Getting Software to Help Researchers Research

Can the Introduction of Software Toolkits Help Researchers Achieve Better Prototypes for Research and Design of Social Robots

Luiz Felipe de Souza Simão: S1905848 – *CreaTe Module 11 & 12*

University Twente

Dr. Ir. E.C. Dertien – *Supervisor*

Dr. Ir. D. Reidsma – *Critical Observer*

Due date: 12 of November of 2021

Word count: *9900 words* – Date of submission: *12 of November of 2021*

Abstract

A constantly increasing amount of research and development must happen for social robots based on the ever-growing market. However, the high dependency on the multidisciplinary knowledge required to do so creates a barrier of entry. One possible solution for minimizing this barrier is the use of software toolkits. As such, this research will study To what extent can the introduction of software toolkits help researchers that lack affinity for software integration achieve better prototypes for research and design of social robots?

For answering that research question, a design study was conducted, and a prototype workshop was developed and taught at the International Summer School of the University of Twente. It was found that there is incentive for companies around the world to invest in consumer software that can potentially help develop prototypes faster and of higher fidelity. However, the incentive seems to still be linked to the sale of their own hardware in closed bundles. It is seen that the range of prototypes that can made with these software is limited and content sensitive, limiting less technologically savvy researchers' prototypes.

Acknowledgment

I would like to thank both my supervisor and my critical observer for being so considered and willing to help and support me through the journey that was to conduct this study. All the time they took from their ridiculously busy schedules to help, motivate, cheer, and teach me will not be forgotten. In addition, I will also like to thank all the extremely dedicated, helpful, and kind people from the International Summer School of the University of Twente.

Contents

I. Introduction

A. Problem Statement

Since 2019, we have been able to see a constantly increasing demand for social robots and we can expect the growth in the market to continue for at least five more years [1]. For the development of social robots, more research needs to be done on robots and their social consequences [2]. However, developing robots is a difficult task that usually requires extensive amounts of knowledge in multiple domains and tends to affect multiple stakeholders at the same time; *e.g.* the increasing market for robots that interact with humans force engineers to take the role of sociologists [2]. The need for extensive multidisciplinary knowledge creates a barrier of entry for research in the field of social robotics.

Studies suggest that there are many benefits to increasing the types of research and to utilizing different approaches of research done in the field of social robotics [2]–[5]. In order to do so, more researchers that are focused in human sciences instead of engineering need to participate in the research process [2], [5]. This implies that there is a need for the development of research in social robots to be democratized, leading to the facilitation of research, development, education, and co-design. One possible way to contribute to this is to lessen the barrier of entry for research in the field of social robotics.

As illustrated by Darriba Frederiks, Octavia, Vandevelde, *et al.* [6], one solution for lessening the barrier of entry is facilitating and providing scaffolding materials to designers and non-experts. This is especially fruitful when done through toolkits [7]. Toolkits are platforms that attempt to package the complexity of developing robots into simple and understandable building blocks. So far, the main focus of these toolkits has been on the integration of hardware elements. One can find all sorts of packages for hardware integration *i.e.* Arduino and Ono platform [7]. However, this focus on the hardware has led to the lacklustre development of software toolkits. As such, nowadays, the software toolkits aimed at non-coders act as bottlenecks for anything more complex than low-level functionality, limiting the ability to address the high-level functionalities necessary for social robot interaction prototyping.

B. Research Question Proposal

Given the proposed problem statement, an attempt will be made to study the research question *To what extent can the introduction of software toolkits help researchers that lack affinity for software integration achieve better prototypes for research and design of social robots?* For the means of attempting to achieve such an objective, this research will be focusing on the aforementioned fact that software tools that aim to provide scaffolding materials for robot prototyping tend to be focused on low-level hardware applications, simplifying just the coding element of the process; or are uniquely focused in extremely specific tasks or operations [7]. Less technologically inclined researchers can be threatened by *e.g.* having to be comfortable with many different platforms to achieve a study with a social robot. More so, each platform, in its own right, has a learning curve, and these collectively, if not accounted for, cause delays in the research as the researchers would have to siphon through different types and levels of scaffolding materials for each individual platform. This still does not guarantee that less technologically inclined researchers, especially if they are not savvy in programming, can implement the minimally required high-level interactability features for a robot to be considered a social robot [7].

To validate this issue and produce an answer to the research question: this research will tackle three unique sub-questions. These questions will be presented as a knowledge, design, and an evaluation one.

1) Knowledge question: What are the advantages and disadvantages of the currently used/available control software tools for the rapid prototyping of high-level features in social robotics such as behaviour and interaction? For the sake of answering such a question, a dive into the literature of available studies that implemented any of these tools will be analyzed; next to this, a query on available consumer products that might not have been used in scientific research will be made; Lastly, all of the found software' features will be tallied and compared to a list of relevant features for social robots.

2) Design question: How can a stimulating environment for the introduction of a software toolkit to researchers not specialized on working on robots be designed? To answer such a question, a design plan will be created which will start with choosing a software based on the robot features that it can simplify the process of applying. After making that choice, a method for introducing the software will be prototyped.

3) Evaluation question: How can the introduction to the software toolkit be shown to be helpful, and how can it be shown to be a hindrance? An attempt to answer this question will be made by elaborating on the implications of the study presented to answer the design question.

C. Research Plan

In order to conduct this research, the following steps will be taken: (i) Desk research into available literature, such that correct and scientifically supported definitions of both a "social robot" and "prototyping" can be acknowledged; (ii) The making of a list of relevant features for general social robots; (iii) The presenting of a selection of software available to the public and with it, some personal notes on their pros and cons; (iv) The concluding of what would be the best software for the user test; (v) Looking into defining the target audience; (vi) The presenting of the details, choices, and assumptions made for the making of the user test; (vii) The presenting of the final structure of the user test and pilot studies for the test; (viii) Coming forth with the results of the test and relevant conclusions obtained; (ix) The evaluation of the user test; (x) Discussing the conclusions found and answering the research question; (xi) Making recommendations for future research. Considering the first four steps of this research, there shall be a considerable amount of background and desk research. This can be found in section II below.

II. Background Research

A study of available literature, public information, and researchers in the field will be used to discuss the knowledge question as presented in section I-B1 (page: 8): *What are the advantages and disadvantages of the currently used/available control software tools for the rapid prototyping of high-level features in social robotics such as behaviour and interaction?*

To be able to answer this question, the following steps will be taken: (i) analysing existing definitions of social robots and formulating one to use throughout the rest of this paper; (ii) determining different types of prototyping methods used in researching behaviour and interaction of social robots; (iii) listing features deemed particularly relevant for the general social robot; (iv) concluding, currently, are some of the most commonly used tools for researching in the field; (v) discussing what makes these tools similar and what makes them different from each other; (vi) presenting what are the advantages and disadvantages that these tools have against the types of prototyping.

I bring forth the note that section II-D will be complemented with the work done by me for the assignment of academic writing for the course of creative technology in the academic year of 2019/2020. This is not to say that the content has not been altered, updated, and/or fitted to better represent the research that is being conducted in this paper.

A. What is a social robot?

1) Process: It is important to make sure that the definition of what is a social robot is made explicit and grounded such that throughout the paper there is no ambiguity when I mention the term social robot. This section will provide a basis for such and for what are the minimally required features that make up a social robot (found in section II-C).

I will present my definition after studying multiple academic papers that provide their own opinion on the matter. A selection of interesting definitions will then be presented and the similarities and differences between these will be contrasted against each other.

2) Introduction to the Chosen Definitions: For this research, three proposed definitions for social robots were chosen based on their relevance and similarities. The definitions chosen are the ones given by Breazeal, Dautenhahn and Kanda in their chapter on social robotics in the Springer Handbook of Robotics; the definition by Hegel et. Al.

in their paper Understanding Social Robots; and the definition given by Onyeulo and Gandhi in their paper What Makes a Social Robot Good at Interacting with Humans [8]–[10].

a) Definition in the Springer Handbook of Robotics: This definition for what is a social robot can be deduced after having read the chapter, though it is not explicitly mentioned. Nevertheless, I hereby present the definition that was deduced based solely and entirely on the content of this chapter on this handbook: A social robot is a robot that is designed to engage people in an interpersonal manner, operate in a human environment, and operate alongside people. All of this is done with the intent of helping people achieve positive outcomes in a wide span of domains. There is an additional dependency on these robots being natural and intuitive for the general public to interact with, communicate with, work with, and teach new capabilities. A distinction has also been made between a social autonomous robot and a [non-social] autonomous robot, most significantly being the aforementioned existence of the social robots in non-hazardous, non-specialized, human environments.

b) Definition in the paper Understanding Social Robotics: For Hegel et. Al. a social robot has as a main difference between it and any other robot the social attributions that are given to it in the form of a social interface. Thusly, a social robot has a dependency on form, function, and context which defines its social interface and enclosing all the features that have been designed to provide the user with cues and stimuli for anthropomorphizing the robot. Doing so, turning the robot from a tool to a social entity within the mind of the person interacting with it.

c) Definition in the paper What Makes a Social Robot Good at Interacting with Humans: This paper chose to uniquely use the traditional definition proposed by Christoph Bartneck et Al., which implies that social robots are robots that interact with humans with the means of integrating in their society. Thusly, this implies that a social robot has a physical body and, added to that, the capability to mimic human activity.

3) Similarities and Differences: Based on the matrix found in table I it is possible to contrast the opinions of the three different works. It was chosen to compare these opinions based on four categories that were conceptualized after reading the the texts and noticing that all mostly touched upon these similar similar topics. The four categories are:

- Social robots in relation to non-social robots;
- Robot-human interaction;
- The robot in contrast with the environment;
- and The robot as a member of a society.

a) Social robots in relation to non-social robots: The biggest similarity between all three texts is the agreement that what makes a robot “social” is its co-existence with humans. This is key for the formal definition of social

TABLE I

MATRIX HIGHLIGHTING THE DIFFERENT DEFINITIONS' INTERPRETATIONS OF EACH LINK.

	Springer Handbook of Robotics	Understanding Social Robotics	What Makes a Social Robot Good at Interacting with Humans
Social robots in relation to non-social robot	A social robot is designed to exist particularly in human environments, unlike a non-social robot.	A social robot must uniquely have some level of a social interface.	Through whatever means, a social robot is capable of integrating into a human society.
Robot-human interaction	Natural, intuitive, and aimed at helping people achieve positive outcomes.	Uniquely tailored to make it easier for the human interacting with the robot to anthropomorphize it.	The robot is composed of a physical body and the capability to mimic human activity.
The robot in contrast with the environment	The robot must be able to simply co-exist in as many different human environments as possible without impacting its normality.	The robot is only as unique as the social context which it was put in.	The robot is capable of seemingly integrating into human society.
The robot as a member of a society	The robot operates in an interpersonal way alongside people.	The robot hosts its utility as a tool, while still being able to act as a member of society	With the ability to mimic human activity, the robot can act its role as a fully functional member of society.

robots. Where the three articles do disagree is in the grade of the social integration. Within the Springer Handbook of Robotics it is brought up that the robot barely needs to integrate itself in the social context, as long as it is operating around humans. Inversely, the article of Onyeulo and Gandhi state that a robot can only be considered social if it is directly effective in integrating into the social context.

b) Robot-human interaction: This is a point that showed a greater variance between the articles. In essence there is the belief that the most natural way for a social robot to interact with humans is by acting human-like while at the same time attempting to facilitate the interaction for the human. The most unique distinction found between the opinions was the point raised by the article of Hegel et. Al. wherein it is implied that it is not what the robot does but how it does it that is important. Specifically it is mentioned that the robot should, by all means, attempt to act in a way that makes the human interacting with it believe that it is interacting with a human-like concept.

c) The robot in contrast with the environment: Between the three articles, there is an interesting bipolarity regarding the relevance of the environment for a robot. While the Springer Handbook of Robotics and the article of Onyeulo and Gandhi imply that a social robot is independent from the environment context, and that it should be able to inconspicuously integrate onto a plethora of predominantly human environments; the article of Hegel et.

Al. implies that a social robot is exclusively defined by the environment that it find itself in.

d) The robot as a member of a society: The last category is one that all three papers agree entirely. Apparently, it is by definition that a social robot can be considered a tool and can achieve its purpose as such, but the social robot has in addition the element that it achieves its purpose without diminishing its established social status.

4) Concluding the Definition of Social Robots: After discussing and evaluating the opinions of the three articles a definition can be made that will serve as a basis for the rest of this research. Mostly the three articles agree with each other, and, as such, the chosen definition does not veer away from theirs. However, in the category "The robot in contrast with the environment" there is a distinction made. Though the other two articles were in favor of the opposite, I believe that the definition of Hegel et. Al. is more appropriate as it highlights how a social robot can solidify the context for a social interaction. The ability to define a context is incredibly important for understanding social innovation [11, Ch. 17]. Inversely, this also implies that, when considering the elements of an innovation within a social domain, the context should be incredibly important.

The definition will be presented in terms of the aforementioned categories respectively. (i) A social robot has all the necessities required to be considered a robot, with the addition of at least some feature that would allow it to participate in a social context/exchange; (ii) A social robot should be able to, in the process of helping someone, make that someone feel as if they are interacting with a social agent instead of a tool; (iii) A social robot is a robot that can complement and be complemented by its environment, consolidating the social context for others around; (iv) A social robot, above all, serves a function within a society.

B. Methods of Prototyping Within Social Robotics

As pointed out by Saul Greenberg [12], prototyping can be reduced to two levels of fidelity: Low-level prototyping and high-level prototyping in the field of Human computer interaction. The term fidelity relates to the degree of which the prototype resembles the final system being designed [13]. Low-level prototyping is what is usually referred to when dealing mock-ups of a system that do not yet resemble a final product (*e.g.* a paper prototype). High-level prototyping is what is usually referred to when dealing with a functional mock-up that includes some level of intractability, like a computer simulation and directly resembles a final product. Given that this research is related to control software tools, the focus will be limited to those that relate to what is considered ideal prototyping methods for high fidelity prototyping as noted by Greenberg [12]. Three types of prototyping in this level of fidelity are Computer-based simulations, Wizard of OZ, and Slide show or video prototyping.

Most software tools available to consumers are designed with some sort of facilitation for puppeteering or reenactment. These features usually come in similar solutions for scheduling movement and choreographing certain actions and reactions. Probably as a consequence of this, a significant amount of literature studied use these facilities and incentives to design prototypes of social robots using the wizard of Oz prototyping method. For instance, Pollmann, Ruff, Vetter, *et al.* [14], Al-Sada, Jiang, Ranade, *et al.* [15], and Helton, Head, and Blaschke [16] all have developed research of social robots rendered using some type or another of wizard of oz prototyping. In conclusion, it is believed that consumer available tools are most practical for doing a middle to high fidelity prototypes utilizing the wizard of Oz technique. As such, it will be important to highlight this limitation when introducing the software to researchers during the design stage.

C. Key features Present in Social Robots

Both to be able to answer the knowledge question and to better form the content of this research, a selection of minimally required features for a general social robot will be made. To achieve this goal the proposed theory of the social interface of Hegel, Muhl, Wrede, *et al.* [9] which was itself based on the Offenbacher approach to the Theory of Product Language [17] is taken as a source of inspiration. The mentioning of the Theory of Product Language is relevant as it highlights how people attribute qualities to objects, in this case, robots. This quality attribution process is what allows for any robot to eventually be considered a member of a social exchange.

As explained by [9], for a social robot, the process of being given social qualities happens as a consequence of orchestrating its practical functions (the objective things that the robot is made to do) and its product language functions (the aesthetic form that the robot takes) such that the anthropomorphization of itself takes place. Based on that theory, it is deducible that social robots are not so dependent on specific features per se but instead rely on well planned out, case and context specific features. This makes it difficult to conclude on a specific selection of features that are all around necessary for a social robot. As such, the following feature set will be assumed and recommended to be required features for a general social robot, and presented as a heuristic.

- Speech recognition;
- Visual recognition;
- Choreographed motion;
- Speech synthesis;
- Artificial intelligence.

This set of features can be additionally separated into two different categories based on my interpretation of the Social Behaviourism Theory of Personality and Behavioural Interaction [18]: behavioral and social-interactional features.

a) Behavioural Features: These relate to the features that the social robot has that serves as a mean to emulate a personality repertoire. A personality repertoire directly affects one's behaviour and how one interacts with others.

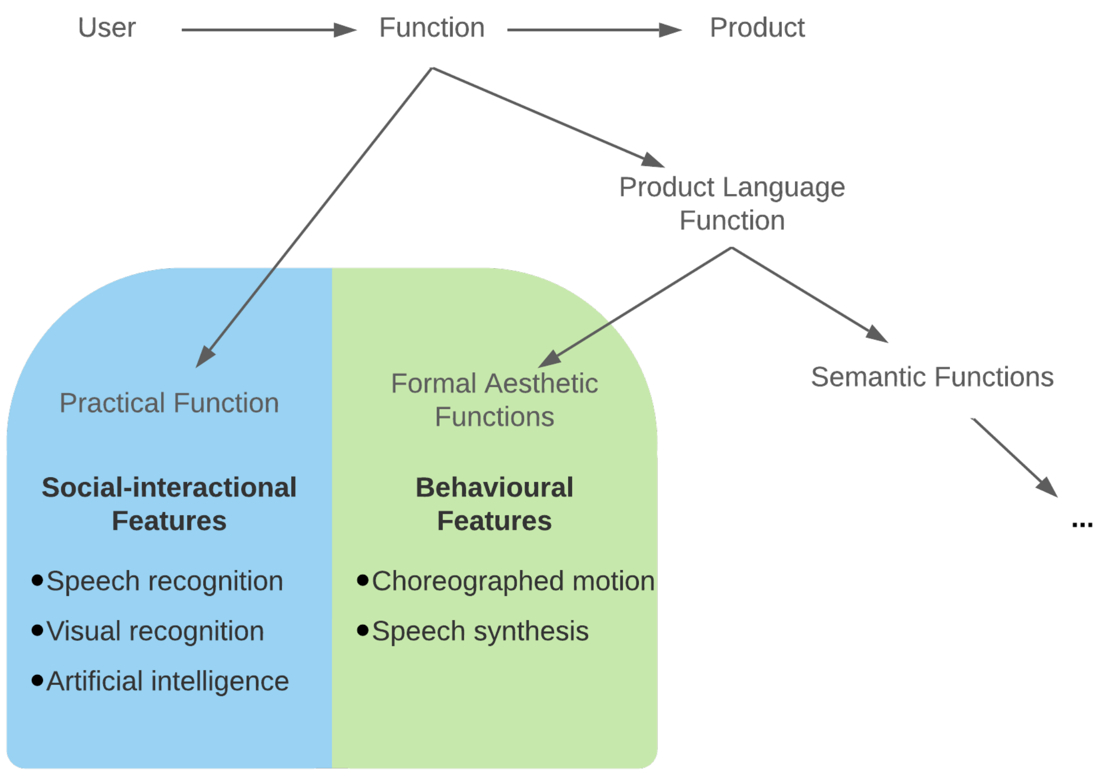


Fig. 1. Visual representation combining my interpretation of the Social Behaviourism Theory of Personality and Behavioural Interaction, and the Theory of Product Language to distinguish between features present in social robots.

Looking at this from the opposite direction, it is possible to see that by defining how one interacts with others, it will consequentially emulate a behaviour and thus a personality repertoire. Thusly, the internal features that control how the robot acts will be considered behavioral features. These include the speech recognition, visual recognition, and artificial intelligence.

b) Social-interactional Features: These relate to the features that the social robot has that serve as social behaviorisms. These behaviorisms represent what one does to elicit a stimulus upon another. The difference from these to behavioural features lies in the ability that these features have in emulating an intention for the interaction. This can be translated into actions that the robot does that directly influence another. As such, the external features: choreographed motion and speech synthesis will be considered social-interactional features.

The final selection of features are expressed in image 1. This image also displays the multiple theories interacting together leading to the final choice.

D. Control Software Tools Used Within the Field of Social Robotics

Since 2000, there has been a significant rise in the interest for educational robotics [19]. It is now becoming common place to have companies invest in education oriented social robots, as brought up by De Nadai Victal and Candido [20]. Considering these robots and the facilities they provide, a list of the software tools can be generated and studied. The collection of consumer available humanoid/puppet robots studied is presented in table II. Having looked through these products and their publicly available information, the following control software tools where isolated:

- Custom Python APKs
- Custom Scratch bootstraps
- ROS
- Choreograph suite
- EZ-Builder
- Robotis' R+ suite
- Coppelia sim

All of these tools have been refined and structured in a way to facilitate the entrance of new users and speed UP the process of creating simple forms of interaction. As such, Despite these being originally dedicated to education or other markets, the fact that they are mostly made with the intention of being puppeteered shows semblance to the needed tools for wizard of Oz prototyping.

