NordDesign 2018
August 14 – 17, 2018
Linköping, Sweden

Open ended problems – A robot programming experiment to compare and test different development and design approaches

Achim Gerstenberg[1], Martin Steinert[2]

[1] Norwegian University of Science and Technology
Achim.Gerstenberg@ntnu.no
[2] Norwegian University of Science and Technology
Martin.Steinert@ntnu.no

Abstract

Evidence-based research in engineering design is in need of well-documented experimental setups that are replicable by other researchers. We describe an experiment setup that involves programming a LEGO Mindstorms robot to solve a complicated open-ended task. This task has a quantifiable performance outcome. The setup is adaptable to investigate different development and design approaches related to engineering performance. It is designed such that direct personal interactions between the participant and the experimenter are minimized to keep the conditions repeatable. We provide a description of the robot and the dedicated software library that makes it easy to control the robot, the open-ended task, the physical setup and the interactions with the participant. We illustrate how this setup can be used in two study design examples.

Keywords: Design Science Experiment, Quantitative Research, Repeatability, Robotics

1 Introduction – need for controlled experiments in engineering design research

Controlled experiments are an essential scientific method for improving the understanding of the design process (Cross, 1994, Blessing & Chakrabati, 2009). These experiments have a falsifiable hypothesis (Popper, 1935), are repeatable under similar controlled conditions and are useful to determine causality (Blessing & Chakrabati, 2009). These conditions include the physical environment, hardware, software, the participant selection and the interaction between the participants and the experimenter (Kriesi et al., 2016). According to Bender et al. (2002) and Kriesi et al. (2016) design research is in need for more and rigourously documented study designs.
In this paper we provide the documentation of an experimental setup and examples of how the setup can be used in two different study designs. The experimental setup is adaptable for testing

different hypotheses related to open-ended problem solving in robotics that use engineering performance as a dependent variable.
In the following section we describe a LEGO Mindstorms robot, a specific library for controlling this robot and the complicated open-ended task that can be solved using the robot.
To allow replicabilty of the experimental setup we provide a detailed description of the physical environment and the hard- and software used. The forth section describes how the participant interacts with the experiment setup and the experimenter and how to keep these interactions controlled and repeatable across different participants and replications of the experiment. Section 5 lists the raw data that can be gathered with this setup whereas section 6 descibes how to utilize this data in two possible study designs. The conclusion summarizes and gives an outlook as to how this experimental setup could be extended.

2 Task with a quantifiable performance metric

We aim to quantify the performance in an open-ended engineering task. Such task does not have a single solution but instead allows several different solution paths.
In this presented experimental setup, the participant programs a robot to solve the task. The time the robot needs from starting the execution of the program until the task is finished is one of the quantifiable performance indicators.

2.1 The robot

The robot is built from LEGO technic and is controlled by a LEGO Mindstorms NXT 2.0 system. Two electric motors can move the robot via two belts; one on the left and one on the right side. The robot has in total 4 sensors. A ultrasound distance sensor is mounted in the center front and two identical light/colour sensors are mounted on the sides. The light/colour sensor can detect light intensity and measure the colour of nearby objects. These three sensors point forward. Near the front and pointing downwards, the robot has a reflection sensor. The sensor emits red light and measures how much of this emitted light is reflected back into the sensor. The reflection sensor can detect differences in reflectivity of the surface underneath the robot. The robot is shown in figure 1.
The robot is programmed by the participant in the NXC language. NXC stands for “not exactly C” and is a programming language very similar to C. The participant is provided with a library that includes functions specifically written for this robot. It simplifies the interpretation of raw data such as converting the time of flight of the ultrasound pulse from the ultrasound distance sensor to a distance in centimeters or the raw light sensor values into a meaningful value for detecting blinking lights. The library allows the participant to quickly use the robot to solve the task instead of spending time on programming the basic robot functionalities.

