Prototyping of Tangible Programming

An Iterative Approach to a Multidisciplinary
Design Problem

Thov Reime

Master of Science in Mechanical Engineering
Submission date: July 2015
Supervisor: Martin Steinert, IPM

Norwegian University of Science and Technology
Department of Engineering Design and Materials

Abstract

The Fibo Car is a game interface consisting of a kit of car parts. A virtual model of the assembled parts is generated on a computer and used in a physics engine based gameplay. This product vision is the source of this master's thesis task where the main goal was to develop a critical function prototype of the physical/virtual interface. The challenge of this task was to create a prototype that contained integrated technology from multiple engineering disciplines.

After six weeks, a functioning prototype was completed using an incremental style of product development process. Based on this prototype and process, two papers were submitted and accepted for publishing at a conference on entertainment computing. The prototype was then further improved and shown at a live demonstration for potential investors. The final prototype is a complete set where external car parts can be attached to a central part and a rotating 3D model will appear on-screen with the corresponding structure. A list of process related learnings form the conclusion of the project.

Acknowledgements

I extend my sincere gratitude to the supplier of this project task: Leonore. A. Nilsen, CEO of Metis Productions. The end result of this project could not have been achieved without the help of several mentors and students who helped me at TrollLABS and gave valuable input. Heikki Sjöman, Achim Gerstenberg, Pekka Abrahamsson, and my supervisor Martin Steinert, whom together with me wrote the two proceedings papers. All the accomplishments of this project are theirs as much as mine. Thank you, all!

Table of contents

ABSTRACT .... **III**

ACKNOWLEDGEMENTS .... **V**

TABLE OF CONTENTS .... **VII**

LIST OF FIGURES .... **IX**

ABBREVIATIONS .... **X**

1 PROJECT OVERVIEW **1**

1.1 TOPIC INTRODUCTION 1
1.2 PROJECT ASSIGNMENT 1
1.3 TECHNOLOGY CHALLENGE 2
1.4 TIMELINE 2
1.5 FINAL PROTOTYPE 3

2 TECHNOLOGY BACKGROUND AND DEVELOPMENT **5**

2.1 TANGIBLE PROGRAMMING 5
2.2 MATERIAL SCIENCE 6
2.3 MECHANICS 7
2.4 ELECTRONICS 10
2.5 MECHATRONICS AND COMPUTER SCIENCE 12
2.6 USER INTERFACE DESIGN 15
2.7 FINAL PROTOTYPE VS. PRODUCT VISION 16
2.8 DESIGN FLAWS IN FINAL PROTOTYPE 17

3 PUBLICATIONS TO ICEC 2015 .... **19**

3.1 PREFACE FOR PAPERS 19

BRIDGING TANGIBLE AND VIRTUAL INTERACTION: RAPID PROTOTYPING OF A GAMING IDEA ....

A SIMULTANEOUS, MULTIDISCIPLINARY DEVELOPMENT AND DESIGN JOURNEY – REFLECTIONS ON PROTOTYPING ....

4 LEARNINGS AND REFLECTIONS **35**

4.1 FROM COMPLEX TO COMPLICATED 35
4.2 INPUT FROM ALL DOMAINS WAS NOT INCLUDED 36
4.3 TAKE MEASURES TO PROTECT IP 37
4.4 DETOURS YIELDED IMPORTANT RESULTS 37
4.5 USE 'BIG PICTURES' AND ASSUMPTIONS WHEN APPLICABLE 37
4.6 FAVOR A 'MOLDABLE' CONCEPT 38

4.7 INTERDEPENDENCIES INCREASE EXPONENTIALLY WITH ADDITION OF DOMAINS 38
4.8 PROTOTYPE AS MANY THINGS SIMULTANEOUSLY AS POSSIBLE 39
4.9 FAVOR EXPERIMENTATION – TACIT VS. EXPLICIT KNOWLEDGE 39

5 PERSONAL REMARKS 41

REFERENCES 43

APPENDIX A: UPLOADED MATERIAL ....

APPENDIX B: POSTER ....

APPENDIX C: PROCESSING CODE ....

APPENDIX D: ARDUINO CODE ....

APPENDIX E: INITIAL PRODUCT VISION FROM PROJECT OWNER ....

List of Figures

Figure 1.1 Left: All physical objects of final prototype. Right: Exploded view of virtual model. 4
Figure 2.1 Painted glue seam around connector 7
Figure 2.2 Magnets on sides of center and F1 parts 8
Figure 2.3 Left: Single magnetic ring connector. Right: Two magnetic ring connectors connected 9
Figure 2.4 3D model of assembled physical shells 9
Figure 2.5 Supporting brackets and LightBlue Bean 10
Figure 2.6 Wheels changing size by tangible input of turning potentiometer (arrows) 11
Figure 2.7 LittleBits Bitsnaps over paper drawing showing principle for measuring resistors 12
Figure 2.8 Left: Circuit board bottom side. Right: LightBlue Bean. 14
Figure 2.9 Iterations of I2C. Left: Two Arduino Unos. Middle: Arduino plus Bean. Right: Two Beans 15

Abbreviations

BLE	Bluetooth Low Energy
COM	Communication Port
I2C	Inter-Integrated Circuit
ICEC	International Conference on Entertainment Computing
IP	Intellectual Property
IPM	Department of Engineering Design and Materials
NTNU	Norwegian University of Science and Technology
OS	Operating System
SCL	Serial Clock
SDA	Serial Data

1 Project overview

This introductory chapter describes the major outlines of the project; where it came from, how it was developed, related topics, and various results.

1.1 Topic introduction

This document contains the explicit results from a master project undertaken at the Department of Engineering Design and Materials (IPM) at the Norwegian University of Science and Technology (NTNU) spring of 2015. The project stems from the start-up company Metis Games that approached TrollLABS at IPM for the development of a prototype of their product idea. This document aims to serve as a framing for the major outcomes of that project: 1) a critical function prototype using tangible programming and two minor prototypes, 2) various digital material, 3) one demonstration paper describing the development of the first prototype during the first six weeks (Reime et al., 2015), 4) one paper on the process used to develop the first prototype (Gerstenberg et al., 2015), and 5) findings and lessons from the entire project. Both papers were accepted for publishing at the ICEC 2015[1]. 2) consists of code scripts, 3D models, and digital footage and pictures that will be made available online (Appendix A: Digital Material). This thesis concerns early project phase product development where specifications are not predetermined but found through an incremental style of development inspired by wayfaring (Steinert & Leifer, 2012). Technology from multiple engineering disciplines was integrated in a flexible, modular architecture (Sanchez & Mahoney, 1996, p. 65) and is described in detail here. As such, this thesis should be relevant to any reader interested in the following: incremental product development; multidisciplinary and/or simultaneous prototyping; product specifications emerging from experimentation and low resolution prototypes; abductive learning; tangible programming.

1.2 Project assignment

The main goal of the project was to create a basic, functioning prototype for presenting the concept to investors. During an initial meeting, the product vision was described: a toy kit consisting of multiple car parts that users select and assemble to design their own car. Upon combining parts, a virtual model of the car would be generated and presented in a game on a PC or tablet. The gameplay should incorporate physical principles to simulate real-world

[1] http://icec2015.idi.ntnu.no/

effects on the car. Utilizing these simulations would create an immersive learning platform fit for STEM[2] education. This principle of a virtual response to a physical action is called tangible programming. The prototype would be made to include the most critical specifications and functions. In addition, additional functions were to be included if applicable and within time constraints.

1.3 Technology challenge

The main challenge of this master project was to integrate solutions from multiple engineering disciplines into one functioning prototype, within a set timeframe. In this specific example of a tangible programming based product, the technology included elements from material science, mechanics, electronics, mechatronics, computer science, and user interface design. The resulting system of components had to function in unison and convey the concept of the product to a user or potential investor. This portrayal of the end product meant that the prototype did not necessarily have to include the same functions as the end product, but rather give a synthesis of the technology. This function synthesis was critical to the success of the project: developing the actual technology would require far more time than available in this project. The final prototype functioned similarly to already existing tangible programming products (Danli Wang, Wang, & Liu, 2014). It was therefore not a direct representation of the patentable IP of the end product vision. Most notable of these similarities is the use of a central module that performs all the identification and communication with a computer. But during the development, measures were taken to accommodate potential expansion of the prototype for better distinction of between existing products and our patent. Due to the project deadline, the prototype was further developed rather than focusing on developing new, patentable technology. But key aspects and learnings about the technology are sure to become relevant when a patentable solution is developed.

