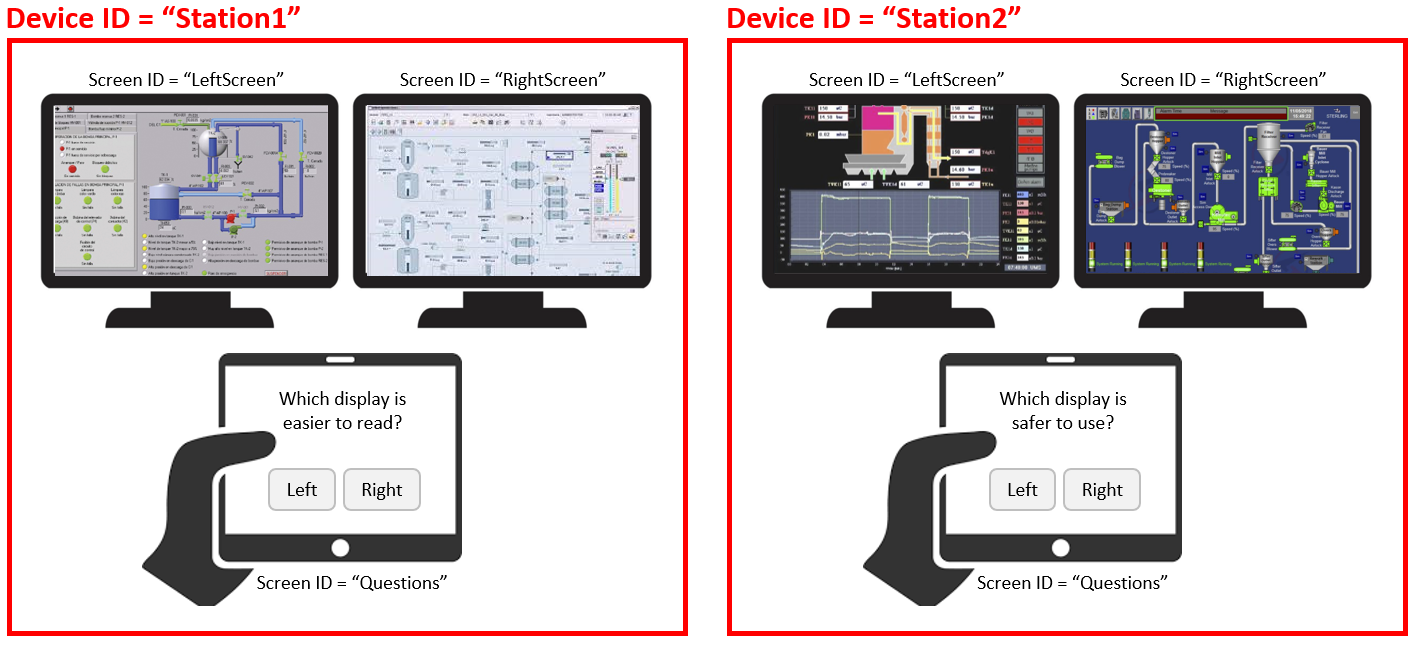


*Figure 18. Screen ID can be set through a button component. The task must be tagged with "setScreenID".*

This Button Component should be displayed towards the beginning of the study. Typically, the researcher will set up the screens before the participant arrives.

***Multi-Screen, Multi-User Studies***

Synquesticon is designed so that multiple participants can run a study at the same time. However, when this feature is combined with multi-screen feature, it creates the problem of knowing which screen belongs to which participant. To solve this problem, a Device ID can be assigned to the screen (see Figure 17). When different Device IDs are assigned to sets of screens, each set will only respond to matching Device IDs, even if the same Screen ID is set.



*Figure 19. A Multi-Screen, Multi-User setup where two sets of screens have the same Screen IDs but different Device IDs.*

***Multi-Screen Means Multi-Tab***

The Multi-Screen feature is implemented so that each tab in a browser can be assigned a separate Screen ID. This way, the same computer can drive multiple displays by dragging browser tabs to the different monitors. This feature is only limited by the number of monitors a computer can support.

If more screens are needed, other computers can be added. If they receive the same Device ID, they will be synchronized together and proceed to the next task when the “Next” button is clicked on any of the screens. Likewise, one or more tablets (e.g. iPad mini) can be added. This feature will work across the Internet as long as all clients are connected to the same Synquesticon server and the same MQTT broker.

