PiNAOqio: Research into Interactive Story-Telling Using Robots and Physical Books

Xavier Laguarta Soler (xl5512), Periklis Korkontzelos (pk1913), Nathan Warner (njw13), Mihnea Rusu (mr3313)

Imperial College London

Abstract

This paper evaluates the effectiveness of the interactive robot storyteller PiNAOqio, boasting the ability to turn the pages of physical books and deliver contextualized gesticulations. An assessment was conducted on 23 volunteers whereby PiNAOqio was pitted against both a static implementation and an audiobook. Performance was measured in terms of information retention and user enjoyment, finding PiNAOqio to be just as, if not more effective than the audiobook across our given test population. Further developments and enhancements are proposed to be implemented in the near future.

I. Introduction

Reading is an integral and fundamental aspect of a child's education and has been shown to have a significant influence on their development (Freire & Slover, 1983). In today's digital era the ways through which children are introduced to books and written content is constantly evolving, placing an increasing prevalence on digitized content and stepping away from traditional hard-copy books. PiNAOqio proposes a new and innovative means to bridge this gap and engage children with literature. Utilizing a synergistic collaboration of an automated page turner, optical character recognition (OCR) algorithm, intelligent gesture association engine and voice recognition, PiNAOqio is well poised to revolutionize conventional story time.

With users being able to load their own pre-existing volumes, PiNAOqio will be versatile enough to be deployed in a range of situations and environments. We feel that children that have to be hospitalized for long periods of time, in particular, would stand to receive the greatest benefit from this technology. Not only will PiNAOqio be able to read them their favourite stories, he can also serve as a companion, supplementing the limited human interaction they are likely to receive and alleviating some of the stress and anxiety these children are prone to experience (Lau, Bernard and Wilson, 1993). Although PiNAOqio is not intended to be a tantamount replacement to a genuine human storyteller, he is anticipated to make story telling a more fun and social experience for young listeners.

II. Hypothesis

The aim of this project was to investigate the effectiveness of a "storyteller robot" as a source of engagement and educational entertainment for children. In order to quantify whether this goal has been achieved, user experience and the ability to retain information were measured. Upon this basis, the subsequent hypotheses are assessed. These figures have been approximated to indicate thresholds for satisfactory performance:

a) Story-telling by PiNAOqio will improve information retention and user experience by 15% and 30% respectively in comparison with story-telling by an audiobook.

b) Making gestures to emulate a human story-teller in conjunction with an active interaction with the user will improve the retention of information and the user experience by 10% and 20% respectively.

III. Related Work

The concept of an interactive storytelling robot has been approached by other groups in the past; equally, there exist several page flipping mechanisms for the purpose of digitizing hard copy books (Qumsiyeh, 2016 and Charlton, 2015). However, an implementation that uses both of these in tandem has heretofore not been attempted.

An instance of a comparable system, The Storytelling Companion, designed by the Personal Robotics Group at MIT (Kory, 2016), was developed as a social character relating to children as a peer through a storytelling game. This game is played on a tablet and the aim is to improve language and vocabulary skills in early stages in life. Another example is TROBO – the storytellinsg robot (TROBO, 2016), a plushy robot that connects to a smartphone or tablet application and tells an education story from a library.

The fundamental drawback to these currently available solutions is that they rely on a tablet to function. Long-term use of tablets and other such devices have been shown to have a detrimental impact on young children, such as addiction. The positive to this approach is that it utilizes technology the user is already likely to own, thereby largely reducing the overall cost of the product.

IV. System Design

1. Overview

From our previous report (Laguerta, Korkontzelos, Warner, & Rusu, 2016), the system has increased in complexity as we discovered insufficiencies with the old setup that would disrupt our desired user experience. Formerly we decided that a Raspberry Pi 3 should act as the main controller, communicating with the NAO robot, page-turning mechanism and camera. Such a setup would have been simpler to execute, and would have yielded a more portable prototype at the cost of limited opportunity for expansion and improvement past basic functionality. The major problem with this setup was the poor quality of the official Raspberry Pi camera for our purpose of optical character recognition (OCR).