1.4 Timeline

The prototype was developed over a total of 14 weeks, of which ca. four were spent writing the two papers for publishing. In retrospect, four major milestones were distinguishable in the project: first critical function prototype; submission of papers for publishing; investor day with live demonstration; final project prototype. The selection of functions and sequence of their prototyping was based on the project owner's input regarding IP (Intellectual Property), input from mentors, and experience from previous project courses focused on concepts in the

[2] Science, Technology, Engineering, and Mathematics.

fuzzy front end. The process was incremental and flexible in nature and used probing of ideas to test concepts, make decisions and reiterations. This process was the topic of (Gerstenberg et al., 2015), as an example of using wayfaring mindset for simultaneous prototyping in a solution space related to multiple engineering principles.

The first prototype to achieve actual, tangible programming was complete after six weeks (first milestone). It was then suggested that two papers on the development process and the prototype itself, should be submitted for publishing at a conference on entertainment computing. Four weeks were spent writing and submitting these papers before work on the prototype was continued (second milestone). Both papers went through review and were edited before final submission until mid-July. The first submissions marked the transition into the improvement phase where work was primarily focused on improving and adding new functions to the existing prototype. Thus, there was a change in process: probing in an undefined space was used in the first phases to sense how the system would interact and develop thereupon; in the final phases, defined boundaries were established by the solutions that the prototype utilized.

In week 12, Metis Productions took part in an 'investor day' where the company would present the product idea for a panel of investors. A live demonstration of the prototype was requested, and the prototype was further improved to be made fit for this occasion (third milestone). At this point, the prototype included a satisfying number of critical functions, and work on developing new critical functions was halted. In the last two weeks before ceasing work on the prototype, efforts were focused on making the prototype ready for handing over (fourth milestone), as well as creating various short prototype presentation movies for Metis Productions. This marked the transition into the documentation phase for this master thesis.

1.5 Final prototype

The final prototype included one central car part called coupe, and eight external car parts to be connected to the coupe. Two of the parts also had potentiometers for changing appearance of the virtual model. The coupe contained a microcontroller with a BLE (Bluetooth Low Energy[3]) unit and a circuit board, while the external parts contained resistors of different values. For a detailed description of the circuit board, see section 4 in (Reime et al., 2015). The coupe was able to identify the external parts by measuring the resistors through electric connectors on all four sides and sending a string of data via Bluetooth to a computer. The

[3] www.bluetooth.com/Pages/Bluetooth-Smart.aspx

computer read the string and processed the data in an algorithm that identified the structure of the assembled parts. The structure was then presented by loading premade 3D models that resembled the physical parts. Thus, tangible programming was achieved by this real time, on-screen construction of the virtual model depending on the physical assembly.

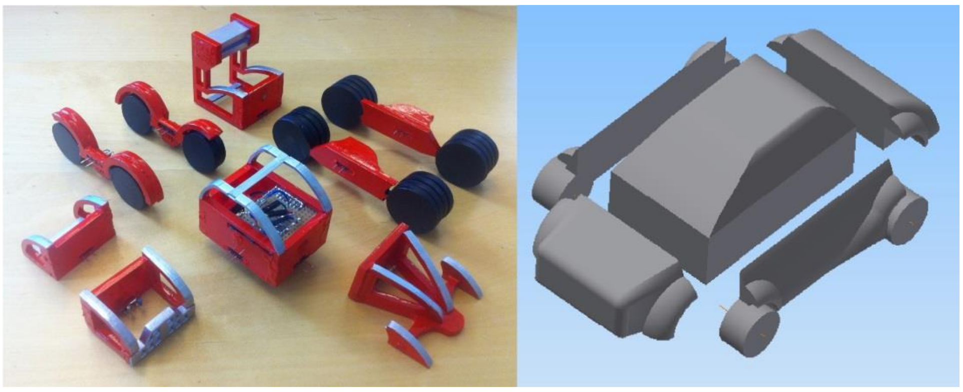


Figure 1.1 Left: All physical objects of final prototype. Right: Exploded view of virtual model.

2 Technology background and development

The product vision belongs in the domain of Tangible Programming, and this serves as a preface for the technology background. The following sections describe facets of the final prototype and the most important functions developed. The prototype development up to the first milestone is described in detail in (Reime et al., 2015). The project task is of a highly interdisciplinary nature with important relations to and between multiple engineering disciplines. Concrete developments are divided among these disciplines to give an overview of their interrelations and the reasoning behind the decisions taken in the process of achieving tangible programming. It is very important to note that the gameplay knowledge domain was unavailable during the entire length of the project. It was initially assumed that gameplay would become relevant at some point during the project. However, due to the product being in its infant state, no gameplay had yet to be developed. Gameplay will naturally become an integral part in the future prototypes of the Fibo Car.

2.1 Tangible programming

The key feature that sets the Fibo Car game aside from any other car game is the use of a tangible interface that correlates *directly* to the gameplay. This principle of game control is called tangible programming. (Sapounidis et al, p. 226) suggest this interpretation: “TUIs (Tangible User Interfaces) in general may be considered as physical objects whose manipulation may trigger various digital effects, providing ways for innovative play and learning for children or novice”. The fundamental goal appears to be utilizing the potential of tangible tools for learning, especially in STEM related education: “A tangible environment can potentially remove both of these obstacles (requirement of computer literacy or letteracy *(sic)* from the user)” (Smith, 2014, p. 430). Following the interpretation of Sapounidis et al., it appears that products need not conform to any specifications other than those mentioned. This means tangible programming can include a vast variety of solutions, something Wang et al. confirm in their review of tangible programming products (Danli Wang et al., 2014). But while the idea of tangible programming for STEM education is sound, there is general consensus that more research is still needed to confirm the positive effects of such tools: “In recent years, educational robotics has become an increasingly popular research area. However, limited studies have focused on differentiated learning outcomes based on type of programming interface” (Strawhacker & Bers, 2014). Sapounidis et al. add the need for design guidelines to the research need:

> Despite the various design efforts and the theoretical frameworks related to Tangible User Interfaces and children education, there is a lack of (a) empirical research investigating the possible advantages of TUIs against graphical interfaces and (b) design guidelines and available tools that combine tangible and graphical programming capabilities.
>
> Sapounidis et al., 2014

Wang et al. attempt to establish such design guidelines, and present a study of how their own T-Maze can improve programming skills (Danli Wang et al., 2014). The T-Maze functions similar to the Fibo Car in that the interface is a screen, but it differs by using a camera and pattern recognition to identify the 2D structure of assembled objects on a desktop. There is no mention of the perhaps more popular LEGO Mindstorms[4] and LEGO MyBot[5], likely because Wang et al. prioritize tools for learning programming skills. Similar to the LEGO products, the Fibo Car aims to use immersive learning through engineering rather than solving problems on an equational level. In fact, Toque is the only real-world based product mentioned by Wang et al., and they found it to not include adequate programming features. This may in fact suggest that the research into tangible programming educational tools may not apply to the Fibo Car simply because it is based on real-world scenarios and engineering, similar to Toque, LEGO Mindstorm, and MyBot. Games have long since been accepted as positive for developing children's task solving skills, but more dedicated research into games with the goal of immersive learning is required in order to determine if the Fibo Car can actually be used in STEM education.

2.2 Material science

The car parts were made from 5mm acrylic plastic sheets, laser cut into pieces that would fit similar to cogged works. The plan was initially to 3D-print all parts in order to make the prototype look as much like an actual car as possible. However, it was pointed out that making the technology visible could be beneficial when presenting to investors. None of the 3D printers available could print with transparent material such as acrylic. Additionally, using 3D printing is very time consuming compared to laser cutting, the parts would likely be fragile due to their size, and drilling holes or cutting away material posed a significant risk to the structural integrity. The choice therefore fell on staying with acrylic plastic, in spite of the challenge of using 2D elements to build 3D shapes.

[4] http://www.lego.com/nb-no/mindstorms/?domainredir=mindstorms.lego.com
[5] http://brickset.com/sets/2916-1/MyBot

Initially, clear tape was used to hold the pieces together because it provided a simple, cheap, and easily modified way of fastening pieces. Later, hot glue was used to get a more permanent, professional looking, and robust solution that also had ability to be reformed. For the sake of safe failing, permanent glue was only applied in the very final prototype before the parts were painted. It proved to be very important to be able to modify, improve, or redo the fastenings: pieces got broken, fastenings had to be strengthened, and access to electrical components often required that parts were disassembled. Choosing the 'prototyping friendly' tape and hot glue in the first stages, and permanent glue only in the very end, was in line with the mindset of incremental prototyping. Additionally, they were accessible, seemingly seamless, and quick to use when compared to using lock pins in holes, or bolts and nuts.