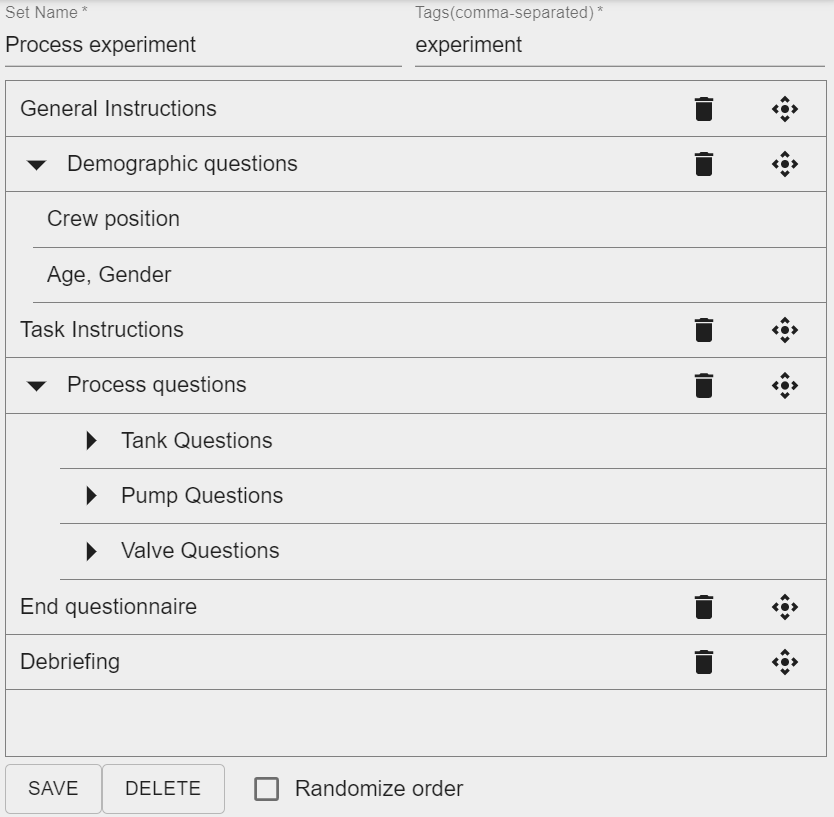
## Set Editor

Below the Task list is the Set section. Sets are used to group Tasks for easier organization, for re-use and for enabling randomization.

New Sets can be added by clicking the (+) icon at the top of the Set list. Tasks can be added to the Set by dragging them into the Set editor. The order of Tasks can be rearranged by dragging them up or down in the Set editor.

In addition to Tasks, you can also add other Sets to the Set editor (nested Sets). For example, you can organize all demographic questions into a sub-Set and then drag them into the experiment set.

The set editor has a “Randomize order” option at the bottom. This will randomize the order of items in the set.



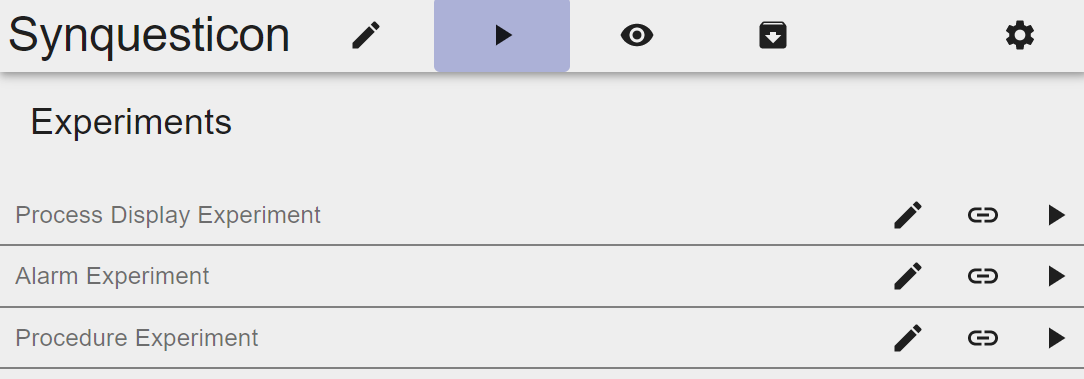
*Figure 20. Set editor, showing Tasks and nested sub-Sets.*

**How to Make a Set Appear in the Experiment List**

Experiments can be run from the list in the Player section (“Play” icon in the task bar). For a Set to appear in this list, it must be tagged with the word “experiment), see Figure 20.