1) Most commonly used tools: In order to determine whether any of the aforementioned tools are more or less relevant for conducting research, a comparison is going to be made based on the amount of articles that have been published about designing social robots with the mentioned software. This approach is going to evaluate whether there has been any academic publication done (peer reviewed or not) using the tools. First: a a search through the Google Scholar database is going to be made to see if there are any publications about using the tools. Second: the search is going to be refined to see how many of the publications found are about designing for interaction or behaviour using said tools. Third: a judgment is going to be made about whether it is relevant to perform further study on the tools, or if there is not enough material to ground any discussions. The results of the first step are found in table III; the results of the second step are in table IV.

a) Ez-builder: The amount of hits highlights this company's obscurity. This is a possible consequence of the company focusing more on quality than on advertising together with being relatively young (founded in 2012).

b) Choreograph suite: The choreograph suite is a well known package of software used primarily for the controlling of the famous Pepper robot and the Nao robot. Sadly, Choreograph is also a commonly used word.

TABLE II

CONSUMER ROBOTS BY THEIR PRODUCT NAMES, WHAT TYPE OF CONTROL TOOLS ARE PROVIDED WITH THE COMPANY, AND IF THE TOOL ALLOWS FOR DIRECT FORMS OF PROGRAMMING

robots	tools	programmable?
lynx	n/a	no
dobi	individual platform	no
ganker EX	n/a	no
zeus	individual platform	no
alpha 1p	individual platform	yes
cosmo	open & individual platform	yes
NAO	open	yes
yanshee	open & individual platform	yes
hexa	individual platform	yes
inmove	open	yes
jd humanoid	open & individual platform	yes
jimmy	open	yes
plen2	individual platform	yes
robotis	open	yes
poppy	open	yes
hugu	n/a	yes?*
e-motion butterfly	n/a	yes?*
snake bot	n/a	yes?*
super anthony	n/a	yes?*

*There was not enough information available to conclude whether the robot is programmable or not.

TABLE III

SHOWS THE RESULTS OBTAINED FROM RUNNING THE QUERY INTO THE GOOGLE SCHOLAR DATABASE ON THE DATE: 16-4-2020.

Original Google Scholar Query	Results found
ez-builder AND robot	23
Choreograph AND robot AND (”nau” OR ”pepper”)	113
robotis AND robot AND (”r+” OR ”IDE”)	284
educational robotics AND ”scratch” AND ”visual programming”	378
educational robotics AND ”ros”	401
educational robotics AND ”python ”	641

Note: There was no relevant query that gave practical results for Coppelia Sim.

TABLE IV

SHOWS THE RESULTS OBTAINED FROM RUNNING THE QUERY INTO THE GOOGLE SCHOLAR DATABASE ON THE DATE: 16-4-2020. THE DIFFERENCE IS BETWEEN THE RESULTS FOUND IN TABLE III AND THIS ONE.

Original Google Scholar Query	Results found	Difference %
ez-builder AND robot AND (behavior OR interaction) AND social	11	52
Choreograph AND robot AND (nao OR pepper) AND (behavior OR interaction) AND social	89	21
robotis AND robot AND (r+ OR IDE) AND (behavior OR interaction) AND social	60	79
educational robotics AND scratch AND visual programming AND (behavior OR interaction) AND social	246	35
educational robotics AND ros AND (behavior OR interaction) AND social	196	51
educational robotics AND python AND (behavior OR interaction) AND social	304	53

Thus, to guarantee that the articles that were found were about the program, the query included the name of the robots. This is known to cause a likely bias in the articles found, since it is now necessary that the use of the software was also used with the robots, which does not have to be the case. Never the less, there were still good signs that this was not going to impact to relevance of the tool, given the sheer number of articles still found.

c) Robotis' R+ suite: This query was more complicated than the rest since Robotis is a company that makes many parts for robotic studies. As such many of the hits found with a simpler query were about the use of those parts, and not the program suite. To attempt to mitigate this, the term (”r+” OR ”IDE”) was included and seemed to solve most of the issues.

d) Custom Scratch bootstraps: Scratch being a programming language, it was taken into account with the same parameters that Python and ROS had. However, unlike Python which is a high level language that allows you to code everything that the robot can do, Scratch is a lower level language that has been developed specifically with the intention of being friendly to new users, with its visual codding paradigm [21]. As such, the addition of the ‘visual coding’ query was kept in, since any research that attempted to use scratch whilst removing the visual aspect is seen as diminishing the ease of use heuristic.

e) Custom Python APKs & ROS: For the queries regarding Python and ROS, two very popular programming and high level environments in robotics, the choice of selecting articles that used them had to include certain limitations [22]. The query as seen in table III looks for articles that specifically use the environments for educational robotics. The argument for this is that both Python and ROS are mostly used in a more industrial setting. This will not do for the case of this research, as one of the criteria that the tools need to account for is ease of use. On that note, the educational robotic paradigm requires the platform to be “usable by all users of the target group regardless of their prior knowledge” [23, p. 208].

f) Coppelia Sim: This software seemed to not give any relevant results independent of which query was made. As such, it was removed from the first step onward.

2) Differences Between tools:

a) What Makes These Tools Similar: Over all the platforms, there is a definite consensus in that hardware integration, especially in the form of control and the scheduling of robot movement is the most important feature to be streamlined. It is present in all the features studied and is shown to be of great influence when it comes to entertainment and prototyping. Especially since wizard of Oz shows requires some form of having specific cues or full control of the robot. One more feature that is quite widely available, being present and used in most of the studied literature is the ability to communicate between the development tool and some form of wireless controller. The streamlining of these features is very optimal for the most appropriate method of prototyping as discussed above: wizard of Oz. Therefor, the unification of the tools is already making great impact in making of future research of social robotics. In spite of this, the fact that the tools are often locked environments is a detrimental factor, as that means that *e.g.* if one company offers one option/solution, it is very possible that this option/solution will not cooperate with other companies’ offerings. This, in turn, limits the research potential of these tools.

b) What Makes These Tools Different: There are critical differences between the tools, as expected due to competition. Unlike the similarities that are well highlighted and clear, the differences are very divergent. Most of the studied tools have a tendency to prefer aiding their own platforms versus facilitating integration. However, some of the more recently developed platforms, *e.g.* EZ-Builder, have in their key concept multi-platform integration. The multi-platform integration is something that comes in handy in most forms of research as it allows for more cost efficient study designs, making use of available resources, and maximising the use of the already available expertise, since no new technologies have to be learned.

The reintroduction of old programming paradigms is something that is sometimes overlooked in these new platforms. Some of the platforms studied, are taking in the initiative of reinventing some computer languages such that they are made “easier”. This so far is used as advertisement point, though through the means that they are approaching it, it is also taking these well established languages and removing some of their key elements. Though it is possible that these reformulated languages might show up as more approachable by a younger audience it is also limiting the maximum performance that can be achieved with such languages. In addition, this reformulation can also limit the amount of available scaffolding material to the amount that the individual developers can output (I.e if the developers making these new languages do not provide good enough material, it is less likely that there will be available material elsewhere, which is the case for more mainstream programming languages and software). Something that can be destructive to the success and continuation of the tools.

E. Conclusion

In an attempt to answer the proposed knowledge question defined in section I-B1: *What are the advantages and disadvantages of the currently used/available control software tools for the rapid prototyping of high-level features in social robotics such as behaviour and interaction?* it was given a definition of what a social robot is, techniques that are often considered for prototyping, a heuristic about features that the general social robot should be able to have, and a selection of used control software. It was found that the consumer available control software mostly benefits medium to high level, wizard of oz prototyping. It is believed that this conclusion is also supported by the companies that are making control software solution as they all share in common facilitations for wizard of oz prototyping such as by providing puppeteering support or allowing for acting queues.

It has also been found that the advantage of the commercially available control software used for prototyping robots is their ability to effectively streamline the development process of some types of control problems particularly revolving around prototyping social-interactions. They are all particularly effective in making the process of controlling a series of motors for emulating anthropomorphic movement, a notoriously tedious and time consuming process, take minutes instead of hours. It was also found that there is a general push towards some of the features required to effectively "wizard of Oz" social robots' behaviours. The disadvantage of these software, however, is that they are more often than not made as part of an enclosed ecosystem. This means that there is a significant preference for working with one brand of hardware (*i.e.* the same brand that makes software also make hardware). This can increase the cost of conducting research and sometimes even limit the amount of research that can be done altogether.

Nevertheless, the existence of control software made to be used by all technical levels that specifically facilitates social robot prototyping, highlights room for academically relevant research using commercially available tools. These tools have shown to be taking the right direction in the aim of helping develop future research in social robotics. With this in mind, a design study will be conducted to evaluate the extent to which introducing said software to a researchers that lack affinity for software integration help them achieve better prototypes for research and design of social robots.

III. Design study

Within this chapter the design study that will serve to better answer the proposed research question is presented. For this, the proposed design question: *How can a stimulating environment for the introduction of a software toolkit to researchers not specialized on working on robots be designed?* will be discussed. To be able to discuss the question, the following will be presented: (i) the ideation steps taken to come up with a method for introducing a chosen software; (ii) a list of design requirements for the proposed prototype method for introducing the software; (iii) the specifications of the pilot prototype; (iv) the reevaluation of the specifications post testing the pilot; (v) and the finalized prototype.

A. Design Plan

The design plan presented (see figure 2) has six phases. The first phase in the design plan represents the use of the knowledge question for undergoing this design study. The implication being that phase one has already been discussed in the paper previously and the outcome of this phase is the collection of currently available control software tools capable of facilitating rapid prototyping of high-level features in social robotics such as behaviour and interaction.

Phases two and three represent the first elements of the design study and correspond to the ideation phase of the design process for creative technology [24]. In phase two, a specific software will be chosen for the development of the design study. From this choice, a selection of social robot features will be argued as relevant for the study. In phase three, a target audience for the design study and possible ways to reach out to them will be defined. Additionally, the method to which the designed prototype will engage with them. These two phases will be put together to define the the list of design requirements for the proposed prototype method for introducing the chosen software, leading to phase four: designing a pilot of how to introduce to the target audience the specifically chosen features on the picked software.

Step four represents the specification stage of [24]. As briefly touched upon before, within this step the choices made during phase two and three will be put together to make the design specifications of a pilot method for introducing the software. Some points are considered and discussed to further refine the pilot. The process for realizing and evaluating the pilot will also be presented.

phase five will bring forth the results of evaluation of the pilot test. These results will be used as material for discussing points of interest within the proposed prototype. The aforementioned will also serve as arguments for reiterating upon the design process so far producing a new list of specifications for the final prototype.

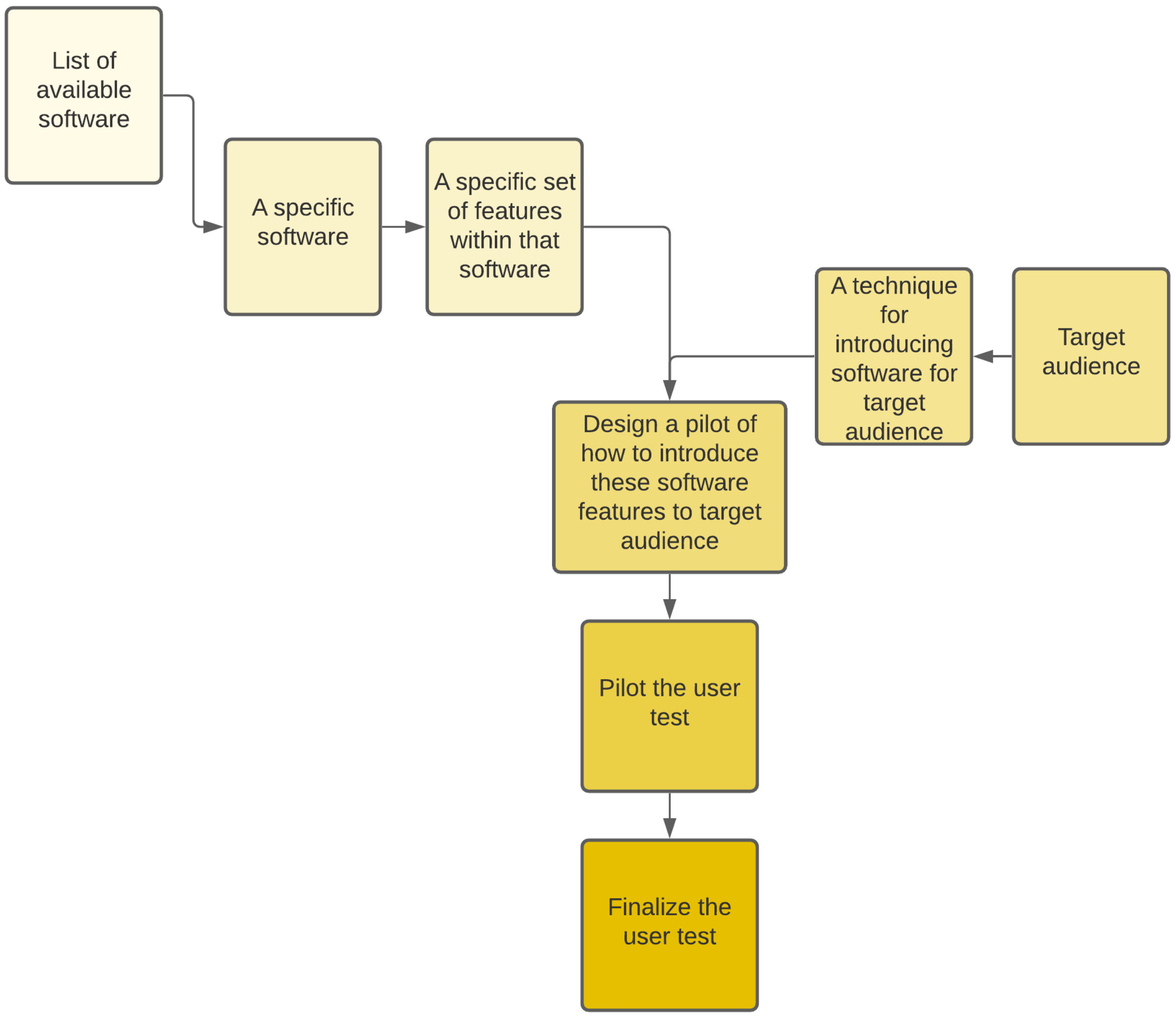

List of available software
A specific software
A specific set of features within that software
A technique for introducing software for target audience
Target audience
Design a pilot of how to introduce these software features to target audience
Pilot the user test
Finalize the user test

Fig. 2. The proposed study plan. It is separated into six phases and eight steps. The phases are differentiated using color and position. The arrows represent the chonological order at which the steps need to take place.

Phase six will be running the user test of the final prototype. This will be further discussed in chapter 4. The last phase hosts a link between the presented design study and the evaluation question and will serve as driver for the discussion section IV.

B. Ideation

As previously state, this section will follow steps two and three of the study plan which imply that (i) the choice for a specific software, (ii) a set of features controllable by that software, (iii) the definition of the target audience, (iv) and the technique for how the software will be presented to the target audience is to follow.

1) The Choice of Software: For making the choice of which software to use, some criteria for making such choice should be discussed. The criteria to be discussed are focused on facilitating prototyping. Specifically, the design question targets academics that are not specialized in working with social robots. These two factors together lead to the proposal of a software tool that offers a low threshold and high ceiling based on the concept of *threshold and ceiling* introduced by Myers, Hudson, and Pausch [25, pg. 6].

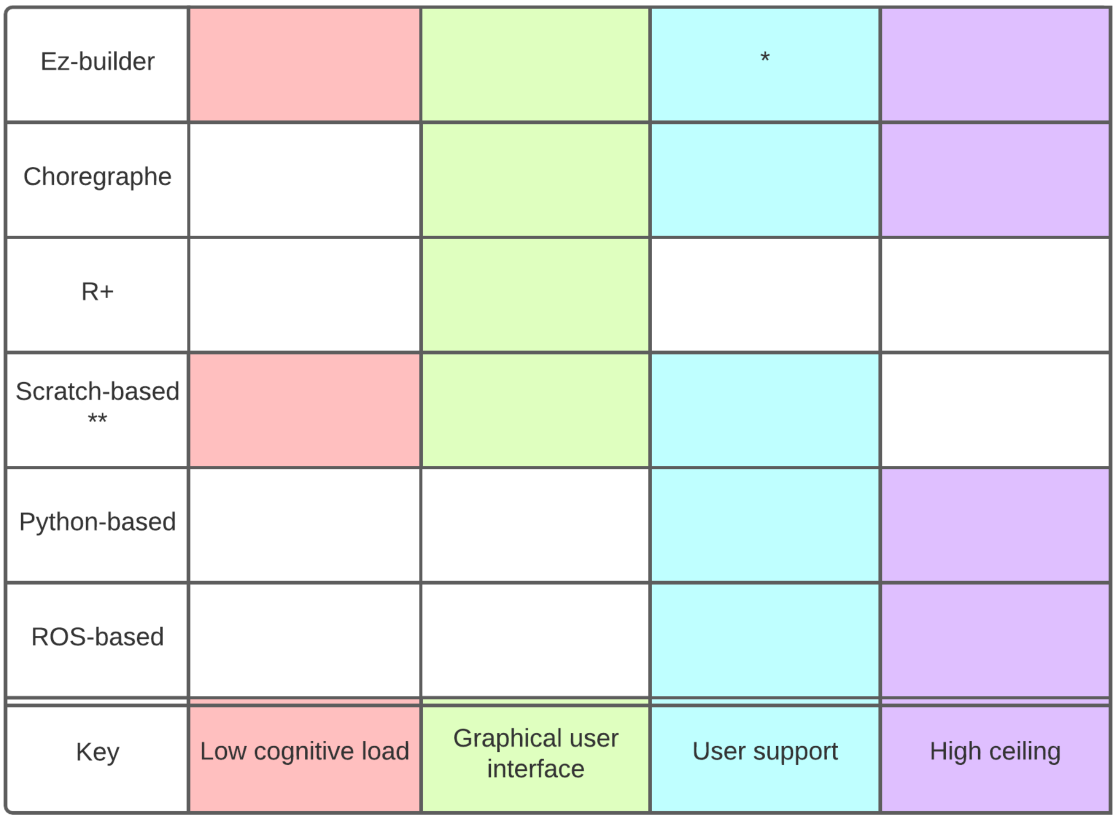
The gold standard, especially when dealing with complicated topic done by a non-specialist, is that of the *low threshold and high ceiling* as mentioned before. A low threshold implies that the software tool is easily understood and ready to go as soon as it is picked up. This element is extremely important for this study. As such, it will be expanded into more detailed design criteria.

a) Low Cognitive Load: The first design requirement to be presented, one specifically relating to the low threshold element, is focused on the cognitive load of using the software tool. As was presented in the article by Chandrasekera and Yoon [26, Pg. 6] cognitive load is directly related to the effectiveness of the learning process. The study shows that the higher the cognitive load of whatever the learning interface is, the more effort the learner has to put into using the interface. Given that cognitive load is a limited and shared resource among different types of tasks, the more effort that has to go into using the interface the less effort will be left for learning with that interface. The previously mentioned problem also applies to prototyping with the aid of a platform *i.e.* if too much effort goes into understanding and using said platform, less effort can be allocated to prototyping. Considering the aim of facilitating prototyping for those that are not used to prototyping, a low cognitive load will be taken as a design requirement.

b) Graphical User Interface: It is known that a graphical user interfaces (GUI) are known to have a drastically variable cognitive load depending on the way that it is designed [27]. Despite this, there is also evidence a novice user can solve task faster and with fewer steps when attempting to solve these tasks with a GUI instead of a text-based user interface [28]. Considering as such, another design requirement will be a platform that offers a native GUI for most standard operations. Together with this design requirement an additional point of interest will be in the quality of the aforementioned GUI. This can be discussed based on the three different aspects as proposed in the study by Yee, Ling, Yee, *et al.* [27].

c) User Support: It is also important for the lowering of a software's threshold to have a well established source of supportive material. Thus, a design criteria will be such an access to supportive material for any level of user. This can be either through forums, videos, tutorials, or other readily available forms.

d) High Ceiling: The last design requirement to follow will be one that is commonly used to evaluate software tools, specifically the second element of the concept of [25]. The ceiling of a software tool implies how much can be done with that software. As such, a tool with a high ceiling is one that allows the user to accomplish a lot.