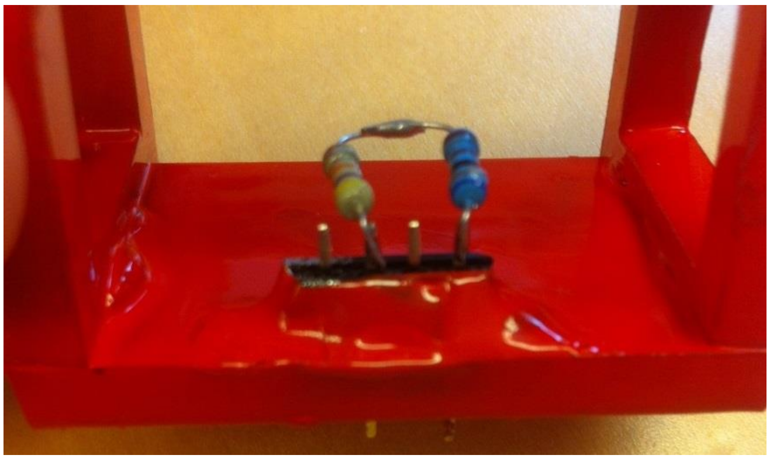


Figure 2.1 Painted glue seam around connector

2.3 Mechanics

Due to the size of the prototype and choosing to use pieces of acrylic plastic for construction material, structural integrity was not an issue apart from when making sure the pieces stay connected. This was solved in unison with developing the part interface for electrical connection: the pin connectors held enough weight to support all parts except the F1 front and rear parts. Magnets had been previously looked into for aiding the orientation of the connectors (section 2 in (Reime et al., 2015)), but were deemed unnecessary when it was decided to not allow orientation of parts. When it became clear that the F1 front and rear parts could not be sustained by the pin connectors alone, magnets became relevant again and were embedded in the coupe sides and on the F1 front and rear walls. Originally, the idea was not to use magnets as a tool for avoiding wrong orientation, but rather for structural integrity not depending on mechanical connection. T-Maze uses the repulsion and attraction of magnets to let users know if the objects are connected with correct orientation (D. Wang, Zhang, &

Wang, 2011, p. 130) (Danli Wang et al., 2014). LittleBits Bitsnaps[6] uses the same principle to avoid wrongful connection of electrical components, as described in section 2 in (Reime et al., 2015). The final prototype contains the same function in that the magnets in the F1 front and rear parts repulse each other, making it impossible to combine these two parts. This it clearly out of line with the end product vision's requirement of complete freedom in combination of parts. But because the prototype only has one 'smart' unit and the F1 front and rear parts serve no function when connected with each other, I deemed magnets for structural integrity to be a permissible solution for this prototype.

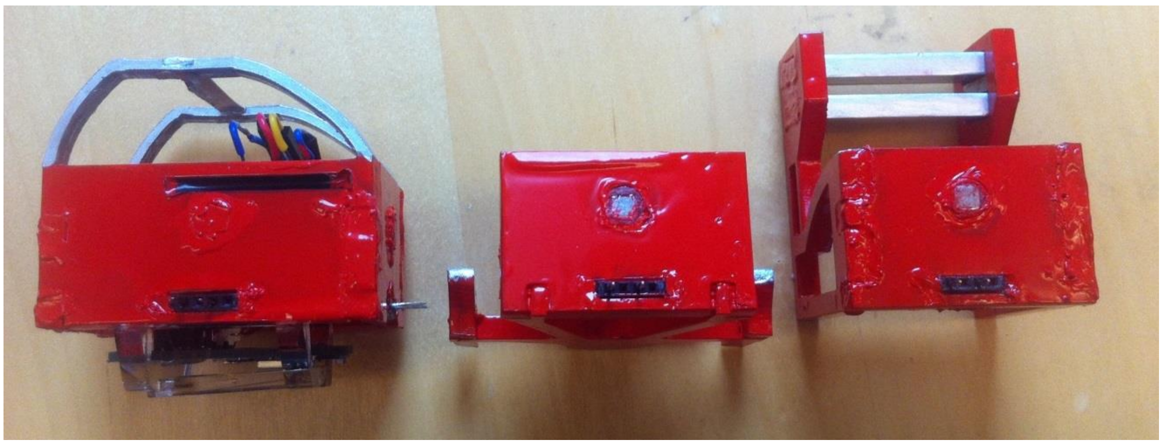


Figure 2.2 Magnets on sides of center and F1 parts

Even though the connectors in the final prototype served the desired specifications (easy assembly, structural integrity, and electrical connection), some ideation was done to improve upon the design in future prototypes: free orientation between parts was always assumed to become a requirement in the future. Note: this must not be mistaken with 'free combination' in the previous paragraph. Various concepts were sketched out during the length of the project, and some proved interesting for future development. Materials were ordered for prototyping, and a low resolution prototype was made: diametrically magnetized ring magnets were glued to a piece of bread board material with three soldered points. The points were connected to three wires on a slip ring so that the rings and points could rotate freely. Two identical prototypes were made to test the connection, but I underestimated the loss of magnetic force when the rings were suspended by the bread boards. In the end, it was decided to keep the old pin connectors in the prototype because the kit would not be able to benefit from a rotatable connection. New sketch designs were made based on the same components,

[6] `www.littlebits.cc/accessories/bitsnaps`

but there were no more physical prototypes of free orientation. This minor prototype is included in the delivery of the final prototype.

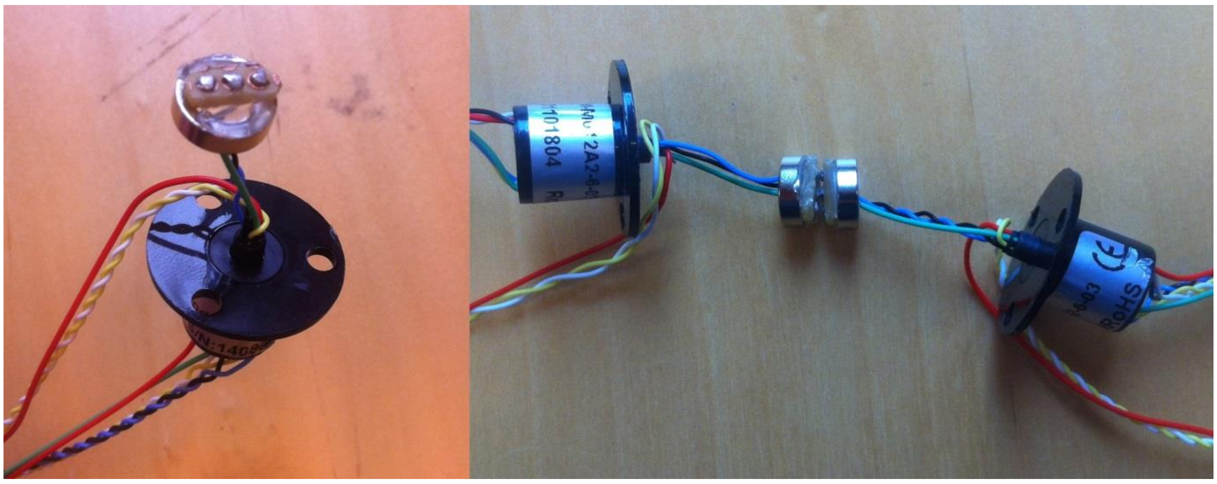


Figure 2.3 Left: Single magnetic ring connector. Right: Two magnetic ring connectors connected

To design the structure of the physical car parts and the virtual models, Autodesk Inventor[7] was used for 3D modelling. As mentioned, the physical car parts came with the challenge of creating a 3D object from 2D pieces, namely the 5mm acrylic sheets. By making all the 2D pieces and assembling them in Inventor, it was possible to verify that each part could be assembled as intended, before printing any pieces with the laser cutter. It was also possible to create assemblies of all the car parts in various combinations to get a preview of what the physical kit would look like.

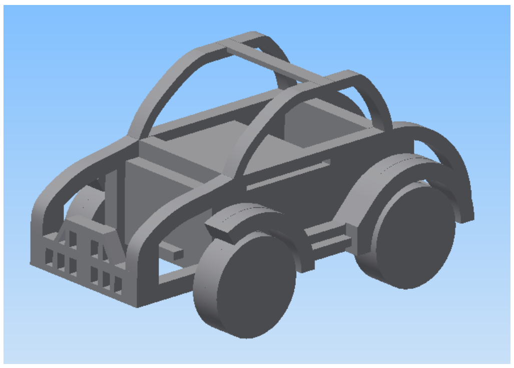


Figure 2.4 3D model of assembled physical shells

The most critical constraint regarding the parts was that the coupe needed to be able to house the circuit board, wires, and four connectors. Detailed 3D models of the circuit board and

[7] `http://www.autodesk.com/products/inventor/overview`

connectors made it possible to verify the size and placement of the components inside the coupe. After the second milestone, a LightBlue Bean[8] was added to achieve wireless communication. This would not fit inside the coupe, and removable supporting brackets were made (figure 2.5).

