CHAPTER 1

INTRODUCTION

The realm of offline device hacking, also known as offline device vulnerabilities, presents a unique challenge in the cybersecurity landscape. Unlike their online counterparts, these isolated systems lack the traditional entry points like network connections, making them incredibly resilient against conventional cyberattacks. We aim to investigate and demonstrate alternative techniques with the potential to bypass security measures and extract sensitive information from offline device vulnerabilities. We aim to investigate and demonstrate alternative techniques with the potential to bypass security measures and extract sensitive information from offline device vulnerabilities. We aim to investigate and demonstrate alternative techniques with the potential to bypass security measures and extract sensitive information from offline device vulnerabilities.

1.1 Unseen Threats.

In our increasingly connected world, where smart devices permeate every aspect of our lives, we often focus on the security of online systems. However, lurking in the shadows are vulnerabilities that affect **offline devices**—those that don’t directly connect to the internet but play a crucial role in our digital ecosystem.

But here’s the catch: offline devices are not invulnerable. They may not communicate directly with the internet, but they often rely on gateways, controllers, or other intermediaries to relay information. These intermediaries bridge the gap between the offline and online realms, ensuring seamless dataflow and centralized management.

1.2 What Are Offline Devices?

Offline devices encompass a wide range of technology, from industrial sensors and legacy machinery to home appliances and medical equipment. These devices operate independently or within local networks, collecting data, controlling processes, and interacting with the physical world. Unlike their internet-connected counterparts, they lack direct access to the web, making them seemingly immune to cyber threats.

1.3 The Silent Threat.

First the analysis will contain information about the state of the art and how the idea of advanced rubber ducky came to be. Secondly, it is described how the hardware and software is implemented for performing the experiment. Thirdly, the results and conclusion will be discussed and finally recommendations for future research are given.

CHAPTER 2

ANALYSIS

2.1 State of the Art.

2.1.1 Introduction.

The objective of this chapter is to find what research already exists, what is still unknown about self-explanation, social robotics and rubber duckies. The research will be used during the ideation phase as inspiration and to give context to the Rubber Ducky as an Advanced Rubber Ducky research.

Firstly, this chapter will discuss what self-explanation is, how it has been used and how it works. After that, examples of social robots, what they have been used for, and how socialibity is implemented are given. Thirdly, what position Rubber Duckies take within scientific research will be looked at. Finally, what can be concluded with respect to the Rubber Ducky as an Advanced Rubber Ducky research will be discussed.

2.1.2 Self-explanation devices and concepts.

The concept of self-explanation was first researched by John Sweller & Graham A. Cooper [5] who found that self-explaining worked out algebra examples improved solving speed more rapidly than conventional methods. Thereafter, several others including Chi et al. [6], O'Neil et al. [7] and Booth et al. [8] have also researched this concept. Pittsburgh Science of Learning Centre defines self-explanation as follows [9]:

“A self-generated explanation of presented instruction that integrates the presented information with background knowledge and fills in tacit inferences.”

Furthermore, Wylie and Chi[10] use the definition.

“Self-explanation is a domain general constructive activity that engages students in active learning and ensures that learners attended to the material in a meaningful way while effectively monitoring their evolving understanding. Several key cognitive mechanisms are involved in this process including generating inferences to fill in missing information, integrating information within the study materials, integrating new information with prior knowledge, and monitoring and repairing faulty knowledge.”.

The definitions are mostly interchangeable but the definition of Wylie and Chi incorporates more facets of self-explanation, and focuses on the activity rather than the explanation. For this research the definition of Wylie and hi will be used since the research involves testing the activity rather than how the self-explanations are formed.

Self-explanation has been used in several different ways. The two ways that will be discussed are prompted self-explanation [6], [7] and corrective self-explanation [8]. With prompted self-explanation the person who performs the self-explanation is asked during or in between the learning process to self-explain. With corrective self-explanation a person is given incorrectly worked out examples with the task to explain how and why they are incorrect.

2.1.3 Sensitive Artificial Listener.

It is a framework for virtual agents that are made to listen and react to a person talking with verbal and nonverbal communication. It has been developed with natural human computer interaction in mind. It has for instance been used for data collection for emotion recognition.[11]

2.1.4 Thinking aloud pair problem solving (TAPPS).

An interesting method of problem solving is Thinking aloud pair problem solving, or TAPPS developed by Lochhead and Whimbley[12]. It is a method which requires a pair of people to sit together and at least one of the people to talk aloud while solving the a problem and the other has to keep its partner talking and asks for elaboration on certain aspects. This causes the person explaining to grasp certain concepts quicker.

2.1.5 Social robots Effectively conveying emotions.

There are a lot of social robots today, some of them are 3d printed open-source designs like, inmoved, ASPIR and Rubber Ducky. These robots are made to advance the field of robotics by the community and make the robot space more accessible for education and makers. These robots can be used to socially interact with humans. A well-known method of adding social abilities is with displaying emotions. There are several methods in literature that are used to have robots convey emotions. How the emotions are perceived could be divided into three main factors: based on visual cues, based on audible cues and depending on the context. McColl and Nejat [13], Erden [14] and Sial et al. [15] all make use of the visual cues with body language. McColl and Nejat try to make the body language of the robot portray that of a human actor while Erden tries to optimise recognition by designing body language specifically for a specific robot. Erden on the other hand uses only the movement properties of the robot (acceleration curve and speed). In this way the robot can still perform tasks without having to invest in extra indicator parts, keeping the cost low. Erden has the most effective method as it come to recognition rates (75% against 42% and 61%) However the method of Sial et al. makes it possible to still perform the tasks it was designed for. McColl and Nejat's method might be in theory the most human like. Another effective way of communicating emotions is by using facial expressions. For facial expressions it is beneficial to be able to make them asymmetric [16] and the mouth seems to be more important for interpreting emotions correctly then the eyes [17]. If a robot needs to convey more complex emotions with facial expressions you can't just mix the different aspects found in human facial expressions [18]. Different colored lights with the help of RGB-LEDs can also be used, but they are only effective to increase the intensity of certain emotions [19].

To convey robotic emotions another way is to use audible cues. Jee et al. [20] discuss how this can be done with the use of music, Crumpton and Bethel [21] discuss how emotion is implemented in synthesized speech and Fernandez De Gorostiza Luengo et al. use emotive sounds [22]. Crumpton and Bethel write that using modern open source speech synthesizer solutions that you can have people recognize certain emotions better than chance just by intonation. They also discuss the fact that certain words within their test data may have influenced they results. This could mean that if a robot would use words with a negative connotation (cried, sad, greed) that its emotions are assumed to be negative. Fernandez De Gorostiza Luengo et al. have found that when using emotive sounds, often the sounds within the same category get confused. The sounds for Agreement, Encouragement and Greeting were mistaken for each other sometimes. Jee et al. experimented with music as the robots' emotions. They only designed music for 4 emotions: Happy, Sad, Fear, and Dislike. The participants of the study reported that the music caused the emotion to be perceived stronger then with facial expressions.

Also, an important aspect for conveying emotions correctly is the context. For instance, Thimmesch-Gill et al. [23] have looked into the perception of robot emotions while under stress. Where, for instance, low arousal poses where rated with relatively higher arousal and high arousal poses where rated with relatively lower. Li and Chignell [24] have also tested the context and found that emotions were better understood in the appropriate context. Fernandez De Gorostiza Luengo et al. [22], Haring et al. [25] and Breazeal [26] all agree that implementing multiple ways of communicating social cues is beneficial for the understanding of the emotion of the robot.

2.1.6 Rubber Duckies.

A rubber ducky is a floating toy made in the shape of duck. The original design was from Peter Ganine and made its first appearance in mid to late 19th century. He patented the toy and sold over 50 million of them. Since then the rubber duck has been popular amongst different people and media. An example of rubber duckies being used in research is the case where a big load of rubber duckies fell from a cargo ship and then the rubber duckies where used to track ocean currents [27].

2.1.7 Conclusion.

Based on current research the idea of a physical reactive listening agent is an interesting and viable concept. Based on anecdotal evidence rubber duck debugging seems to be helpful for some people, and research shows the effectiveness of self-explanation and TAPPS. It is also very well possible to make a robot as Rubber Ducky social. Besides that, there has been few research about rubber duck debugging and whether it can be improved. Rubber Ducky as advanced rubber ducky would use prompted as well as corrective self-explanation. By utilizing the eyes and body movements inspired by the robot Nao it is possible to create convincing portrayal of emotions which can help the communication process between man and robot.

2.2 Ideation.

The first stage was to get the plan of what to do with Rubber Ducky. The brainstorm was setup to find an answer to the following question: “What meaningful purpose can be fulfilled by Rubber Ducky using its nonverbal social potential and how can this be implemented in an effective manner?” After quite some brainstorming and looking at what was out there the ideas were put into a mind map.

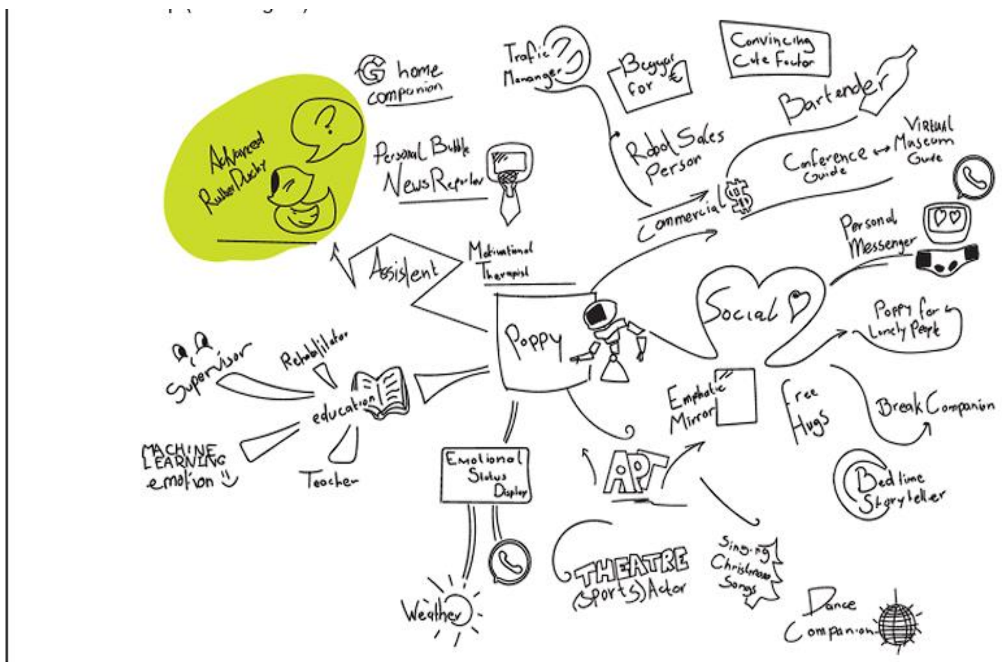


Fig 1: Mind map of possible purposes of Rubber Ducky.

The next phase was to find out which ideas were novel and meaningful and could be tested within the scope of this project. The concept of advanced rubber ducky was chosen for it didn’t need a too high skill level in programming, could provide actual benefit to people and was relatively easy to test. Also, there is not a large amount of research in this field and thus could fill a knowledge gap.