Table 1. Library of specific functions

Functions	Short Explanation
motor(left belt speed, right belt speed)	Moves the belts with belt speeds from -100% to 100% of the maximum motor speed
turn(turning speed, degrees of turning)	Turns one belt forward and the other belt backward with the specified speed to turn the robot on the spot by the specified angle
ultrasound()	Returns the distance to an ultrasound reflecting object in cm.
blink	Is an automatically updated global integer variable that corresponds to the measured blink intensity combined from both light sensors.

reflectionDown()	Gives a value based on the reflectivity of the surface under the robot.
ReflectionRedLeft(), ReflectionRedRight()	Returns an integer value corresponding to the red light reflectivity in front of the left or right light/colour sensor
wait(waiting time)	Interrupts code execution for the waiting time in milliseconds
startTimer1(), readTimer1()	Starts/resets a timer and reads the current timer value in milliseconds. Timer 2 and 3 work analogous. The timers can run in parallel with the execution of other codes.
random(lower limit, upper limit)	Generates a random integer between the lower and the upper limit.
dispNum(x, y, value)	Displays the input value on the screen of the robot at pixel location x,y.
dispText(x, y, text)	Displays the input text on the screen of the robot at pixel location x,y.
playTone(frequency, duration)	Plays a tone with the specified frequency in Hz for the duration defined in milliseconds.

The participant is given a more detailed description of these functions and a data sheet about the robot that provides information about movement behavior and sensor properties such as driving speeds, turning precision, precision and angle dependencies of the ultrasound and light/colour sensor.

2.2 The task

A rectangular cardboard playground platform of approximately 1,50 m by 1,20 m has a white area rectangle in the centre surrounded by a 17 cm cardboard fringe. On top of the white area three coloured cubes (red, green and blue) are placed (figure 1). The participant programs the robot so that the robot removes the cubes entirely from the white area in the shortest possible time.
The cubes have a base and top plate with a side length of 14 cm and two diagonal plates that create a retroreflector, which is required for detection from all directions relative to the cube by the robot’s ultrasound sensor. In the top of the cube a cut-out allows inserting one blinking light per cube that can be detected by the robot from any direction.
The participants can optionally place up to three blinking lights anywhere on the playground including inside the top of the cube. The blinking lights are detectable with the light/colour sensor and using the blink variable from the library. The robot can detect the difference of reflectivity between the cardboard surface and the white area with the downwards pointing reflection sensor. If the robot drives off the cardboard playground the height of the cardboard inhibits the robot from driving back onto the cardboard playground and the task is failed. The robot shall be capable to solve the task from any starting position and orientation inside the white area. Therefore, the participant is presented with a new starting location and orientation of the robot for each performance evaluation. These starting configurations are unknown to the participant to avoid solutions specialized to a specific setup but are the same for all participants to allow comparisons between participants. The starting positions and orientations of the cubes remain the same every time a code is evaluated.

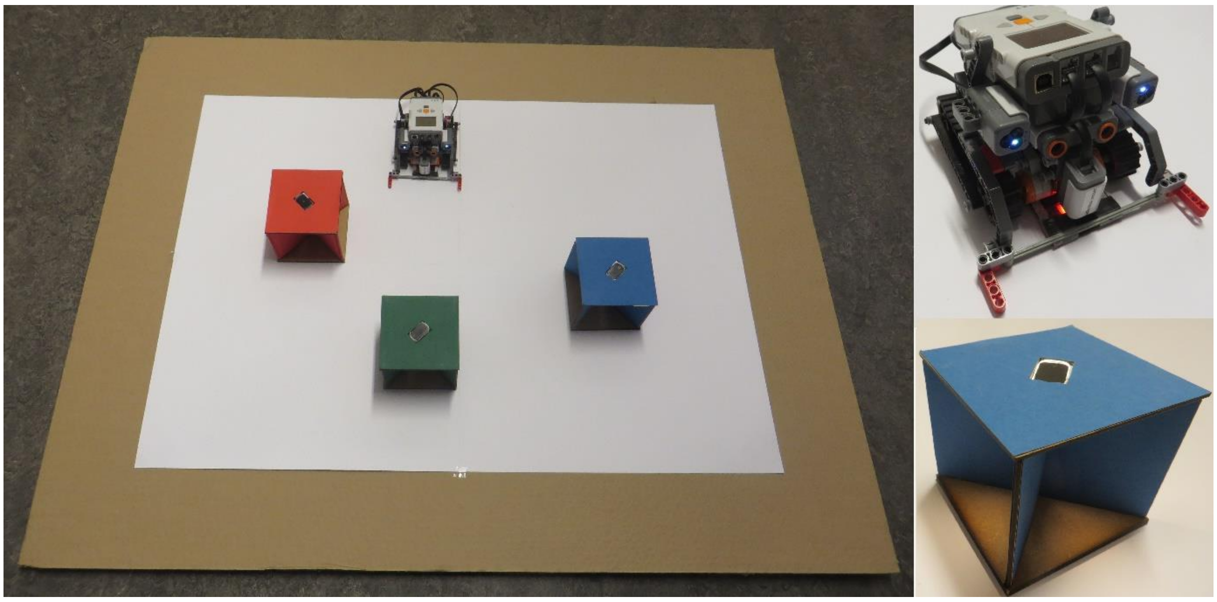
Figure 1. Playground platform with cubes (in detailed view with blinking light turned on) and robot.