Our first attempt at solving the problem of the camera, mentioned in one of our individual reports (Rusu, 2016), was to install a light-plate fashioned from a liquid crystal display (LCD) backlight directly above the book. While this boosted the accuracy of OCR slightly, we were still troubled by the shallow depth of field of our camera, which made it so that only a handful of pages could be turned before a (manual) refocus was needed. After experimenting with exposure settings and filter parameters on the Raspberry Pi camera, we were ultimately unable to produce reliable, repeatable OCR results.

Brief experimentation with an iPhone camera using the existing rig resulted in OCR accuracy figures of approximately 90% following brightness and contrast image enhancement filters. This observation led to our decision of using a smartphone as the camera, despite the increase in system complexity posed. At this point, the Raspberry Pi was only required for controlling the page-turning action; as such, we decided it would be best that a laptop with far more computational power than a single-board computer handled the OCR engine and communications to the NAO robot. An iOS app would be needed, as well as a means for transmitting images from the phone to the laptop and page-turning instructions from the laptop to the Pi board.

The final system was comprised of five elements: iPhone, Raspberry Pi, laptop, the NAO robot, and a server running on the cloud as an Amazon Web Services (AWS) instance to enable communication between the components. A high-level diagram of the system is shown in figure 1. Communication to the server is done via (Hypertext Transfer Protocol) HTTP. The role of the server is to manage the data being transmitted to each server-connected device given what has been received from all devices up to that point in time.

Devices operate in half-duplex mode: either transmitting or receiving. During autonomous system operation only two of three devices should ever be simultaneously in receive mode on the network, otherwise the system reaches rest. Receive mode utilizes a technique known as HTTP long-polling that reduces the amount of unnecessary network traffic when polling for data. The client begins by sending a request for data from the server, to which the server only responds after data is available for that device or a time-out has occurred (in which case the request should be re-initiated). Devices expect to receive single actions from the server, thus the data being transmitted is simply the action, or data object (e.g. image) as a string. Transmission data is sent using (JavaScript Object Notation) JSON strings in the body of the HTTP requests.

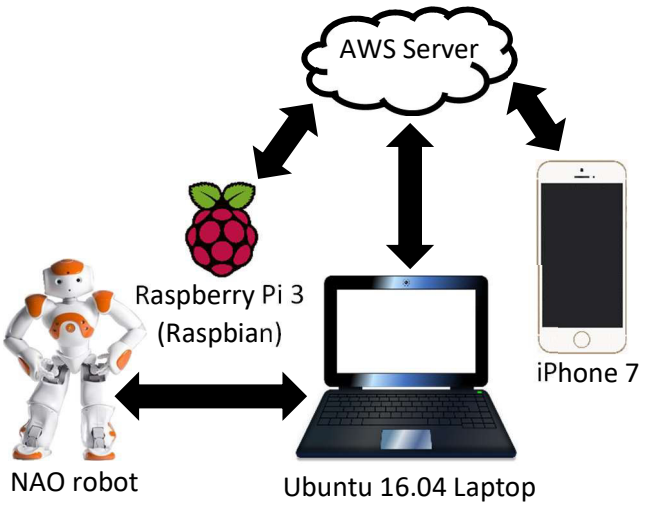

Figure 1 - System High Level Diagram

2. Server

The elements of our system need a way of communicating with each other in an organized manner in order to emulate the behavior of a state machine. One thing all our components have in common is a network connection – the obvious facilitator for our goal is thus a centralized hub (or server) to which all devices report updates to their state. A t2.micro Elastic Compute 2 (EC2) instance on AWS running Ubuntu 16.04 was used as the platform for our server. The cloud server route was taken due to the hosting restrictions imposed on the college network. With access to an isolated network, the server component can be integrated within the laptop component and the system design simplified.

Python was used to implement a multithreaded HTTP server with three valid addresses: /pi, /phone and /laptop. The HTTP protocol defines several possible request types

that may be sent by a client device. Of importance for our application are the GET and POST requests. A GET request to one of the aforementioned addresses initiates the long-polling process, and the server would only fulfill the request if data is available for that device or the time-out threshold has been reached. Multithreading is necessary for handling long-polling as this is a blocking operation, meaning the thread of execution is blocked until a result is available. POST requests from a device to the server contain JSON data about the state of that device. The JSON fields are specific to each device and are agreed upon by both the client and the server before runtime.