2.2.1 Advanced Rubber Ducky.

The idea of the rubber ducky comes from the term “rubber ducky debugging” which is used in the programming community. Rubber ducky debugging is debugging by explaining your code to a rubber ducky, or some other lifeless object, so that you understand your code and the interactions between them better and find where is goes wrong. The idea of the advanced rubber ducky is that these programmers talk to a humanoid robot instead of to a lifeless object. For coming up with a way to test the concept of advanced rubber ducky different scenario where setup and evaluated.

2.2.1.1 Scenario 1.

Heather works for Firedroid, a mobile app development company. Times have been rough for Firedroid since the last game they brought out wasn’t as good received as they had thought. Heather had been stuck on this one bug where main character wouldn’t jump whenever the menu was closed. After an hour of staring at her code she leans back in her chair. Rubber Ducky sees her and turns his towards her and starts to wave. Heather laughs, gets up and lifts Rubber Ducky to place him on her desk. She starts explaining him what the code does. Rubber Ducky listens carefully and looks confused at her when she explains what the pause boolean does. She takes another crack at explaining it and noticed that when the pause boolean is set to false the controls never get re-enabled. It all becomes clear to her and she wonders why she didn’t see that sooner.

2.2.1.2 Scenario 2.

Carlos is a first year Electrical Engineering student. He is working out a diagnostic test and comes across a problem he couldn't solve immediately. He had always learned to solve electrical schematics using the rules about parallel and series. But he got confused with this one problem. He is afraid that he needs to use Kirchhoff's current and voltage law which he doesn't yet master. He grabs Rubber Ducky and sets him on the desk. He explains Rubber Ducky what he knows about the voltage and current laws then grabs his book and explains everything he missed in his explanation. Rubber Ducky acts to be listening. After doing this Carlos returns to the problem and solves it within ten minutes.

2.2.1.3 Scenario 3.

Simone wants to tell her parents about her sexual orientation but she has been to shy for some time. She grabs Rubber Ducky from the book shelf and takes him to her room. She begins to tell him about how she really likes this girl in her class. Rubber Ducky scratches the back of his head and looks confused. "Right" thinks Simone. "This is not going to work like this." She dials Rubber Ducky in to react mad. And she tries again. With seeing the angry eyes of Rubber Ducky, she tries to relax him in the same way she would do for her parents. She then dials in that Rubber Ducky reacts sad and tries again, then also for disappointment and finally for acceptance. She now feels well prepared to talk to her parents. She goes downstairs and begins to talk to her parents.

2.2.1.4 Conclusion.

The first scenario shows the most potential for testing. That is because there was excess to a target group with a known skill level in programming that would make it possible to make a challenging test while not being too difficult. The third scenario could be used for a wide target group; however, it would involve confronting people on a personal level which makes it more difficult to set up an ethical test.

2.2.2 Requirements.

In this part a list of requirements will be set up to be able to perform the experiment. For Rubber Ducky to be used as an advanced rubber ducky, it will need to be able to react to the person explaining things to in a meaningful manner. This can be achieved by controlling the servo motors and screens in the head. This could be done by making Rubber Ducky respond automatically, however this is probably too difficult to realize within the time frame. A more reasonable approach would be to make Rubber Ducky controllable and have the researcher puppeteer the movement and eyes from a distance.

For the research to be valid:

1) Rubber Ducky must able to react to a person by being remote controlled.

2) Rubber Ducky must able to react understandable by the test participant. To be able to be controlled remotely.

1) There must be an interface the researcher can interact with.

2) The method of controlling must give the researcher the ability to let Rubber Ducky express intentions/emotions understandably within a reasonable time frame.

The scenario that will be tested is the scenario where Rubber Ducky is used as a debugging tool. For this Rubber Ducky will need to react like a person who is listening to someone explaining his or her code. In other words, Rubber Ducky needs to be able to display standard conversational cues and some emotional cues that fit within this context.

1) Rubber Ducky should be able to nod (say yes)

2) Rubber Ducky should to be able to shake it head (say no)

3) Rubber Ducky should be able to look confused

4) Rubber Ducky should be able to look like it’s thinking

CHAPTER 3

IMPLEMENTATIONS

3.1 Hardware setup.

3.1.1. USB ATtiny85.

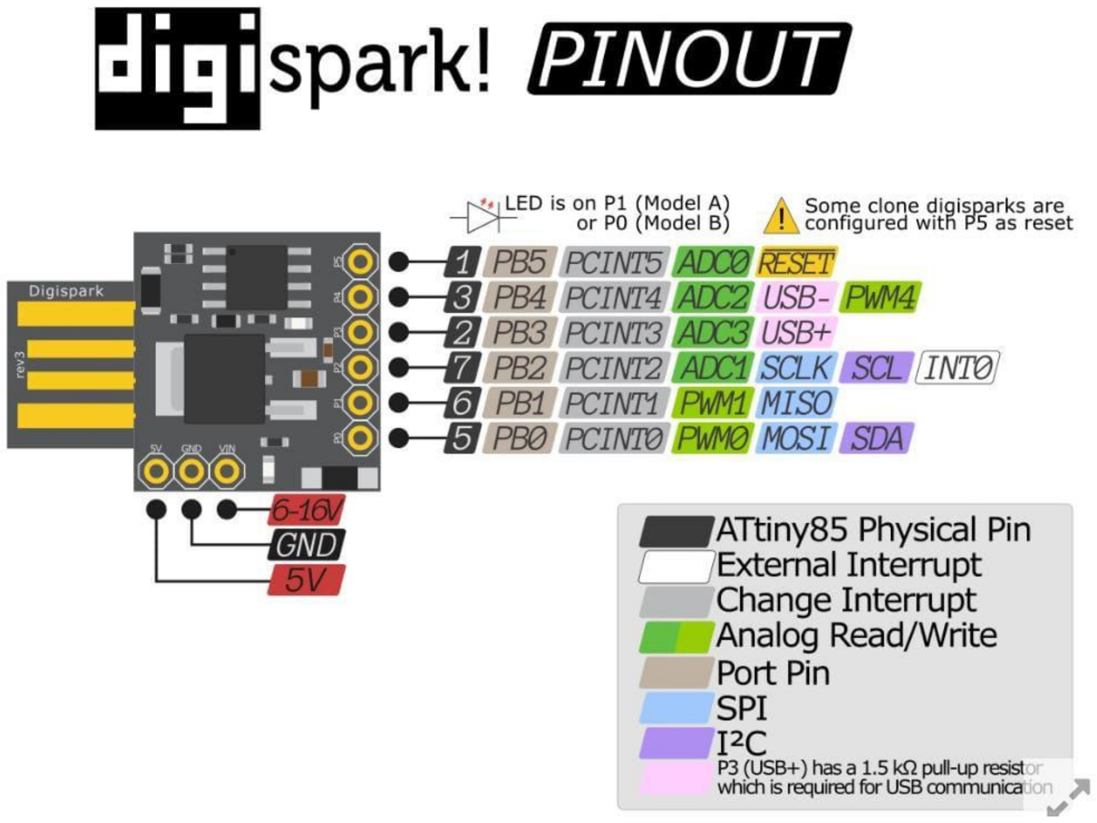

digispark! PINOUT
LED is on P1 (Model A) or P0 (Model B)
Some clone digisparks are configured with P5 as reset
Digispark
rev3
P5 P4 P3 P2 P1 P0
5V GND VIN
1 PB5 PCINT5 ADC0 RESET
3 PB4 PCINT4 ADC2 USB- PWM4
2 PB3 PCINT3 ADC3 USB+
7 PB2 PCINT2 ADC1 SCLK SCL INT0
6 PB1 PCINT1 PWM1 MISO
5 PB0 PCINT0 PWM0 MOSI SDA
6-16V
GND
5V
ATtiny85 Physical Pin
External Interrupt
Change Interrupt
Analog Read/Write
Port Pin
SPI
I²C
P3 (USB+) has a 1.5 kΩ pull-up resistor which is required for USB communication

Fig 2: Pin diagram.

ChapeKeylogger software has the capability to record every keystroke a user makes to a log file. It can record information such as user id, password, instant messages, and e-mail. Detail of Keyloggers performance and whether they need administrative access to the target machine or not are discussed in [19]. In recent years there has been some hardware development that enhances the task of ATtiny85. In this section we describe the specification of one of that hardware that we use in this research.

The USB Rubber Ducky has been developed by Hak5 [4]. This USB key includes a 60MHz programmable microcontroller and an SD slot. It behaves like a keyboard and it looks like USB flash drives. It can be easily hidden on a computer's device port. Another feature of this device is that it may be hidden in the task manager; it is assumed that its power consumption may be revealed by physical measurements. However, to use the USB Rubber Ducky we need physical access to the victim's machine and we need to write a malware to be injected into the device.

Computers inherently trust devices that claim to be a HID. It's through these devices that humans interact with and accomplish their daily tasks on all computers including desktops, laptops, tablets, and smart phones. The USB Rubber Ducky is a keyboard emulator disguised within a USB thumb drive case. It has been used by IT professionals, penetration testers and hackers since 2010 and has become the most widely used commercial keystroke injection attack platform in the business. Combined with its scripting language, malware payloads can be written and deployed.

Many people leave their computers unattended, even if only for a few minutes. These few minutes is all it takes for personal information to be stolen from the victim's machine by a malicious hacker using the USB Rubber Ducky or a similar device. Whether it is a local account or a Microsoft account, vulnerability exists in Windows and many other operating systems.

Clear text passwords are stored in the computer’s main memory that can be extracted using a program called Mimikatz designed by Benjamin Delpy [22]. One of the many functionality included in mimikatz is the sekurlsa function, which specifically targets logon passwords and hashes.

This research exploits Windows vulnerability utilizing the USB Rubber Ducky. In this project the machine has windows defender for its antivirus. An account is created on the victim’s machine and all activities are targeting this account. In the next section we describe the details of the tools and technology needed to construct the malware payload and for launching an attack

3.1.1.2 Electronics.

Different electronics were used than in the Stock version of Rubber Ducky, except for the servos. Instead of the stock ODROID-XU4 and an LCD screen, two arduinos were used with breakout boards for driving the servos and the adafruit 8x8 Bi-Color LED Matrices. That is 14 because Arduino is very popular open-source hardware with a lot of community and library support. Also, it is embedded electronics and thus there is no operating system that takes up extra processing power and could cause instability issues. One of the Arduinos and all the servos received power from a 12Vdc power supply with a maximum output of 45W. The other Arduino received power from a USB connection. Some extra ICs were used to have everything working together.

ATtiny85 connection circuit.