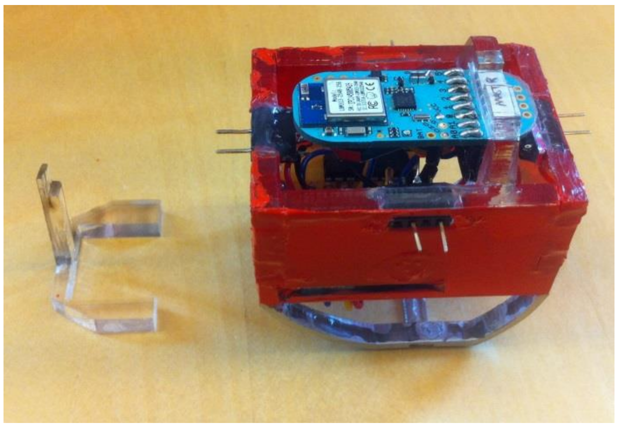


Figure 2.5 Supporting brackets and LightBlue Bean

2.4 Electronics

At the very heart of the prototype was the ability to identify car parts. During initial ideation, resistors came up as a means of creating a passive, measurable size for identification. The idea was to use thresholds to identify the parts. E.g. if the measured resistance in a given part was between 20 and 25kOhm, the system would recognize this as a specific car part. Other suggestions, such as RFID and various 3D scanning methods were discarded for the sake of simplicity and personal knowledge about the technology. (See Appendix A – document named 'Prototype 1.0: Concept repository') After some promising prototyping, I decided to move forward with this solution, and it remained throughout the project. Not looking into other ways of identification may seem out of line with the exploratory nature of the wayfaring mindset, but 'never go home to early' is another rule that certainly applied in the case of resistors. Later in the project, work was done on exploring other ways of identification, specifically how to send data directly between parts (see section 2.5).

Ability to interact with car parts to enhance appearance or performance in the gameplay was a key part of the product vision. This was an optional function to be included in the prototype only if appropriate. It turned out that resistors were ideal for this: using potentiometers and switches as part of the identification of resistors, they would bring the measured resistance

[8] `https://punchthrough.com/bean/`

from one threshold to another, effectively swapping car parts virtually while keeping the same physical part. A side-version of the first prototype used a switch to change the 3D model of the engine, while potentiometers changed from small to large wheels (figure 2.6). In the final prototype, only the wheel-changing attribute was kept because there were no more available resistor thresholds. It is noteworthy that the original decision to base the system on something as physically basic as resistance was partly reason why the system had such a flexible nature. If a more 'high tech' solution had been chosen, modification and improvement of the prototype might have proven significantly more difficult.

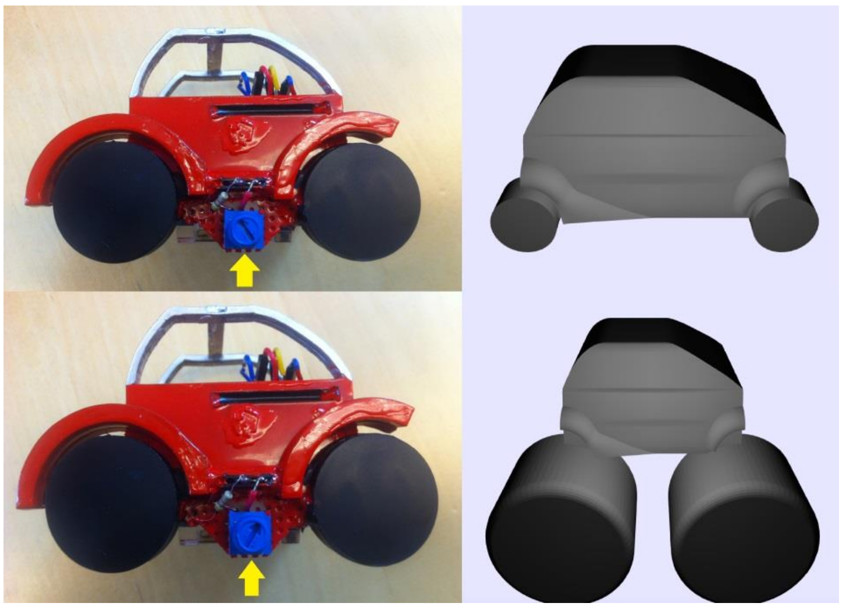


Figure 2.6 Wheels changing size by tangible input of turning potentiometer (arrows)

One of the first priorities of the project, after having decided to go forward with resistors, was how to develop a connection between parts. I developed a principle for measuring resistance across a three pin connection, and the first physical prototype built on LittleBits connectors (figure 2.7). The principle was designed so that other 'smart' parts could be created based on the exact same connectors, but the coupe remained the only car part with ability to recognize other parts (i.e. measure the resistance in a neighboring part). This was an important finding early in the project, and a result from using simple and fast probing. One of the earliest prototypes was built on this principle and is described more in detail in week two in (Reime et al., 2015). The remaining components of this prototype are included with the final prototype.

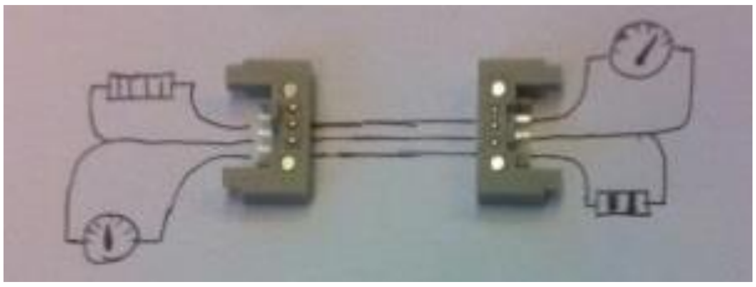


Figure 2.7 LittleBits Bitsnaps over paper drawing showing principle for measuring resistors

A key aspect of the system was using only one analogue port for measuring the resistors in all connected parts. This required the ability to decide which connector was active at any point. The solution was to make a circuit board containing a shift register and transistors. Detailed description of the board is found in (Reime et al., 2015). The entire solution of the electrical system was made to fit an eventual combination with the LightBlue Bean, which it did successfully after significant modification (see section 2.6).

2.5 Mechatronics and computer science

There were two major components in the prototype related to computer science: the code used to program the microcontroller for measuring resistors and sending data; and the code used to receive data from the microcontroller, process the data, and present the structure of the car in a 3D model format. The first prototype used an Arduino Uno[9] connected by wires to all components inside the coupe. Processing 2.2.1[10] was used to receive and analyze data from the coupe, and display the 3D models corresponding with the physical structure.

After the first prototype was finished, the project owner made a list of prioritized functions that were to be attempted to include in the prototype before the Investor Day, i.e. the third milestone. The function of highest importance was wireless communication with a computer, and this posed some major challenges for the identification system: a LightBlue Bean was chosen as wireless microcontroller platform, while the first prototype used Arduino Uno with USB connection directly to a computer. The major advantage of using a Bean was that it contained the same type of microcontroller (ATmega328p[11]) and was programmable in Arduino Java program. Thus, the specified interface between microcontroller and computing unit would require minimal change to the modular architecture (Sanchez & Mahoney, 1996,

[9] http://www.arduino.cc/en/Main/HomePage
[10] https://processing.org/
[11] http://www.atmel.com/devices/atmega328p.aspx

p. 66). Other ways of wireless communication were briefly explored (transceivers, XBee[12]), but the Bean was deemed superior for its prototyping friendliness and compatibility with the system. The first major challenge presented itself when it turned out that Windows 8.1 OS (Operating System) protocols did not allow setting up virtual COM ports (Communication port). The solution was to switch to a computer with OS X. As a consequence, the prototype became unable to function on a Windows OS. This posed a limitation in presentation of the prototype because any computer using Windows could not run the software. The time cost of developing a system with a different way of wireless communication would likely have been exceedingly higher.

The first prototype used direct link from car parts to computer, which meant that the sending and receiving of data was stable and fast. Although the codes in the first prototype and the Bean were identical at first, they measured different resistance values on identical resistors. Additionally, there was a vast reduction in the number of resistor thresholds available for different parts. The most harmful effect, however, was that some car parts affected the measured resistance in other car parts, thus making some specific combinations malfunction. (It was later suggested that this was because the analogue gates were not automatically grounded in the Bean, something I had no knowledge of at the time.) As a consequence, major modifications were needed in the Processing code, Bean code, in-part resistors, and even circuit board: a new series of algorithms in the Processing code actively changed the received data before going through the ‘part recognition’ algorithm; the Arduino code in the Bean was changed from continuously measuring and sending data to only sending data at timed intervals if three sequential measurements were identical, thus filtering out spikes or individually different measurements; new wires on the circuit board bypassed critical components in the first prototype; new resistors with larger span between resistance values were placed in the external car parts. The wires were made from jumper wires with headers so that the bypass could be removed by hand if it became necessary to go back to Arduino (figure 2.8, left). But although the system worked after all these modifications were done, it became impossible to completely revert to the old setup from the first prototype in case the Beans failed. This was a significant risk that involved much time spent developing the algorithms and lacked the option to fail safely. It was nevertheless necessary to achieve the critical function of wireless communication. Also, at the time, it was seen as creating a totally new prototype that happened to use some of the same components as the first prototype.