State machine functionality is emulated on the server through (First In First Out) FIFO queues. JSON data combinations from a device's POST request produce one or more work items, or actions, that need to be passed on to the other devices. Each device has its own FIFO queue of work items; the resulting actions are pushed on the corresponding device's queue. When a device long-polls the server and its queue is not empty, the first element on its queue is popped and parsed in the body of the HTTP response.

Emulating a state machine as such offers a high degree of modularity and flexibility in adding functionality to the system as a whole. Additional JSON fields may be defined for a device, as well as the work items they generate. Handling the new functionality can be implemented simply on the respective target device side

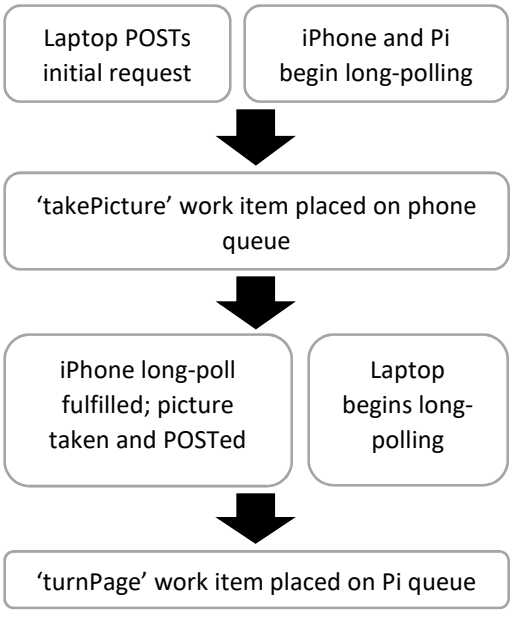

Figure 2 - Example System Operation

by scripting the desired action in response to the received work item. Before deploying the server, we first ensured successful operation using dummy clients on the physical laptop. Example operation from rest is briefly shown in figure 2.

3. iPhone App

There are two main tasks the iPhone element of the project must perform: capture and filter an image using the rear-facing camera; and implement the functionalities of the phone client. A skeleton camera app, written in Swift, provided by an online article [Caldari, 2015] was used as a starting point. Since modern iPhones feature powerful Graphics Processing Units (GPUs), we decided to take advantage of said feature by firstly configuring the base app to use the GPU pathway for rendering the image, and secondly to filter the shot on the GPU. This enabled a real-time post-filtering viewfinder that did not introduce any visible lag to the graphical user interface (GUI), even on an older generation GPU-equipped handset.

In terms of image filtering, the Apple iOS Core Image library provides a large number of GPU filters, with easily configurable parameters as well as the option of user-written kernels in a limited version of the OpenGL language. Our final filter stack was composed of a saturation reduction filter (for converting the image to black and white), followed by brightness and contrast booster filters and ending in a custom GPU kernel for a thresholding filter. The thresholding filter determines whether the intensity of a given pixel is below or above one of two thresholds. When below the respective threshold, the pixel is forced to black; when above, the pixel becomes white. Even without our dedicated lighting rig, OCR accuracy levels of 100% were common occurrences in our experiments, proving to us that switching to the iPhone camera was indeed the right choice.

Client-side code was written using the aid of the Alamofire networking library and the SwiftyJSON library for generating the JSON inside the POST request to the server. Instructions for only a single server action were provided – 'takePicture', and only a small number of data fields were required. The filtered image was sent across to the server as a base-64 encoded string in the JPEG format with minimal compression factor, under the 'picture' data field of the JSON POST request. Previously (Rusu, 2016), we mentioned requiring lossless compression for the picture on the Raspberry Pi camera, however this proved to be unnecessary in the case of the higher quality iPhone camera.

4. Raspberry Pi

The main role of the Raspberry Pi is to sequence the servos of the page turner in order to flip pages of the book, when instructed to do so by the server. A secondary role

of the Pi is to control the brightness of the lightplate in order to improve OCR performance. However, given the speed and accuracy of the iPhone camera's auto-exposure function and the performance of our filter stack, we found that the only use of this functionality is to preserve battery life. In the final prototype, the Pi was not to be connected to a screen, mouse or keyboard therefore we created pass-through modes on the server and laptop sides that would allow control over it out of story-telling context.