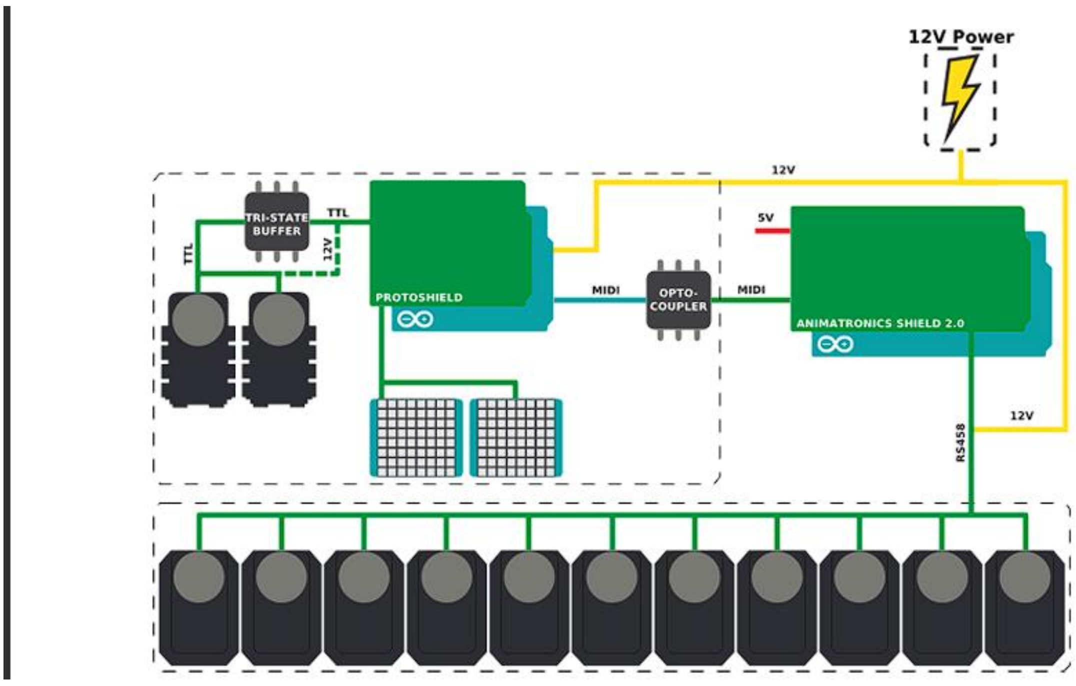


Fig 3: ATtiny85 connection circuit.

3.1.1.2.1 Head electronics.

In the head of Rubber Ducky an Arduino Uno was used. This Arduino drives two servos in the neck of Rubber Ducky as well as the LED Matrices that are used as Rubber Ducky's eyes. On the Arduino there is a Proto Shield v2 with all the supporting circuitry.

Arduino Uno:

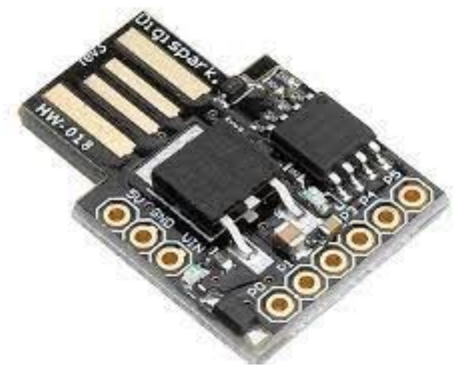


Fig 4: ATTiny85

List of hardware on the Arduino UNO.

1) This is the TTL connector for the two Dynamixel AX-12 servos in the neck. This is also the point where the 12Vdc power input is.

2) This is the SN74AS241N, a tri-state buffer, this is used to improve the out- and input impedance of the TTL connection between the servos and the Arduino.

3) This is the TSR 1-2450, this is used to efficiently step down the 12Vdc to an Arduino friendly 5Vdc.

4) This is the 6n137, an optocoupler, this is used to receive MIDI signals from the other Arduino with a galvanic separation.

5) These are the headers for the two LED matrices.

6) This is the connector for a MIDI connection from the other Arduino.

Adafruit 8x8 Bi-Color LED Matrix.

These LED Matrices were used to to display emotive robot eyes on the head. These LED Matrices were chosen because library and community support and that they give enough flexibility to display eyes in a sufficient manner while still being easy to program.

3.1.1.2.2 Base electronics.

In the base of Rubber Ducky an Arduino Mega ADK is used in conjunction with an animatronics shield 2.0. This Arduino was chosen because it has an extra female USB connector which can be used for a midi controller. This Arduino is used to drive all the servos in the body, it interprets the instructions from the MIDI controller give the head Arduino instructions which servos to move and what to display on the LED matrices.

Arduino Mega ADK.

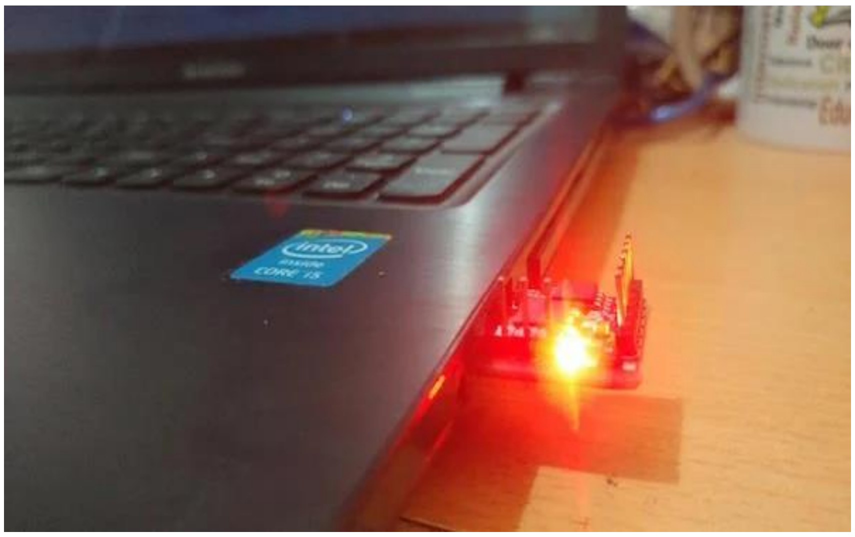


Fig 5: ATTiny85 plugged into a computer.

List of hardware on the Arduino Mega ADK with animatronics shield.

1) This is the Midi out connector. Here the wires that go to the head Arduino are coming from.

2) This is the circuitry that translates the serial signal from the Arduino to the RS485 protocol with the help of a LTC485 integrated circuit.

3) This is the connector to the servos.

4) This is the USB connection for reprogramming and to power the Arduino.

5) This is the USB connection to connect with the midi controller.

Korg nanoKontrol 2 This controller (see image 8) is used because it has enough actuators, it uses the widely implemented midi protocol, the layout of all the actuators is clear and is not too expensive (see image 10 for controller mapping).

Dynamixel MX-28 & Dynamixel AX-12 The dynamixel servos are useful for robotics and mechatronics because they are a fully integrated solution. All the control mechanics and electronics are integrated and all the instructions that it needs are send to the onboard controller by TTL (for the AX-12) or RS485 (for the MX-28). They can also be daisy-chained. The main differences between the two models are that the MX-28 has more power, a larger operating angle and a higher resolution.

3.2 Software setup.

3.2.1 Dynamixel Firmware setup.

First index numbers need to be assigned to controlling all the Dynamixel servos. I assigned the index numbers according to the motor naming convention [29] using the dynamixel wizard in the RoboPlus software[30] and a USB2Dynamixel(see image 9). Also, in this software the maximum position values of the dynamixels is filled in for each motor. These values were determined by putting the different joints in maximal positions and reading out the value using the dynamixel wizard. The process was similar for determining the neutral position for each joint.

3.2.2 Rubber Ducky software.

There are 2 arduinos in this version of Rubber Ducky and thus also two arduino programs were created.

3.2.2.1 Rubber Ducky base script.

This script:

Interprits midi input from the midi-controller using the MIDI library [31] and the USBH_MIDI library[32]. This data is then used to determine program actions.

Stores the position values of certain animations.

Calculates servo positions from stored values for different intensities of animations.

Sends position data for the servos in the neck and which eye animation that needs to be displayed to the Arduino in the head with the midi protocol.

Offsets the chest and arm position values to make Rubber Ducky appear to be breathing.

Sends position data to the dynamixel MX-28's using the DynamixelReader library [33].

The base script is based on code of Edwin Dertien that has been used in other projects with the Animatronics Shield. In the main loop of the program there is a 100 Hz clock for addressing the Dynamixels for not overloading the RS485 connection. This timer is also used for sending Midi signals to the Arduino in the head for the same reason and it is used to time the animations. At setup all the serial/midi connections are initialized. As soon as there is a connection with the midi controller all the servos are set to the default position and the P-gain on the internal memory of the servos is lowered. This done to have the movements look smoother and more natural (since the P-gain has the most influence on the acceleration near the setpoint).

The midi controller sends out a midi message every time a button is pushed or a slider is changed. This message contains a value unique for every slider, button and knob and a value to determine what the slider, buttons or knob is set to. These values are used to determine the action of the program. For instance, the sliders are used to change the position values of the joints in the arms. If the slider is in the middle position the joint is in the default position and all positions of the slider higher than that are mapped between the default and maximum value. For lower values vice versa. This is to make it easy to return to the default position of the joint.

This script also stores different positions in a certain sequence to display body language to the user. The animation data is stored as an array of arrays, of positions of all the joints, and a timestamp variable. For playback of the animations a parametric approach was chosen. This was achieved by first designing an animation and then later increasing or decreasing the difference between sequential joint positions with a certain value. By controlling this value, the animation seems more or less intense. To make sure that the animation looks like a continuous movement without too many data points, linear interpolation between the stored values was used.

1) previous position +c urrent time * next position − previous position next timestamp − previous timestamp = c urrent position

Current time stands for a variable that counts up using the 100 Hz clock mentioned earlier from 0, the beginning of an animation, to a value that stand for the end of the animation. Each position has a timestamp for when that position should be assumed according to the current time 19 variable. With every animation there is also a value that stands for the eye animation send to head Arduino.

3.2.2.2 Rubber Ducky head script.

This script:

Interprets midi input from the Arduino in the base using the MIDI library [31]. This data is then used to determine program actions.

Stores bitmap data and sequences for different eye animations.

Sends the right bitmap information to the Led-matrices for playback of eye animations using the Adafruit_GFX [34] and Adafruit_LEDBackpack libraries [35].

Sends position data to the dynamixel AX-12's using the DynamixelSerial [36] library.

The head arduino listens midi signals from the base arduino. Midi has different channels and the different channels are used for controlling different functions in the head. Channel 1 and 15 are used to set the angle set point for the two servos in the neck and channel 3 is used to determine which eye animation should be displayed.

Eye animations are timed using the millis() function. Every eye animation is a loop of a couple seconds with 2 - 14 different frames.

The Dynamixels are directly controlled with the midi input from the Arduino in the base. With the midi value 64 corresponding to the default position of the servos and 0 and 127 corresponding to the maximum positions of the servos.

3.3 Animation design.

For designing the animations, a couple of steps were taken. First there was a small brainstorm phase to what kind of responses where needed for this application. Secondly there was a search for references by google images, drawings or performing the response myself to see what the response actually looks like. Thirdly designing each frame/intermediate position. Then playing the animation back and working out the timing and finally tweaking the positions, frames or timing.

For the eyes a eye design tool was made in Processing. This design tool exported an image that was made in the tool to an array of bytes that is understandable for the Arduino. There was tried to keep some conformity between the eyes. The suggested eyebrows are good to show most of the emotions.

3.4 Experimental Design.

For the experiment some sub research questions were written. RQ1: Do test participants find it helpful to explain their problem to an animated Rubber Ducky, a non-animated Rubber Ducky or nothing? RQ2: Do the test participants feel comfortable talking to Rubber Ducky? RQ3: Are the test participants able to connect in a social way with Rubber Ducky? The quality of puppeteering will also be measured to validate the research questions.

After signing an informed consent form (Appendix C), one participant at the time will make two assignments of seven minutes each. For each assignment, the Participant is going to solve it in a different way. One time the participant is asked to explain what they are doing to an animated Rubber Ducky and the other time it is asked to explain what they are doing to a stationary Rubber Ducky. A researcher for controlling Rubber Ducky and a camera will be present in the room as well.