Performance measures for this task are how many cubes the robot removes and how quickly. The task difficulty can be increased by taking the cube colours into account.
For example the task can be extended to detecting the colour of the cubes before they are pushed out. The participants can gain a 10 seconds time bonus by correctly and a 10 second penalty by falsely detecting the cube colour. Another way of increasing task difficulty is to remove the cubes in a specific order or only removing two cubes but not the third of a specific colour.
The task is open-ended because it is not clear to the participant which approach leads to the best performance. For example, it is not clear from the beginning of the task if using the light detector and the blinking lights or the ultrasound detector leads to a faster and more reliable detection of the cubes.
The task with removing all cubes and detecting their colour was tested on third year undergraduate cybernetics students. The majority of those students were able to find a solution that removed all cubes and fewer than half were able to successfully detect the colours of the cubes.
The task is complicated but not complex according to Snowden and Boone (2007) as it has multiple solutions and it is not easy to see which solution performs best. It calls for investigating several solutions but all necessary data for solving the task is available from the data sheet alone. A participant who had no prior experience with this robot and this task was able to solve the task at the first test. This verifies that the provided information was sufficient to solve the task.

3 Detailed description of the experimental setup

For replicability of the setup of the experiment, we provide a detailed description of the physical environment, the hard- and software used in the setup and the interactions between individuals and objects in the experiment. More details can be found on github (citation, excluded for blind review).

3.1 Physical environment

The experiment room is divided by honeycomb structure cardboard walls into three areas. Behind the door lies the testing area with the playground platform. Next to the testing area in the corner of the room is the programming booth where the participant programs the robot. The programming booth and the testing area can be temporarily separated by a sliding cardboard wall. The experimenter controls the experiment from the experiment control area adjacent to the programming booth. This area is also separated from the other two areas by cardboard walls.

Table 1. Physical environment

Location	3rd floor of a university building
Total room size	4,0 m by 4,5 m = 18 m2, ceiling height: 2,8 m
Programming booth size	1,6 m by 1,4 m, height 2,0 m
Lighting in the playground area	two 28W warm white neon light bulbs in the centre above the cardboard playground, no direct sunlight
Room temperature	22°C
Background noise	Noise from PC ventilation, indistinguishable conversation sounds outside the experiment room

3.2 Hardware

Table 2. Hardware inside the testing area (4,0m by 3,1m)

Cardboard playground	152 cm x 118 cm x 3 cm, honeycomb cardboard
Cubes	14 cm x 14 cm x 14 cm, with laser cut 6mm MDF bottom plate and crossed 3mm MDF plates as retroreflectors and cavity for blinking light
3 blinking lights	Lalizas Safelite 2, lifejacket light, blink frequency of 0.92 Hz
Webcam 1	Logitech, HD 720p, connected to PC2

Table 3. Hardware inside the programming booth (1,6m by 1,4m)

Working table with chair	Wooden table top 1,20m x 0,55m, height 78 cm
Programming booth walls	honeycomb cardboard sheets, height 2.0 m
Lighting	IKEA LED strips, reflection enhanced with aluminium surfaces on the programming booth walls, halogen reading lamp: TYPA9904, 20W, IKEA
Main screen	LCD display, Philips Brilliance 240B, 24”, 1920 x 1200, connected to PC1
Instruction screen	LCD display, Benq FB731, 17”, 1920 x 1200, connected to PC3
Keyboard	Dell RT7D20, PS/2, Norwegian layout; connected to keystroke logger and experiment control device
Mouse	Input device to PC1
Webcam 2	Logitech, HD 720p, connected to PC2
Robot	Based on LEGO Mindstorms NXT 2.0 Firmware: 1.28, AVR: 1.01, BC4: 1.01, Build:1902091856 The code is transferred to the robot from PC1 via a USB cable
Written documentation for the participant	Consent form, programming exercise, questionnaires, task description
Robot datasheet and library instructions	Explanation of the functions provided in the programming library, robot data sheet providing information about the robot’s properties and performance
Pen and paper	1 ball pen and 1 A4 white paper sheet for note taking