# Player

The Play section of Synquesticon shows a list of all Sets that were tagged with “experiment”.



*Figure 21. Player screen with experiment list.*

Next to each experiment is an edit button that will open the Set in the editor. To the very right is a play icon that will start the experiment in the browser and on all other devices that are connected to the same Synquesticon server, the same MQTT broker, and have the same Device ID (see section 5.1.2.3).

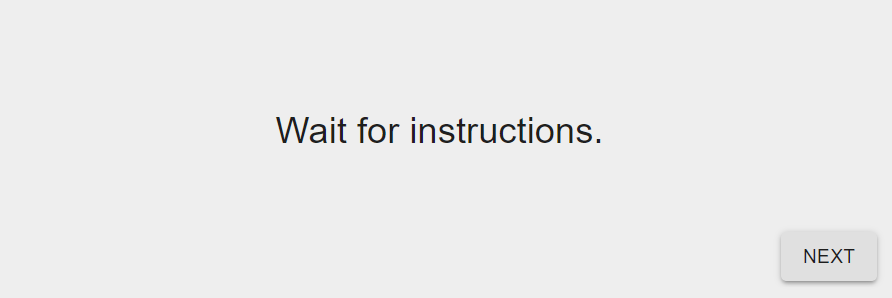
**How to Generate Links for Remote Studies**

Between the edit and the play icon is the link icon. It generates a URL that can be sent to remote participants. The link encodes the experiment ID and a participant ID. A separate link should be sent to each participant. Along with the link, participants need to enter the correct MQTT settings. Participants need to enter these settings before starting the experiment. In a future version of Synquesticon, the MQTT settings will be encoded in the link.

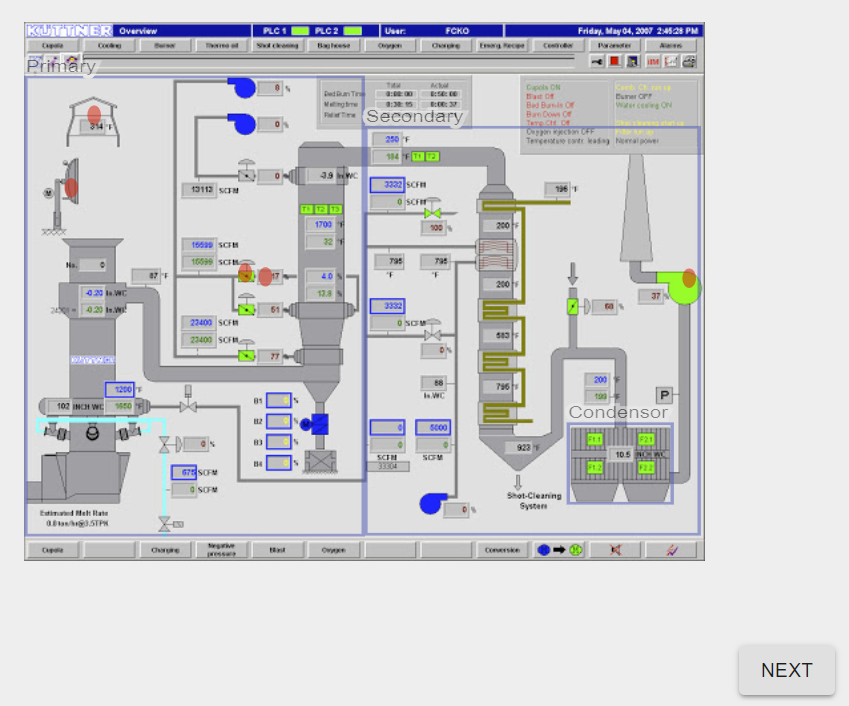
Multi-screen functionality can be used with remote studies launched through a URL. In that case it is strongly recommended to set the screen IDs through a button component (see section 5.1.2.2).

**Component Types in the Player**

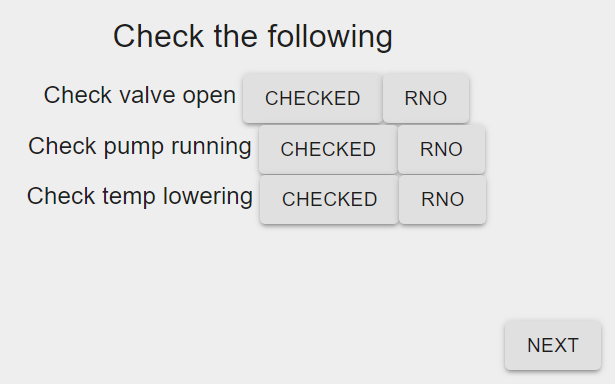
This section shows examples of how each component type is rendered in the player.