[12] `http://www.digi.com/lp/xbee/`

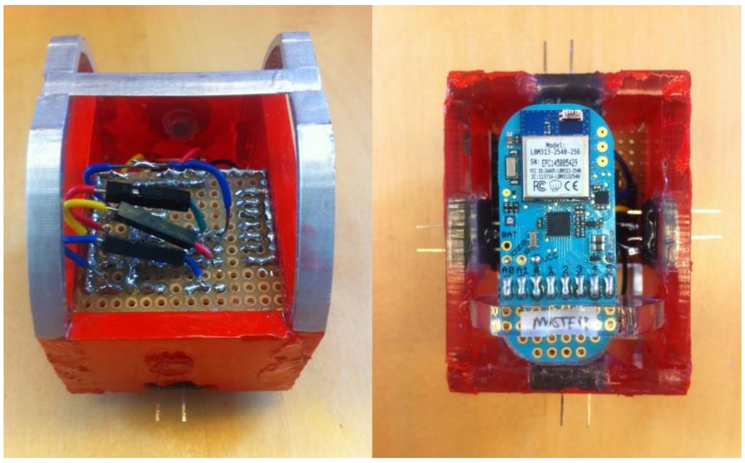


Figure 2.8 Left: Circuit board bottom side. Right: LightBlue Bean.

After having confirmed that the LightBlue Beans were suitable as wireless platforms, time was taken for a digression into an alternate way of intercommunication between car parts. While resistors proved to be a successful means of identification, its limitations in terms of available thresholds meant that it was likely unsuitable to be a part of the end product. Direct sending of data between parts appeared to fit well with the learnings from the resistor based prototypes, and it was decided to explore using Inter-Integrated Circuit, also known as I2C, for data exchanging between microcontrollers. I2C utilizes a ground port and two specific analogue ports called SDA (Serial Data) and SCL (Serial Clock). SDA and SCL can be found on both Arduino Uno and Bean. To avoid the risk of I2C failing on Beans, Arduino Uno was first used to learn how to use I2C: two Arduinos were connected and used I2C to communicate with each other (figure 2.9 left). Each Arduino was connected to a computer for monitoring the data transfer on their serial monitors. If implemented in the prototype, the connectors between objects would require three pins similar to the resistor connectors. But orientation of the pins would not be the same, which pointed out how symmetry also needed to be taken into account in the future.

The first test using Arduinos and I2C was successful, and the next step was to start using Beans. One of the Arduino Uno's was switched with a Bean (figure 2.9 middle). The Bean sent the serial data over Bluetooth for monitoring, and the system worked just like before. Another Bean was promptly added, removing the Arduinos from the setup (figure 2.9 right). The data exchanging worked flawlessly there as well, confirming that I2C could be used for inter component communication on the Beans. In addition, the Beans used principally identical codes. This was necessary in order to deviate from the 'master component-slave components' relationship found in other tangible programming products. This deviation in development of the final prototype served mostly as a probing into future possibilities, but it also pointed out potential challenges in the future. E.g., how to cycle between which connectors are active at any point. The scalability of I2C was not a coincidence, and again prototyping with safety to fail and ability to scale led to success. Much of the ideation and development that followed was based on using I2C or a similar technology for intercommunication. At the end of the project, this concept remained the most promising for future development.

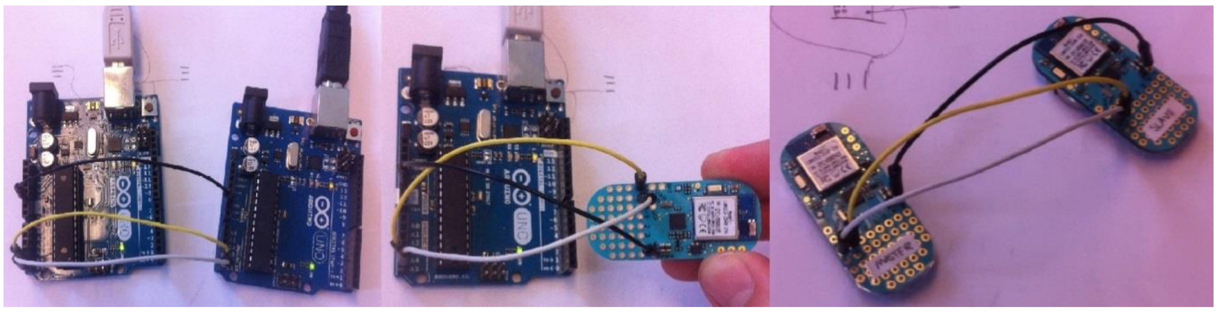


Figure 2.9 Iterations of I2C. Left: Two Arduino Unos. Middle: Arduino plus Bean. Right: Two Beans

2.6 User interface design

While most of the solutions developed during this project are invisible to most users, the core goal of the whole project was still to convey a concept to a user or investor. Therefore, certain decisions that may seem arbitrary to this point were in fact ultimately made to accommodate the user experience. The three pin connectors supplied electrical connection and structural integrity, but they were also designed for simple assembly of car parts. Apart from the script initializations, the part of the Processing code needed to import and convert the data from Arduino is only nine lines:

```
while ( StrFromArd.available() > 0) {
  myInts = StrFromArd.readStringUntil('\n');

  if (myInts != null) {
    //Splits myInts into array of strings
    String[] SplitMyInts = split(myInts, ";");

    for (int i = 0; i <= 3; i++) {
      //Converts the array of strings into array of ints
      float floatVal = Float.parseFloat(SplitMyInts[i]);
      intVal = int(floatVal);
      Base[i] = intVal;
    } //for i
  } //if
  println(Base);
} //while
```

The remaining 76 lines of the Processing code consist of algorithms for placing the correct 3D models in a presentation window (Appendix C); the algorithm for receiving data was simple and small by comparison. If the goal was to create a prototype able to receive simple raw data from the physical prototype and not a presentation friendly prototype, the project would never have required these algorithms to be made. The two sets of 3D models (one to make the physical car parts and one loaded in Processing to present structure) did in no way affect the core technology apart from how the components would fit in the car parts: the models and parts could just as easily have been of boats, cities, molecules, or anything module based. The very last modifications to the prototype were focused on making the final prototype presentable: a removable supporting bracket for the Bean was needed for simple battery replacement; a coat of red paint (issued by the project owner) with silver and black details made the previously transparent prototype look more like an actual toy car.

The only facet of the final prototype that I was unable to improve the user experience in, was the interaction with the necessary software on the computer: the Bean required the Bean Loader[13] application for accessing the Bean. This did not integrate with Processing, and both programs had multiple ‘bugs’ that could make interaction with the prototype very difficult for an untrained user.

2.7 Final Prototype vs. Product Vision

The vision for the end product describes a complex system of car parts that all communicate with each other and a computer simultaneously (Appendix E: Initial Product Vision). As explained in section 2.1, this is a new concept compared to other tangible programming products, which are primarily based around a central module that does all the identification of parts and communication with computer; without the central module, the system has no function. On the surface, the final prototype is identical with such existing solutions in that the coupe is the only ‘smart’ part that identifies other objects and communicates with the computer; the other parts only contain resistors, making them passive. What differentiates this prototype from existing products is that it contains features intended to give all the parts ability to identify and communicate: the connectors and electrical connection principle were initially designed for having an ohmmeter and resistor on each side of the connection (figure 2.7). This was a deliberate design meant to make it easier to create new ‘smart’ car parts in the future. Naturally, one such part had to be made before an entire system could be in place.

[13] `https://punchthrough.com/bean/bean-ios-loader/`

But the project progressed always with the expectation that additional ‘smart’ parts would require too much time for developing and integrating with the existing prototype. Thus, the prototype was deliberately designed for potential expansion, even though it seemed unlikely that it would occur. Clear distinctions can be made between how the final prototype works and which functions that are needed in the end product. This can in fact be seen as an advantage when presenting to investors: sharing and showing off technology before a patent is secured can give investors the impression that the company does not sufficiently protect its interests.

2.8 Design flaws in final prototype

While the system operated at a satisfactory level upon completion, mistakes were made during the process, and some design flaws deserve to be commented on.