Ez-builder	🟥	🟩	🟦 *	🟪
Choregraphe		🟩	🟦	🟪
R+		🟩		
Scratch-based **	🟥	🟩	🟦	
Python-based			🟦	🟪
ROS-based			🟦	🟪
Key	🟥 Low cognitive load	🟩 Graphical user interface	🟦 User support	🟪 High ceiling

Fig. 3. The matrix used for choosing which software toolkit fit the proposed criteria. In the left are the software, the bottom the meaning of each column. Each cell coloured in highlights that the row's software fits the column's criteria. User support for EZ-Builder is technically less prominent than that of the other coloured in cells, but it is existent and highly supported by the company that runs that software.

Due to the subjectivity of 'a lot', the design criteria to be proposed here will be mostly relating to how limiting a software is.

Continuing with the process of choosing what software to use for the design study, a breakdown of how many design criteria the six previously discussed platforms meet. The tally can be seen in figure 3. It is found that, together with some additional comments other than just meeting the design criteria, EZ-builder (which changed its name to ARC throughout the development of this study, thus will be referred to ARC from now on) was found to be the most appropriate software tool for this design study.

As mentioned, other than just meeting the design criteria, ARC has some special elements which add to why it was chosen. Namely, ARC supports the utilization of of python and scratch within its catered environment. It does not have a very strong user based support however, the developers of the software do the best they can to provide feedback to any and all questions thrown their way.

2) The Features to Study: For the upcoming design study, a selection of relevant features will be considered. The aim is that the features chosen are particularly relevant features to be able to prototype social robots in a wizard of Oz style for conducting research. The hypothesis is that it is possible to conduct a user test where academics that lack affinity for software integration show to be able to comfortably and/or efficiently produce a prototype usable for research.

The features considered to use all fall under the categories: speech technology, image recognition, and movement.

a) Speech Technology: As explained by [29], speech is both necessary for social robots to be able to establish its social standing, and it is also a complicated field that could benefit form having more research take place. Because of this, the ability to prototype speech technology is going to seen as a critical element for my design. ARC has multiple ways to simplify the process of implementing speech technology into your prototype. For example, you can simply add speech synthesis modules and speech recognition. It also allows for speech to text for further more complicated analysis. Lastly it also has very streamlined processes for connecting to existing cloud based solutions for all of the above, if it would benefit you to do so.

b) Image recognition: Image recognition is a much more general term than speech technology. It encompasses many things that would be useful or even necessary for a social robot. Things such a face recognition, emotional recognition, people detection, etc... . Due to the utility of this feature, it will be one of the required features to evaluate. Within ARC you can do quite a bit with image processing as well. The program naively supports video ingestion and post processing which in turns already streamlines a significant amount of the work for most cases of image processing

c) Movement: You cannot really have a robot that is entirely incapable of doing any form of motion. For this case movement controllers and similar behaviours are necessary for a high fidelity prototype. ARC does support many different facilities to model, choreograph, and puppeteer robots that are made using many different types of hardware. Despite this, due to the COVID-19 epidemic, there is a need to prepare this design study with as little contact as possible. As such, to include a test that would account for users trying out movement based features would require hardware to be presented to participants or be expected that they would have access to such. Because of this, it was decided to not include movement features in the study to best account for minimizing unnecessary contact between researcher and researchee.

3) Defining Target Audience: Within this section, the specific audience of this study will be defined. It is imperative that the target audience is both relevant for answering the research question and the design question, such that the outcome of this study is relevant. Within both of those questions a general target has been proposed: “academics that lack affinity for software integration” and “academics not specialized in working on robots” respectively. Both already encompass a similar intersection of groups; specifically the niche within the “academics” cluster: ones that lack affinity for software integration AND are not specialized in working on robots.

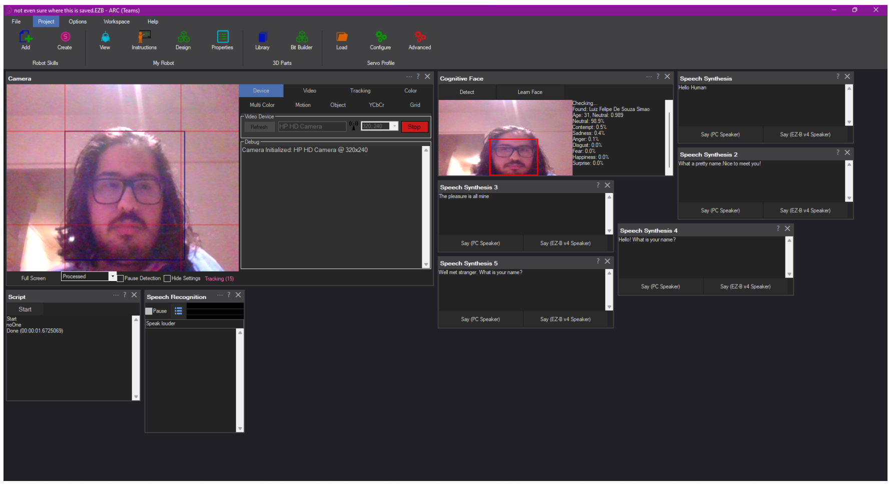
Dealing with the “academics” cluster is most straight forward. The study that will be designed will focus on testing students participating in a masters course within the University of Twente. However, since that University is a technical university, a big percentage of the students there will be either engineers that work with robots and the like or programmers. Thus an additional filter category will need to be put into place. The chosen approach for dealing with this will be to host the user test as an elective within the University of Twente’s international summer school. This event is designed for international students in the end of their bachelor or start of their masters. It is also designed to be able to introduce to these students a new prospective outside of their original studies. As such, the students that are participating in the summer school are encouraged to get outside of their comfort zone, thus, by making my user test an elective those students can choose from, it is highly likely that they will not already be comfortable with nor specialized in working on robots.

4) Specifying the Technique for Presenting the Software: Based on the limitation of hosting my user test as an elective in the international summer school of the University of Twente, there are few possible options for attempting to introduce a software and get the participants to actively attempt to reproduce the features that have been discussed. Luckily though, one of the few ways that would work for the limitation is also a viable option to achieve the desired goal [30]. This would be to make the user testing a workshop where the goal is to introduce the concept of prototyping social robots.

C. Specifications

In this section steps two and three will be put together and lead to the development of step four as described in the study plan. Additionally, this section relates to the second stage of the creative technology design process [24]. This implies the exploration of the design space. In the case of this research, the design space is the culmination of what was proposed in the ideation phase. Particularly, this means the exploration of how to design a workshop for the international summer school of the University of Twente which serves to instruct how to develop a prototype of a social robot using ARC.

1) Early Prototyping: In the first stages of this explorations, a selection of low fidelity prototypes were made. This was done with the intention of evaluating both the complexity of the task to be presented and the time it would take to achieve it. It is important to note that these low fidelity prototypes were primarily made by me and evaluated by me in discussion with other university students around me. The development of these prototypes were the result of three tinkering sessions done by myself [31]. For these sessions, I set a timer of one hour where I would be alone, equipped with: my laptop running the ARC software, paper and pens, and some Arduinos. Within this hour, a sample social robot that included some sort of the previously discussed features was made. After the



```
Camera Initialized: HP HD Camera @ 320x240
```

```
Start
noOne
Done (00:00:01.6725069)
```

```
Speak louder
```

```
Checking...
Found: Luiz Felipe De Souza Simao
Age: 31, Neutral: 0.989
Neutral: 98.9%
Contempt: 0.5%
Sadness: 0.4%
Anger: 0.1%
Disgust: 0.0%
Fear: 0.0%
Happiness: 0.0%
Surprise: 0.0%
```

Speech Synthesis: Hello Human

Speech Synthesis 2: What a pretty name.Nice to meet you!

Speech Synthesis 3: The pleasure is all mine

Speech Synthesis 4: Hello! What is your name?

Speech Synthesis 5: Well met stranger. What is your name?

Fig. 4. The sample social software used. The image shows the ARC software's desktop. Here it is possible to see what the implementation of face tracking, face recognition, speech synthesis, speech recognition, and scripting.

hour has passed, I would gather two or more colleagues and discuss with them about the prototype. Lastly, I would take the feedback obtained through the discussion, make myself some coffee, and attempt to restart, re-plan, and remake the sample prototype. During this second stage, the complexity to integrate and understand, and the time taken to implement were observed. The best sample social robot is presented in 4

2) *Pilot Test:* After the tinkering sessions, a plan for what to teach in the workshop had been made. Following the "cookbook and guidelines" approach to workshops, an instructional document was composed highlighting the steps to be taken (see appendix A) [30, pg. 72]. Within this document was the instructions to get the user up and running from the ground up. It instructed how to implement object detection, face tracking, face recognition, person recognition, speech synthesis, speech interpretation, speech to text, and AI. This selection of tasks were chosen as they covered most of the necessary features and was still deemed to not take too long to implement nor was was it deemed too complex.

The workshop will include a lecture, mini lesson, guided solution, introduction of bigger assignment, and finally propose students to work on the software themselves while still provide a few minutes for questions. This gives three reasons to pilot test the aforementioned document: (i) every step needs to be explained during the lecture times, thus knowing how much effort it takes to explain this is important for knowing if it fits in the time frame; (ii) it is important to know if a target user would find the task too uninteresting, either because it is too hard, take too long, or has a non-meaningful result [32]; (iii) to know if the text document to could be given to the target user and have it understood without supervision. As such, the pilot took place by having a student in the University of Twente that has finished his bachelor go through the document after a brief introduction of the software. It was not mentioned what they would eventually do with the software to see how they would react to the exercises. The amount of time the participant took to finish each task, and with it the total amount of time, was recorded. Lastly

Fig. 5. Here you can see the pilot test taking place. The user is comfortably sitting down. At the moment, looking through the ARC software's skill library (See in the screen of the laptop) in the process of running the pilot.

it was requested of the participant to think out loud what they were thinking, and I was in the same room listening and taking subtle notes.

D. Realisation and Evaluation

The pilot took place in a comfortable environment with an emphasis on a relaxed state given that the the document would be intended to be used as a follow-up to the workshop where the students would be asked to follow it in their own time. I did little to no interventions while the participant was calmly going through the document and resolving the tasks on a laptop on their lap as represented in figure 5.

During the pilot testing, it was found that it took the participant around forty minutes to fully complete the assigned task, which is too long for the amount of time that has been allocated to me for the international summer camp. Thus, that is the first point that needs to be adjusted for the user test. There were a some bits of text that seemed confusing or distracting from the activity in the document that were noted from the participant thinking out loud. Some of them will be changed to avoid possible ambiguity that could arise. However, not all text will be changed as some of the confusions that were noticed will serve as changes to the lecture moments of the workshop. There were quite significant signs that the participant felt stimulated to tinker with the UI of the software. This did slow down the first few steps, but did make the experience seem more interesting, and eventually when the participant returned to following the tasks, they seemed much more comfortable and efficient. The participant did seem to struggle with the coding element of the assigned task, seen by the fact that this is the task that took longest for them to solve (approximately nine minutes). Partially because the program presented multiple possible languages to code in and partially because the code that was recommended to the participant to write seemed unintuitive. It was noticed, however, that after the participant made a choice of language and figured out how and why to use the code, they seemed to quickly grasp the idiosyncrasies of programming in ARC. Despite this realisation, and based on the strict time element, this task was simplified for the final workshop.

IV. Realisation

This chapter is separated into two parts. First will be resolving the final step of the study plan wherein the conclusions from having the pilot test will be put together to produce the final user test. Second the presentation of the results of conducting the user test.

A. Presenting the Final Design

The general structure of the workshop did not change after the pilot test. It will still be presented in the order: "a lecture, mini lesson, guided solution, introduction of bigger assignment, and finally propose students to work on the software themselves while still provide a few minutes for questions" as described in section III-C2 (page 26). These sections need to be split and molded to fit under the regulations proposed by the international summer school.

1) Regulations to Follow: As the workshop was planned to be hosted during the international summer school of the University of Twente, some rules had to be followed and limitations applied due to logistics. The organization of the summer school offered me two classes of one hour for the same group of a maximum of twelve students. Due to the COVID-19 pandemic, the summer school chose to host the entire event online, thus my workshop would also have to comply to a fully remote operation. The registration and enrolment of the students would have to be entirely handled by the summer school organisation. The registered students' information would be kept anonymous to me with the exception of a few minimal demographic information and their first names.

Given the need for two sessions, the original piloted content was edited. It was both split into two sections based on content similarities and had some additional content added (to be addressed later).

2) The Lecture and Mini-Lesson: In the lecture element of the first session of the workshop a brief explanation of what will be presented in the workshop to prime the class on what to expect. After, an introduction of what is a social robot will be given. First, the relevance of form and how to isolate and prototype different aspects of form. Then the concept of context and how to isolate and prototype different aspects of social robots in context. Last will be presented the concept of features. The students will then be told that they would have to come up with different features ways to prototype for features. At this stage the students would be made ready to discuss this among themselves.

The second session's lecture will discuss the use and importance of an interface. Additionally, the more complex topic of the social interface will be introduced. After the introduction, the lecture will be directed towards how to turn a standard interface into a social one. The mini-lesson will be introduced as a challenge to the students to design a GUI into something that could be considered a social interface.

The lecture element would end in the twentieth minute in the workshop's dedicated time. This is chosen to account for any slowdowns that could happen during the start. The mini-lesson would be set to take five minutes.

3) Guided Solution: After the five minutes have passed, I will join in on the discussion with the hope to redirect the answers towards the next topic if the necessary elements have not been brought up already. in the approach of doing this, I will also attempt to get the students to maintain in the discussion by asking them to think a little bit more critically about what they are talking about. This guided discussion will be kept going until the thirty-fifth minute of the workshop.

4) End of the Workshop session: From the thirty-fifth to the fiftieth minute I will introduce how and what the assigned task for after the workshop would be. In the first session I will go over what is ARC and how to search, add, and edit skills. I then proceed to explain in more detail the speech recognition skill. In the second session, this part will be more focused in explaining the slightly more advanced skill used to make the gaze behaviour-like GUI. This includes a quick demonstration of how to quickly generate the avatar using a collection of PNGs and how to use the full screen video player skill to display these images.

With the aim of ending the class with some time for the students to already start on the assignment while still online, some closing words where said at around the fiftieth minute. The students were then given a PDF of the session's assignment. Both sessions' assignment document have been put together and presented in the appendix.

B. Results

During the realisation of the workshop ten students enrolled and six participated. Of the participants three were female and three were male. Throughout the class there were quite active students that had not worked with the subject before but were very engaged and willing to try their luck. It was clear that in general the students felt quite comfortable with admitting to not knowing how to prototype social robots. Despite this, the general environment led them to be able to effectively answer the challenges proposed during the mini-lessons. This is based on the fact that in the five minutes where the students would try to answer without my help, they managed to answer what I needed to proceed with the lessons, not needing me to complement their answers during the guided solution and leaving time to further the discussion. This also led to the opportunity to further their critical thinking on the subject. It was quite visible that there was an appreciation of this, based on some comments left on the chat. Overall there was general appreciation of the course. I was made aware that during the general survey about the international summer camp, two comments from two different students mention the workshop positively. One calling the workshop "one of their favorite parts" and the other saying "Furthermore, before attending this Summer School, I've never had the opportunity to learn about robotics, so I really enjoyed the Designing a Social Interface workshop."

V. Conclusion and Discussion

In this section the design and the evaluation question will be presented and an answer to both will be given. After that, the answers will be put together and used to conclude the research question. Lastly, contributions, limitations, and future work will be presented.

A. Answering the design question

The proposed design question this study attempted to solve was *How can a stimulating environment for the introduction of a software toolkit to researchers not specialized on working on robots be designed?*

To answer this question, a design study was conducted and it was found that there are consumer software designed with the intent of helping users of all sorts of levels in relation to technology achieve all sorts of designed. It is possible that because of this, the chosen software for conducting the study with, ARC, seemed to be able to be picked up in less than an hour by the six participants of the workshop, the participant of the pilot, and myself. I believe that this shows that there is incentive for companies to put in a lot of effort into making a tool that is extremely efficient. Again in relation to ARC, the platform supports very high level features, and with the integration of python and community supplied "skills" it is possible to create many different topical environments worth its introduction. Sadly though, it was also seen that it is more beneficial for a company to develop software, only after they have developed and are selling hardware already.

B. Answering the evaluation question

The proposed evaluation question this study attempted to solve was *How can the introduction to the software toolkit be shown to be helpful, and how can it be shown to be a hindrance?* After going through with the user test, and evaluating the workshop and the response to the workshop, some benefits can be seen with introducing ARC to the target audience. Primarily, the fact that ARC is a cheaply available platform that works well with third-party applications provides a great research potential to researchers on a lower budget. This is very important as it opens way for researchers that would otherwise be constrained by their budget to be able to take on the challenge of researching on social robots.

Additionally, it was found during the design study that the platform does offer the required materials to pick it up be conducting research with it in a short amount of time. This is something that, in the field of iterative design, can be seen as a huge benefit. It allows for more prototypes and tests to be conducted in a tighter time frame.

However, this platform does have hindrances that need to be considered. Primarily, it is a replacement for a more complex and engineered solution. This is mostly to design, but it should not be forgotten. By this, it is implied that, since prototypes can be made fast and tested fast, there is also the possibility that a product can be made with this stage of functionality as its final. This would create a market of not optimized or less capable social robots, diminishing the benefits of increasing the research in the field. From another perspective, it is possible to imagine that a selection of researchers (otherwise not in charge of robots and programming) could develop a dependency on the software and its large (but finite) set of skills after being presented with it.

C. Answering the research question

This research presented the question *To what extent can the introduction of software toolkits help researchers that lack affinity for software integration achieve better prototypes for research and design of social robots?*

Today there are currently available software toolkits that can effectively benefit researchers and designers that lack affinity for software integration achieve higher fidelity prototypes. It was found that the platform ARC by the company Synthiam is an example of such a toolkit. The benefits of which, are not limited to, a decreased cost of software integration, low threshold, and increase in development speed. However, it was also found that the utility of consumer available software tends to be context sensitive and have a limited range of prototype fidelity. Because of this, the introduction of software for the means of prototyping has to be done in very catered manner to minimize the dependence that it could create, possibly worsening the prototypes made.

D. Contribution, Limitations, and Future Work

a) Contribution: This research highlighted a field where there is a lack of publications taking into consideration. That being, the gap in accessibility for non-technical researchers to conduct studies on social robots. It also attempted to provide an alternative to mitigate the problem such that it does not become bigger. It has done this by shedding light on possible commercially available solutions and demonstrating how they can have significant impacts in future research.

b) Limitations: It is necessary to be aware, though, that this research is exploratory in nature and due to multiple limitations, lack in depth. An example of such limitation is the necessity to make the research be able to take place remotely. In terms of output of this research, it is also important to keep in mind when reading the answers that the pool of participants was very limited due to the reliance on the summer school.

c) Future Work: The limitations mentioned above do open ways for future work. Particularly, it is believed that a study conducted on a wider scale and with a more targeted audience could be used to produce a lot of insights in the limitations faced by non-technically inclined academics in the modern days.

References

[1] 360iResearch, “Social Robots Market Size, Share, Growth and Trends Analysis Report By Type, By Application, and Segment Forecasts, 2019-2025,” Tech. Rep., 2021.

[2] R. Campa, “The Rise of Social Robots: A Review of the Recent Literature,” *Journal of Evolution & Technology*, vol. 26, no. 1, pp. 106–113, Feb. 2016. [Online]. Available: https://www.researchgate.net/publication/308787576_The_Rise_of_Social_Robots_A_Review_of_the_Recent_Literature.

[3] L. Royakkers and R. van Est, “A Literature Review on New Robotics: Automation from Love to War,” *International Journal of Social Robotics*, vol. 7, no. 5, pp. 549–570, 2015, ISSN: 18754805. DOI: 10.1007/s12369-015-0295-x. [Online]. Available: http://dx.doi.org/10.1007/s12369-015-0295-x.

[4] W. Q. Koh, S. A. Felding, K. B. Budak, E. Toomey, and D. Casey, “Barriers and facilitators to the implementation of social robots for older adults and people with dementia: a scoping review,” *BMC Geriatrics*, vol. 21, no. 1, p. 351, 2021, ISSN: 1471-2318. DOI: 10.1186/s12877-021-02277-9. [Online]. Available: https://doi.org/10.1186/s12877-021-02277-9.