A test protocol is written and can be found in appendix D. The order of assignments and whether or not Rubber Ducky is animated will be shuffled between participants according to the table below. This will be done to suppress/distribute the influence of difficulty, order and fitness to the solving method of the assignments. Every solving method has the same number of the three assignments and every possible order of solving methods is present 4 times.

3.4.1 Assignments.

In literature there is not much to find about rubber ducky debugging or similar methods to base the assignments on. However when searching for rubber ducky debugging, anecdotal evidence often leads to the use in programming. For both assignments the challenge is to understand how the different elements on screen are drawn. During the design of the assignments they were tested with colleagues and the aim was to not have the difficulty in the syntax. The assignments can be found in appendix E and are designed in and for the IDE processing.

3.4.2 Participant.

The participants are students from the program creative technology. This user group was chosen because the level of programming skills is known. The test was setup for within participant analysis.

3.4.3 Questionnaire.

For designing the questionnaire some research was done to the different things that need to be tested. So questions were set up that test the scales: puppertering quality, perceived social connection, perceived effectiveness and perceived comfort. For every scale different subscales are chosen and every subscale is tested with at least 3 different questions (see table below). Some subscales are taken from other research. C2, S3 and U2 are taken from Heerink [37] and Q3 and S2 are taken from Harms and Biocca [38]. All the questions where is the form of statements and the possible answers that could be given where 1, 2, 3, 4 and 5 with 1 being marked as fully disagree and 5 as fully agree. Besides that, there were also some demographics questions. Full questionnaire can be found in Appendix F.

3.4.4 Observations.

The video footage will be used to look for: Time to complete task in seconds(finish time). ---- Time the participant is telling Rubber Ducky what he or she is doing in seconds (narrating). Time the participant is explaining to Rubber Ducky what the code means in seconds (explaining). Time making communicative sounds in seconds (communicative sounds). Time talking to Rubber Ducky about not assignment related things (talking). Whether there are any anomalies. For the data analysis time for narrating and explaining are combined and called solving behavior, since it was not always clear what the difference between narrating and explaining was. The communicative sounds and talking are also put together for the data analysis and called other behaviour together with everything that the participant says that is not understood.

CHAPTER 4

EVALUATIONS

Test Data can be found in Appendix G.

4.1 Evaluation Method.

First off all the Cronbach's Alpha[39] of all the subscales are determined to validate the questionnaire. The Cronbach's Alpha is a method to measure whether different questions measure the same subscale. Everything above 0,7 is considered adequate and above 0,8 signals a good correlation between the questions. The answers can then be averaged into a new variable of the subscale. All the subscales of the same scale were then tested with a correlation analysis and if they significantly correlate the subscale variables are averaged into a scale variable. These scale variables can then be used to test different hypothesis. Different suspected hypothesis are connected with the sub research questions and then tested with SPSS using sample t-tests and correlation analysis.

The quality of puppeteering to validate the measurements.

H0: Just as many people are able to finish the first assignment as the second assignment

H1: It takes just as long to finish the first assignment as the second assignment

H2: People found the quality of puppeteering convincing

RQ1: Do test participants find it helpful to explain their problem to an animated Rubber Ducky or a non-animated Rubber Ducky?

H3: When Rubber Ducky is animated assignments are finished more quickly

H4: When Rubber Ducky is animated assignments are finished more frequent

H5: People find it helpful to use Rubber Ducky

H6: If people perceive Rubber Ducky as useful they finish assignments more frequent.

H7: If people talk more to Rubber Ducky they finish the assignment more frequent

RQ2: Do the test participants feel comfortable talking to Rubber Ducky?

H8: When people feel a social connection with Rubber Ducky they feel more comfortable talking to Rubber Ducky –

H9: If people say that they feel comfortable talking to Rubber Ducky they talk more to Rubber Ducky

H10: People feel comfortable talking to Rubber Ducky

RQ3: Are the test participants able to connect in a social way with Rubber Ducky?

H11: People connect socially with Rubber Ducky

H12: When Rubber Ducky is animated people talk more with Rubber Ducky

4.2 USB Ducky Hardware.

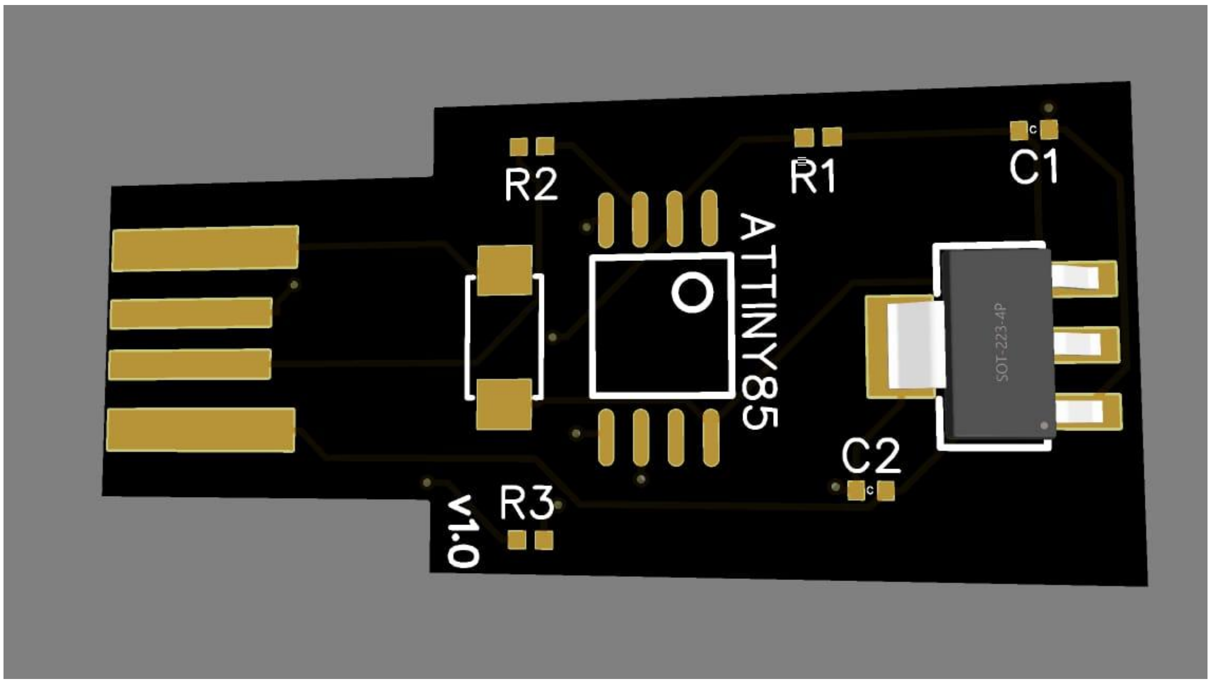


Fig 6: ATTiny85's back side's circuit structure.

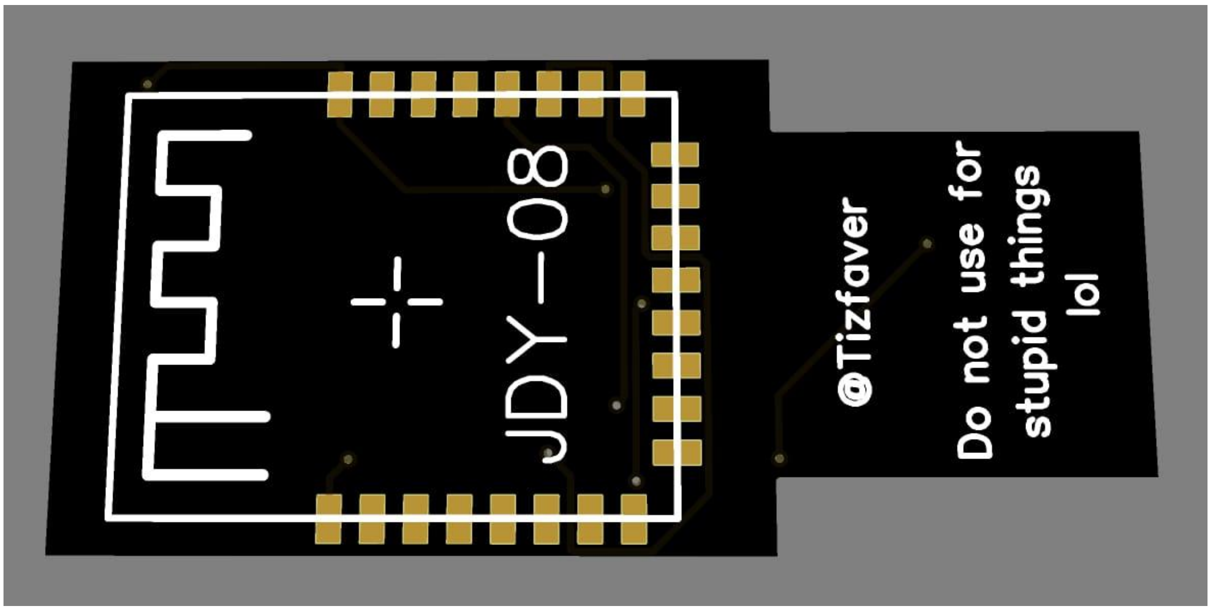


Fig 7: ATTiny85's Front side's circuit structure.

We used a USB Rubber Ducky for attack media (Hak5 [4]), this looks like a USB flash drive which can be plugged into the victim's machine. The average USB Rubber Ducky includes a 60MHz programmable microcontroller and a SD slot. Some of the features of this device include behaving like a keyboard; it does not show in the task manager and its power consumption may be revealed by physical measurements.

Form Factor: The USB Rubber Ducky typically resembles a standard USB flash drive, making it inconspicuous and easy to conceal.

Keystroke Injection: It functions by emulating a keyboard, allowing it to send pre-programmed keystrokes rapidly to a target computer. This capability enables various automated tasks and penetration testing activities.

Programming: Users can program the USB Rubber Ducky using simple scripting languages, such as Ducky Script, to define sequences of keystrokes to execute on the target system.

Payloads: The device can execute a wide range of payloads, including opening reverse shells, launching applications, downloading and executing files, and more, depending on the user's intentions and objectives.

Cross-Platform Compatibility: USB Rubber Ducky is compatible with multiple operating systems, including Windows, macOS, and Linux, allowing it to target a broad range of systems.

Community Support: There is an active community of users who share scripts, payloads, and techniques for using the USB Rubber Ducky effectively, making it a versatile tool for both security professionals and enthusiasts.

Legitimate Uses: While USB Rubber Ducky is primarily associated with security testing and penetration testing, it's important to note that its use for malicious purposes, such as unauthorized access or data theft, is il

CHAPTER 5

QUESTION RESEARCH

The research question is:

"Is it more helpful and/or pleasant for someone to explain a concept to a socially reactive entity compared to a non-reactive entity for said someone to get a better understanding of the concept?"

The Perceived usefulness scale scored lowest in the questionnaire with a mean of 3.0 and an SD of 0.7. This probably means that most people don't perceive it as useful to talk to an robot in general. The statistical tests that test the actual usefulness are not significant possibly due to a sample size that wasn't big enough or the difference in programming skills between the participants. And when looking for a correlation between effectiveness and and Interaction (see image 10) there does not seem to be one.