Drinking water	Ca. 1 L with paper cup for each participant

Table 4. Hardware inside the Experiment Control Area (1,4m by 2,4m)

Main screen duplicate	LCD display, Dell U2410, 22”, 1920 x 1200, duplication of the main screen from inside the programming booth (PC1)
Camera screen	LCD display, Samsung, 1920 x 1080, for webcam 1 & 2
PC1	Dell, Intel Core Duo CPU @ 2.33 GHz, 4GB RAM, 64-bit system running Windows 7 Enterprise, service pack 1
PC2	Apple MacMini, Intel Core I5 dual core @ 1.4 GHz, 4GB RAM, 500GB hard disk, running OSX
PC3	Dell Laptop, Intel Core i7-4910MQ CPU @ 2.9 GHz, 16 GB RAM, 64 bit system running Windows 8.1 Enterprise
Keystroke logger and experiment control device	Arduino Leonardo with protoshield, experiment routine control and data saving onto a 4GB, 4 MB/s SD card, received keystrokes from the keyboard and emulates a USB keyboard sending keystrokes to PC1
Paper documentation	Pre-experiment checklist, templates for manually documenting task performance, lab book for note taking
stopwatch	Used for measuring the task completion time

3.3 Software

Table 5. Software

Bricx command center	Version 3.3, build 3.3.8.9, running on PC1, development environment for programming the robot
Instruction slides	Power point 2013, Microsoft Office, running on PC3
Screen recording	ShareX 11.9.1, used for recording the main screen, running on PC1
Audio player	for playing audio instructions, itunes 12.1.2.27, running on PC2
Arduino IDE 1.8.4	Development environment for programming, starting and reading out the keystroke logger and experiment control device, running on PC1. additional libraries used: Keyboard 1.0.2, SD 1.1.1, PS2KeyAdvanced 1.0.2, PS2KeyMap 1.0.2 adapted to Norwegian keyboard, Time 1.5.0, SPI 1.0.0

4 Interactions with the participant

Interactions with the participant can consciously and subconsciously influence the participant’s behaviour during the study (Kahnemann, 2011). Direct human-human interactions between the participant and the experimenter are especially difficult to standardize. Therefore, this experimental setup minimizes direct human-human interactions by using exactly repeatable pre-recorded and computer generated voice instructions, video and text displayed on the instruction screen and paper handouts with printed information. The interactions that were not digitized, such as the timing for handing out paper information, are kept as similar as possible by following a scripted interaction protocol.
In the following we present some concrete interaction examples for this experiment.

The participants get invited by a standardized email message through a person who is otherwise not involved in the study. This avoids introducing the experimenter and thereby biasing the participant already before the experiment. For example, the participants cannot know if the experimenter is male or female. To avoid confusion, the invitation message includes that the participant will not be greeted by a human and that the experiment starts automatically when the participant enters the room.
The task is presented to the participants through illustrations on the instructions screen and assisted with computer-generated voice explaining the illustrations. The participants also receive a printed task explanation on a paper handout.
Written documentation is exchanged between the participant and the experimenter through a slit in the programming booth wall above the main screen. The slit is designed such that it does not allow visual contact between the participant and experimenter. The timing of handing out papers follows a protocol. For example the printed task description is handed out before the other explanations start to avoid that the handing out distracts from the other instructions.
As the aim of this experimental setup is to investigate problem solving methods and not to test coding fluency debugging support is given. When the participants compile the code and the compiler issues a syntax error the participants receive help in finding and correcting these errors. This help is limited to syntax errors and does not include logical errors. The help is given by computer generated voice and positioning the mouse cursor at the location of the syntax error in the code.
Participants can ask questions that can be answered by computer-generated voice with either “yes” or “no”. The participants receive answers to organisational questions and do not receive technical help in solving the task. All new questions are noted so that a similar question will be answered equally to other participants in the future.
Before an evaluation, the experimenter closes the cardboard door towards the programming booth to avoid visual contact with the participant and then places the cubes on the cardboard playground. To indicate the starting position and orientation of the robot the experimenter places a paper with a sketch of the robot. The participant replaces this paper with the actual robot before starting the robot code for the evaluation as instructed by a computer generated voice.