When designing the placement of the pin connectors on the car parts, the assumption was that they would have enough strength to support all suspended parts through this connection only. This was true for the first set of car parts, but the F1 engine and rear needed extra support. If the connectors had simply been oriented vertically instead of horizontally or placed higher on the walls (given enough room for the circuit board), I believe they would have supported all parts. However, it did lead to the inclusion of magnets in the parts, which answered the concern of if magnets would interrupt the electrical or Bluetooth signals. They did not, and it proved a valuable opportunity to explore using magnets in the future. Another design flaw regarding the connectors was that car parts can be attached up-side-down without the virtual model changing the orientation of the part. This could have been avoided by making it physically impossible to attach parts up-side-down, although I believe it is a negligible fault.

As mentioned in section 2.4, when modifying the prototype to use Beans, the circuit board needed new wires to bypass certain components. The wires were made with headers for easy removal of the bypass, which made it necessary to place them on the visible side of the circuit board. These visible wires do not contribute to the appeal of the prototype, and a more elegant solution should have been found.

The very last modification of the final prototype was the painting. The Project Owner wanted a more toy-like look on the final prototype, and the prototype received coat of red and black paint with silver details. This task was saved for the very last because any other task would have to wait until the paint had dried. In the end, the prototype did look more realistic, but the paint job could have been improved greatly if more time had been available.

3 Publications to ICEC 2015

3.1 Preface for papers

The following two chapters contain the papers accepted for publishing at the ICEC 2015. The first is a demonstration paper where concepts related to entertainment computing is described. The second paper is a poster paper where the process used in the six weeks is described in the paper along with a poster (Appendix B). Both the physical prototype and the poster will be shown at the conference. Both papers function as subchapters of this chapter 3.

The papers are presented as similar as possible to their original Springer format, though some minor alterations in paragraph length may occur due to formatting differences.

Bridging Tangible and Virtual Interaction: Rapid Prototyping of a Gaming Idea

Thov Reime[a], Heikki Sjöman[a], Achim Gerstenberg[a], Pekka Abrahamsson[b], Martin Steinert[a]

[a] Department of Engineering Design and Materials, NTNU, Richard Birkelands vei 2B, 7492 Trondheim, Norway
{Heikki.Sjoman, Achim.Gerstenberg, Martin.Steinert}@ntnu.no, Thov@stud.ntnu.no
[b] Department of Computer and Information Science, NTNU, Sem Sælands vei 9, 7491 Trondheim, Norway
pekkaa@ntnu.no

Abstract. The Fibo Car is an example for a game interface that allows a user to modify a virtual car in a racing game through assembling tangible car parts. This paper describes the 6 week development journey towards a fully functional proof of concept prototype, reflections on the process as well as the technical details of the prototype.

1 Introduction

The basic idea of the Fibo Car game project is that the player can construct a real world car model out of tangible building blocks. The structure of the model is digitally recognized and it influences the properties of the virtual model in the car racing game.

In this paper we focus solely on the development of the tangible objects, the structure recognition, and a virtual representation without any gameplay. The game idea is related to games like Kerbal Space Program [1] or Besiege [2] except that the constructing takes place tangibly like in LEGO Mindstorms [3].

The solution presented here has one central part that can detect the attached neighboring parts. The identification is realized by measuring and identifying part-specific resistances with an Arduino Uno microcontroller board [4] wired to a PC. A virtual representation of the identified tangible model is then shown on a screen. The latest version of the prototype is shown in figure 1.

The upcoming section describes the development journey of the first 6 weeks. It includes failures, dead ends, and gives reasons for the actions taken. We concentrated on development speed using a process with rapid iteration cycles in favour of fast learning and quick improvement without project control by predefining requirements and a priori budgeting.

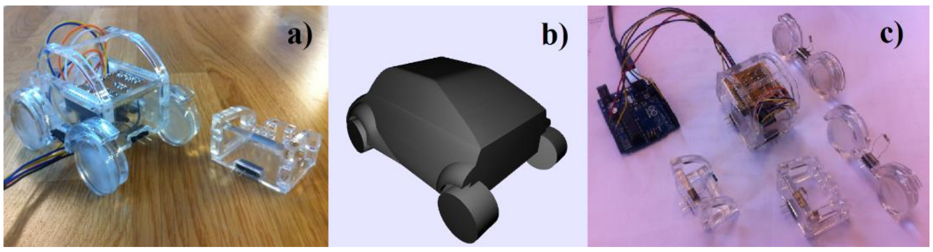


Fig. 10. Latest proof of concept prototype showing the tangible model in a) and the corresponding virtual representation in b). All available tangible car parts of the latest prototype are shown in c).

2 Development Journey

The project started with a presentation of the basic game idea by the problem owner to the developers. The aim was to reach a common ground on the project vision and the reasons behind the idea. This initiated a brainstorming about possible solutions. The main challenge was perceived to be the structure recognition. Therefore, we started by exploring possible technical solutions on paper. When considering radio communication and information through light signals, measuring resistances turned out to be the easiest to develop and cheapest alternative. Concerning the algorithm for structure recognition, we realized that it is simpler if each car part is only detecting its nearest neighbor instead of all parts detecting the entire structure. Therefore, resistance identification and neighbor detection were chosen to be pursued. The resistor solution was tested with resistors on a solderless prototyping board measured with ohmmeters. The principle was confirmed as functional. The idea here was to make sure as early, simple and fast as possible that principles worked with components already available in the lab in order to minimize the amount of time wasted in case it did not work. In the beginning of the **second week** of the development journey, we determined that we require three electrical connection points for connecting two car parts. This fit with the already existing BitSnap connectors [5] that had three electric connection points. They used magnets and their own shape to ensure a consistent and non-ambiguous electrical connection. Those BitSnap connectors were designed to be soldered onto a circuit board, and solderless prototyping boards were too large to fit into the car parts. Knowing that the principle worked, we decided that spending time for manufacturing a soldered circuit board version was a safe investment. The result is shown in figure 2 on the left. From this prototype we learned that the BitSnap connectors were mechanically not rigid enough to support the weight of the car parts. Furthermore, the connectors were not symmetrical, meaning that only matching pairs were combinable. This was in conflict to the fundamental idea of allowing any given car piece to connect. Anyhow, the structure recognition technically worked. Therefore, we continued developing the remaining critical functions such as measuring the resistors with a microcontroller, sending this data to a PC and displaying the measurement on a screen. We decided to not bother with improving the connectors at this point in time to save time towards achieving the critical functions of our envisioned game idea. The microcontroller measurement was prototyped using an Arduino Uno because it is easy to develop, immediately accessible in the lab, and already offers the software to display the results of the measurement on a computer screen. After merging the existing development stages and fulfilling the critical functions as early as possible (digitally recognizing a structure of mechanically attached objects, transferring this data to a PC, and displaying it) we could now focus on improving the existing solution. During **week three**, we intended to focus on shrinking the Arduino microcontroller solution to a size that is suitable for embedding in a car part. Light Blue Beans [6] appeared to be a suitable solution that is already available in the lab. They also had the advantage of replacing the wire connection between the Arduino and the PC by wireless Bluetooth communication. One upcoming problem with Light Blue Beans was that they only have two analogue inputs. This lead to the use of shift registers to channel many measurements through few input pins on the microcontroller. The shift register also work in combination with the Arduino Uno and the Arduino was kept because it is more convenient to program.

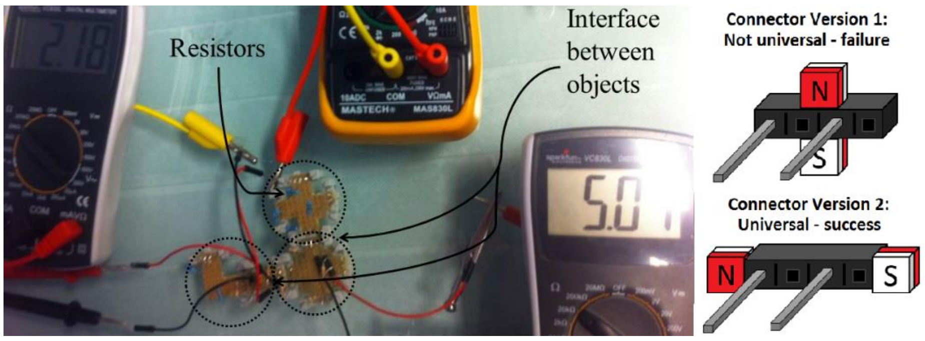

Resistors
Interface between objects
Connector Version 1: Not universal - failure
N
S
Connector Version 2: Universal - success
N
S

Fig. 11. Left: the resistances on the connected circuit boards in the middle are unique for each connection and measured by ohmmeters. Right: two sequent designs of a mechanically more stable connector.

Week four started with developing mechanically stronger and symmetrical connectors. This was implemented by using larger and stronger magnets and using pin connectors to further stabilize mechanically. The first design is shown in figure 2 in the top right. However, this design turned out to be impossible to connect to an identical connector because the magnet orientation would not match. It required a matching counter piece and was thus no improvement to the previous solution with the BitSnaps. The bottom right design solved this problem. This design flaw was discovered by building and testing the design in a very rough way instead of technically drawing and machine producing the parts. This decreases the risk of design errors and thereby saves more costly resources at a later development stage when such errors have more profound implications.