Whether it was more pleasant to talk to a animated robot or a stationary robot is not clear from the statistics since all nothing shows a significant difference between animated and not animated and the questionnaire only answered the questions for both animated and not.

Some people seem to have a greater affinity of talking to Rubber Ducky then others. Whether it works might be highly personal. There were also reports that participants were confused when Rubber Ducky didn't react, which could have influenced the results. One participant said to have already used the rubber duck debugging concept before with a gnome statue. This participant reported finding it less difficult to speak to something that looked alive than to a gnome statue, whether Rubber Ducky was animated or not. There might also have been a bias since most of the participants knew me personally and a few also knew about the research. Finally, the time that participants spend on solving the assignment might also be of interest.

CHAPTER 6

RECOMMENDATIONS

6.1 Design recommendations.

A noticeable problem was that sometimes, the dynamixel AX-12's in the neck would shut down after being on for a couple minutes. They also don't respond to a message from the Arduino to change the compliance slope. It is recommended to use dynamixel MX-28's for these servo's instead. Another benefit that brings is that all the servo's behave the same.

To make Rubber Ducky seem more engaged, having Rubber Ducky look to the participant with face tracking is recommended.

The code currently doesn't allow for smooth transitions between animations, which can lead to Rubber Ducky making sudden movements that might scare people.

6.2 Further Research.

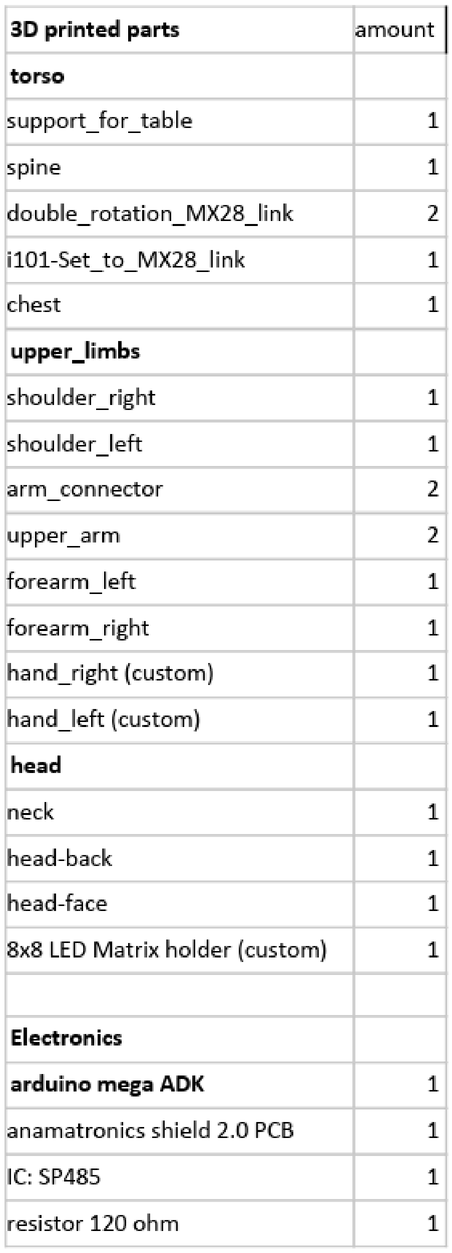
The experiment that was deducted barely held any significant results. To change this, a larger sample size and/or a longer experiment is recommended. The rubber ducky debugging concept doesn't have to be tested with a humanoid, if it seems to work trying it with a different cheaper to produce robot might be beneficial. The experiment design would probably benefit from a bigger focus on the differences between the two states of animated and not animated. This could be achieved by having the participants fill in the questionnaire twice for instance.

Some control test are needed for comparing a 'real' rubber ducky with a robot since there is no research to be found on this concept.

CHAPTER 7

APPENDIXES

7.1 Appendix A: Bill of Materials.



3D printed parts	amount
torso	
support_for_table	1
spine	1
double_rotation_MX28_link	2
i101-Set_to_MX28_link	1
chest	1
upper_limbs	
shoulder_right	1
shoulder_left	1
arm_connector	2
upper_arm	2
forearm_left	1
forearm_right	1
hand_right (custom)	1
hand_left (custom)	1
head	
neck	1
head-back	1
head-face	1
8x8 LED Matrix holder (custom)	1
Electronics	
arduino mega ADK	1
anamatronics shield 2.0 PCB	1
IC: SP485	1
resistor 120 ohm	1

Table No: 1. Bill of Materials 1.

resistor 220 ohm	2
resistor 1k ohm	1
led	2
flexifuse	1
capacitor 100nF	1
micro-match connector	1
flexible wire 1m	1
molex 22-03-5045 connector	1
Arduino Uno	1
Arduino ProtoShield v2	1
Adafruit 8x8 Bi-Color LED Matrix	2
IC: SN74AS241N	1
IC: TSR 1-2450	1
IC: 6n137	1
Female header pins	8
molex 22-03-5035 connector	1
2 pin male molex connector	1
2 pin female molex connector	1
resistor 1.2k ohm	1
resistor 120 ohm	1
misc wires	x
Other	
Dynamixel	
MX-28AT	11
AX-12A	2
Parts	
HN07-N101 set	11
HN07-i101 Set	6
Visserie	
Wrench Bolt M2*3 (200 pcs)	1

Table No: 2. Bill of Materials 2.

Wrench Bolt M2.5*4 (200 pcs)	1
Wrench Bolt M2.5*6 (200 pcs)	1
Wrench Bolt M2.5*8 (200 pcs)	1
BIOLOID Bolt Nut Set BNS-10	1
Nut M2.5 (400 pcs)	1
N1 Nut M2 (400 pcs)	1
M5 nuts	2
M5x20mm screws	2
Cables	
SMPS2Dynamixel	1
Robot Cable-3P 60mm 10pcs	1
Robot Cable-3P 100mm 10pcs	1
Robot Cable-3P 140mm 10pcs	1
Robot Cable-3P 200mm 10pcs	1
Other	
frosted plexiglass screen	1
piece of office paper with light pencil marking	1
Lasercut box as base (custom)	1
Korg NanoKontrol 2.0 with cable	1
12v 3.75A Power Supply	1
SMPS2Dynamixel Adapter	1
USB2AX	1

Table No: 3. Bill of Materials 3.

7.2 Appendix B: Electronics Schematic.

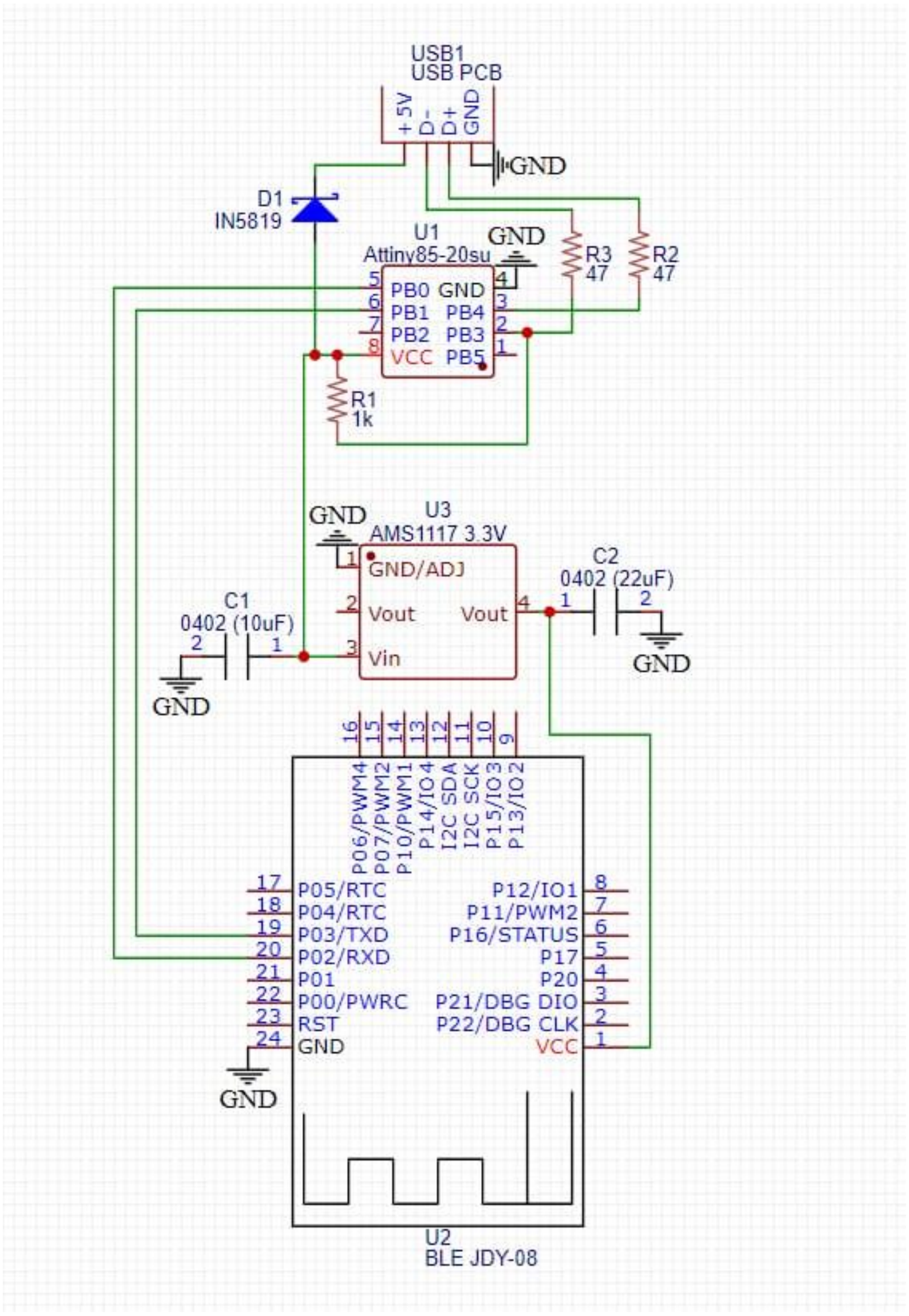


Fig 8: Circuit Structure.

ATtiny85 and Arduino Interfacing.

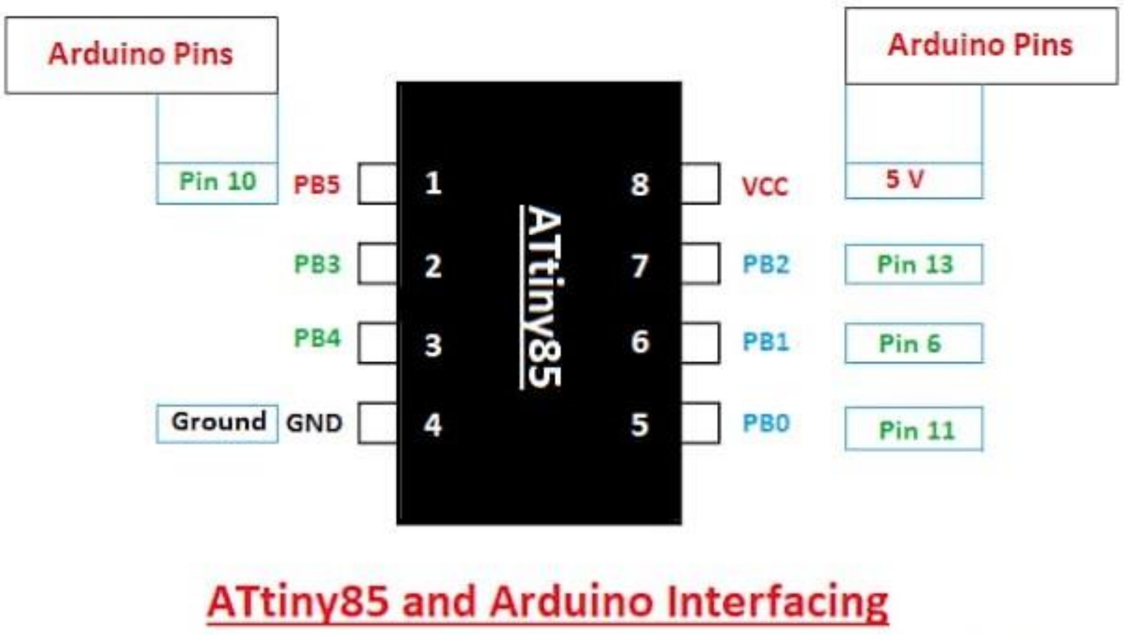


Fig 9: ATtiny85 and Arduino Interfacing.