[5] A. Henschel, G. Laban, and E. S. Cross, “What Makes a Robot Social? A Review of Social Robots from Science Fiction to a Home or Hospital Near You,” *Current Robotics Reports*, vol. 2, no. 1, pp. 9–19, 2021, ISSN: 2662-4087. DOI: 10.1007/s43154-020-00035-0. [Online]. Available: https://doi.org/10.1007/s43154-020-00035-0.

[6] A. Darriba Frederiks, J. R. Octavia, C. Vandevelde, and J. Saldien, “Towards Participatory Design of Social Robots,” in *Human-Computer Interaction – INTERACT 2019*, D. Lamas, F. Loizides, L. Nacke, H. Petrie, M. Winckler, and P. Zaphiris, Eds., Cham: Springer International Publishing, 2019, pp. 527–535, ISBN: 978-3-030-29384-0.

[7] C. Vandevelde and J. Saldien, “An open platform for the design of social robot embodiments for face-to-face communication,” in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Mar. 2016, pp. 287–294. DOI: 10.1109/HRI.2016.7451764.

[8] C. Breazeal, K. Dautenhahn, and T. Kanda, “72. Social Robotics,” in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2nd ed., Berlin: Springer Nature, 2016, ch. 72, pp. 1935–1972, ISBN: 978-3-319-32550-7. DOI: 10.1007/978-3-319-32552-1. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2F978-3-319-32552-1.pdf.

[9] F. Hegel, C. Muhl, B. Wrede, M. Hielscher-Fastabend, and G. Sagerer, *Understanding Social Robots*. Feb. 2009, pp. 169–174. DOI: 10.1109/ACHI.2009.51.

[10] E. B. Onyeulo and V. Gandhi, “information What Makes a Social Robot Good at Interacting with Humans?,” DOI: 10.3390/info11010043. [Online]. Available: www.mdpi.com/journal/information.

[11] J. Ortiz Nicolás, "Understanding the Context of Design for Social Innovations: A Methodological Case Study," in *Handbook of Research on Ergonomics and Product Design*, Mar. 2018. DOI: 10.4018/978-1-5225-5234-5.ch017.

[12] S. Greenberg, *Prototyping for Design and Evaluation*, 1998. [Online]. Available: http://grouplab.cpsc.ucalgary.ca/saul/681/1998/prototyping/survey.html#techniques.

[13] J. Sauer, H. Franke, and B. Ruettinger, "Designing interactive consumer products: Utility of paper prototypes and effectiveness of enhanced control labelling," *Applied Ergonomics*, vol. 39, no. 1, pp. 71–85, 2008, ISSN: 0003-6870. DOI: https://doi.org/10.1016/j.apergo.2007.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003687007000245.

[14] K. Pollmann, C. Ruff, K. Vetter, and G. Zimmermann, "Robot vs. voice assistant: Is playing with pepper more fun than playing with alexa?" *ACM/IEEE International Conference on Human-Robot Interaction*, pp. 395–397, 2020, ISSN: 21672148. DOI: 10.1145/3371382.3378251.

[15] M. Al-Sada, K. Jiang, S. Ranade, M. Kalkattawi, and T. Nakajima, "HapticSnakes: multi-haptic feedback wearable robots for immersive virtual reality," *Virtual Reality*, 2019, ISSN: 14349957. DOI: 10.1007/s10055-019-00404-x. [Online]. Available: https://doi.org/10.1007/s10055-019-00404-x.

[16] W. S. Helton, J. Head, and B. A. Blaschke, "Cornering law: The difficulty of negotiating corners with an unmanned ground vehicle," *Human Factors*, vol. 56, no. 2, pp. 392–402, 2014, ISSN: 00187208. DOI: 10.1177/0018720813490952.

[17] D. Steffen, B. Buerdek, V. Fischer, and J. Gros, *Design als Produktsprache: Der "Offenbacher Ansatz" in Theorie und Praxis*. Jan. 2000.

[18] A. W. Staats, "'Behavioural interaction' and 'interactional psychology' theories of personality: Similarities, differences, and the need for unification," *British Journal of Psychology*, vol. 71, no. 2, pp. 205–220, May 1980, ISSN: 0007-1269. DOI: https://doi.org/10.1111/j.2044-8295.1980.tb01738.x. [Online]. Available: https://doi.org/10.1111/j.2044-8295.1980.tb01738.x.

[19] M. Blancas, C. Valero, V. Vouloutsi, A. Mura, and P. Verschure, "Educational Robotics: A Journey, Not a Destination," in Dec. 2020, ISBN: 9781799867173.

[20] E. R. De Nadai Victal and A. P. Candido, "Learning programming with robotics using arduino: Practice and interdisciplinarity," in *Proceedings - 2019 Latin American Robotics Symposium, 2019 Brazilian Symposium on Robotics and 2019 Workshop on Robotics in Education, LARS/SBR/WRE 2019*, Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 498–503, ISBN: 9781728142685. DOI: 10.1109/LARS-SBR-WRE48964.2019.00094.

[21] M. Resnick, J. Maloney, A. Monroy Hernández, *et al.*, "Scratch: Programming for Everyone," *Communications of the ACM*, vol. 52. Pp. 60–67, Nov. 2009. [Online]. Available: http://scratch.mit.edu.

[22] *Different Types of Robot Programming Languages — Plant Automation Technology*. [Online]. Available: https://www.plantautomation-technology.com/articles/different-types-of-robot-programming-languages.

[23] C. Giang, A. Piatti, and F. Mondada, “Heuristics for the Development and Evaluation of Educational Robotics Systems,” *IEEE Transactions on Education*, vol. 62, no. 4, pp. 278–287, Nov. 2019, ISSN: 15579638. DOI: 10.1109/TE.2019.2912351.

[24] A. H. Mader and W. Eggink, “A Design Process for Creative Technology,” Undefined, in *Proceedings of the 16th International conference on Engineering and Product Design, E&PDE 2014*, E. Bohemia, A. Eger, W. Eggink, A. Kovacevic, B. Parkinson, and W. Wits, Eds., ser. E&amp;PDE, The Design Society, Sep. 2014, pp. 568–573, ISBN: 978-1-904670-56-8.

[25] B. Myers, S. Hudson, and R. Pausch, “Past, Present, and Future of User Interface Software Tools,” *ACM Trans. Comput.-Hum. Interact.*, vol. 7, pp. 3–28, Oct. 2000. DOI: 10.1145/344949.344959.

[26] T. Chandrasekera and S.-Y. Yoon, “The Effect of Tangible User Interfaces on Cognitive Load in the Creative Design Process,” Oct. 2015, pp. 6–8. DOI: 10.1109/ISMAR-MASHD.2015.18.

[27] C. Yee, C. Ling, W. Yee, and W. M. N. Zainon, “GUI design based on cognitive psychology: Theoretical, empirical and practical approaches,” in *Proceedings - 2012 8th International Conference on Computing Technology and Information Management, ICCM 2012*, vol. 2, Oct. 2012, pp. 836–841, ISBN: 978-1-4673-0893-9.

[28] J.-W. Chen and J. Zhang, “Comparing Text-based and Graphic User Interfaces for Novice and Expert Users,” *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, vol. 11, pp. 125–129, Oct. 2007.

[29] M. Marge, C. Espy-Wilson, N. G. Ward, *et al.*, “Spoken language interaction with robots: Recommendations for future research,” *Computer Speech & Language*, vol. 71, p. 101 255, 2022, ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2021.101255. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0885230821000620.

[30] R. Ørngreen and K. Levinsen, “Workshops as a research methodology,” *Electronic Journal of e-Learning*, vol. 15, pp. 70–81, Oct. 2017.

[31] A. H. Mader and E. C. Dertien, “Tinkering as Method in Academic Teaching,” English, in *DS 83: Proceedings of the 18th International Conference on Engineering and Product Design Education*, E. Bohemia, Ed., Sep. 2016, pp. 240–245.

[32] J. Eastwood, A. Frischen, M. Fenske, and D. Smilek, “The Unengaged Mind: Defining Boredom in Terms of Attention,” *Perspectives on Psychological Science*, vol. 7, pp. 482–495, Nov. 2012. DOI: 10.1177/1745691612456044.

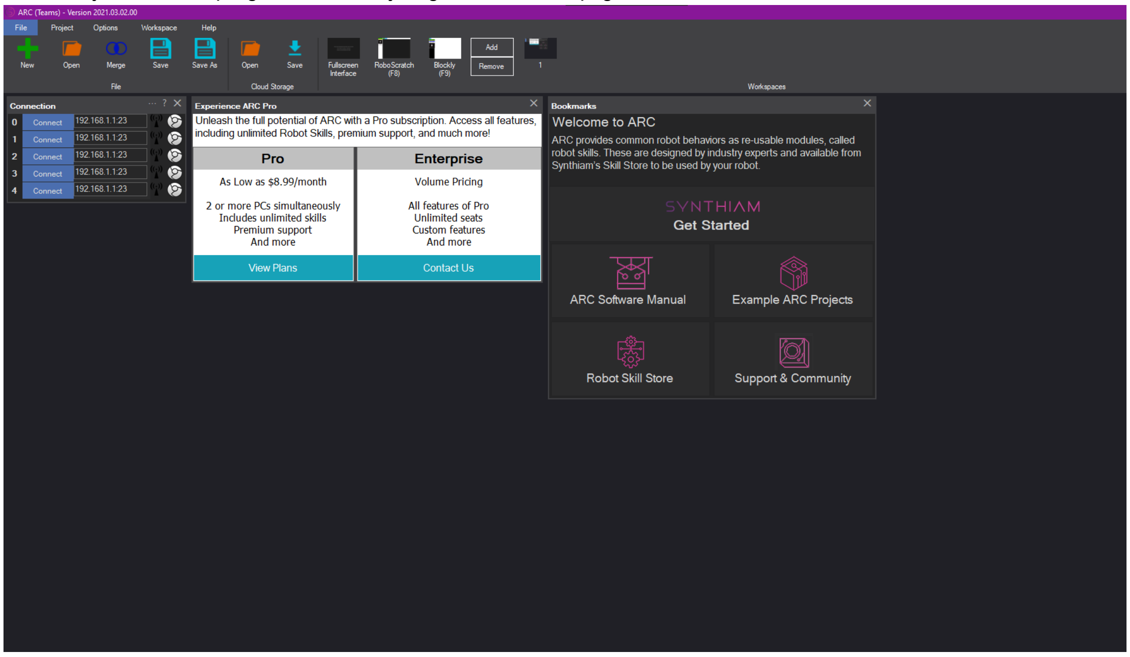
Appendix A

Full tutorial for ARC given in the workshop

A workshop on designing a social interface by Luiz Felipe de Souza Simão

Installation guide

- The first step in the process of making a quick and dirty social robot is going to be getting the platform that is going to be "pulling strings." In this case, we are going to be using a platform called ARC [by Synthiam].
    a. To install this software, start by clicking here (note this installation only really works on Windows computers, for it to run on Mac, you are going to have to take the few extra steps required by your favorite way of emulating Windows apps).
    b. When you click on the link, you are going to need to scroll a bit down and get the free version of the app. You are going to need the *Teams* variant of the program.
    c. When you finish downloading the app, run the executable (click the *.exe* file you just downloaded) and go through the installation process.
        i. For the program to work, you are going to need to specify a preferred coding language. This step is not as important as it may look. For this tutorial, I will be using *Blockly* language, so maybe it is best to select that now.
        ii. For the software to work, you are going to need an account.
- Now that you have the program installed, you get to the home page that should look somewhat like this:



Unleash the full potential of ARC with a Pro subscription. Access all features, including unlimited Robot Skills, premium support, and much more!

Pro	Enterprise
As Low as $8.99/month	Volume Pricing
2 or more PCs simultaneously Includes unlimited skills Premium support And more	All features of Pro Unlimited seats Custom features And more
View Plans	Contact Us

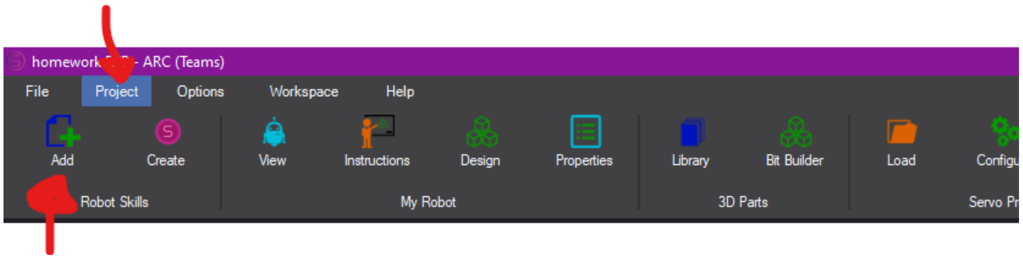
Welcome to ARC

ARC provides common robot behaviors as re-usable modules, called robot skills. These are designed by industry experts and available from Synthiam's Skill Store to be used by your robot.

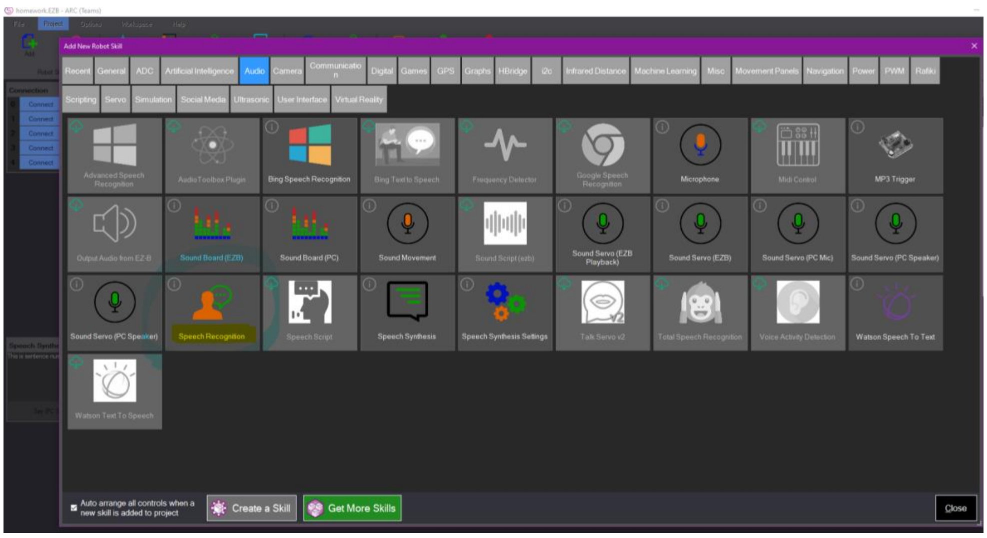
    a. This is the most important page of the program, and it acts as a central hub where you, the developer, have access and vision of everything that is going on in the "brain" of your social robot. On that note, everything here is going to be represented in terms of these windows that you now see (e.g., Connection, …). Here you are encouraged to add and remove things as you wish – if they are there, that means your robots will consider them; if they are not, then your robot will not.
    b. Note that I say *consider* and not *use*; for it to be used, it needs to have some active and used part to the window, else the robot will just ignore it. We shall get to understanding the active elements of the windows later in the tutorial.
    c. Feel free to remove (done so by clicking the *X* next to the question mark in the top right corner) any of the windows that come open by default as they are mostly useless for this tutorial. Any window you put into the central hub that you later no longer feel like you want can be removed in the same way. Be wary that there is no undo to this action, but you can always just add the window back in a glance.

Making a speaking robot

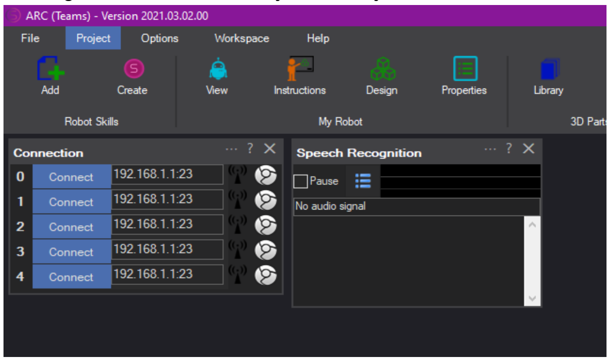
- For this tutorial, we are going to make a simple conversation agent. This interaction only relies on using your laptop (or desktop if you have a functioning webcam and microphone). The first step to have an agent converse is the ability to listen.
  a. In terms of robots, that means that we need to set up some sort of speech recognition. This sounds complicated, but it is as simple as a few clicks with ARC. That is because there is a "window" that does the whole speech recognition for you. We just need to find it.
  b. In ARC, the thing that I have been calling a "window" is called "a skill" for that purpose I will be calling it that from now on. To get the speech recognition skill, you are going to have to go to "Project → Robot skill → Add."



  c. Clicking this page should have opened a library separated by tabs of possible "skills" that you can add. For now, you want something in the likes of a speech recognition agent. Under the tab for audio-based skills, you might find something that suits your needs. I went with the "Speech Recognition" skill for this tutorial.



  d. Clicking it should automatically add it to your hub

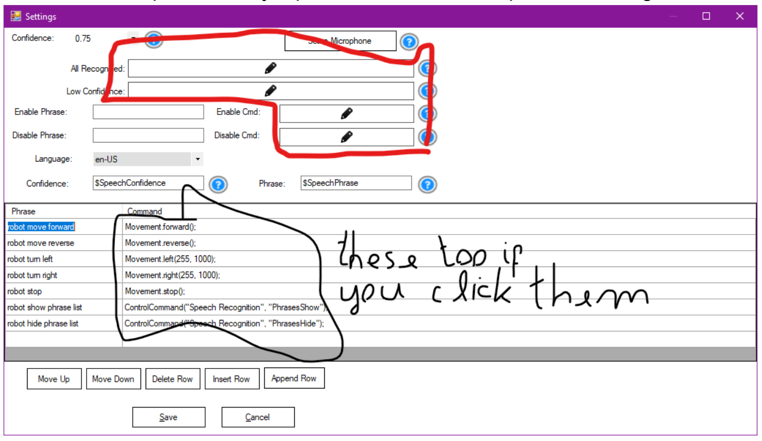


0	Connect	192.168.1.1:23
1	Connect	192.168.1.1:23
2	Connect	192.168.1.1:23
3	Connect	192.168.1.1:23
4	Connect	192.168.1.1:23

No audio signal

  e. To figure out what you can do with this skill and any other skill, you must click the "…" button. This should open a popup window titled "Setting" with all sorts of things. Here you are introduced to the inner workings of ARC. Most skills in ARC come with some *functions* and with some [global] *variables,* and this one is no different.

i. Functions are represented by squares that have a little pencil-like thing in the middle



Settings

Confidence: 0.75 | Select Microphone

All Recognized:

Low Confidence:

Enable Phrase: | Enable Cmd:

Disable Phrase: | Disable Cmd:

Language: en-US

Confidence: $SpeechConfidence | Phrase: $SpeechPhrase

Phrase	Command
robot move forward	Movement.forward();
robot move reverse	Movement.reverse();
robot turn left	Movement.left(255, 1000);
robot turn right	Movement.right(255, 1000);
robot stop	Movement.stop();
robot show phrase list	ControlCommand("Speech Recognition", "PhrasesShow");
robot hide phrase list	ControlCommand("Speech Recognition", "PhrasesHide");

ii. These functions are snippets of code that run when the condition defined by the skill is met. <u>Clicking them opens up the code editor, which will be addressed later.</u>

iii. Variables are sometimes not directly visible here, but when they are, it means that you can name them, and the skill will create and update this variable with whatever utility it might have. The variables are always created and named automatically with a placeholder name (e.g., $SpeechConfidence).

f. In this skill, you can assign phrases, and the program will do the best it can to detect those. Each phrase also comes with a function that is triggered whenever that phrase is detected with enough *confidence* (if the program is not confident enough that it heard the correct text, it will report that and ignore whatever it heard). All these pre-existing phrases seem to not be relevant for this project, so it is nice to delete them to not risk the algorithm detecting them by accident.

i. It is possible to delete any row you select by clicking the “Delete Row” button. The action of deleting the row cannot be directly reversed. This is not a big issue though, if you delete the wrong thing, then you can always click the “Cancel” button on the bottom of the page. The only issue here is that it will revert all unchanged settings.

ii. Optionally (if you chose to ignore this step, continue to step *g.*), once you are done deleting the unnecessary phrases, you can click the “Save” button. This will save all edits to the settings and close the page. <u>If you close the page without saving, all changes will be reverted</u>.

iii. Now that you have saved the settings return to them by, again, clicking the “…” button.