Multithreading was used to enable control of several servos in parallel. One problem we faced was with the built-in (General Purpose Input-Output) GPIO library on the Raspberry Pi. This library is not designed for multithreaded operation, and as a result we were troubled by segmentation faults when trying to access the same servo multiple times in a program. Happily, the solution was simple - run each set of parallel servo operations as a standalone script.

As in the case for the other devices on our network, the Pi requires client-side code that mirrors what the server expects from it and reacts to all possible actions received from the server. Since the mechanics of the page flipper have no way of feedback, a way for the laptop to adjust a small set of servo parameters was needed as the robot would reach further into the book. Fortunately, the software architecture allowed us to implement this with ease, by defining new work items for the Raspberry Pi, together with the corresponding JSON fields on the laptop side.

5. Laptop

Arguably the central element of our system was the laptop. Its responsibilities included: speech processing the user's voice greetings and commands through an external microphone; requesting images from the server; doing OCR on images of text from the book; parsing gestures into the text obtained from OCR; and lastly controlling the NAO robot. A state machine was in charge of requesting server data and performing the operations in the needed order.

As for the case of the Raspberry Pi and the server, Python was used for writing the laptop code. The C++ based OCR engine Tesseract proposed in the analysis of one of our individual reports (Rusu, 2016) was used in the final prototype. Thanks to the library's popularity in the open-source community, a Python wrapper was readily available for it. An outdated version of the NAOqi Python Software Development Kit (SDK) was used, as the most recent official one at the time of writing was not functional. Thankfully, we were not missing any features of the newer library to complete our project. Finally, user speech processing was handled using the latest official Python Speech Processing library.

In our design report (Laguerta, Korkontzelos, Warner, & Rusu, 2016) we mentioned the need for doing the page turn and OCR with a short time in advance such that NAO can speak without interruptions. Due to NAO's speaking and OCR being blocking operations, multithreading was required to achieve this purpose. Unfortunately, due to time constraints with the project we were unable to implement this feature. However, thanks to the high performance laptop used, the delay between an image being received and OCR being complete was reduced compared to the alpha prototype featuring the Raspberry Pi as the main controller.

6. Page Flipper

The page flipper module is required to flip the pages of the loaded volume once it receives its cue from the server, allowing for a seamless transition in the flow of the passage. Each turned page will need to be presentable enough for OCR to take place, entailing minimal creasing and curvature. A custom servo actuated mechanism was designed and built, with each servo operated sequentially, modelled as a state machine on the raspberry Pi. The system receives an action request from the server, the sequence executes and then awaits its next command. The sequence is shown in figure 3, below.

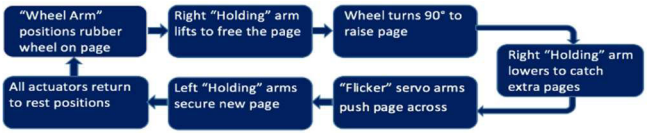

Figure 3 - Page Turning Sequence for Servos

The pivotal component of the system is a small rubber wheel that produces traction in order to lift the page. After passing through several iterations and permutations, the final mechanism to accomplish the page turning utilised seven servo actuators, these are illustrated in figure 4.

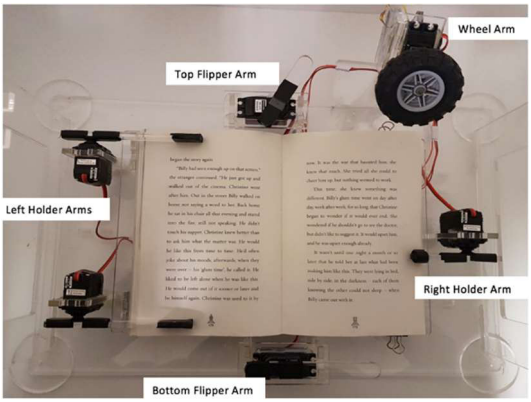

Figure 4 - Final Design of Page Turning Module

The whole module was made wireless and portable, using an 11.1V Lithium Polymer battery to power the light plate, with a 5V buck converter to supply both the servos and Pi. The entire assembly was constructed from 5mm laser cut acrylic, allowing for both a stylistically pleasing and structurally functional design, as well as fast development time. Once loaded by the user with any standard size fiction text and properly calibrated, the system was capable of autonomously turning in excess of 10 consecutive pages without error. The final system assembly is shown in the appendix.