7.3 Appendix C.

Consent Form

Consent Form: Robot guided self-explanation

Please read and sign this form.

Outline:

This informed consent form is for research in robot guided self-explanation. An example for guided self-explanation is rubber ducky debugging. This is when a programmer explains his or her code to a rubber duck to understand the code better themselves.

In this test, the participant is going to do two small programming assignments in Processing. During the assignments, it is encouraged for the participant to explain a robot what he/she is Doing.

Rights:

During this user test I agree to participate in doing two small programming challenges with the help of a humanoid robot named Rubber Ducky.

I understand and consent to the use of video recordings and the results of the questionnaire by the faculty RAM of the University of Twente for their social robotics research.

I understand that the information and recordings are for research purposes only and that my name and image will not be used for any other purpose.

I understand that if I want my video to be removed I can request that to Kasper de Kruiff.

I understand that participation is voluntary and if I want to seize my participation I can do that at any time without need for explanation.

If there are any concerns I can discuss them with Kasper de Kruiff.

Please sign below to indicate that you have read and understood the information on this form and that any questions you might have about the session have been answered.

Date :_________

Name:____

Sign :____

Thank you!

We appreciate your participation.

7.4 Appendix D: Test Protocol.

1. Welcome the participant into the room.

2. Introduce the participant to Rubber Ducky.

3. Give them a short explanation what the test entails.

a. What we are going to test here is robot guided self-explanation. An example for guided self-explanation is rubber ducky debugging. This is when a programmer explains his or her code to a rubber duck to understand the code better themselves. In this test, we're going to see how this phenomenon holds up with a robot. In a moment, you are going to do two small programming assignments in Processing. During the assignments, I want you to explain Rubber Ducky what you are doing. But first I would like you to fill in this consent form. By signing it you give me permission to use your data and you allow me to film you for the purpose of this research. The footage will only be used for this experiment and you can ask us to delete it any time. Also, you can leave anytime you want without need for explanation.

4. Ask whether the participant has any questions and answer them.

5. Give the participant a consent form, give him or her time to read and sign it.

6. After the participant signed the form turn on the camera

7. Show the participant where he/she needs to sit and what Assignment he/she needs to do.

8. Sit at the control table to control or not control Rubber Ducky.

9. Wait until the participant is ready with their assignment or stop him/her after seven minutes.

10. Tell the participant we're going to do the test again but this time Rubber Ducky will react/not react.

11. Sit at the control table to control or not control Rubber Ducky.

12. Wait until the participant is ready with their assignment or stop him/her after seven minutes.

13. Thank the participant for participating.

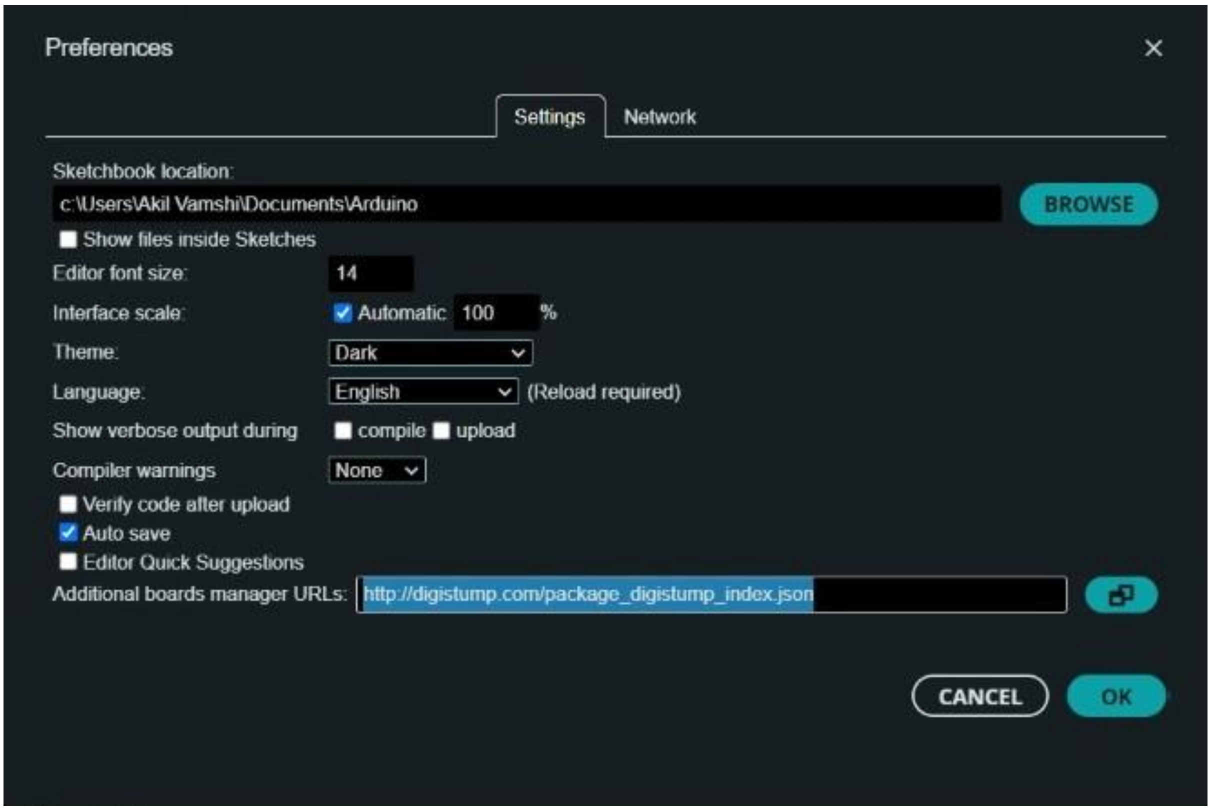
14. Turn off the camera.

15. Offer the participant a piece of cake.

16. Let him leave the room and get the next participant.

7.5 Appendix E: Programming Assignments.

Assignment 1.


Preferences

Settings | Network

Sketchbook location: c:\Users\Akil Vamshi\Documents\Arduino

- [ ] Show files inside Sketches

Editor font size: 14

Interface scale: [✓] Automatic 100 %

Theme: Dark

Language: English (Reload required)

Show verbose output during [ ] compile [ ] upload

Compiler warnings: None

- [ ] Verify code after upload
- [✓] Auto save
- [ ] Editor Quick Suggestions

Additional boards manager URLs: http://digistump.com/package_digistump_index.json

Fig 10: Installation of Digistump Package.

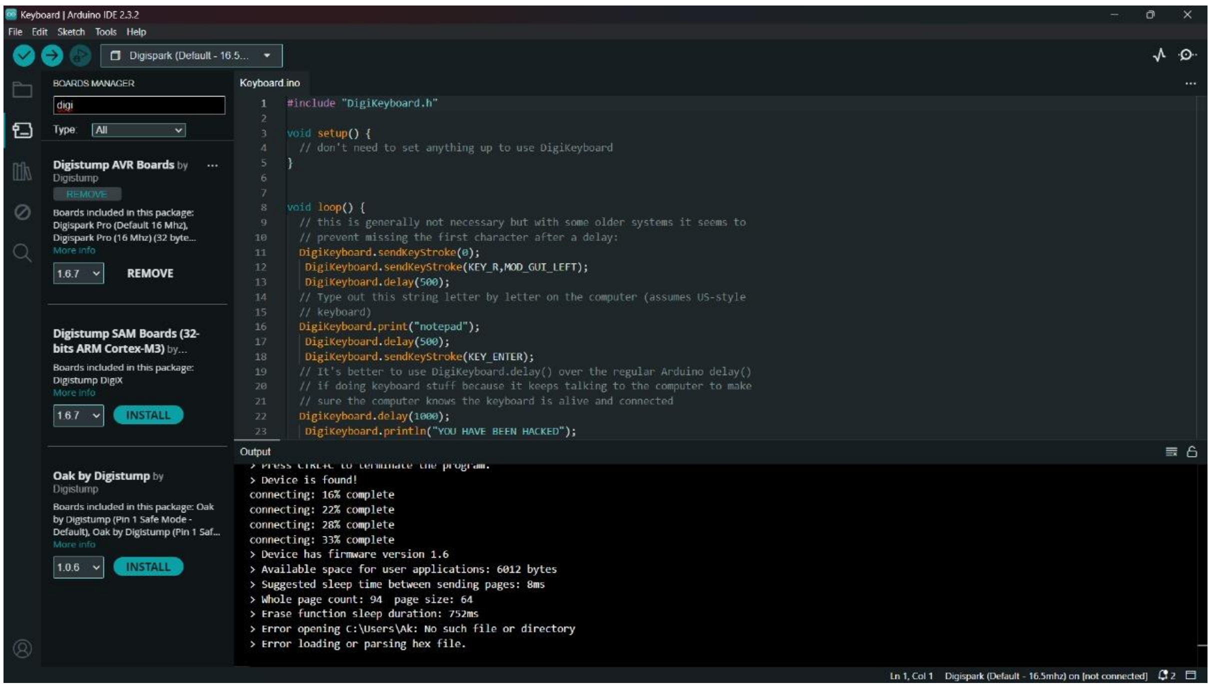
Source Link:

http://digistump.com/package_digistump_index.json

Program:

```
void setup() {

  pinMode(0, OUTPUT);

}

void loop() {

  digitalWrite(0, HIGH);

  delay(1000);

  digitalWrite(0, LOW);

  delay(1000);

}
```

Assignment 2.


Keyboard | Arduino IDE 2.3.2

Digispark (Default - 16.5...

BOARDS MANAGER

digi

Type: All

Digistump AVR Boards by Digistump

Boards included in this package: Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte...

1.6.7 REMOVE

Digistump SAM Boards (32-bits ARM Cortex-M3) by...

Boards included in this package: Digistump DigiX

1.6.7 INSTALL

Oak by Digistump by Digistump

Boards included in this package: Oak by Digistump (Pin 1 Safe Mode - Default), Oak by Digistump (Pin 1 Saf...