All electrical components were now on two large breadboards that required a lot of space. The components had to be merged on one platform so that they would all fit safely inside a physical shell. This was accomplished by soldering all components (transistors to control a shift register, reference resistors and header connectors) compactly onto a custom circuit board.

The next issue to be tackled was to advance the virtual representation from a line of text to a car look-a-like representation. We took two approaches into account: The first was a photograph based version where the PC would display a corresponding picture for every possible combination of parts. The second was using 3D models for representation of the car structure. We decided to develop the second option because the number of pictures needed for the first was inconveniently large when scaling up the number of car parts. We used Processing [7] to process the data coming from the Arduino, determining the structure and displaying the models on the PC screen. The system was first tested by displaying a rocket and a chair as substitutes for the virtual car model. Only after verifying the concept, we continued to make virtual representations of the car parts using a CAD software and importing those models to Processing. After confirming that Processing was a reasonable option for displaying a digital representation, **week five** began by drawing the car model parts and implementing them within Processing. At the same time, we also pursued the implementation of Light Blue Beans to make the physical model wireless. However, we experienced that Windows 8.1 did not allow importing serial data via Bluetooth. We could not instantly resolve this problem with the resources at hand and therefore decided to move back to the proven technology to not lose more time with this issue. During **week six** we explored switches, buttons and potentiometers as extra tangible inputs to alter the car parts. Since the gameplay did not yet exist, the visual representation was the only possibility to make adjustments to. We showed that this extension was technically functional and could also be used to change non-visual properties in the game later on. But there was no meaningful reason to develop something further that had no use at the current development stage. Therefore, we stopped after the proof of concept and continued to make a laser cut physical car model in acrylic. The acrylic car model was combined with the existing technology and combined all aspects from physical model to structure recognition and virtual representation. Figures 1 show the prototype after these six weeks of development.

3 Reflections on the development process

It turned out that our process was very similar to the wayfaring process described by Steinert and Leifer [8]. Both processes are largely based on rapid iteration cycles of design, build and testing ideas as early and quickly as possible. We tested the most critical functions with the resources that are readily available in the lab to fail early and mitigate the risk of losing advances that become unusable due to a later design changes. The early testing lead to learnings that shaped the development journey; the design emerged over time.

4 Detailed description of the latest prototype

The final prototype consists of one central part connected to a PC, and four external objects that can be attached to the central part. The central part has four connectors, one on each vertical side, on which external parts can be attached; each external part has only one connector. When no external parts are attached to the central part, a 3D model resembling the central part is displayed on the connected PC screen. Upon attaching an external part to the central part, a virtual 3D model resembling the attached part is automatically updated.

The identification of the neighboring car parts is achievedby the measurement of resistors through the connectors on the sides of the car parts. All connectors are made from 4 pin headers where two alternating pins are pulled out (see figure 2, bottom right). In the external parts' headers, a resistor is placed with one pin hole between its two legs. In the central part headers, two wires are connected to the female pins that the external connectors will fit into. Thus, when an external part connector is connected to a central part connector, we get a closed loop that runs through one wire into one of the central part header pins, through the male pin on the external part header, through the resistor, and back across to the other wire. This design is made with the intention of having multiple 'central parts' in the future that can measure each other's resistors. So far in this prototype, the central part pins that connect to the external header serve only for structural integrity.

The connector wires are connected to an analogue gate and ground on an Arduino Uno. The Arduino is able to calculate the resistance between ground and the analogue gate by comparing it to a reference resistor between its 5 volt supply and ground. Because there are four connectors and we use only one analogue gate, a shift register is used to control which connector has current at any time. The shift register is placed on a custom made circuit board along with the reference resistor, four transistors, two rows of headers for the connector wires, and a series of headers for easy connection of wires from the Arduino. Three wires connect three digital pins on the Arduino to the shift register. The shift register is connected to the gate pins on the transistors which open the current through the various connectors. Thus, the loop through ground, resistor, and analog gate is controlled. The circuit board is placed inside the central part and connected to the Arduino through a total of six wires (three digital, 5V, ground, and analogue).

When measuring the resistors, the Arduino uses as sequence of North, West, South, and East when the central part is seen from above. For each measurement, the value is serial printed, and a semicolon is added between the values. Processing 2.2.1 imports the string through the COM port on a PC. Before Processing can use the data for anything, it must convert the string into integers and store them in an array. The semicolons act as delimiters for the values. Processing then takes each value in the array and compares it to a set of thresholds.

Processing displays a rotating 3D model resembling the central part in a window. Depending on which interval between thresholds a certain measured value is, Processing displays a corresponding 3D model next to the central part. The correct position is acquired by the position of the value in the array, thus the reason for the compass sequence in the Arduino.

All 3D models are made in Autodesk Inventor [9] and converted into an .obj format. Processing loads the models in the setup of the script, and only displays them when receiving not NULL values from the Arduino.

The physical objects are made from laser cut pieces of 5mm thick acrylic plastic sheets. All pieces are modeled and assembled in Inventor before converting to a 2D format fit for cutting. The pieces are then assembled together with circuit board, central part connectors, and external connector. The pieces are held together with hot glue and clear tape so that broken pieces can be removed and retrofitting is easier.

5 Future plans

In the near future, we will focus on improving the existing prototype by including wireless communication, universally orientable connectors, alternate modes of inter-object communication, more than one ‘smart part’, and how to merge our tangible programming prototype with actual gameplay. We will continue to use a wayfaring mind set as we are satisfied with the results it has yielded so far. Looking further ahead, developing and testing of real gameplay is needed before we can undergo user testing and subsequent reiterations.

Acknowledgments. We extend our most sincere gratitude to project owner Leonore Alexandra Nilsen, CEO of Metis Productions AS, for giving us this challenge and introducing us to the Fibo Car Project. This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

1. Web page about “Kerbal Space Program”. Available at: https://kerbalspaceprogram.com/en/?page_id=7 [Retrieved April 28, 2015]
2. “Besiege” sandbox game. Available at: http://www.besiege.spiderlinggames.co.uk/ [Retrieved April 28, 2015]
3. LEGO Mind Storms EV3. Available at: http://www.lego.com/en-/mindstorms/?domainredir=mindstorms.lego.com [Retrieved April 28, 2015]
4. Arduino Uno microcontroller board. Available at: http://www.arduino.cc/en/Main/HomePage [Retrieved April 28, 2015]
5. Little Bits Electronics, BitSnaps. Available at: littlebits.cc/accessories/bitsnaps [Retrieved April 28, 2015]
6. LightBlue Beans. Available at: https://punchthrough.com/bean/ [Retrieved April 28, 2015]
7. Processing programming language. Available at: https://processing.org/ [Retrieved April 28, 2015]
8. Steinert, M., Leifer, L.: ‘Finding One’s Way’: Re-Discovering a Hunter-Gatherer Model based on Wayfaring. Int. J. Eng. Educ. 28, 1, 251-252 (2012).
9. Autodesk Inventor. Available at: http://www.autodesk.com/products/inventor/overview [Retrieved April 28, 2015]

A Simultaneous, Multidisciplinary Development and Design Journey - Reflections on Prototyping

Achim Gerstenberg[a], Heikki Sjöman[a], Thov Reime[a], Pekka Abrahamsson[b], Martin Steinert[a]

[a] Dep. of Eng. Design and Mat., [b] Dep. of Comp. and Info. Sc.
NTNU, Høgskoleringen 1, 7491 Trondheim, Norway
{Achim.Gerstenberg, Heikki.Sjoman, Thovr, Pekkaa, Martin.Steinert, }@ntnu.no,

Abstract. This paper proposes a wayfaring approach for the early concept creation stage of development projects that have a very high degree of intended innovation and thus uncertainty. The method is supported by a concrete game design example involving the development of a tangible programming interface for virtual car racing games. We focus onto projects that not only have high degrees of freedom, for example in terms of reframing the problem or iterating the final project vision, but are also complex in nature. For example, these can be projects that allow for the exploration and exploitation of **unknown unknowns** and serendipity findings. Process wise we are primarily focusing onto the early stage that precedes the requirement fixation, which we see as more dynamic and evolutionary in nature. The core conceptual elements that we have derived from the development experiences are: **simultaneous prototyping** in multiple disciplines (such as computer science, electronics and mechanics and engineering in general, **abductive learning** based on the outcome of rapid cycles of designing, building and testing prototypes (**probing**), and the importance of **including all the involved disciplines** (knowledge domains) from the beginning of the project on.