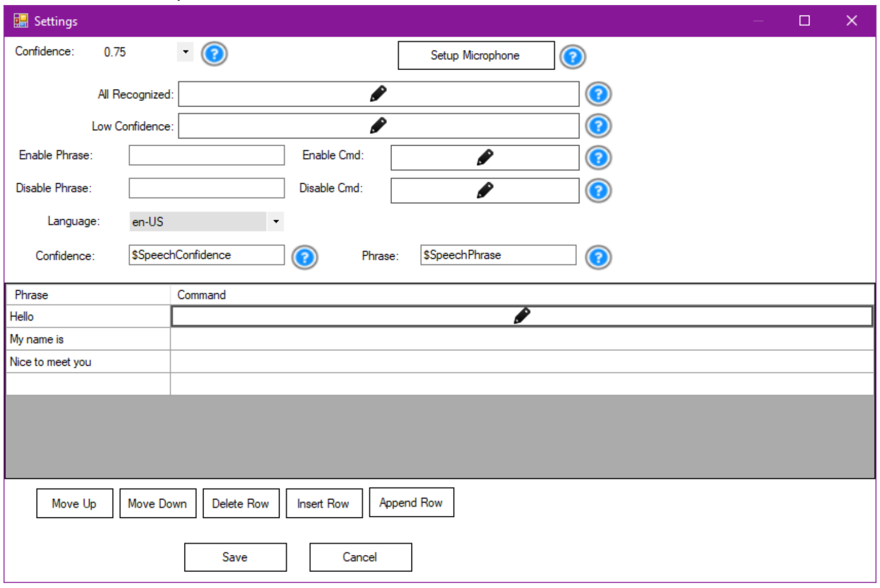
g. Now let us teach the software what we are interested in listening to.

i. Feel free to add any handful of conversational queues and phrases for the skill to listen to. This can be done by clicking the blank cell under “Phrase” and just typing.

ii. Make sure to include in the list of phrases (that you are listening to) the phrases:

- Hello
- My name is
- Nice to meet you

- As these will be important for the rest of the tutorial.



Confidence: 0.75

All Recognized:

Low Confidence:

Enable Phrase: Enable Cmd:

Disable Phrase: Disable Cmd:

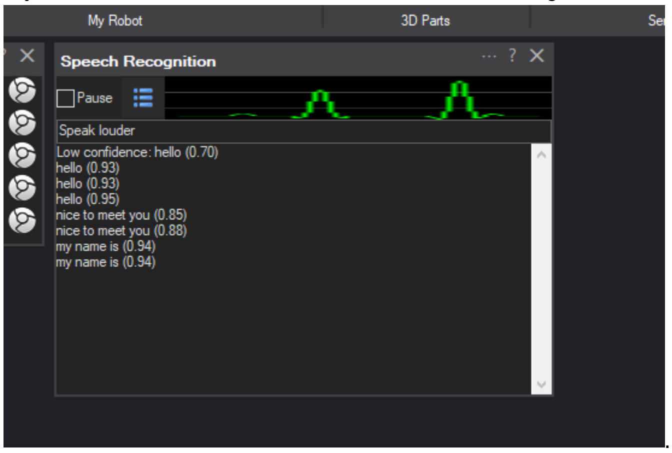
Language: en-US

Confidence: $SpeechConfidence Phrase: $SpeechPhrase

Phrase	Command
Hello	
My name is	
Nice to meet you	

iii. Now let us test if the skill can actually hear these phrases. Click save and return to the hub.

iv. In the window of the skill, you should see some squiggly red or green curves moving next to the "Pause" button. Click the pause check box if the line is red to make them green; if it is already green, then your "social robot" is already capable of hearing you. Try saying all the phrases included in your list and see if you get that the dialog box to spell it out. That means it heard you correctly (inversely, if it says low confidence, it means that the skill heard something, but it is not sure what it is)

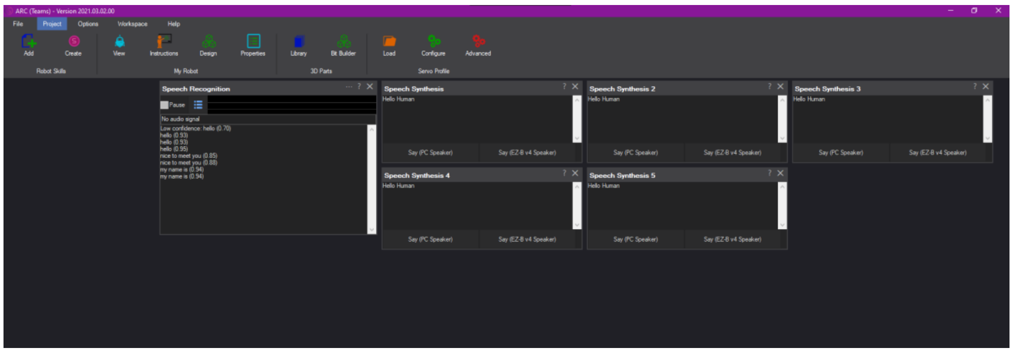


```
Speak louder
Low confidence: hello (0.70)
hello (0.93)
hello (0.93)
hello (0.95)
nice to meet you (0.85)
nice to meet you (0.88)
my name is (0.94)
my name is (0.94)
```

.

v. If you seem to not get the squiggly line and it says no audio signal detected, check that your microphone is on correctly and is not muted.

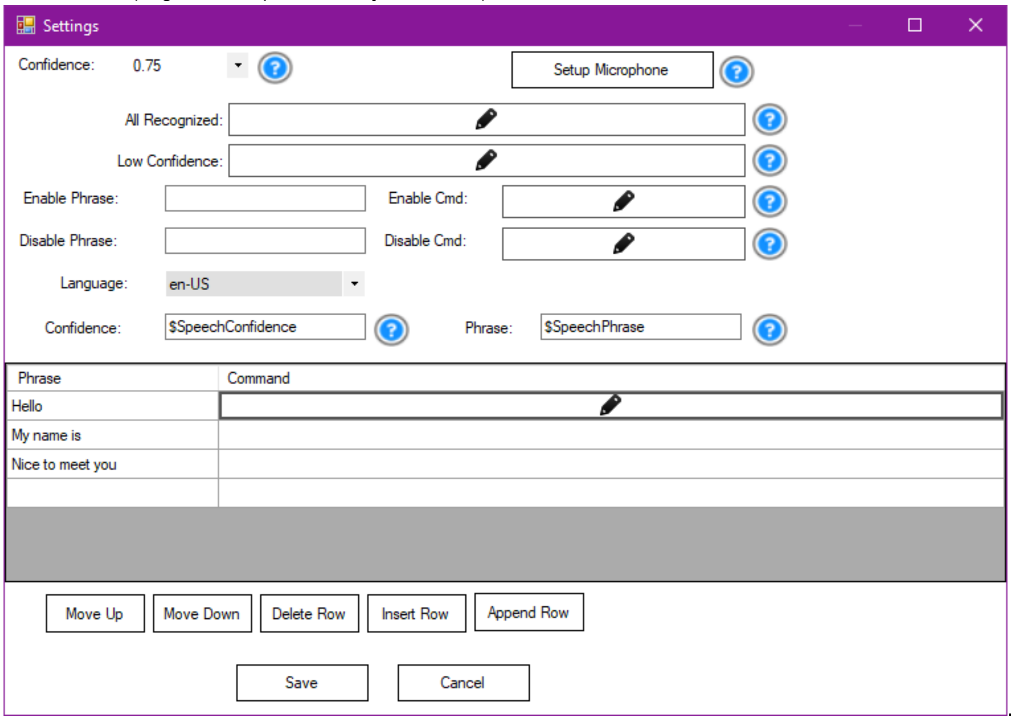
- Now that your "social robot" can hear you, we need to let it be able to respond to you. For this, I recommend using the skill "Speech Synthesis." This skill is a simple (as in, it does not even have a settings page) mediator that lets you preload a sentence and play it through the speaker.
    a. Let us connect these sentences to the phrases such that they can be used as responses. First, we will need to have at least one sentence per phrase, so let us add more of the same skill. Feel free to add as much as you would like ~~(I think there is a skill limit since we are using the free option, but I don't really know)~~. I personally used five.



```
No audio signal
Low confidence: hello (0.70)
hello (0.93)
hello (0.93)
hello (0.95)
nice to meet you (0.85)
nice to meet you (0.88)
my name is (0.94)
my name is (0.94)
```

Hello Human

b. The next step is to link the sentences to the dedicated phrase. For this, we are going to use the functions that the "Speech Recognition" skill created. Go back to its settings and open the code editor for "Hello" by clicking the blank cell (might have a pencil-like symbol there) under "Command"



Confidence: 0.75

All Recognized:

Low Confidence:

Enable Phrase: Enable Cmd:

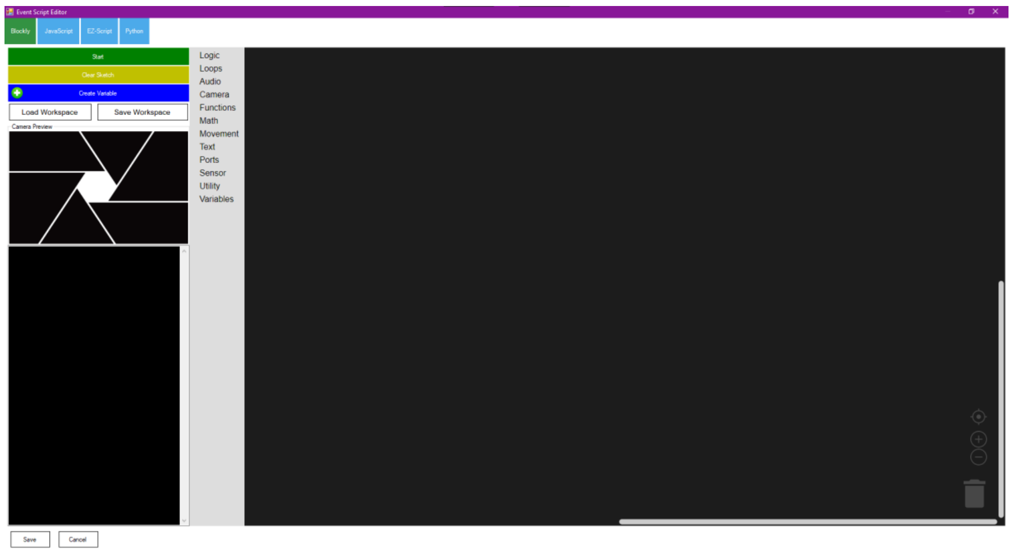
Disable Phrase: Disable Cmd:

Language: en-US

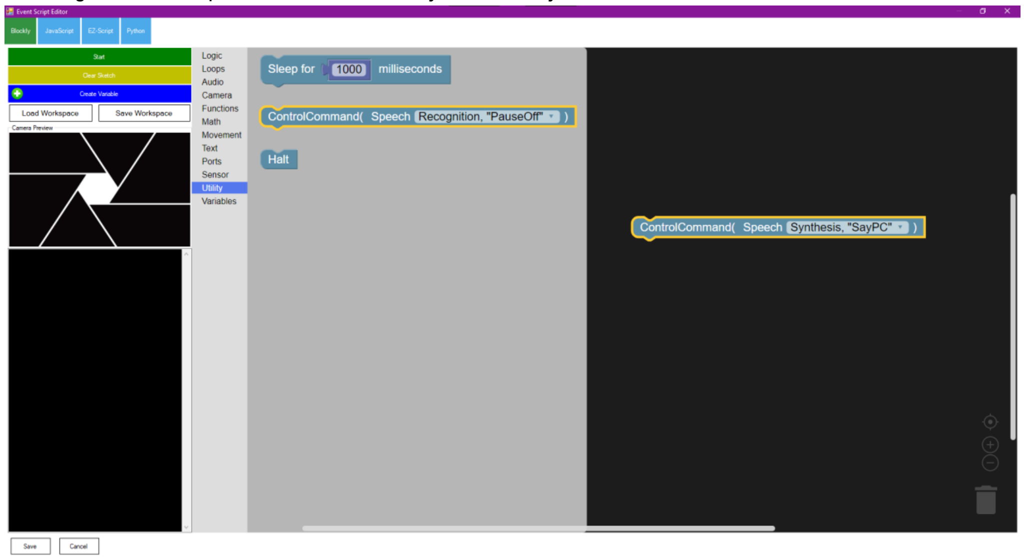
Confidence: $SpeechConfidence Phrase: $SpeechPhrase

Phrase	Command
Hello	
My name is	
Nice to meet you	

c. Here in the code editor, you can see a lot of things are going on. On the top left corner, you might recognize that the editor can take in four different programming languages: Blockly, JavaScript, EZ-Script, and Python. For simplicity, this tutorial will focus on the quick and dirty Blockly language, but feel free to use any of the other languages if you feel more comfortable with them. Each language has its own glossary of predefined functions on the right side of the screen, which should be analogous to what I do in Blockly. Moving forward, if you are not in the Blockly editor, click on the "Blockly" button on the top left, which should hopefully take you to this page

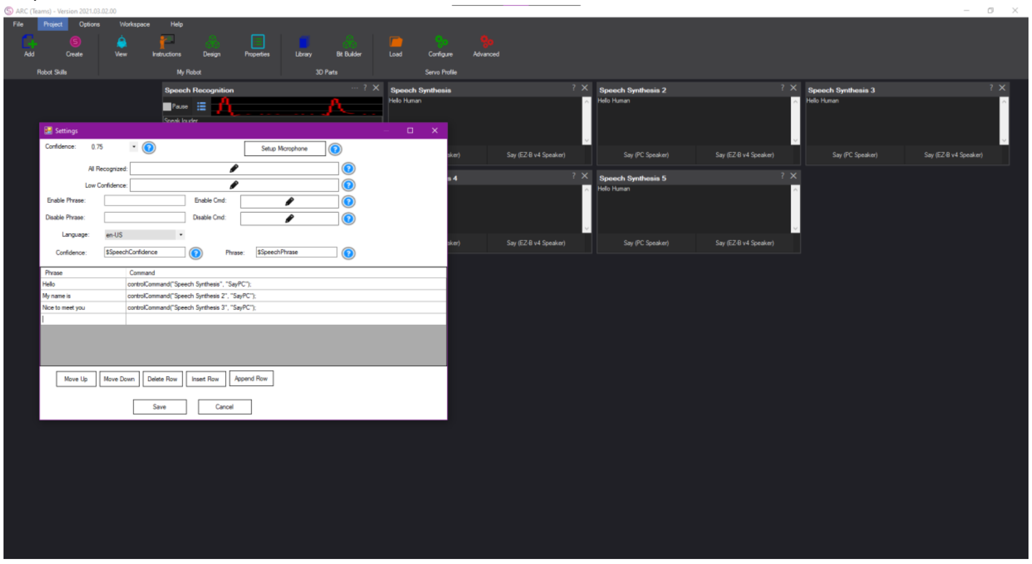


d. Since we got here through the function of the "Hello" phrase, we should be aware that whatever code we put here will be called as soon as the skill detects that phrase. As such, it might be best to avoid any loops here. So far, all we really want is to activate one of the sentences. For that, all you are going to need to run is one of the "Control commands." These refer to actions that all skills in your hub can have. We want our function to call the skill "Speech Synthesis," and we want to tell it to say its sentence through the PC speaker. In Blockly, this is done simply by dragging the "ControlCommand()" block from the "Utility" section into the main work area and selecting from the drop-down of that block: "*Synthesis, "SayPC*."



e. Now you should be able to save your edit, and it will return you to the Settings window, where the cell that you just edited will probably be highlighted green. That means that the edit was saved. Here is a perfect moment to save the setting edits as well and test if – when you say hello, that the sentence in *SpeechSynthesis* (specifically the not numbered one [assuming you have multiple]) is called (make sure that you have an active speaker on your PC).

f. If things seem all right, now is the time that you repeat the steps above until all your phrases have at least one response.



Confidence: 0.75

All Recognized:

Low Confidence:

Enable Phrase: Enable Cmd:

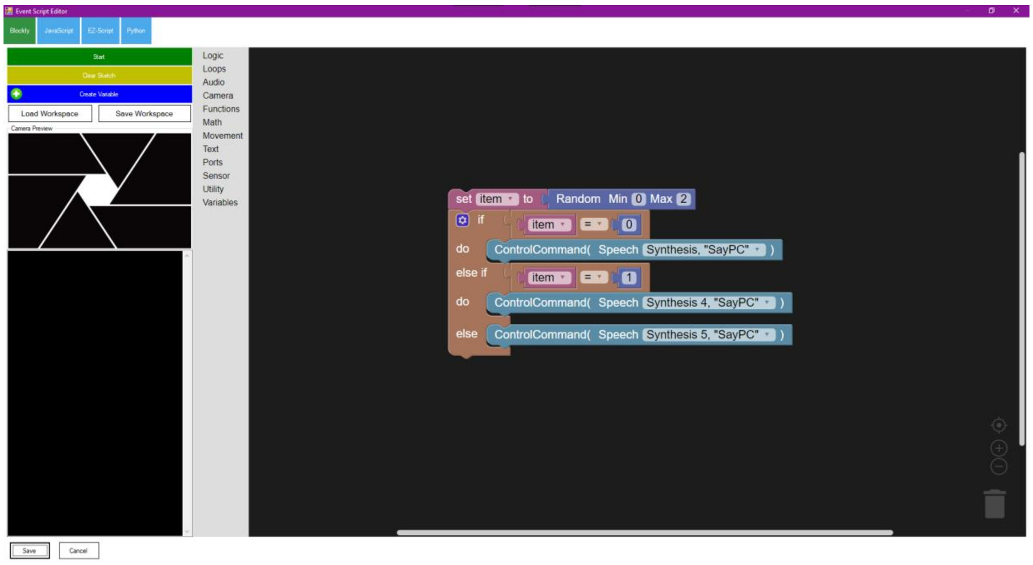
Disable Phrase: Disable Cmd:

Language: en-US

Confidence: $SpeechConfidence Phrase: $SpeechPhrase

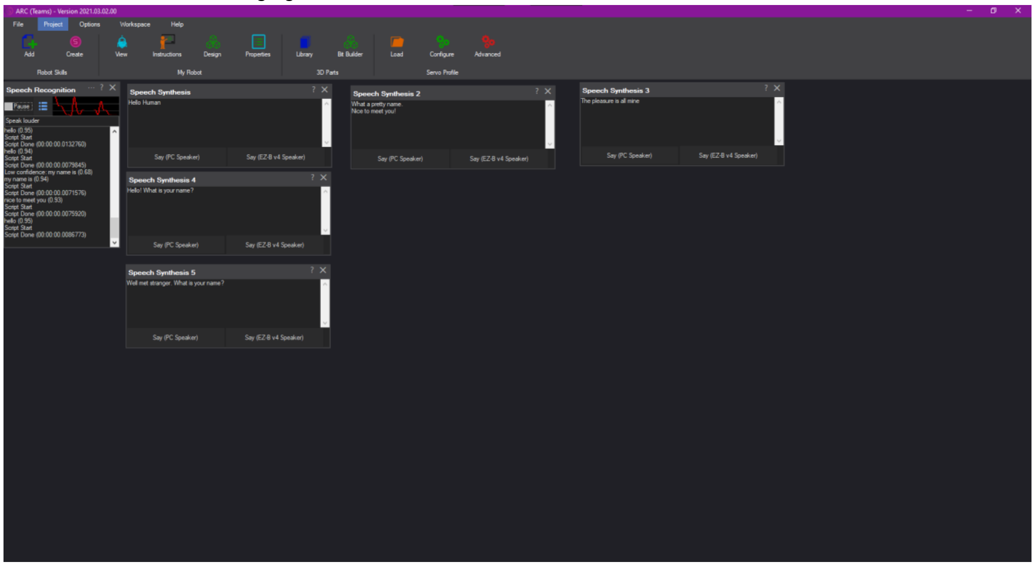
Phrase	Command
Hello	controlCommand("Speech Synthesis", "SayPC");
My name is	controlCommand("Speech Synthesis 2", "SayPC");
Nice to meet you	controlCommand("Speech Synthesis 3", "SayPC");

g. For some extra credit, you can always try to make the social robot switch between responses to make the conversation sound more natural like this



```
set item to Random Min 0 Max 2
if item = 0
do ControlCommand( Speech Synthesis, "SayPC" )
else if item = 1
do ControlCommand( Speech Synthesis 4, "SayPC" )
else ControlCommand( Speech Synthesis 5, "SayPC" )
```

h. The last step here is to, of course, edit the actual sentences that are spoken. That is done simply by going back to the central hub and changing the text field on the individual skills.



```
Speak louder
hello (0.95)
Script Start
Script Done (00:00:00.0132760)
hello (0.94)
Script Start
Script Done (00:00:00.0079845)
Low confidence: my name is (0.68)
my name is (0.94)
Script Start
Script Done (00:00:00.0071576)
nice to meet you (0.93)
Script Start
Script Done (00:00:00.0075920)
hello (0.95)
Script Start
Script Done (00:00:00.0086773)
```

Speech Synthesis: Hello Human

Speech Synthesis 2: What a pretty name.
Nice to meet you!