1.0.6 INSTALL

Keyboard.ino

```
#include "DigiKeyboard.h"

void setup() {
  // don't need to set anything up to use DigiKeyboard
}


void loop() {
  // this is generally not necessary but with some older systems it seems to
  // prevent missing the first character after a delay:
  DigiKeyboard.sendKeyStroke(0);
   DigiKeyboard.sendKeyStroke(KEY_R,MOD_GUI_LEFT);
   DigiKeyboard.delay(500);
  // Type out this string letter by letter on the computer (assumes US-style
  // keyboard)
  DigiKeyboard.print("notepad");
   DigiKeyboard.delay(500);
   DigiKeyboard.sendKeyStroke(KEY_ENTER);
  // It's better to use DigiKeyboard.delay() over the regular Arduino delay()
  // if doing keyboard stuff because it keeps talking to the computer to make
  // sure the computer knows the keyboard is alive and connected
  DigiKeyboard.delay(1000);
   DigiKeyboard.println("YOU HAVE BEEN HACKED");
```

Output

```
> Press CTRL+C to terminate the program.
> Device is found!
connecting: 16% complete
connecting: 22% complete
connecting: 28% complete
connecting: 33% complete
> Device has firmware version 1.6
> Available space for user applications: 6012 bytes
> Suggested sleep time between sending pages: 8ms
> Whole page count: 94  page size: 64
> Erase function sleep duration: 752ms
> Error opening C:\Users\Ak: No such file or directory
> Error loading or parsing hex file.
```

Ln 1, Col 1   Digispark (Default - 16.5mhz) on [not connected]

Fig 11: Searching Digispark Extension.

Program:

```
int dim;
void setup() {
  size(640, 360);
  dim = width/9;
  background(0);
  colorMode(HSB, 360, 100, 100);
  noStroke();
  ellipseMode(RADIUS);
  frameRate(1);
}
void draw() {
background(0);
for (int x = dim/2; x <= width; x+=dim) {
drawGradient(x, height/2);
}
}
void drawGradient(float x, float y) {
int radius = dim/2;
float h = random(0, 360);
for (int r = radius; r > 0; --r) {
fill(h, 90, 90);
ellipse(x, y, r, r);
h = (h + 1) % 360;
}
}
```

7.6 Appendix F: Questionnaire.

O: open question

C: closed question

R: rate whether agree with the statement (strongly disagree to strongly agree)

Demographics

O: Participant number

O: What is your age?

O: What is your gender?

O: What is your nationality?

C: Are you a Student at the University of Twente

R: Relative to an average CreaTer who finished their first year, I am a good programmer

Quality of Puppeteering

R: Rubber Ducky reacted to me in a credible way

R: Rubber Ducky reacted directly (without delay) to me

R: Rubber Ducky's emotions where neither too small nor too big

R: I like the appearance of Rubber Ducky

R: I like Rubber Ducky as a robot

R: Rubber Ducky's emotions where convincing

R: My feelings have changed positively about Rubber Ducky during this experiment

Perceived Affective Understanding – Harms & Boicca

R: I could tell how Rubber Ducky felt

R: Rubber Ducky could tell how I felt

R: Rubber Ducky's emotions were not clear to me

R: My emotions were not clear to Rubber Ducky

R: I could describe Rubber Ducky feelings accurately

R: Rubber Ducky could describe my feelings accurately.

Social Connectedness

R: I had a connection with Rubber Ducky

R: Rubber Ducky had a connection with me

R: I am willing to talk to Rubber Ducky more often

Perceived Message understanding

R: My thoughts were clear to Rubber Ducky

R: Rubber Ducky's thoughts were clear to me

R: It was easy to understand Rubber Ducky

R: Rubber Ducky found it easy to understand me

R: Understanding Rubber Ducky was difficult

R: Rubber Ducky had a difficulty understanding me

Perceived Sociability – Heerink

R: I consider Rubber Ducky a pleasant conversational partner

R: I find Rubber Ducky pleasant to interact with

R: I feel Rubber Ducky understands me.

R: I think Rubber Ducky is nice Comfort

R: I felt comfortable talking to Rubber Ducky

R: I had no problems explaining the code to Rubber Ducky

R: I felt at ease when explaining the code to Rubber Ducky

R: It was easy talking to Rubber Ducky

R: It felt awkward talking to Rubber Ducky

R: I felt forced to talk to Rubber Ducky

Anxiety – Heerink

R: If I should use Rubber Ducky, I would be afraid to make mistakes with it

R: If I should use Rubber Ducky, I would be afraid to break something

R: I find Rubber Ducky scary

R: I find Rubber Ducky intimidating Usefulness

R: I understood the code better after I explained it to (the animated) Rubber Ducky

R: I felt like Rubber Ducky was distracting

R: I felt like Rubber Ducky was a useful addition in understanding the code

R: I felt like talking to Rubber Ducky was helpful

R: I would've needed more time if Rubber Ducky wasn't there

R: I would consider explaining something to someone else more often to get a better

understanding myself

R: I would consider explaining something to Rubber Ducky more often to get a better understanding

myself Perceived usefulness – Harms & Boicca

R: I think Rubber Ducky is useful to me

R: It would be convenient for me to have Rubber Ducky

R: I think Rubber Ducky can help me with many things.

7.7 Appendix H: Reflection Report.

7.7.1 Introduction.

7.7.1.1 Project overview.

This document is an ethical review about the "Rubber Ducky as advanced rubber ducky" project. It will discuss the ethical aspect of the project and what is morally good and bad about it. This Project started with Rubber Ducky. Rubber Ducky is a child sized humanoid and open source robot designed for educational artistic, engineering, humanities and life science research. The goal of this project is to investigate social interaction with Rubber Ducky. The way that this is going to be done in a meaningful manner is by exploring the rubber ducky effect. The term comes from the computer science scene. The story goes that there was a programmer stuck on his code and asked his supervisor about it. The supervisor gave him a rubber ducky and said to explain his code to it. By explaining the code to the rubber ducky, the programmer understood his own code better and was able to work out the bugs and thus the term 'rubber ducky debugging' was born. The plan with Rubber Ducky is to have it act like an advanced rubber ducky. In other words, it will be used to speak to and help you understand a certain problem better: a listening robot. However, it might also be useful support ordering thoughts, practice conversations and presentations, or just to get something off your chest. The difference between an actual rubber ducky and a humanoid robot is that, instead of doing nothing, the robot will try to motivate the explaining behavior by interacting with, and reacting to the person in question. This builds on the premise that someone would rather converse with something that acts alive and aware than something that does not. This report is going to consider hypothetical versions of the further developed Rubber Ducky as an advanced rubber ducky in different scenarios.

7.7.2 Stakeholders.

To identify the stakeholders there will be looked at what Rubber Ducky as advanced rubber ducky could and might be used for. Firstly, it might be used for: social interaction to order thoughts (programming help, study help or support during emotionally difficult times), social interaction for practice (practicing a job interview, practicing a presentation or a conversation with an acquaintance) and intrinsic social interaction (for people that need more social interaction in general of socially challenged people). Secondly, it can be used for research and possible stakeholders that interact directly with Rubber Ducky are: students who could use Rubber Ducky as a learning tool, programmers that could use Rubber Ducky as a debugging tool, Other stakeholders might be: IT businesses, universities and personal trainers, because they might have to provide a Rubber Ducky. Thirdly people in need of more social contact who could use Rubber Ducky as a substitution for other social contact, and people who need to rehearse with "a listener". Fourthly, because Rubber Ducky has the size of a child and it is not illegal to have sexual interactions with a robot Rubber Ducky is of use to pedophiles for a sexual outlet and finally robot enthusiasts (artists, tinkerers, researchers, educators) who want to hack Rubber Ducky to do the things they want it to.

Possible ethical consequences.

The use of Rubber Ducky might lead to morally undesirable consequences. Where all people should benefit from its use, poor or disabled persons are excluded because new technology is expensive. Another consequence can be, that by using Rubber Ducky as a substitute for interaction with human beings, social connections might become obsolete and people get socially excluded. Moreover, people can get dependent on the robot, by getting used to it or even falling in love with it. Lastly, as all technology, it can be used to bother other people in a socially unaccepted way.

Exclusion.

The first cause of exclusion is money. This robot is expensive for most people [1]. The people who might need a study helper the most are probably also the people who can't pay for one: (poor) students. On the other hand, if this concept of Rubber Ducky as an advanced rubber ducky seems promising enough there can be research on how to make this concept cheaper. Universities might be inclined to buy more robots for their students if it really proofs to help. This all however still does not directly translate to poorer countries where the schools do not have enough money to buy robots. Also, Rubber Ducky's system depends on the use of eyesight to create effect, and 0.5% [2] of the world population is blind. In this case it becomes the question if a design is supposed to be used by everyone. A utilitarian approach [3] would be to only design and develop things that contribute to the general happiness. In other words, to create something that gives the most happiness to the most amount of people. In this case developing Rubber Ducky as an advanced rubber ducky would probably be morally sound, since most people are not blind and people who are not proficient in using the potential of Rubber Ducky probably are not interested in using it in the first place. So, the net total of happiness would go up. On the other hand, there is the deontological [4] approach, which is more duty bound and states that if something is bad, it will always be bad regardless of the consequences. In this case one could argue that since this concept of Rubber Ducky can be seen as discriminating, and since discriminating is morally unjust, the development of Rubber Ducky as an advanced rubber ducky is unjust. However, it is morally good to help people, and, within this concept, Rubber Ducky is built to do that, which makes it, using the same logic, morally good.

Social Exclusion.

A different form of exclusion that might occur is that Rubber Ducky might replace some social connections. For instance, a son might always have needed his father to help him with his homework and thus they created a good social bond with each other. If suddenly the father is not needed anymore because a robot replaced that need, it can have a serious impact on their relationship. And because robots can always be improved they might become close to the perfect study partner and makes this exclusion inevitable because the robot is therefore more preferred. If this happens, the son's idea about social interaction might change to the point that interacting with his father about homework is just tiresome because the robot is a 'perfect' partner to interact with. This renders humans less capable than they used to be in this aspect of life. However, this has already happened to some extent. A larger part of the population used to be more physically fit due to food not being readily available and getting said food was physically demanding. Besides that, allergies are becoming more present in current times because children get less exposure to dirt and filth [5]. Also, there seem to be more visual impairments due to modern lifestyles [6]. The pragmatic ethics movement [7]is one that that fits this phenomenon. The hypothesis is that it is bad to replace social interactions between humans with human robot interaction but in due time the hypothesis might be rejected for a hypothesis that states that interaction between humans is worse than human robot interaction. Besides that, robots might also provide social interaction for people that don't have access to it.

Social Implications.

The concept of Rubber Ducky as an advanced rubber ducky requires a person to speak to Rubber Ducky. Because of this, Rubber Ducky can be used to perform selfish acts that can be harmful for others. For example, if a student brings Rubber Ducky to a place where also other students study, because the student feels that studying is done best at that place, and the student starts to talk to Rubber Ducky the others can be distracted which could cause them to fail a class. If the concept of Rubber Ducky works as it should and this is not regulated the above is a reasonable possibility. But regulation is essentially taking away a little freedom. There is a good chance that the student of the example really takes the utmost care to not bother the others by talking softly and sitting far from the others which might result in all the students to pass their exams. Another possibility might be that it works very well for most students, when someone talks to a robot, but then that single person that like to work in silence needs to adjust to rest. This is in line with the utilitarian approach though. Because in this case, the advanced rubber ducky concept causes more happiness to more students than working in silence. But there might just be a sweet spot between regulating and freedom in this example where both silence and rubber duckies can coexist.

Creating dependency.