A more complete list of individual to object (I – O) and individual to individual (I – I) interactions can be found in the table 6.

Table 6. Interactions

Type	Interacting items	Description
I - O	Participant - Movable cardboard walls	Walls create a repeatable work environment and control the view onto the testing area and the experimenter.
I - O	Participant - Written documentation	Participant fills out the consent form, the programming exercise for testing his/her programming knowledge and questionnaires on self-reported experience levels and returns the documents to the experimenter. The participant is provided with the task description which he/she can keep until the end of the experiment.
I - O	Participant - Robot	The participant can upload code to the robot and start the execution of a program by pressing a button on the robot. The participant places the

		robot on the cardboard playground and can observe the robot's behaviour.
I – O	Participant - Main screen	The main screen presents the development environment for programming the robot to the participant.
I – O	Participant - keyboard, computer mouse	Input devices for programming the robot
I – O	Experimenter - main screen duplicate	Experimenter can see what the participant is writing in the programming environment. This is needed for debugging help.
I – O	Experimenter - Camera screen	Used for keeping an overview of the experiment progress and the robot task performance
I – I	Participant - Experimenter	Exchanging written documentation through the slit in the cardboard wall.
I – I	Participant - Experimenter (through predefined computer generated voice)	When syntax errors occur the experimenter plays a predefined computer generated voice instruction and moves the mouse cursor on the main screen to the location of the syntax error. The predefined voice instructions are: 1. "a semicolon is missing" 2. "a bracket is missing" 3. "there is a spelling mistake" 4. "watch the mouse cursor for a hint" for cases that do not match one of the above Furthermore, the experimenter starts a voice instruction for: 1. Greeting the arriving participant 2. Ask to close the door 3. Sit down in the programming booth 4. Making the participant aware the he/she has received a new instruction on the instruction screen if they do not read it immediately 5. Instruct to connect the robot to PC1 6. Answer "yes" or "no" to organisational questions

5 Raw data examples

This experimental setup is designed to evaluate the performance of the participant's solutions to the programming task and the participant's behaviour. The main quantitative performance measures are task completion time, number of removed cubes and time until the task failed. These data are taken by the experimenter through observation via webcam2.
Each evaluation during the experiment gives only data for one solution and for one starting position. As some starting positions can favour one solution over another, more representative results can be generated by rerunning each participant code post experimentally from several different starting positions. Averaging over the performance indicators allows a more precise performance evaluation; the standard deviation shows the robustness of a solution and this becomes a performance indicator in itself. Driven distance and how often the robot reached the

edge or touched a cube without removing it are further indicators for the efficiency of the solution.
The keystroke logger helps quantifying the participant’s behaviour. It timestamps each keystroke with millisecond resolution and thereby allows a detailed analysis of when and how often the participant typed the commands to load the code onto the robot for testing.
The code the participant wrote, screen recordings of the programming screen and recordings of the cameras showing the participant’s and the robot behaviour are gathered as qualitative data. The behaviour measurements allow looking for correlations to the performance measurements and the qualitative data allow for understanding individual cases, help to find causes of correlations and to explain outliers.

6 Study design examples

The aim is to provide an experimental framework that is useful to answer different research questions and allows testing of different hypotheses that relate to problem solving performance in open-ended robotics tasks. The following examples are to illustrate the possibilities of the experimental setup but the reader is encouraged to find their own hypothesis related to their research questions.

6.1 Study design 1: Influence of abductive learning on solving complicated problems

For complex problems a known approach is to conduct experiments, analyze the results and abductively conclude how to change the design (Snowden & Boone, 2007, Gerstenberg, 2015). How can this abductive learning based process be used for solving complicated instead of complex problems? Is testing different designs helpful for selecting the better performing concept? How does early testing influence concept selection?
In this study the participants are randomly assigned to a planning and a testing condition. The planning group is not allowed to load their codes onto the robot and thus cannot test their code for the first eighty minutes of the programming time. The testing group is allowed and encouraged by voice instructions to test their codes at least every five minutes. The solution performances from both groups get evaluated from 80 to 120 minutes in 10 minute intervals. After 80 minutes, during the evaluation phase, both groups are allowed to test their codes.

This study design allows testing two hypotheses.

- Participants who are allowed to test in the first 80 minutes remove in average more cubes in evaluation 1 (after 80 minutes of programming).
- Participants who are allowed to test in the first 80 minutes remove in average more cubes in evaluation 5 (after 120 minutes of programming).