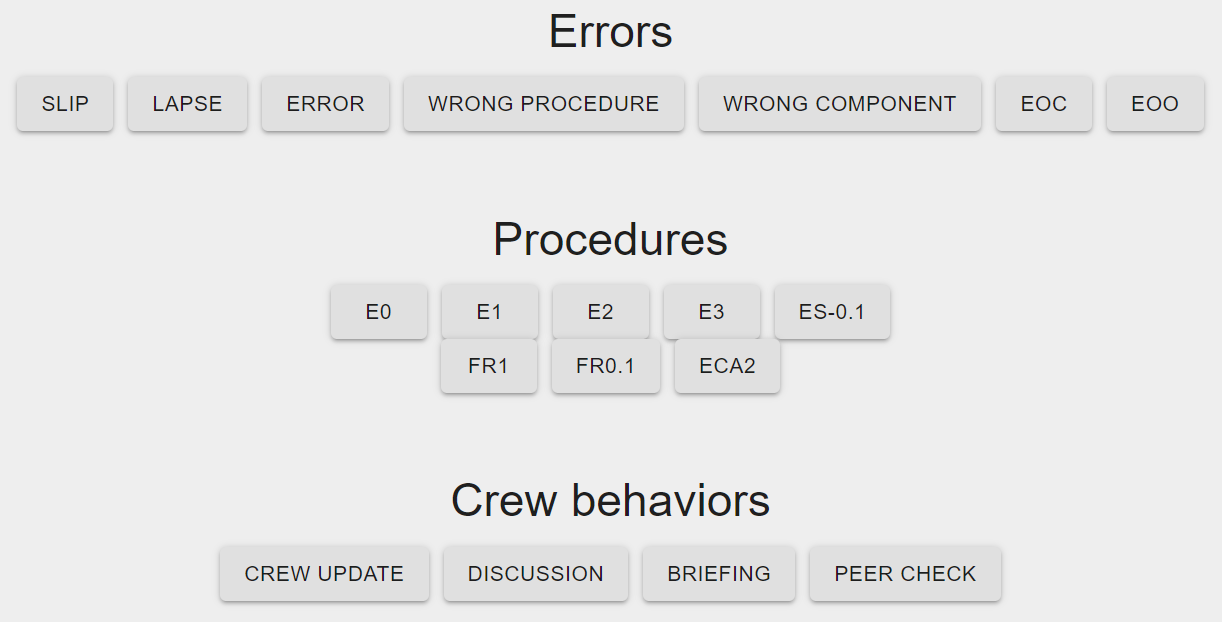
*Figure 22. Example of an Instruction Component.*



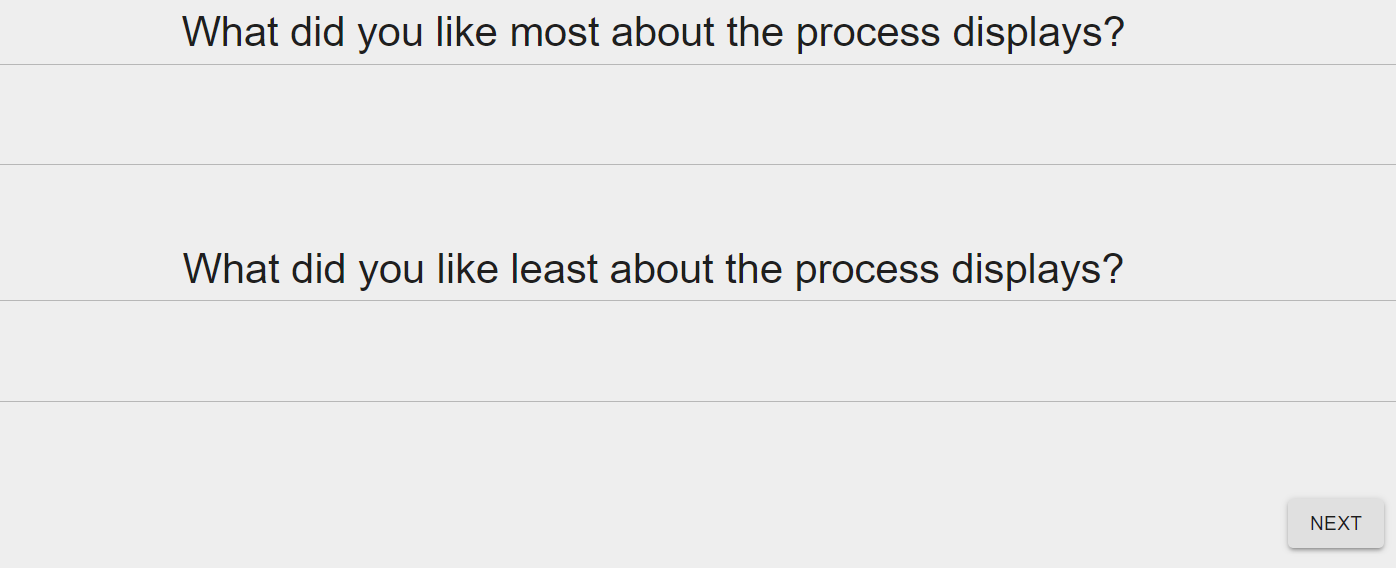
*Figure 23. Example of an Image Component with options "Show AOIs" and "Record Clicks" turned on.*