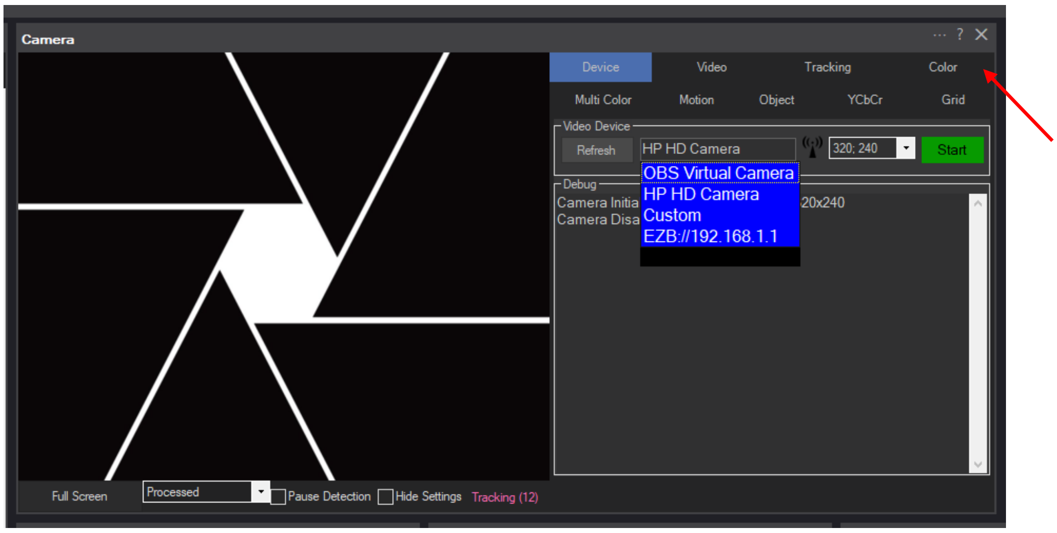
Speech Synthesis 3: The pleasure is all mine

Speech Synthesis 4: Hello! What is your name?

Speech Synthesis 5: Well met stranger. What is your name?

Image recognition

- At this stage, you have created a “social robot” that can listen and respond. One more step to making the interaction more realistic is to make sure that it is not listening and responding when there is no one there. The best way to do this is some simple object detection magic.
    a. To do any form of object detection, you must make sure that the hub has some camera feed coming in. Of course, there is a skill for that. This one needs to be the skill “Camera.” When you have it open, make sure to provide it with any camera that works (by clicking through the drop-down).



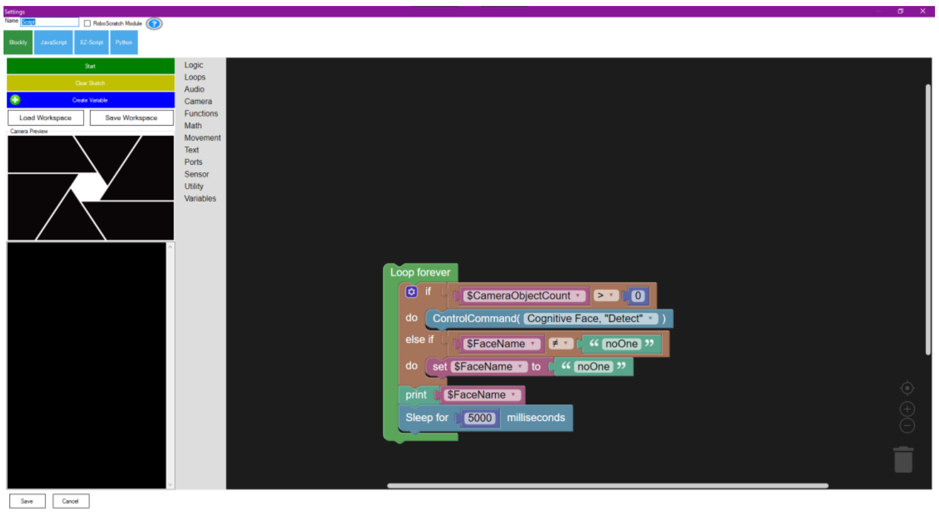
    b. For now, you are only going to need to know two things with this skill. The first thing is that when the aperture image is up, it means that the camera skill is paused and not taking in any images. <u>To start it up, you need to click the “Start” button in the “Video Device” cell</u>.
        i. Note that unless you have the paid version of the program ARC, you are not allowed to use a video stream that is *standard definition* or higher. This does imply that if the resolution is anything greater than 320:240, you cannot use it. Some terrible laptop webcams do let you run the video this low, but if you do not happen to own such a poor camera, there is a possible [hacky] workaround. You can find that in the appendix.
    c. The second thing is that you need to tell it to track faces. You do this by going to the “Tracking” tab and select the “Face” check box.
    d. Now we are done with the camera skill, and we are going to go to the more exciting element: recognizing people. For this, I recommend using the oddly named “Cognitive Face” Skill. It is capable of quickly learning people’s faces and assigning them names, and then looking for them in *stills* (pictures taken from the video stream)
        i. After adding the skill and checking that the “Camera” skill is not paused, click the “Learn Face” button while looking directly to the camera (bonus point for checking if the camera is tracking a face when you take the still).
        ii. Having done that, give a name to that face. Full name is necessary [apparently].
        iii. This should be it. Now your social robot can recognize [probably] you. You should, as always, click detect a few times and see if it recognizes you correctly (you might need to expand the default window of the skill to see the given name of the face show up in the dialog box).
    e. There is one last issue that needs to be addressed before we can use the recognition in the social robot. This is the fact that the “social robot” only checks if someone is there through stills. This means that we need to automate something to press the “Detect” button for you. This can easily be done by using the skill “Script.”
        i. Once you have added this skill, you get the lamest window: it is a start button and a dialog box. That is because this skill simply runs a bit of code, as to be expected. Open its settings by clicking the “…” button, and you will see that you get brought straight to the code editor.
        ii. Opposite to the function’s code editor, this code only runs when you click the “Start” button in the hub. This means that it is usually preferable to have some sort of loop here. If you don’t, you are going to need something to start it (e.g., you are going to need to physically press the start button all the time).
        iii. For this code, we are going to use the *Variables* that I had previously mentioned. These are created by default by the skills. They serve as ways of trading information with other skills and share the fact that

they start with ‘$.’ I propose a loop to check if the camera is tracking a face and then checking if that face is the recognized person.

iv. To check if the camera is tracking a face. One way to do this is to see the number of objects that the camera has detected. This can be done by using the variable created by the Camera skill “$CameraObjectCount.” It can be deduced from this variable’s name that it returns the number of objects *tracked* by the camera skill. This variable also seems to be updated every time there is a change in the number of objects detected by the skill. If you remember, the only thing that the camera has been set to track is faces, so if the number of objects that the camera found is greater than zero, it means that there is at least one face.

v. Now that we know when we are looking at a face, we need to know to whom the face belongs. This can be done by clicking “Detect” in the *Cognitive Face skill*. For the code, though, this can be done by using those “ControlCommand()”s. However, I do not recommend clicking that button too often since it does take time for it to respond.

vi. Lastly, we actually need to know the name found by the skill. This is saved in the skill’s variable “$FaceName.” However, unlike the camera that updates its variable multiple times a second, this carriable is only updated when a name is found, aka. If you click detect and nothing is found, the variable will not be changed. It is essential to do this ourselves if it detects nothing so we can account for it.

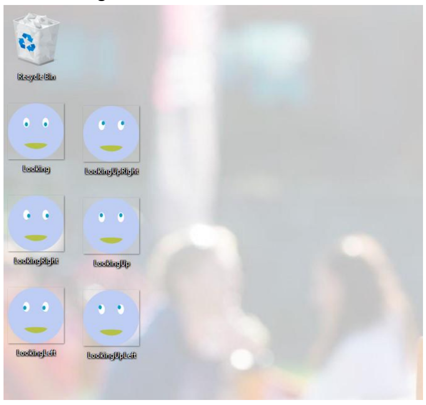


```
Loop forever
    if   $CameraObjectCount > 0
    do   ControlCommand( Cognitive Face, "Detect" )
    else if   $FaceName ≠ “ noOne ”
    do   set $FaceName to “ noOne ”
    print $FaceName
    Sleep for 5000 milliseconds
```

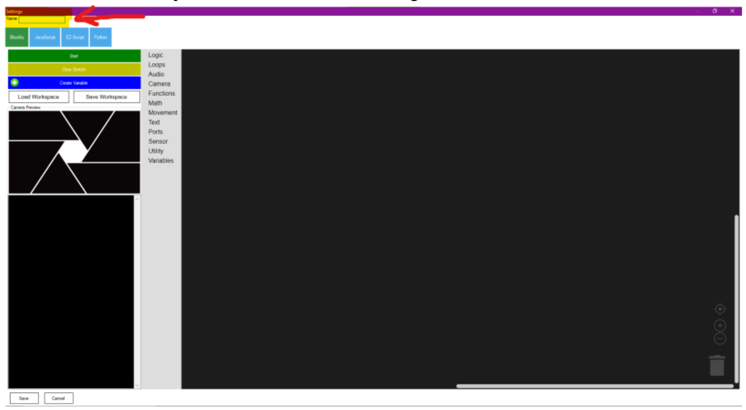
f. Now, if you save the script and click the “Start” button back in the hub, you should see that your social robot will start checking if you are there or not.

Visual interface

- In order to convince everyone that you have a fully fledge social robot, you are also definitely going to need to give your robot some shape: a face [at least]
    a. The first step in getting a visual interface for your robot is going to be making a face. This can be a PNG roughly sketched on any drawing software. The only thing that you need to account for is that you would also like to attribute to this face some super simple gaze behavior, so you are going to need to save multiple copies of your face looking at different directions that we will later link to wherein the video stream [possibly] you are standing.

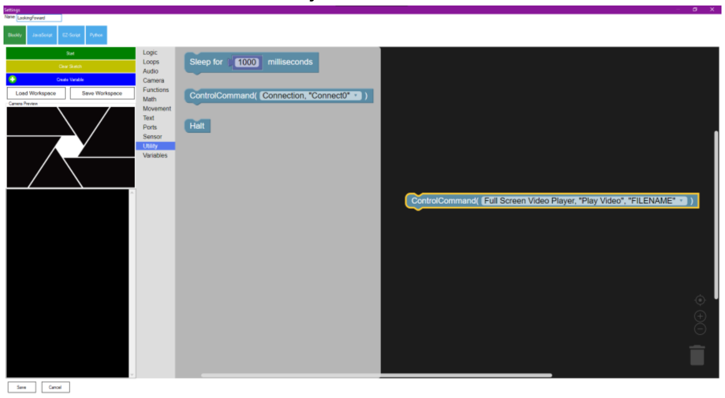


    b. To run that so-called video stream, we are going to use a skill that is called "Full Screen Video Player," which you should be able to find under "user interface." This skill has the ability to allow you to run images or videos in the foreground and lets you start and stop them at a whim. What this does to us is let us show the face you made when we think it is appropriate, basically creating a simple animation that we can make it feel like the face is paying attention to us.
        - Note that when a video is playing is on Fullscreen, you can always click 'esc' to disable the skill doing so, stopping the video and preventing any new video from playing until the skill is reenabled (done so by pressing the *new playback paused* button)
    c. However, for this skill to work, we are going to need to have some coding that sends to the skill what video it should play and when.
        i. To send the video that you are going to be playing to the skill, we are going to need a script that does that. There are many possible solutions, I am going to present one that is effectively not the best, but the one that I think is most appropriate for those that are not comfortable with coding.
        ii. The first thing we are going to need to do is to get a script skill ready for this. Since this recommendation will require you to make quite a few scripts, it will be nice to keep them all organized; so if you haven't done so already, this is the time to use the "script collection" skill instead of just many "script" skills.
        iii. After adding a script to your collection, you want to make sure that you start naming everything well (with names that you can easily remember what they mean). To name things, you need to edit them, which should take you to the standard coding screen.



        iv. This particular script is going to act as a function that serves to play an individual image; in my case, this is going to be the looking forward script. To do that, we are going to call the utility block

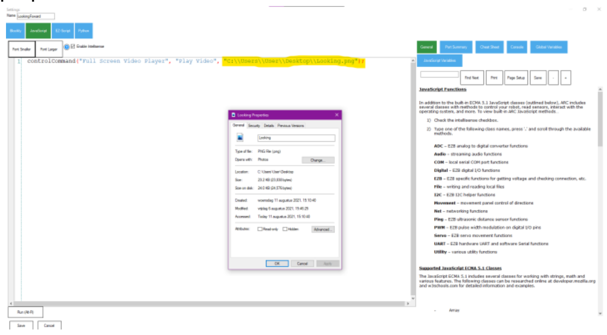
ControlCommand for the skill we just added.



```
Sleep for 1000 milliseconds
ControlCommand( Connection, "Connect0" )
Halt
ControlCommand( Full Screen Video Player, "Play Video", "FILENAME" )
```

v. As you might expect, we need to input the "FILENAME" somehow. Sadly, this cannot be done through block coding. For this, we are going to quickly look at the JavaScript language tab. Quickly switching to this tab should show you some already written code that you should recognize to be precisely what you had in blockly. Here we **can** edit the "FILENAME" variable. This is, as the name suggests, the location of what image or video you want to display.

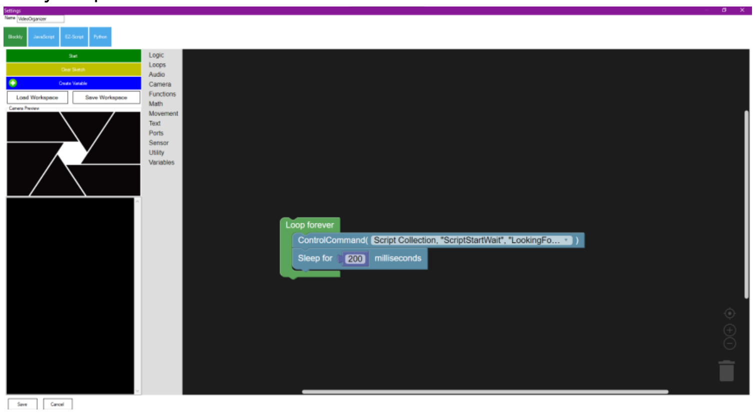
- Note: the languages are not interchangeable, despite the javascript already have been written for you from the blockly code. As soon as you edit anything on the written side, you are going to lose the blockly (and inversely, if you were to put anything in blockly again, you would lose the changes to the written languages). The program is going to give you a warning, but in this case, where we *need* to edit the code, you are going to have to accept.
- If you are not sure what the file location of your images is, you can always check around the properties window of that image. You get there by right-clicking the image and opening properties.



```
controlCommand("Full Screen Video Player", "Play Video", "C:\\Users\\User\\Desktop\\Looking.png");
```

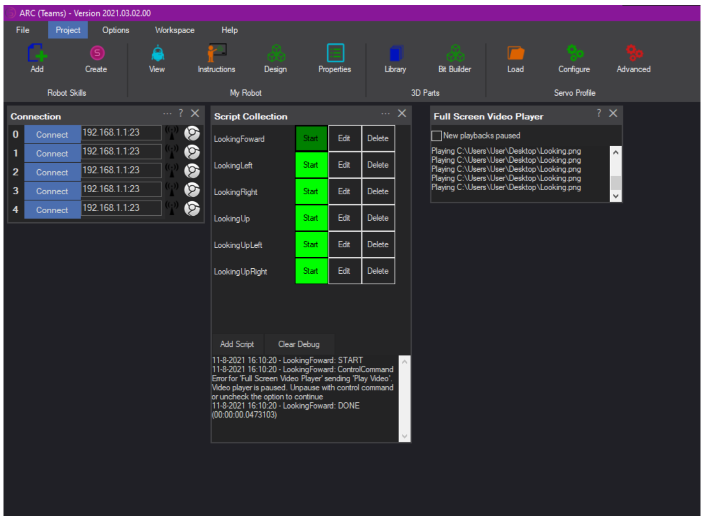
- Note that in JavaScript, the command '\' is actually spelled '\\', which might cause confusion

vi. Now, if you save the script and try to start it, you should be rudely interrupted by a large playback of your image for a split second and then white. If that is the case, you did good (press esc to close that "video"). Let us run one final test before we go out to making new scripts for every different picture. This final test will be to see if we can call this image multiple times from the comfort of our own *other* blockly script.



```
Loop forever
    ControlCommand( Script Collection, "ScriptStartWait", "LookingFo... )
    Sleep for 200 milliseconds
```

vii. If running this last script showed you the picture that you were looking for consistently, then it is now up to you to make the rest of the images into their own little scripts that we shall go over how to call later.



0	Connect	192.168.1.1:23
1	Connect	192.168.1.1:23
2	Connect	192.168.1.1:23
3	Connect	192.168.1.1:23
4	Connect	192.168.1.1:23

LookingFoward
LookingLeft
LookingRight
LookingUp
LookingUpLeft
LookingUpRight

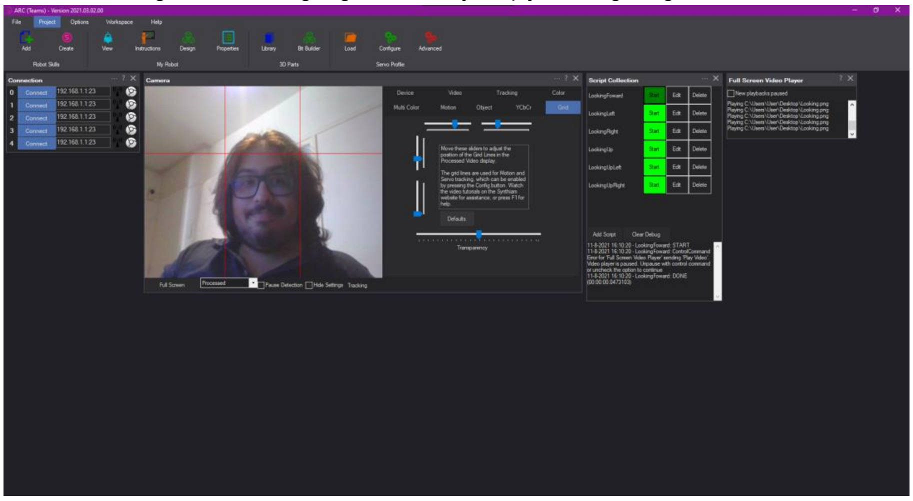
```
11-8-2021 16:10:20 - LookingFoward: START
11-8-2021 16:10:20 - LookingFoward: ControlCommand
Error for 'Full Screen Video Player' sending 'Play Video'.
Video player is paused. Unpause with control command
or uncheck the option to continue
11-8-2021 16:10:20 - LookingFoward: DONE
(00:00:00.0473103)
```

☐ New playbacks paused

```
Playing C:\Users\User\Desktop\Looking.png
Playing C:\Users\User\Desktop\Looking.png
Playing C:\Users\User\Desktop\Looking.png
Playing C:\Users\User\Desktop\Looking.png
Playing C:\Users\User\Desktop\Looking.png
```

d. There is one more thing that we need to adjust before we can get the user interface ready and running.

i. In the Camera skill, you are going to use one thing that you have not done yet. You are going to need to create the segments that you are going to later use as coordinates for where the person the robot should be looking at is. You are going to do this by simply enabling the grid in the camera skill.



Move these sliders to adjust the position of the Grid Lines in the Processed Video display.

The grid lines are used for Motion and Servo tracking, which can be enabled by pressing the Config button. Watch the video tutorials on the Synthiam website for assistance, or press F1 for help.