If people use Rubber Ducky as an advanced rubber ducky extensively, some might get so used to it or find it so helpful that they are unable to go without it. People might not think clearly anymore without talking out loud to something or someone. This can have effect on their daily life if they need to keep secrets or follow informative presentations where people are expected to keep silent.

Another way of dependency is if people fall in love. This can happen because Rubber Ducky as an advanced rubber ducky is made to listen which might not be self-evident for everybody. It is very much possible that falling in love with robots is not accepted in this society and this might have an effect on how to deal with it. They might try to keep it a secret and feel embarrassed or this person might not be taken seriously because he is the weird one that has a robot as significant other. Another option is that they suppress their feelings and get depressed.

People who might have this problem can be pedophiles, because Rubber Ducky kind of looks like a child. This can be a good thing if Rubber Ducky is used as an outlet for them so that they do not have sexual or romantic tendencies towards children who are not ready for romance or sexuality. What also could happen is that they are not fully satisfied with a robot and get more frustrated which can lead to illegal activity.

About Rubber Ducky:

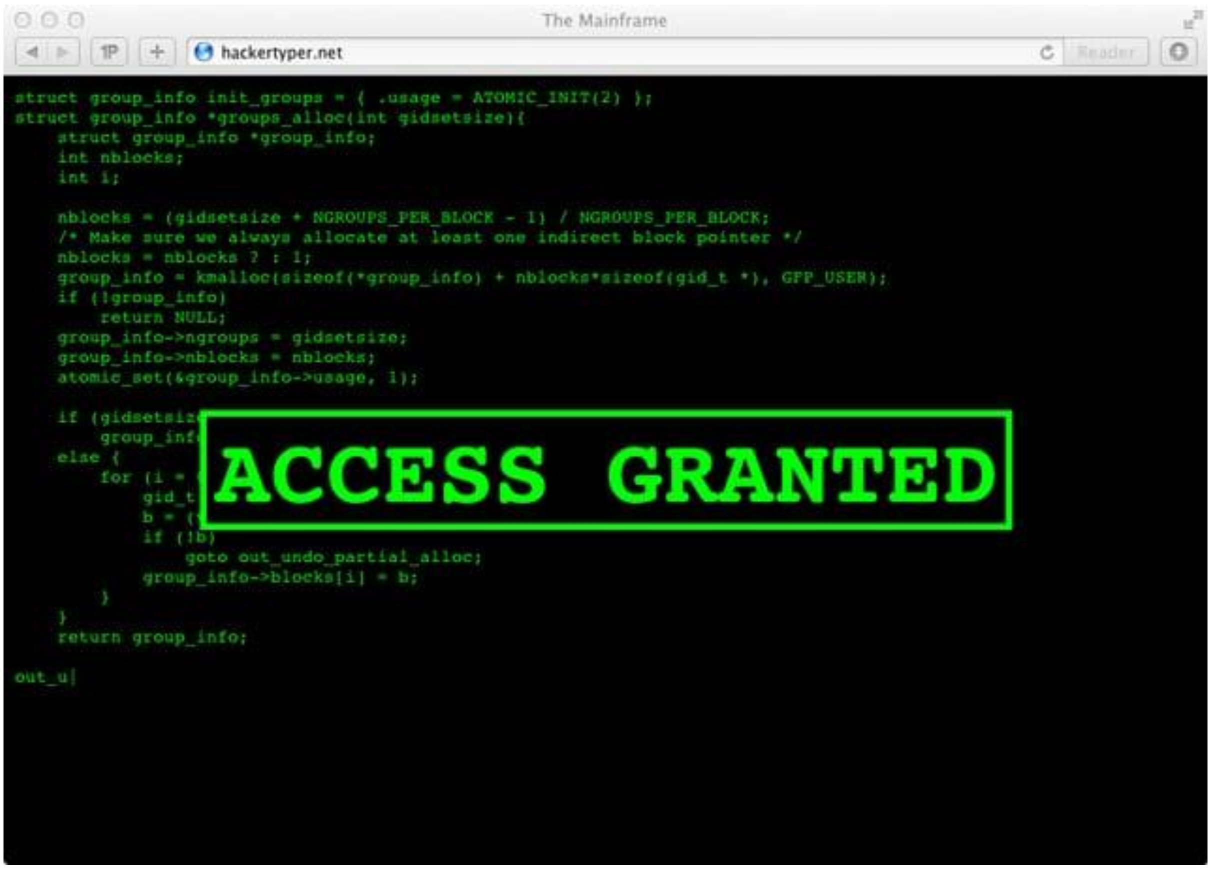
The Rubber Ducky as an advanced rubber ducky project is a research project to see whether advanced rubber duckies are a useful concept. Although the research outcomes are yet unclear, the concept is promising and it could help out a lot of people. In this paper the consequences of the “Rubber Ducky as an advanced rubber ducky” project is explored on moral issues. Found was, that exclusion, social exclusion, application in a socially unaccepted way and dependency could follow when the undertaken research would turn out to be fruitful and Rubber Ducky was taken into production. Of these, some problems are concerned with the prize of technology and others with the use of Rubber Ducky as a rubber ducky. To prevent most of these consequences is to not use Rubber Ducky for the rubber duckie concept because Rubber Ducky is expensive and Rubber Ducky is a small humanoid which people relate to easily. Although this was exactly the reason it is used to test this concept: since people can easily read social cues from a humanoid and that is very useful for the proof of the concept. Unless it appears that for the concept to really work the robot needs to be similar to Rubber Ducky, it is also possible to use something that can be lend or leased or is available at the place where it is used. This would solve the issue of people using Rubber Ducky at inappropriate times andplaces. However, in this small-scale research these problems could not a threat, while the robot has only been used in the experimental setup.

Reflection:

The writing of this report was interesting because I could explore the edges of the consequences of my research. And would this concept be used in a product, then it is a good thing that the extremes already have been considered. Everything that I came up with how ever is not that likely to happen, especially not in the scope of this research. For the experiment I chose to have people interact with Rubber Ducky for approximately fifteen minutes and measure their experience and behavior. I felt that there were at least no moral complications with the research I've done so far. Except maybe that I have gathered some personal information about people for the user test, but that will be officially as anonymized as needed. Making this report and thinking about the ethical issues has not made me change any design decision. However, it has strengthened my opinion about trying the rubber ducky effect with different robots that are less expensive and not necessarily resemble a human.

CHAPTER 8

Attacks on Windows 10 Machine



```
struct group_info init_groups = { .usage = ATOMIC_INIT(2) };
struct group_info *groups_alloc(int gidsetsize){
    struct group_info *group_info;
    int nblocks;
    int i;

    nblocks = (gidsetsize + NGROUPS_PER_BLOCK - 1) / NGROUPS_PER_BLOCK;
    /* Make sure we always allocate at least one indirect block pointer */
    nblocks = nblocks ? : 1;
    group_info = kmalloc(sizeof(*group_info) + nblocks*sizeof(gid_t *), GFP_USER);
    if (!group_info)
        return NULL;
    group_info->ngroups = gidsetsize;
    group_info->nblocks = nblocks;
    atomic_set(&group_info->usage, 1);

    if (gidsetsiz
        group_inf
    else {
        for (i = (
            gid_t
            b = (
            if (!b)
                goto out_undo_partial_alloc;
            group_info->blocks[i] = b;
        }
    }
    return group_info;

out_u|
```

Fig 12: Penetration is completed.

Description.

As can be seen in Figure 6, the username testdummy8585@hotmail.com and password Dummydummy have been retrieved in clear-text.

After Windows 7, Microsoft changed the way that their operating system handled passwords. This vulnerability is not easily exploitable on Windows 10 without a registry edit. Due to the unique platform of attack, since we have physical access to the system, we can make a registry edit and allow this vulnerability to be exploited again.

Because the way Windows registry works, performing this all-in-one attack is very difficult. Therefore, on Windows 10, this attack must be split into two parts.

In the first part we have to edit the Windows Registry to enable the vulnerability in Windows 10 and make it susceptible to the second part, which is the attack we have already created. Once we make the registry edit, the Windows account must be locked, signed out, or restarted before the registry changes go into effect. We utilized the "reg add" command to recreate the registry value that Microsoft has removed, and setting its value to "1" for true.

Once we add this value and the account is logged into once again, logon passwords will be stored in memory. The Ducky script for this part is quite similar but shorter than the previous script. We ran PowerShell as an administrator again, then performed the proper registry edit, and then launched the attack similar to the one we did before. Similar to Windows 7, we were able to retrieve the user's credentials from the memory.

CHAPTER 9

CONCLUSION & FUTURE IMPLEMENTATION

9.1 CONCLUSIONS.

In this paper we demonstrated the process of writing a malware payload which can exploit Windows vulnerability to launch an attack on a victim's machine. The payload can be executed from the victim's machine or remotely. Our aim in this project was to launch the attack remotely targeting a Windows 7 machine. To create the malware and launch the attack, we utilized various tools, such as USB Rubber Ducky, scripting language, powersehll, mimikatz, Ducky toolkit NG and a web server account.

For Windows 10 machine, we had to take an extra step of editing the registry to create the vulnerability that existed in Windows 7. However, in both cases of Windows 7 and 10 the malware was injected into a USB Rubber Ducky device, the device was attached to the victim's machine, and the payload was executed remotely. As a result, we were able to retrieve the victim's user id and password from the memory of the victim's computer.

9.2 FUTURE IMPLEMENTATION.

This project can be extended I several ways: 1) instead of running remotely, one can run the payload locally. 2) More analysis of main memory and locking for more information. 3) More experiment with Windows 10 and looking for other possible implementations rather than editing the registry.

REFERENCES

[1] G. Fournier, P. Matousswoski and P. Cotret. Hit the KeyJack: stealing data from your daily device incognito. CS.CR, France, Oct. 2016.

[2] R. Schilling and F. Steinmetz. USB Device Phoning Home. Hamburg University of Technology, February 2016.

[3] R. Bhakte, P. Zavarsky and S. Butakov. Security Controls for Monitored Use of USB Devices Based on the NIST Risk Management Framework. Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual.

[4] OLEA Kiosks, Inc. Malware Scrubbing Cyber Security Kiosk. http://www.olea.com/product/cyber-security-kiosk/, 2015.

[5] D. Tian, A. Bates and K. Butler: Defending Against Malicious USB Firmware with GoodUSB. ACSAC ’15, December 07-11, 2015, Los Angeles, CA, USA.

[6] BlackHat USA 2014, Karsten Nohl and Jakob Lell, BadUSB - On Accessories that Turn Evil, https://srlabs.de/badusb/, Accessed on 07 Jan 2015.

[7] S. Kamkar, USBDriveBy, http://samy.pl/usbdriveby/, Jan 2015 [13] Nikhil "SamratAshok" Mittal, Kautilya, https://github.com/samratashok/Kautilya, Jan 2015.

[8] S. Vouteva, Feasibility and Deployment of Bad USB. University of Amsterdam, System and Network Engineering Master Research Project, Feb 2015.

[9] Arduino Micro, http://arduino.cc/en/ Main/ArduinoBoardMicro, 2015.

[10] M. Kang. USBWall: A Novel Security Mechanism to Protect Against Maliciously Reprogrammed USB Devices. M.S., Computer Science, University of Kansas, 2015.

[11] KeyScrambler, https://www.qfxsoftware.com/.

[12] KeyGrabber, http://www.keelog.com/usb_hardware_keylogger.html

[13] Mimikatz, https://github.com/gentilkiwi/mimikatz.