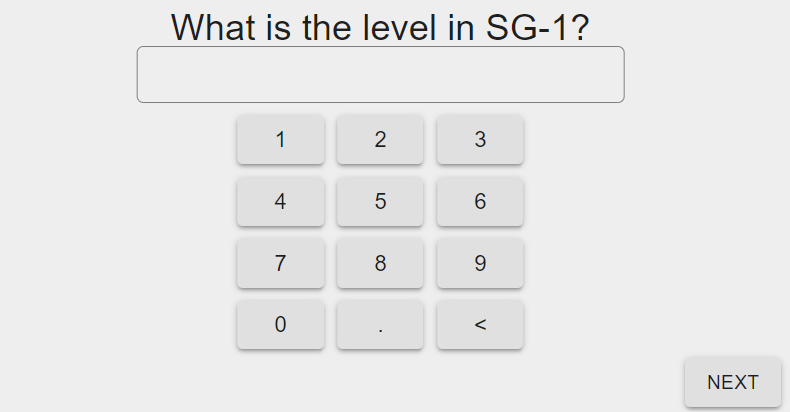
*Figure 24. Example of a Button Component.*



*Figure 25. Example of Task with multiple Button Components.*



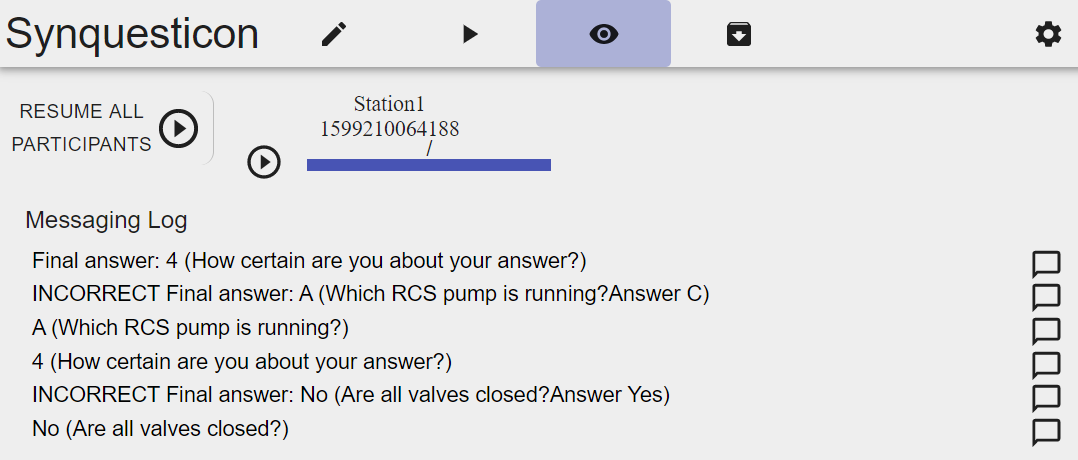
*Figure 26. Example of a Task with two Text Components.*



*Figure 27. Example of a Numbers Component.*

# Observer

The observer shows all element-level events (e.g. button clicks, image clicks) and component-level events (i.e. final answers when the Next button is clicked). If different Device IDs are defined in the Settings menu, a new tab for each device (i.e. participant) is created in the observer.