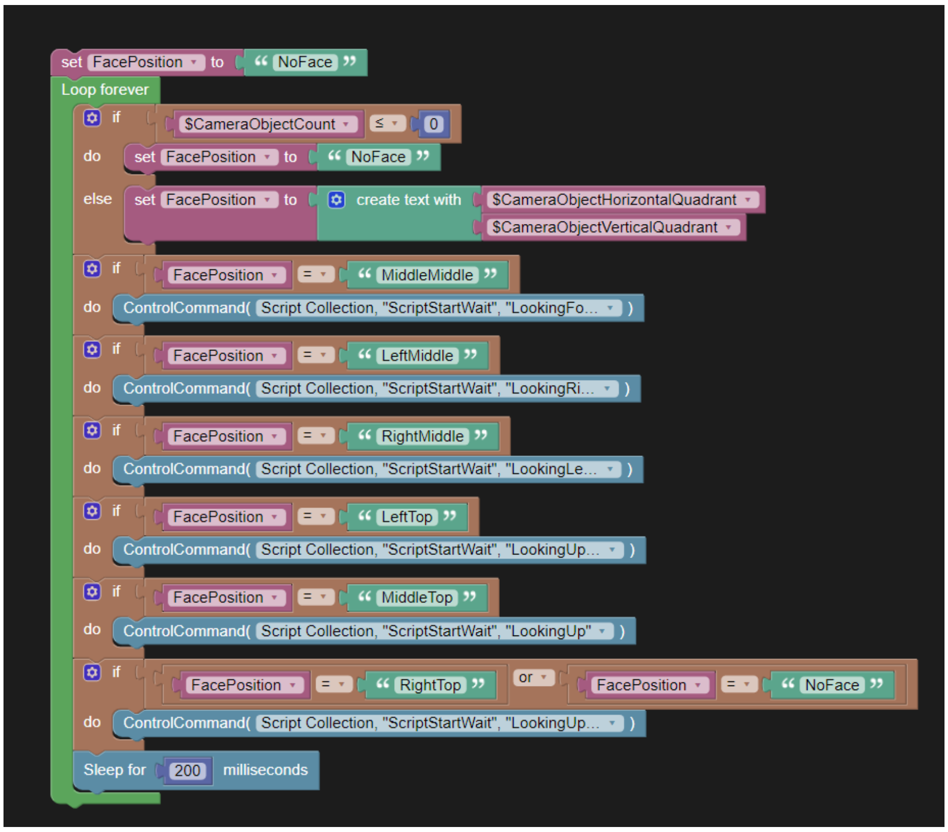
ii. The grid should be enabled by default; you just need to adjust their positions according to the faces you have (I demoed here the six orientations I have).

e. Now to put the final piece of the puzzle in place make a functioning gazing inter**face,** we are going to set up the script that starts all these videos accordingly.

i. For this script, we need to learn where the person that the video is looking at is (in relation to those quadrantes that we have defined), and we need to launch the videos accordingly.

ii. The camera skill has two variables related to where the person is; these are '$CameraObjectHorizontalQuadrant' and '$CameraObjectVerticalQuadrant'. These are updated every time a detected object moves from one quadrant to another. This should provide a big chunk of the information that we need. The only thing it does not know is whether there is a face moving around there or not since the variables are only updated when a face crosses the grid. However, we have discussed previously that a solution for this would be to check how many objects are being detected and if there are none, then we can update the variables ourselves.

iii. I would also suggest that instead of working with many variables being edited all over the place, it is more practical to make a local variable that only reads from the ones by the camera and holds all the information that we plan on using



```
set FacePosition to " NoFace "
Loop forever
    if  $CameraObjectCount ≤ 0
    do  set FacePosition to " NoFace "
    else set FacePosition to  create text with  $CameraObjectHorizontalQuadrant
                                                 $CameraObjectVerticalQuadrant
    if  FacePosition = " MiddleMiddle "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingFo... )
    if  FacePosition = " LeftMiddle "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingRi... )
    if  FacePosition = " RightMiddle "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingLe... )
    if  FacePosition = " LeftTop "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingUp... )
    if  FacePosition = " MiddleTop "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingUp" )
    if  FacePosition = " RightTop "  or  FacePosition = " NoFace "
    do  ControlCommand( Script Collection, "ScriptStartWait", "LookingUp... )
    Sleep for 200 milliseconds
```

by itself.

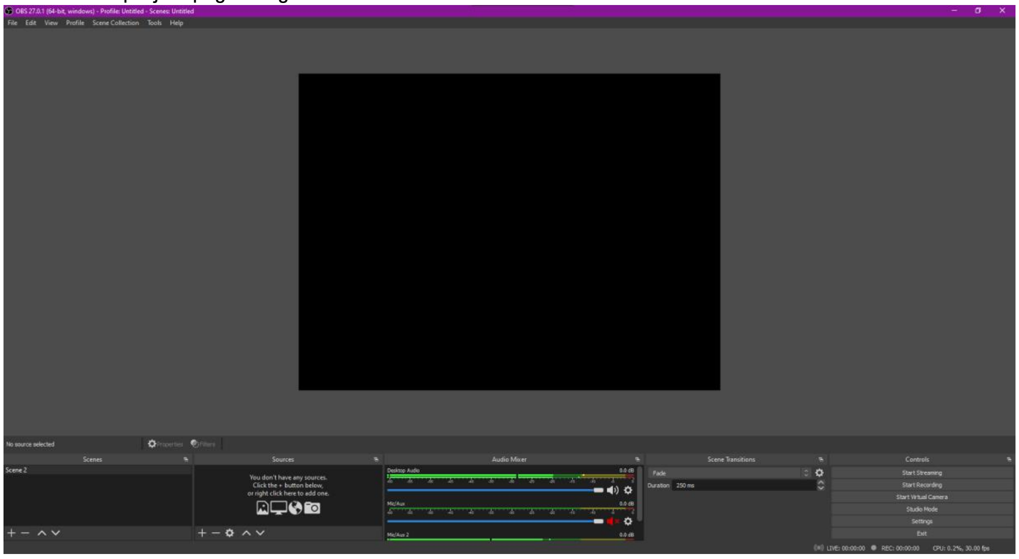
iv. Lastly, we just need to add all the ifs and calls the right way, and it should look something like this:

With this, you should have all the pieces to set up a friendly and exciting interaction. It is up to you now to tie all the loose ends and make something that actually is capable of conversing with you! Good luck and have fun!

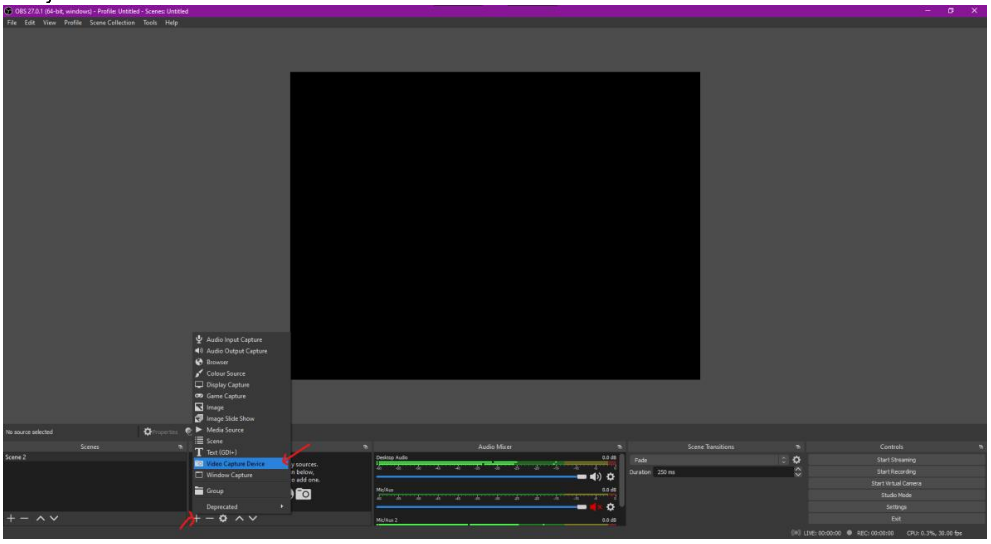
A quick hack to downscale your webcam resolution

For this solution, you are going to need to download an additional program that is going to serve as a virtual camera replacement. You can get the app here. Just clicking the windows button should do the trick. After getting the app up and running, you are going to need to do the following:

1. Start a fresh project page thing

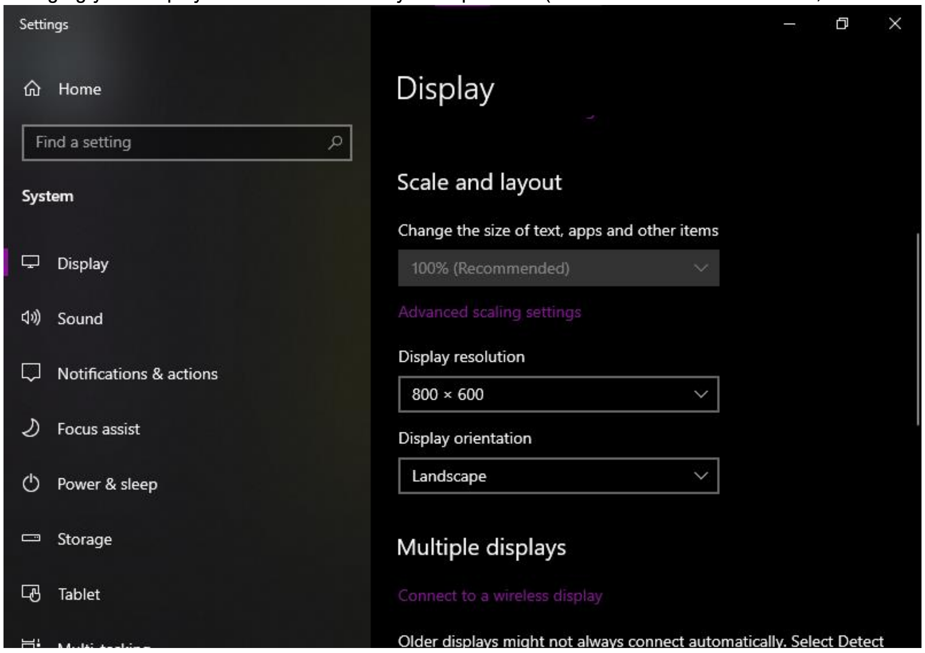

OBS 27.0.1 (64-bit, windows) - Profile: Untitled - Scenes: Untitled
File Edit View Profile Scene Collection Tools Help
No source selected
Properties
Filters
Scenes
Sources
Audio Mixer
Scene Transitions
Controls
Scene 2
You don't have any sources.
Click the + button below,
or right click here to add one.
Desktop Audio
Mic/Aux
Mic/Aux 2
Fade
Duration 250 ms
Start Streaming
Start Recording
Start Virtual Camera
Studio Mode
Settings
Exit
LIVE: 00:00:00 REC: 00:00:00 CPU: 0.2%, 30.00 fps

2. Add your camera feed



3. Click through, accept all the random popups, and leave everything the way it is. If you did it right, the “canvas” (the display thing in the center of the screen) should be populated by your webcam feed

4. Now you have to go to your display settings in windows (one way to get there is by right-clicking the desktop) and changing your display resolution to the very least possible (it looks horrible for a little bit, but we shall revert back soon)



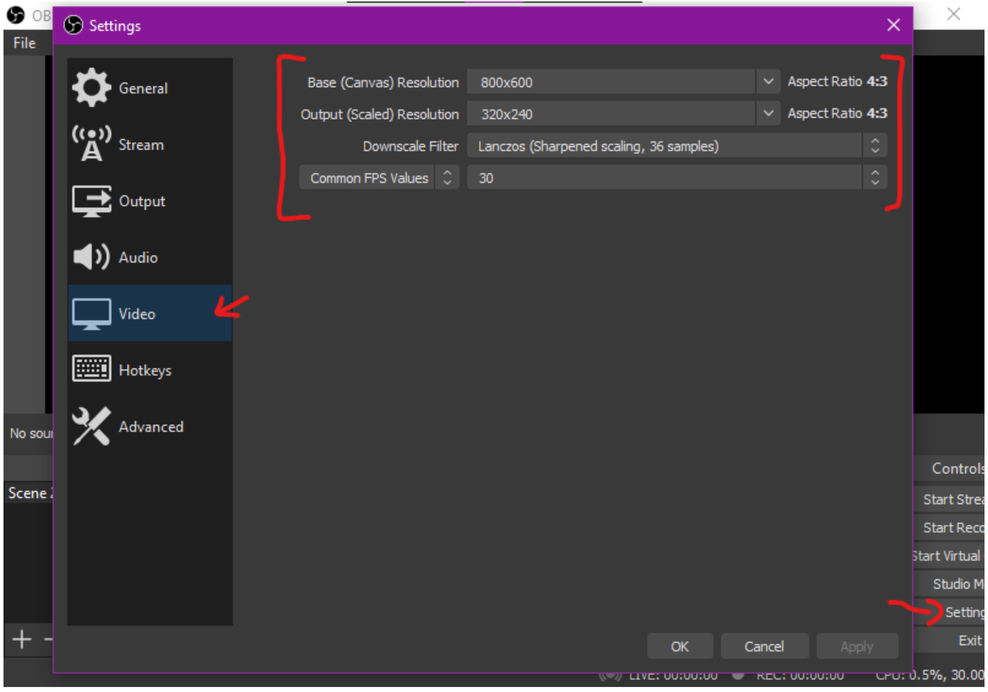
Change the size of text, apps and other items: 100% (Recommended)

Display resolution: 800 × 600

Display orientation: Landscape

Older displays might not always connect automatically. Select Detect

5. Now with your changed resolution, return to OBS and go to the settings button and navigate to the video settings



Base (Canvas) Resolution: 800x600 Aspect Ratio 4:3

Output (Scaled) Resolution: 320x240 Aspect Ratio 4:3

Downscale Filter: Lanczos (Sharpened scaling, 36 samples)

Common FPS Values: 30

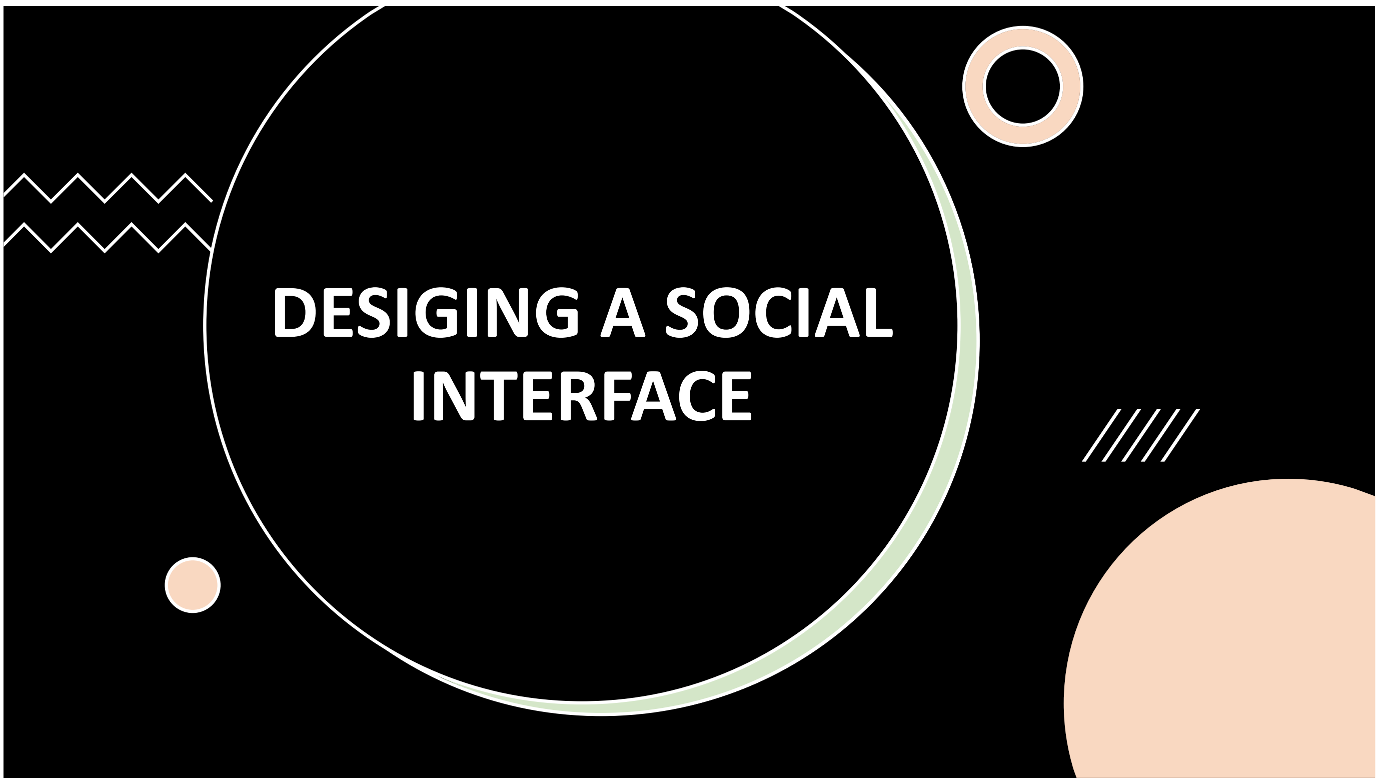
6. It might be a little bit difficult but when you do get to the video settings, change the base, output, and filter settings to the ones in the picture

7. After clicking Ok and closing the settings in OBS, you may return the screen resolution back to normal in the display setting

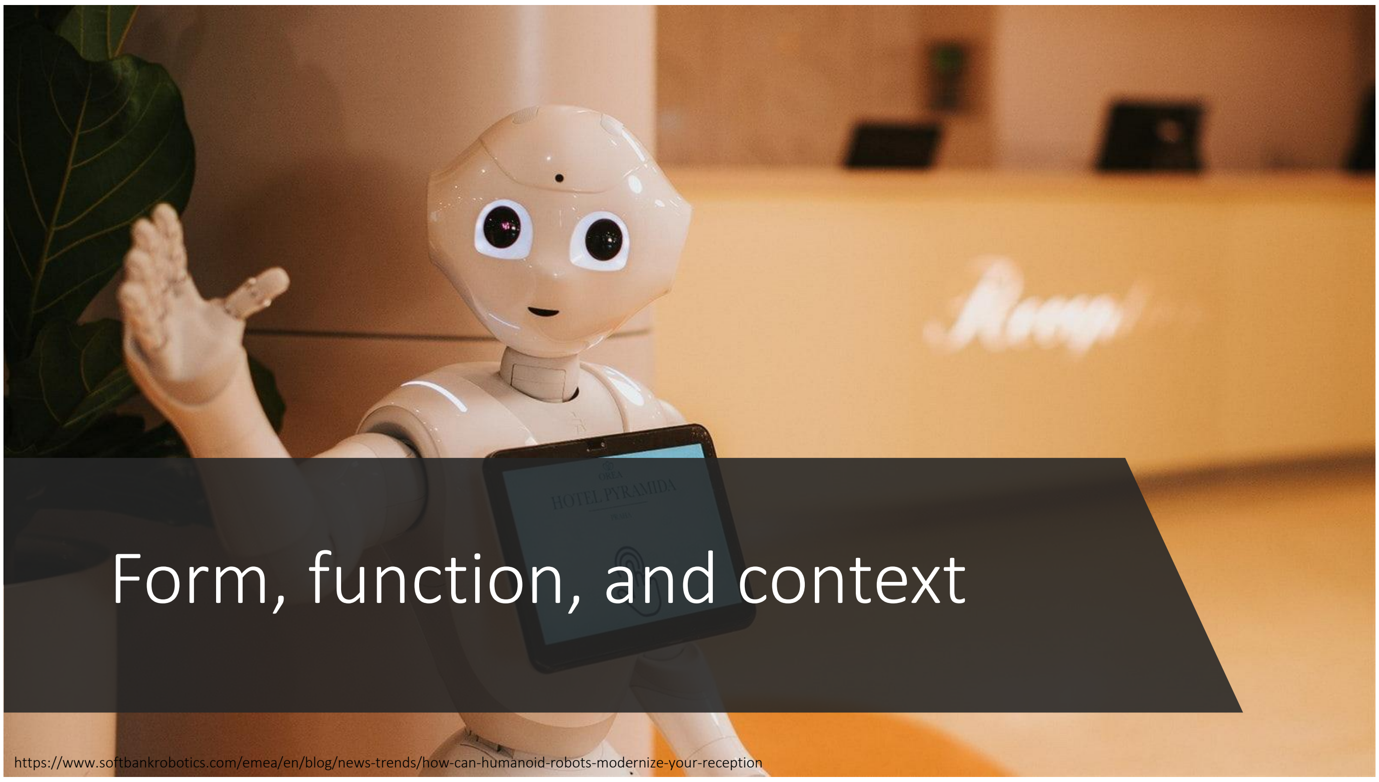
8. Lastly, now, if you click the “start virtual camera” button above the “settings” button, it should stay active. That means that now you are running the aforementioned virtual camera that displays exactly what is on your canvas, so feel free to make it look all neat before *reopening arc* (It is important to note that you will only be able to find the virtual camera feed in ARC if you launch ARC after it is up and running)