6.2 Study design 2: Better understanding for solving complicated tasks through desirable difficulties

Bjork (1994) introduced the idea of adding desirable difficulties for the learner to improve long term learning. Do desirable difficulties also improve learning and understanding needed to reapply the learned to solve a complicated problem in a new context?
In this study design participants are randomly assigned to two groups. Both groups are introduced to the functionalities of the robot by short and simple exercises that introduce each functionality. For example, the ultrasound distance sensor could be introduced by giving the exercise to drive the robot towards a wall and stop 20cm in front of it. The control group receives a flawless code that solves this exercise. The debugging group receives a flawed code that drives the robot into the wall. They need to understand to repair the code and solve the

exercise. Both groups get the same time for each exercise and are able to change and test the given codes. After completing the introduction exercises, the participants are given the complicated task presented earlier in this pap

Hypothesis: The debugging group removes in average more cubes after 80 minutes of programming the complicated task.

7 Potential sources of error and limitations

Replicating exactly equal conditions is difficult. Executing the same code from slightly changed starting conditions can yield different results. This can be caused by deviations in placing the robot and the cubes before starting the robot. Additionally, deviations in the robot's behaviour over time (due to sensor noise, friction variances in the actuators, varying alignment of the drive belts, battery drainage) lead to fluctuating trajectories and varying performance results.
In order to obtain data comparable between participants it is necessary that codes from similar times during the development process are saved. This means that the participant is requested to deliver their best code at predefined times. These evaluations interrupt the participant's work flow, consume valuable time and we therefore recommend not to do it too frequently. This usually means that each new code is only tested once during the experiment. This does not provide sufficiently many data points to statistically evaluate this one code. The data gathered while the participant is present is not sufficient to conclude reliable performance results.
These two mentioned limitations can be overcome by post-experimentally executing the code that was saved at the times of the evaluations. Thereby it is possible to acquire sufficiently many data points to obtain results with statistic validity. The drawback is that this process is time consuming.
Pretest have shown that about 2 hours of programming time is a sufficient length for undergraduate cybernetics students to solve the task and have time to make some improvements to their solutions. Including introduction to the task and other circumstances the overall time the participant needs to spent on the experiment can be almost three hours. Due to the experimental duration it is difficult to find sufficiently many participants.

8 Conclusion

We present an experimental setup for investigating the participant's behavior in relation to the performance in solving a complicated open-ended robotics task. We describe how this experiment can be conducted under very defined and repeatable conditions to avoid unequal biasing of the participants using automated interactions. We illustrate how this experimental setup can be adapted to investigate different research questions by providing two study design examples.
The automated interactions allow instructions at several locations simultaneously. This makes it possible to extend the experimental setup for investigating remote collaboration.
In the future we hope that other researchers adapt this experimental setup to their research questions and produce results under these controlled conditions to allow comparisons across studies.

Acknowledgement

This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

Citations and References

Bender, B., Reinicke, T., Wünsche, T. & Blessing, L.T. (2002) Application of methods from social sciences in design research, Proceedings of DESIGN 2002, the 7th International Design Conference, pp. 7 – 16

Bjork, R. A. (1994) Memory and metamemory considerations in the training of human beings, Metacognition: Knowing about Knowing, pp. 185 – 205

Blessing, L.T., Chakrabarti, A. (2009) DRM, a design research methodology. Springer London

Cross, N. (1993) Science and design methodology: A review, research in engineering design, vol. 5, no. 2, pp. 63-69.

Gerstenberg, A., Sjöman, H., Reime, T., Abrahamsson, P., Steinert, M. (2015) A simultaneous, multidisciplinary development and design journey – Reflections on prototyping, Proceedings of Entertainment Computing - ICEC 2015, Springer International Publishing, pp. 409-416

Gerstenberg, A. (2018, May 15) RoboExpSetup. Retrieved from https://github.com/AchimGerstenberg/RoboExpSetup

Kahnemann, D. (2011) Thinking fast and slow. Farrar Straus and Giroux

Kriesi, C., Balter, S., Steinert, M. (2016) Experimental studies in design science and engineering design science – A repository for experiment setups, Proceedings of NordDesign 2016

Popper, K. (1935) Logik der Forschung. Springer

Snowden, M.E., Boone, D.J. (2007) A leader's framework for decision making, Harvard Business Review, November Issue