Although the page flipping was shown to work well, there are several potential enhancements through which the performance could be improved significantly. Namely, feedback could be introduced on the wheel using a micro load cell, allowing it to consistently assert the appropriate amount of force on the page. For testing the ideal wheel height was determined experimentally, programmed into the system and adjusted as needed. The drawback to this approach is that the angle by which the wheel came down by had to be user adjusted every 10 pages or so as this height changes. Another weakness with the current configuration is that there is considerable jitter in the servos. Since we are using a software based controller, the timing of the pulses sent to the servos is likely to be thrown off by other processes running on the operating system. To circumvent this, it would be advantageous to control the servos instead using hardware; this can be done using a dedicated PWM chip, such as PCA9685 that interfaces directly with the Pi via hardware I2C.

In addition, with the current setup two or more pages may be turned at once due to irregularities in the binding of the book, pages being stuck together, or the wheel asserting too much force. Currently the system has no means to deal with this eventuality. One possible approach could be to keep track of the page numbers involves the OCR module. If an inconsistency in the page count is detected, PiNAOqio can then alert and instruct the user to turn back to the correct page. Better yet would be the ability to actively check and detect whether more than one page has been lifted and turn back to prevent the error from occurring. Commercial printers are faced with a similar challenge, some transmit ultrasound through the page and measure the transmission coefficient, thereby any discrepancy in this reading can trigger an error condition.

7. Gesture Controller

The NAO robot is controlled by a gesture controller, which is implemented as a python class. The gesture controller interfaces between the laptop and the robot and is responsible for implementing all talking and gestures made by PiNAOqio during initial interaction and storytelling, using the API (Aldebaran, 2016) provided by Aldebaran.

The gesture controller is initialized along with the main program at the initialization of the main state machine. In the initialization phase the motors of the robot turn and PiNAOqio stands up to signal the user that testing is about to commence. In addition, the desired pitch, speed and volume are set to their default value.

For the initial interaction, the gesture controller is passed to the speech controller module and called upon whenever needed, to avoid the risk of runtime errors whereby more than one modules try to access the NAO robot. The questions and responses of PiNAOqio during this phase are predetermined and thus, a text-to-speech function handles the articulation of responses. At the same time, the *BodyLanguageMode* is set to 'contextual' and allows the robot to make gestures that can be associated with the responses given. This is a built-in function from the NAOqi API and works on a basis of a dictionary associating words with gestures. This is not optimal as some gestures appear to be out of context, yet it enhances the animacy and likability of the robot, which is the aim.

During storytelling, the gesture controller is being used for the fully interactive and the non-interactive robot versions. For the non-interactive version, only the text-to-speech function is being used as no gestures are required. In order to acquire statistically meaningful data for the evaluation of the hypothesis, it is required that the fully interactive version PiNAOqio emulates a human storyteller as closely as possible. Thus, the 'contextual' *BodyLanguageMode* is not adequate but two more dictionaries associating key words/phrases with gestures are built to provide a more refined gesture library. The first library is built using some already available gestures that accompany words such as "me", "you" and "I don't know". To further enhance the animacy of PiNAOqio custom gestures are added by defining joint angles as a function of time through choreograph. These gestures are more specific and complex to generate including gestures to "wipe forehead" or change eye colour LED, so the custom dictionary was optimized for the specific, fully-interactive passage.

A string parsing algorithm has also been developed as part of the gesture controller and is responsible for handling the gestures of custom built dictionaries. Before the interactive storytelling commences, the story string is parsed according to key words and phrases of the two dictionaries. The gesture controller is then able to determine the timing and the place to input gestures associated with speech.

8. Speech Recognition

A speech interaction component, Speech Controller, was introduced to the interactive version of the robot before the reading began. Aside from basic salutations, the interaction allowed for the robot to get feedback on the user whether the volume and speed were fine or should be changed before the reading began.