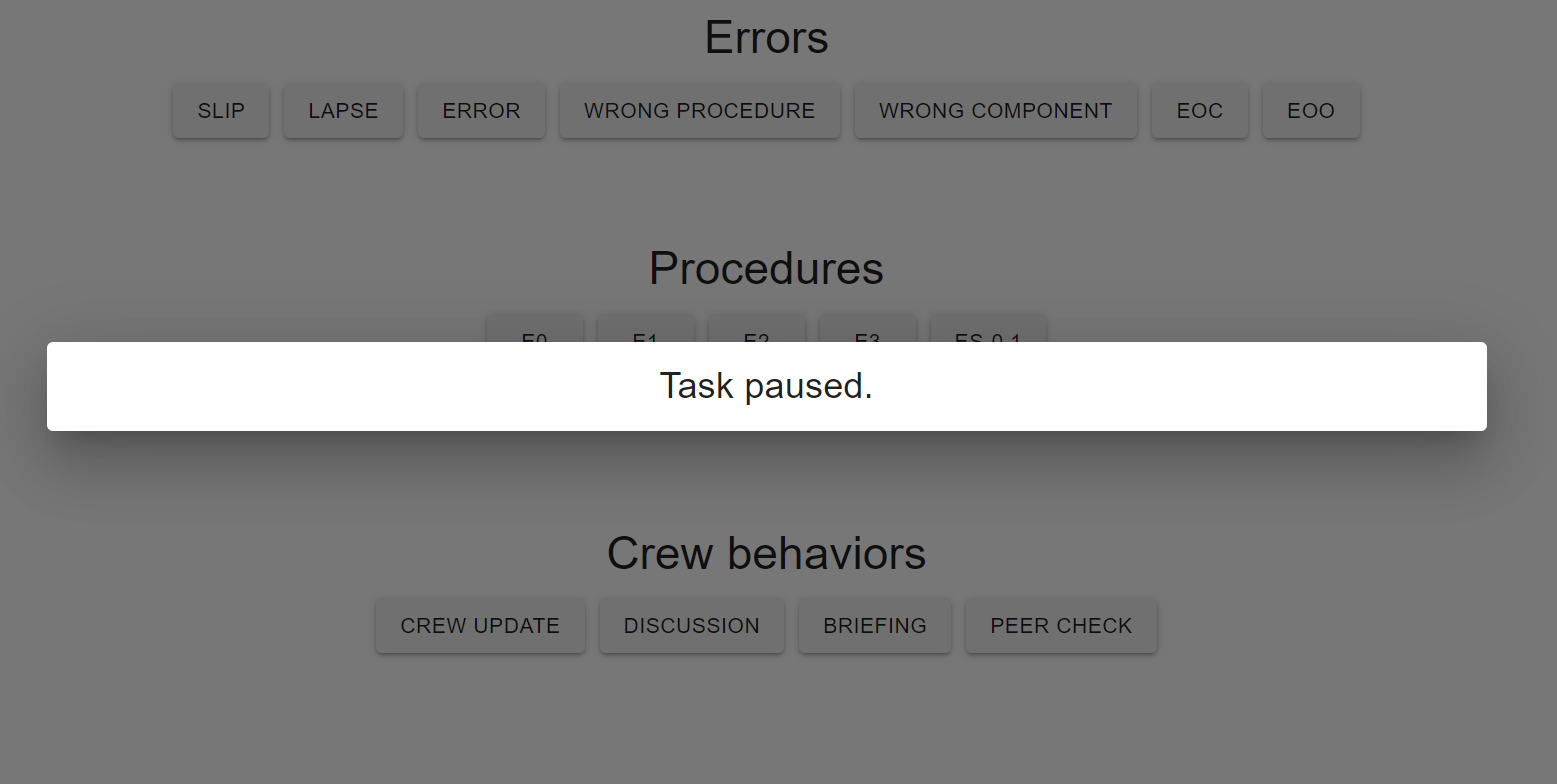
*Figure 28. Observer.*

Element-level events are shown with the response and the question text (e.g. “B (Which RCS pump is running?)”. The log will also show when the participant changed an answer (e.g. “Un-click B (Which RCS pump is running?”).