Appendix B

Presentation used during first day of the workshop

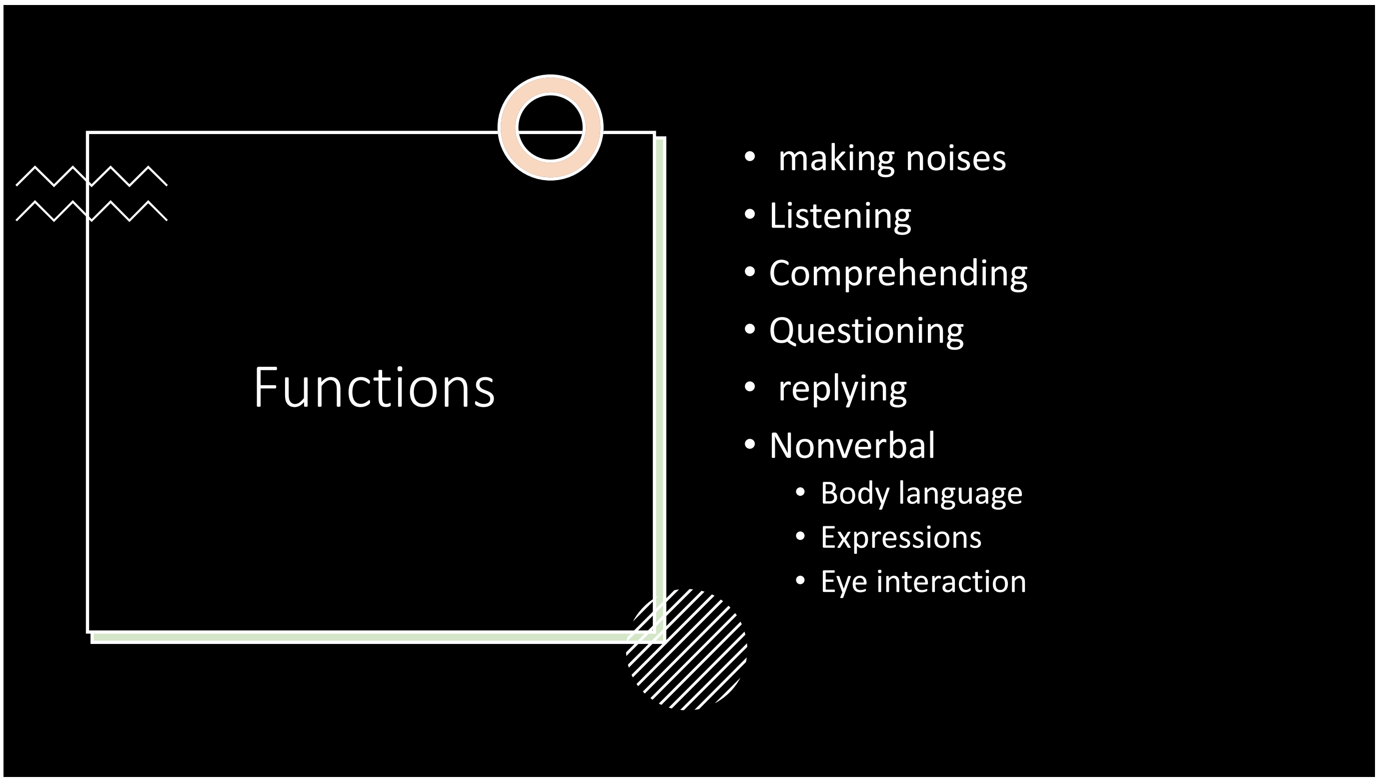


DESIGING A SOCIAL INTERFACE



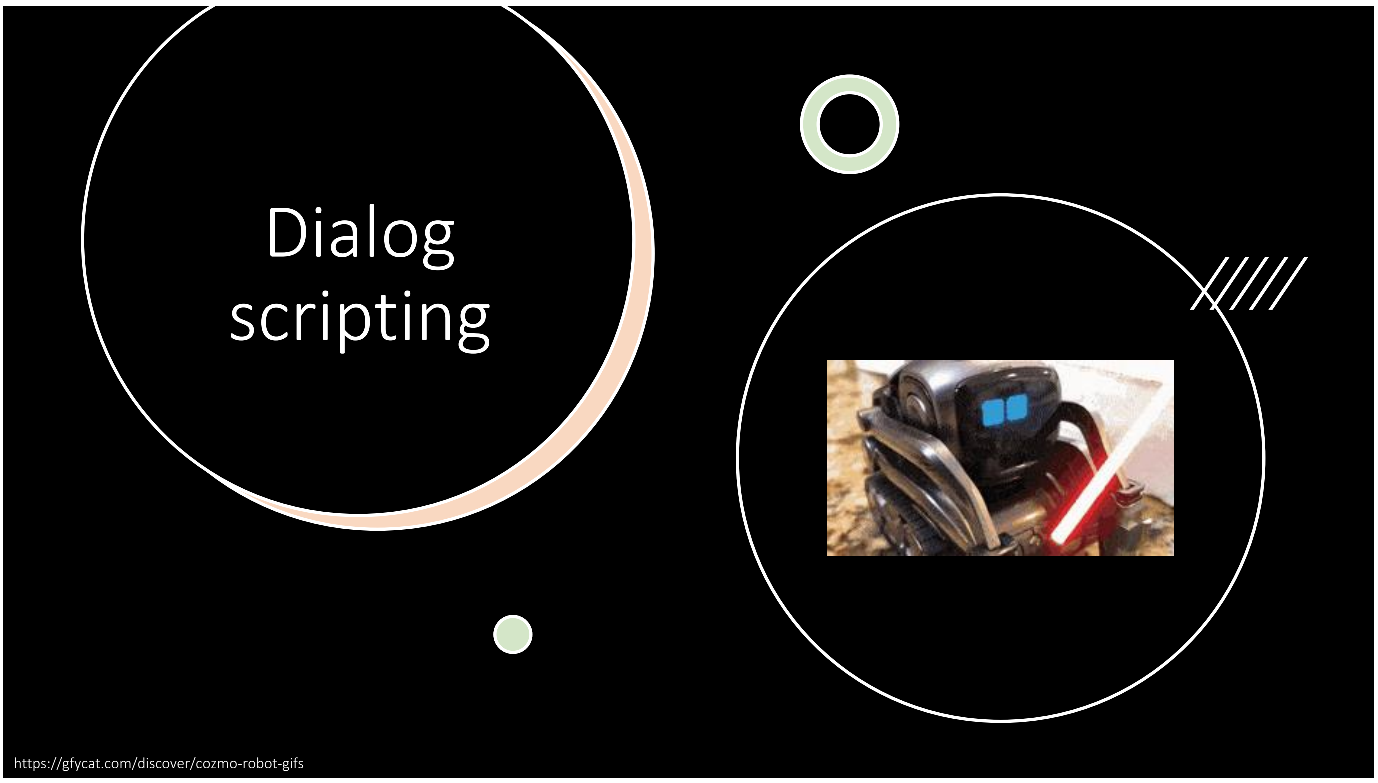
Form, function, and context

https://www.softbankrobotics.com/emea/en/blog/news-trends/how-can-humanoid-robots-modernize-your-reception



Functions

- making noises
- Listening
- Comprehending
- Questioning
- replying
- Nonverbal
  - Body language
  - Expressions
  - Eye interaction



Dialog scripting

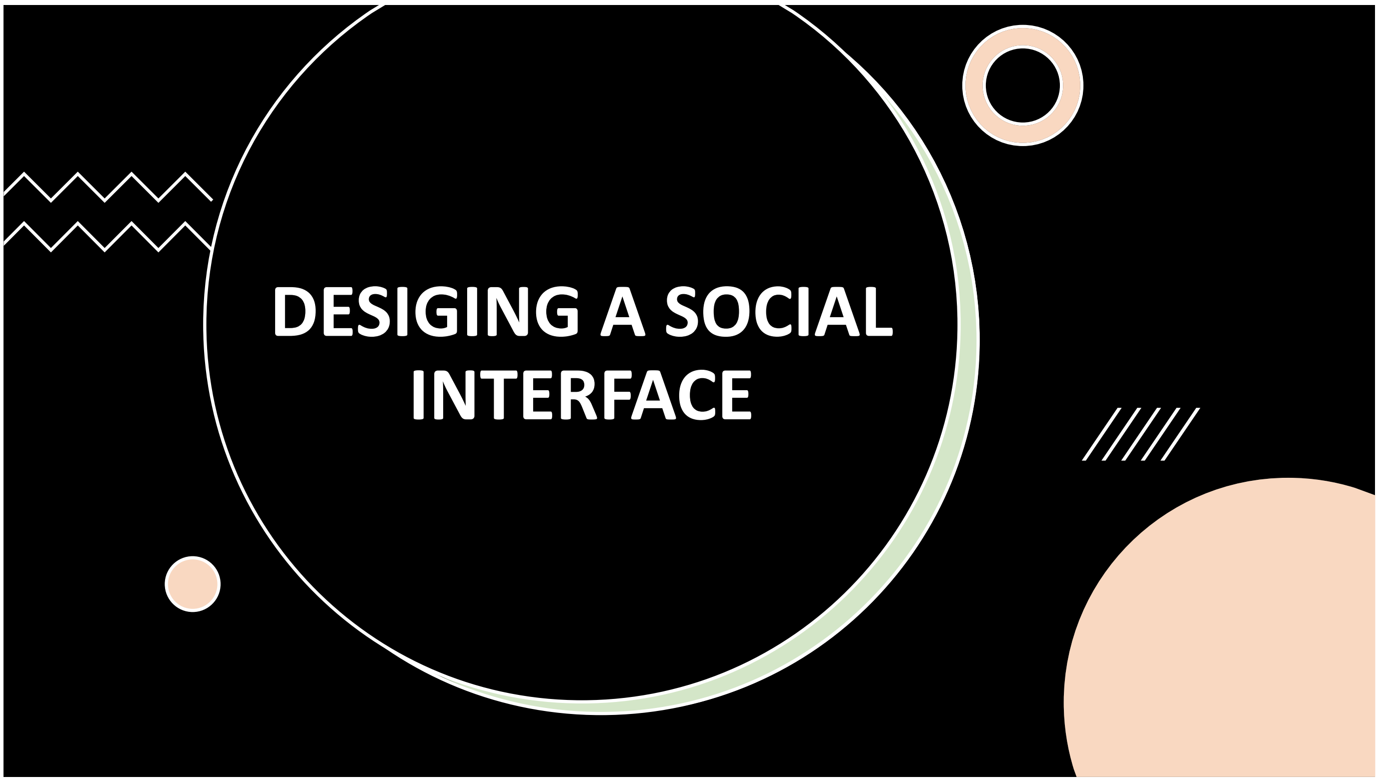
https://gfycat.com/discover/cozmo-robot-gifs

Control software

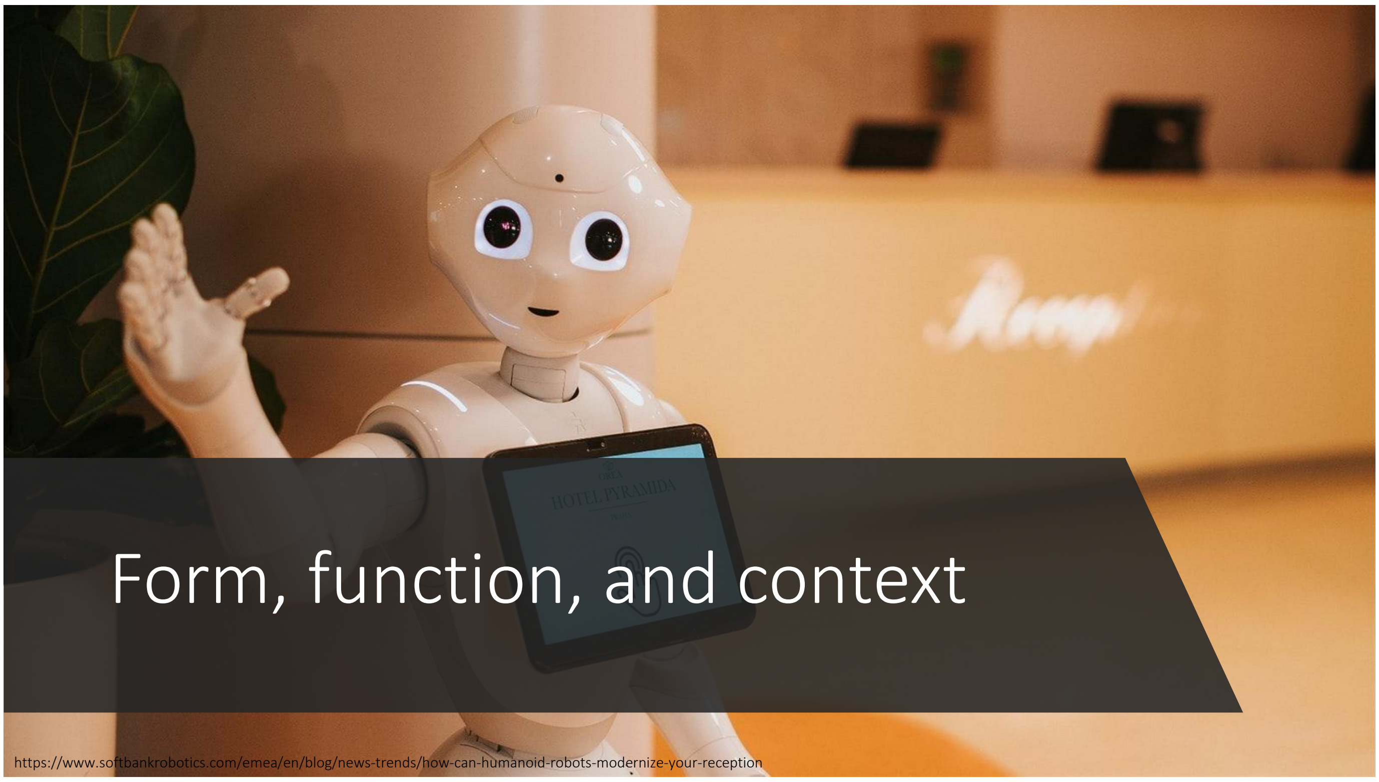
https://synthiam.com/Products/ARC

Appendix C

Presentation used during second day of the workshop

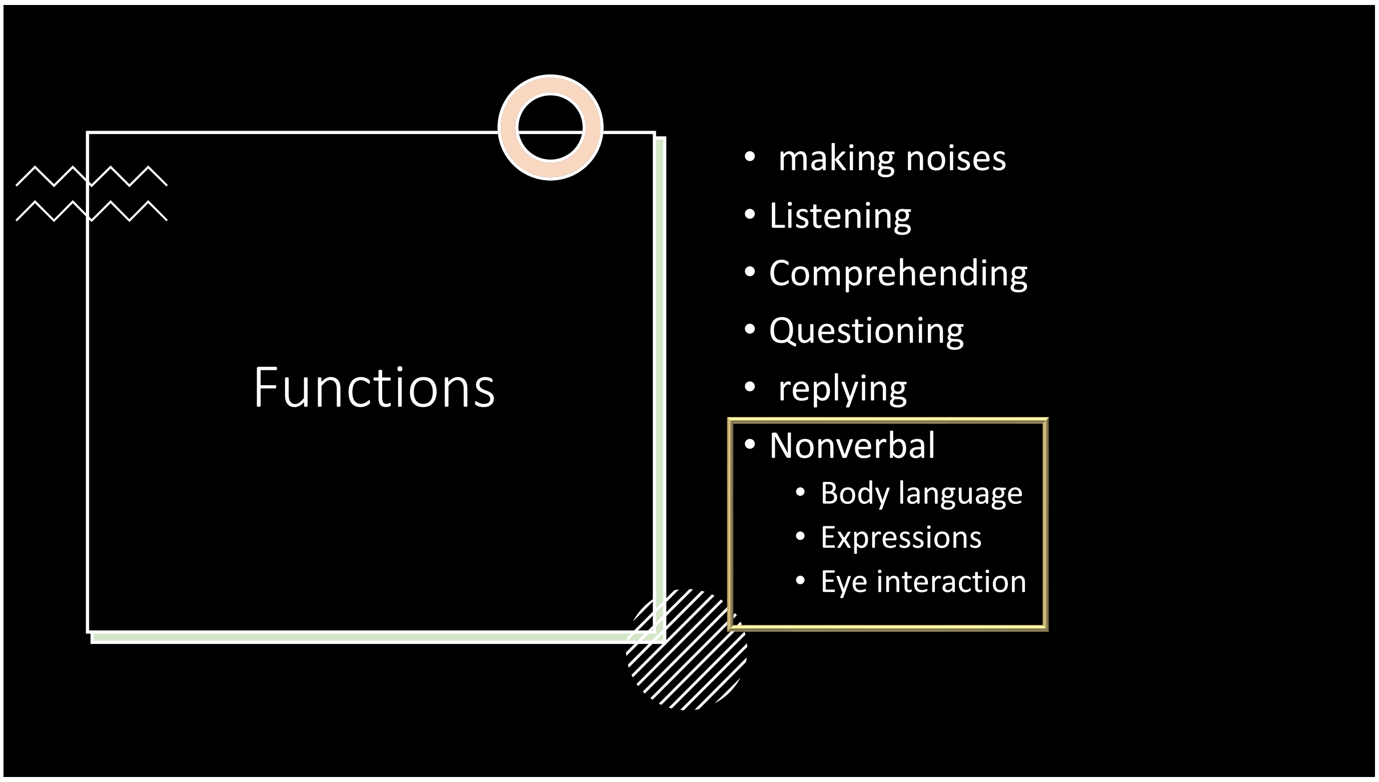


DESIGING A SOCIAL INTERFACE



Form, function, and context

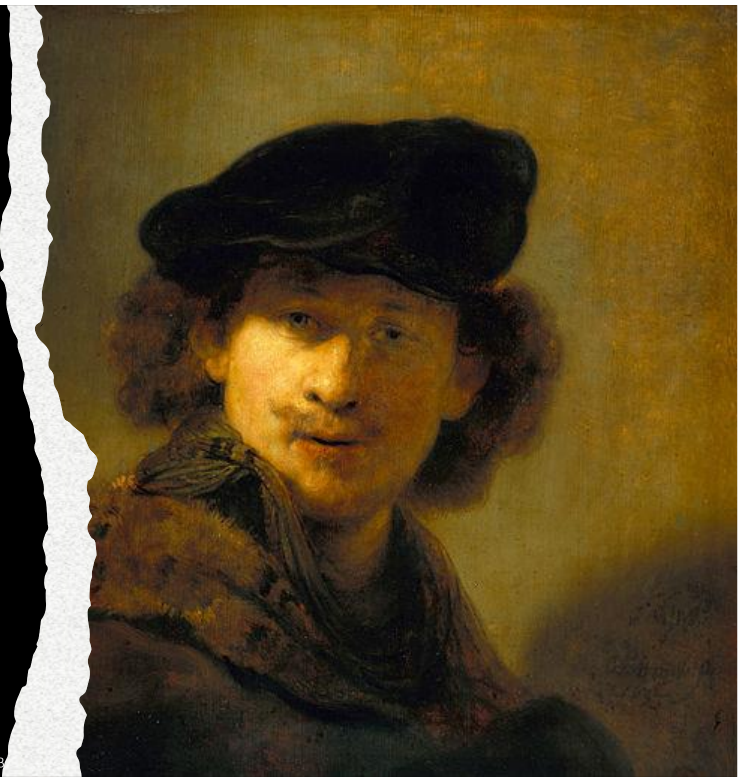
https://www.softbankrobotics.com/emea/en/blog/news-trends/how-can-humanoid-robots-modernize-your-reception



Functions

- making noises
- Listening
- Comprehending
- Questioning
- replying
- Nonverbal
  - Body language
  - Expressions
  - Eye interaction

GAZE BEHAVIOUR



https://artsandculture.google.com/asset/selbstbildnis-mit-samtbarett/mQGjCu2ESqQc_w?hl=en-GB