This was implemented as a python class, which was called from the main state machine at an initial stage. Google speech was used as the speech processing tool, as it gave the highest accuracy when compared to other easy to implement tools, such as sphinx. Once the speech was recognized, the string of text was checked with a dictionary in order to detect the key words in each sentence, which varied as the conversation progressed. Depending on the result, one answer or another would be given by the robot, along with a specific action if required, such as increasing the volume or speed of speech.

V. Experiment

Pinaoqio was tested over the course of three days in room 508 of the EEE building in Imperial College London to have a testing environment of complete silence. Keeping a low noise level was important for the reliable operation of the speech recognition module, but also to make the story-telling experience more enjoyable by reducing distractions.

The book of choice was "An Eagle in the Snow" by Michael Morpurgo, primarily because it is a child's book. However, this book also has large font and thick pages facilitating optimal performance of the page-flipping and OCR modules. The testing consisted of reading three passages of the book, firstly using an audiobook recording taken from Audible, secondly a fully interactive PiNAOqio with gestures and speech recognition and finally, a non-interactive PiNAOqio version. Each of these methods consisted of different story telling versions that were then compared. The passages were also chosen so that there is a logical link between them and to allow the user to follow the story.

After each passage, a questionnaire was completed by the users to test the text comprehension of that passage and the user experience for the version used. The questionnaire was organized into three main sections, covering personal information (age, sex, robot exposure), passage comprehension and information retention through multiple choice questions and user experience through a semantic differential scale assessing Animacy, Likability, Perceived Intelligence and Safety.

VI. Results

Of the 23 people that participated in the experiment, the average age was 20.35, with a standard deviation of 1.41. From the whole population, 26.1% were females and the rest males. On average, the users that undertook the evaluation session read 10.41 books annually, so one could argue in favour of a familiarity with books.

The results can be broken down into two main sections: information retention and user experience. For information retention, users had to answer four multiple-choice questions for each passage and their score is determined as a percentage. User experience on the other hand, is assessed through a semantic differential scale, in which users were asked to indicate their position on a scale between two bipolar words, the anchors (Bartneck et al., 2008). This scale measured the user experience for the narrators' Animacy, Likeability, Perceived Intelligence and Perceived Safety on a scale of 30, 25, 25 and 15 respectively. Furthermore, for the user experience results were separated into different groups according to gender and robot experience in order to investigate any correlation between a group and an experience.

The results are summarized below, in table 1.

	Audiobook	Interactive	Non-Interactive
Information Retention (%)	73.53	77.94	58.82
Animacy (/30)	19.88	21.59	12.53
Likability (/25)	18.11	20.35	15.94
Perceived Intelligence (/25)	18.76	18.12	15.41
Safety (/15)	10.47	11.12	9.65
Averages (/95)	67.22	71.18	53.53
Male UX (/95)	67.77	71.92	53.54
Female UX (/95)	65.5	81.75	53.5
Robot Exp. 1 UX (/95)	68.75	74.5	53.25
Robot Exp. 2 UX (/95)	71.25	70.75	57.75
Robot Exp. 3 UX (/95)	64	70.63	50.75
Robot Exp. 4 UX (/95)	71	64	60
Robot Exp. 5 UX (/95)	n/a	n/a	n/a

Table 1 - Results

VII. Discussion

According to the hypothesis, the fully-interactive version of narration implemented by PiNAOqio would improve amount of information processed and remembered and also enhance the user experience compared to both the audiobook version and the non-interactive version. Based on the above results, the validity and accuracy of our hypothesis is evaluated, while an attempt is also made to explain the factors leading to such results.

In terms of information retention, the interactive version entails the highest percentage of correct answers with 77.94% of them being correct, 4.41% more than the audiobook version and 19.12% more than the non-interactive version. When comparing the interactive with the non-interactive versions, it is expected that any variation is a result of the additional gestures implemented of the former. This results supports the Social Agency Theory, according to which the combined effect of multiple interaction techniques exceeds the individual effect of each one (Ham, Bokhorst & Cabibihan, 2011). Moreover, the difference between the audiobook and the fully interactive version is not as significant. While, gesturing creates a more rounded experience engaging vision in addition to hearing, the voice used by PiNAOqio is not able to achieve the same expressive articulation used in the audiobook. In fact, being narrated by a professional human storyteller, the audiobook incorporates emphasis at critical point, pauses for dramatic effect and other narration tools to enhance the storytelling experience. It is interesting to note, that the benefit of adding gestures outweighs the drawback of using a what some called a "robotic voice" lacking the natural aptitude of expression.