Component-level events are shown with the text “Final answer” and, if a correct answer is defined for the component, with an evaluation of correctness (e.g. “CORRECT Final answer: Yes (Are all valves closed? Correct: Yes)”.

**Pause Function**

Next to each participant tab, there is a “Play/Pause” icon. When clicked, an overlay with the text “Task Paused” will appear on the participant’s screen, and no buttons will be clickable. The experimenter can resume the study by clicking the pause button again. There is also a global pause button in the top-right corner that will display the “Pause” message on all screens.



*Figure 29. "Pause" function.*

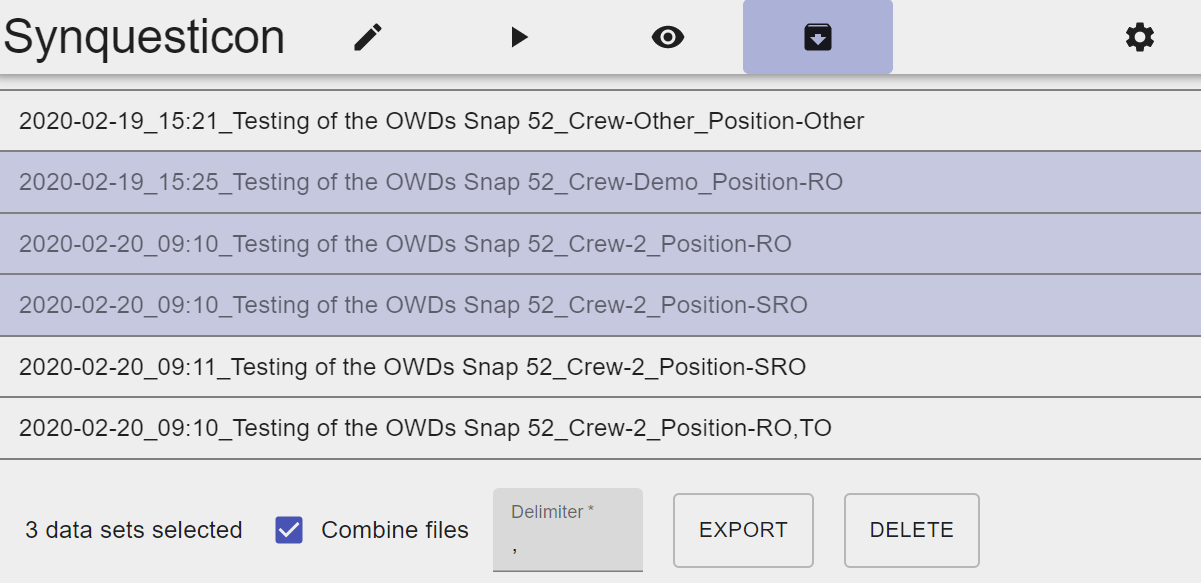
**Comment Function**

To the right of each entry in the observer is a comment icon (see Figure 28). Clicking it will open a dialog where the experimenter can enter a comment. This comment is saved in the log file. This function can be used to tag data points that should be discarded.

# Export

A separate dataset is logged for each participant. Datasets can be downloaded as .csv files through the export function. Each dataset is labeled with the time of the recording, the name of the experiment (i.e. the name of the top-level Set), and with any global variables. It is recommended that variable such as crew ID and participant ID are set as global variables, so that datasets can be easily distinguished (see section 5.1.1.3).

One or multiple entries can be selected by clicking on them. The user can select if the datasets should be exported as individual files or as one combined file by checking the “Combine files” option. The default delimiter is “,” but this can be changed in the “Delimiter” field. The “Export” button will start the export. The browser will show the file(s) in the status bar once download is completed. Data files are saved in the sub-folder /backend/exported\_data of the Synquesticon installation path.



*Figure 30. Export function.*

**Variable Labels**

The saved .csv data files can be imported into a statistics software. The first line of every file contains the variable labels. The following data is available:

* **Global variables.** A colon-separated list of global variables.
* **Family tree.** Names of the sets and subsets in which this task occurred.
* **Task content.** The question text or image name.
* **Start timestamp.** Timestamp when the task was presented.
* **First response timestamp.** Timestamp of first response.
* **Time to first answer.** Time from when the task was presented to first response.
* **Time to completion.** Time from when the task was presented to when the participant clicked “Next”.
* **Answer.** The participant’s response.
* **Correctly answered.** “Correct”, “incorrect”, or “notApplicable” (in case no correct answer was defined).
* **Comments.** Any comments that were entered through the observer module

Note that the Synquesticon data structure is currently being extended. Check the documentation on github.com/synopticon/synquesticon for the latest version.

# Settings

The settings are accessed through the icon in the top-right corner of the screen.

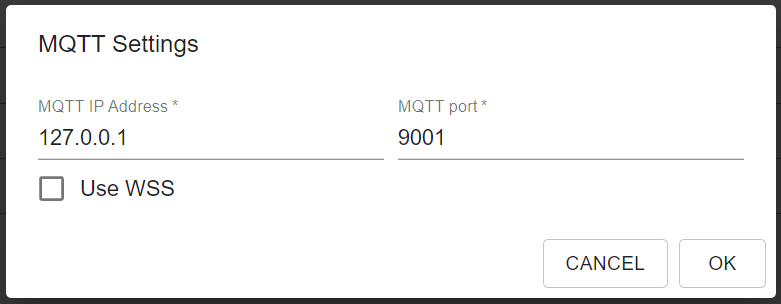
**Device Settings**

Device settings are explained in section 5.1.2.

**MQTT settings**

Synquesticon controls experiments through command messages sent over MQTT, an Internet of Things messaging protocol. Without a connection to an MQTT messaging broker, Synquesticon does not work. An MQTT broker is installed by default and started when Synquesticon is started (in case of a local installation on Windows, this is done when “launcher.bat” is double-clicked).

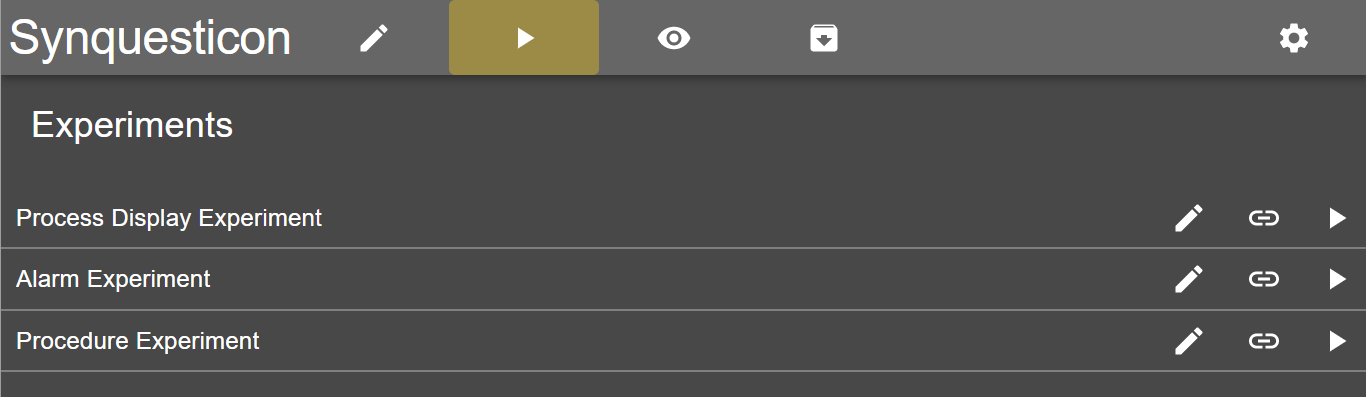
The default settings for a local installation of Synquesticon are. address 127.0.0.1 and port 9001. If port 9001 is blocked, try port 8883. MQTT settings for a server installation are explained in Appendix A.



*Figure 31. Default MQTT setting for a local installation of Synquesticon.*

**Dark Theme**

The next entry in the settings menu is “Dark Theme”. It toggles between the default color scheme and a dark theme.



*Figure 32. Dark theme.*

**Fullscreen**

The next entry in the Settings menu is “Fullscreen”. When collecting data with Synquesticon, it is advised to switch to fullscreen mode. In this mode any unwanted user interface elements, such as the browser’s back button, are removed. This mode also allows images to be shown seamlessly when the option “show fullscreen” is checked in the image editor (see section 5.1.1.2).

Note that on iOS devices, fullscreen mode is only available in the Safari browser, not in other browsers such as Chrom or Firefox.

**Eye Tracker**

This entry provides a drop-down list of all connected eye tracking devices. The user can select an eye tracker for the data collection. Currently only Tobii eye trackers are supported. The eye tracking functionality in Synquesticon is currently being updated. Check github.com/synopticon/synquesticon for the latest documentation.