The fully interactive version also had the highest ranking in the overall user experience evaluation, 4.17% and 18.58% more compared to the audiobook and non-interactive versions, respectively. It also ranks the highest in the subcomponents of Animacy, Likability and Perceived Safety, while it only lags to the audiobook in terms of Perceived Intelligence. There were two main factors contributing to that; firstly, the interaction designed was relatively straightforward, not including a refined artificial intelligence module to interact with user. Moreover, at certain point during storytelling by PiNAOqio, pauses would be inserted into incorrect positions, thus making the robot appear less intelligent.

An interesting result emerges by separating the group by sex. While the overall user experience, including Animacy, Likability, Intelligence and Safety, is approximately the same for both males and females in the audiobook and non-interactive versions, this is not the case for the fully interactive version. Females, ranked the user experience for the latter with an average of 81.75/95, which 10.3% higher than the males' average score. This could be explained by the appearance of the robot which appeals more to the female population, possibly due to the 'cuteness' factor of the robot being appreciated more. However, since the female population was only 26.1% of the total population, more data would be required to present a statistically significant result.

A final point that can be drawn from the results of the evaluation session has to do with user experience as a function of exposure to robots. For the fully interactive version, which is the one with the most added functionalities, there seems to be a clear, negative relationship between user experience and exposure to robots. People that are more familiar with robots are less fascinated by features they have already seen and experienced and are more likely to notice an inconsistency or a malfunction. These two factors contribute to the decline of user experience by 11.1% as robot exposure increases from 1 to 4 on a scale of 5, as shown in table 1.

VIII. Future Work

Despite the success of the project in building a system which is more than adequate to be tested on users and support the main hypothesis, there are still opportunities for further enhancements. Further work could be carried out to make the robot smarter, perform a quantitative engineering analysis on users and compare the system with state-of-the-art rival systems.

A major difference between PiNAOqio and a human storyteller is the ability of the human to interpret literary tools used by the author to convey meaning. One solution for making PiNAOqio 'smarter' could including saving the text read by the OCR module in memory and using it for future user-interaction. Currently, strings are fed to PiNAOqio to tell the story and are then discarded, which makes it impossible to acquire a generic perspective. If, however, larger passages were being stored, then more refined algorithms could be developed aiming to grasp the main points of the book, such as who is the main character and the mood of the story. Thereafter, PiNAOqio could engage in a more personal manner with the user, asking him questions specific to the story. It is expected, that by enhancing PiNAOqio's functionalities, it will be able to emulate a human storyteller more closely and provide a better experience to the user.

While a lot of care was put into making a fair questionnaire, this method of evaluating a system will always withhold some degree of subjectivity. A questionnaire would be even less reliable in the case of the system being evaluated on children, as they are less likely to be objective when completing it due to their lack of interest [Goodman, 2001]. Another measure of user experience involving emphasis in engineering techniques

could include recording the motions of the user during storytelling as well as his gaze and pupil dilation. Looking at the overall motion of the user during storytelling could be indicative of the attention he is paying at the narrator while tracking gaze and pupil dilation could provide a measure of the user's interest in the story (Wang, 2009).

Finally, through research of the relevant work in the area of robotic storytelling, the avatar storyteller version has been identified as the most commonly implemented solution. This is mainly due to the low cost of a virtual storyteller which can be implemented in the form of a tablet application, compared to an actual robot associated with significant fixed costs. By comparing our proposed solution with the current benchmark for a virtual storyteller, we would be in a position to assess whether the development of a robotic storyteller is a viable project in the market and what is the true value added by this implementation.

IX. Conclusion

PiNAOqio is designed to provide educational entertainment for young people that would otherwise be at risk from the adverse effects of long-term use of electronic devices. It consists of a novel storytelling system using the NAO robot, a page flipper and an OCR module to read physical books. After being tested, it was found that the fully interactive version improves information retention by 4.41% and 19.12% and user experience by 4.17% and 18.58%, compared to audiobook and non-interactive version, respectively. Further enhancements are being proposed to improve the intelligence of PiNAOqio, perform a quantitative analysis of results and to test the system against an avatar solution.

X. References

1. Bartneck, C., Kulić, D., Croft, E. and Zoghbi, S. (2008). Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. International Journal of Social Robotics, 1(1), pp.71-81.
2. Wang, Joseph. (2009). Pupil Dilatation And Eye Tracking.
3. Kory, Jaqueline. (2016) Storytelling Companion - Personal Robots Group. Personal Robots Group, 2016. Web. 28 Oct. 2016.
4. TROBO. (2016). TROBO The Storytelling Robot: What Are Trobos?
5. Aldebaran. (2016). Naoqi Apis — Aldebaran 2.1.4.1 Documentation. Doc.aldebaran.com
6. Goodman, Robert. (2001). Psychometric Properties of the Strengths and Difficulties Questionnaire. *Journal of the American Academy of Child & Adolescent Psychiatry*, 40(11), pp.1337-1345.
7. Qumsiyeh, D. (2016). Linear Book Scanner - The open-source automatic book scanner. *Linearbookscanner.org*, [Online]. Available: https://linearbookscanner.org. [Accessed: 17- Nov- 2016].
8. (2016). Book scanner ScanRobot® 2.0 MDS - The automatic page turner solution. *Treventus.com*, [Online]. Available: http://www.treventus.com/automatic-book-scanner-scanrobot.html. [Accessed: 17- Nov- 2016].
9. M. Dredge, T. Stuart. (2013). Children's Reading Shrinking Due To Apps, Games And Youtube. *The Guardian.*
10. Laguerta, X., Korkontzelos, P., Warner, N., & Rusu, M. (2016). PiNAOqio: Story-telling for children using physical books and HRI. Imperial College London.
11. Caldari, M. (2015). Camera Capture on iOS. *Objc.*
12. Charlton, J. (2015). Mechanical System For Lifting And Turning Bound Pages. Web. [Accessed: 28 Oct. 2016].
13. Rusu, M. (2016). Making PiNAOqio Read: Digitizing Text From Physical Books. Imperial College London.
14. W. K. Lau, Bernard and Wilson W. C. Tse. (1993).Hospitalised Childer Stress: Psychologi cal Effect Of Physical Illness And Hospitalization On The Child And The Family.
15. Freire P., & Slover, L. (1983). The Importance of the Act of Reading. *The Journal of Education* 165(1).

XI. Appendix

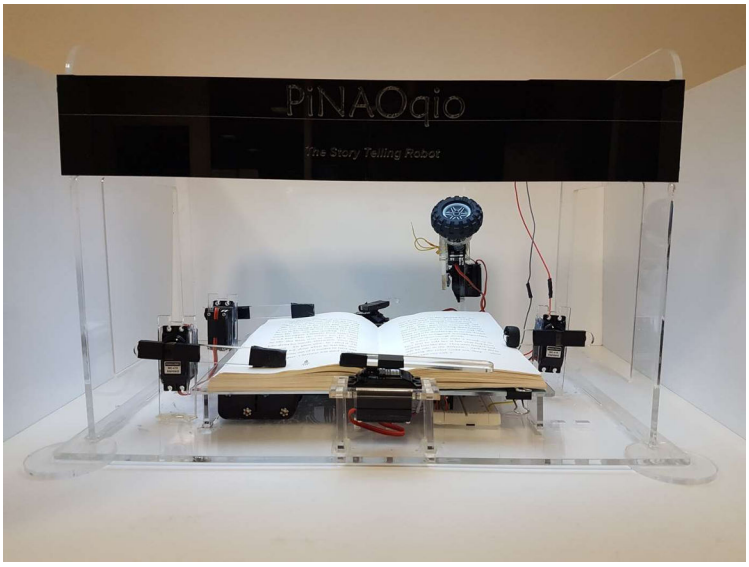

Figure 5 - Final System Assembly