1 Introduction

To innovate incrementally is hard, to innovate “radically” harder still. Many an engineering project is fixating their requirements very early and then focus onto executing these predefined (and often unproven) specs as fast, as good, and as cost effective as possible. The usual outcome is a cost and/or time overrun if the innovative specs are to be met or a decrease in result quality. In a sense people perceive the innovation game often as a game under certainty with fixed variables and attribute values, fixed rules and thus predictable outcomes, hence it can be modeled, simulated and optimized. We argue that the innovation game is a game under uncertainty, with unknown unknowns that need to be discovered, evaluated and then discarded or embodied. The game is also played in a dynamic environment (opponents may counter and react) and even the rules are technically not fixed - take the Kobayashi Maru test situation as an example.

We argue that the development of highly innovative/uncertain and products is rather like an exploration journey. You have a vision where you want to end up and a general idea where your project is heading. However, neither can you know all the "moves" required to get there, nor can you accurately anticipate the effects and responses that one move will have in the future. Your expertise is your toolbox and it greatly helps in "playing your way through the project". Nevertheless the project is dependent on many unforeseeable events. In fact **unknown unknowns** (variables that are part of your problem/solution that you are neither aware off nor do you know their value) arise, serendipitous events present themselves, turning a complicated problem into a complex one - too complex to be planned out beforehand. We subsequently argue that sequential process models are not fitting for any innovative projects. [1,2].

The reference case [3] of designing a tangible game interface for racing games is used to extract reoccurring patterns during the design process and propose a method based on the experiences [4,5]

Our proposed method is based on abductive learning [6,7,8] and includes all involved disciplines from day one. This wayfaring model based on Steinert & Leifer [9] aims to allow the rapid requirement dynamics that become necessary during the development process.

2 Use of Wayfaring in the Example Case

Our example case is based on the **vision** of developing a physical car model as a tangible interface for manipulating/shape a digital car model in a virtual car racing game. A description of the project and the technical solution can be found in [3] This vision as an overarching goal was given to the developers instead of a precise list of requirements on how the technical solution is supposed to look like and the project architecture was allowed to emerge. This meant that the space of possible solutions is open, ambiguous and uncertain. In our example case, the problem became to identify car parts in the physical model that are attached to each other and to recognize the assembled structure. We explored the solution space by trying to come up with as many possible solutions to the problems as possible (divergent thinking). Possible ideas for solutions included measuring resistance, power dissipation of wireless communication devices or pulsed light communication for identifying connected pieces. For determining the structure we looked into a centralized structure with one central part that collects the data from all assembled parts and a decentralized structure that only required the detection and identification of neighboring parts. However, with no, or only little, experience it is unknown to us which of the suggested ideas was feasible to pursue. We call these unsolved uncertainties **unknown unknowns** because these open questions emerged during the development process and were in itself unknown to us before engaging the problem. We argued that resistors were the cheapest, simples and most reliable proposition and that just detecting neighboring car parts simplified the algorithm. Furthermore, having to use a specific center part restricted the liberty of freely using any car part separately in the virtual game. However, these are only arguments based on limited experience and in order to converge on the most promising proposal one has to build and test ideas to gain new knowledge. This repeating cycle of divergent and convergent thinking with designing, building and testing ideas is called **probing**. The probing cycles lead to **abductive learning** where the test result leads to design requirement changes and ideas for the next probing cycle. In our case, we realized that the measurement of the resistors fluctuates significantly. This lead to changes in the programming of the microcontroller that processes the measurement. The abductive learning from repeating cycles of probing leads to a **wayfaring** of opportunistically finding one’s way through the project. This means that the test results of the last probing cycle shape the future development. Figure 1a) shows the first test of the resistor principle. There we discovered that the idea is feasible and that three electrical connections between car parts are needed. This lead to the development of the setup shown in figure 1b) that uses BitSnap connectors that serendipitously already had exactly three electrical connections and allow the user to easily manipulate the physical car model. Testing these BitSnap connectors revealed that these are mechanically not sufficiently robust and not genderless, thus limiting the combinations of mountable car parts. This learning resulted in the development of the customized connectors shown in figure 1 c) and d) where the first version in figure 1c) turned out to be also not genderless and subsequently lead to the development of the second version in figure 1d). This train of subsequent probing cycles showcases the wayfaring journey that can be successive or dead ended.

Progress is achieved by the emergence of new ideas as a result of previous probing cycles. Therefore, it is important to minimize the time spent and to maximize the learning outcome for each probing cycle. This accomplished by concentrating on just testing the **critical functions** by building a **low resolution prototype** that is reduced to the properties that are necessary to only test the critical function. An example for this is the testing of the resistor principle as it is shown in figure 1a). The critical function was to find out how resistors can be used to identify connected parts unambiguously. To save time we compromised robustness, automation of the measurement, looks and compactness of the system to focus only on the critical function and thus used prototyping boards, header wires and ohmmeters that were readily available in the lab.

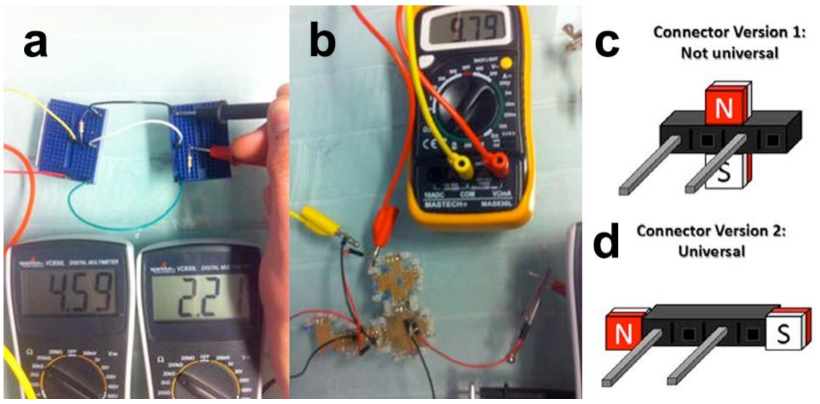


Fig. 1. a) first test of the resistor concept, b) testing with the BitSnap connectors, c) failed version of universal connector, d) successive version of a universal connector.

Imposing this train of thought to the entire project yields that prototypes that fulfill critical functions within different disciplines are merged as soon as they are available to test the system at large. The aim is to test and discover **interdependencies**. In our case, we combined the resistance measurement with a microcontroller, the information transmission to a PC and the virtual representation on the PC screen as soon as they were available in their most rudiment form. This means that all components from possibly different disciplines need to be **prototyped simultaneously**. Testing the entire system creates an **interlaced knowledge** between different disciplines. The structure recognition algorithm for example influenced the shape of the connectors and these changes had to be made in agreement with mechanical design of the car parts. This was possible because the developers of all disciplines were integrated from day one.

3 A wayfaring approach to early stage concept creation

In this part we describe a method that we derived from the project described above. The method has potential when finding and tackling previously unsolved engineering design problems that have no known existing solution. These problems are not necessarily complicated but rather complex according to Snowden and Boone [10]: they cannot be solved by asking experts to plan the final solution because they require the use of previously unproven and maybe even unknown concepts. In this context the development process becomes a wayfaring journey where the path towards fulfilling the vision emerges from making educated guesses and testing concepts, rather then a navigation journey along predefined waypoints. An optimum solution cannot be predicted when doing things that have never been done before. This method concerns only the early part of product development, the fuzzy front-end of concept creation, where the requirements of the product are not yet fixed. Figure 1 depicts such a wayfaring-inspired product development journey. This is a systematic and heuristic approach to developing something radically novel. The path to the end result will only be explored and discovered during the project. The journey consists of many probes. A probe is a circle of designing, building and testing of an idea or a prototype. In the figure 2, probes are depicted as multiple circles and may contain branching of ideas and prototypes on a multidisciplinary level or even dead ends. Each circle level corresponds to a role or a discipline in the project. At first, the team takes the best-guess direction based on the initial vision. Through multiple probing and prototype cycles the team then tries to find the big idea worth implementing. This journey can be long or short, but the main point is to learn fast with low-resolution prototypes. Through these prototypes one develops the requirements dynamically as perception of the problem and the vision of the solution will change during the journey. In a nutshell, we increase the degrees of freedom in the early design phase, develop requirements dynamically, and only then switch into classical engineering/project management mode.

While researching radical innovation projects, our chess analogy is lacking because in chess it is theoretically possible to calculate the move with the highest probability of winning the game. However, in the product design “game” the possible future moves, players, even the boundary conditions are often neither comparable nor foreseeable. There are **unknown unknowns** that create opportunities for extremely innovative solutions but also prevent us from predicting or simulating an optimum solution. In this analogy, the rules of the chess game can change without notice and we can only provide a journey overview in hindsight, roadmaps do not apply. The Hunter-Gatherer model by Steinert and Leifer [9] and Ingold [11] inspired this wayfaring concept.

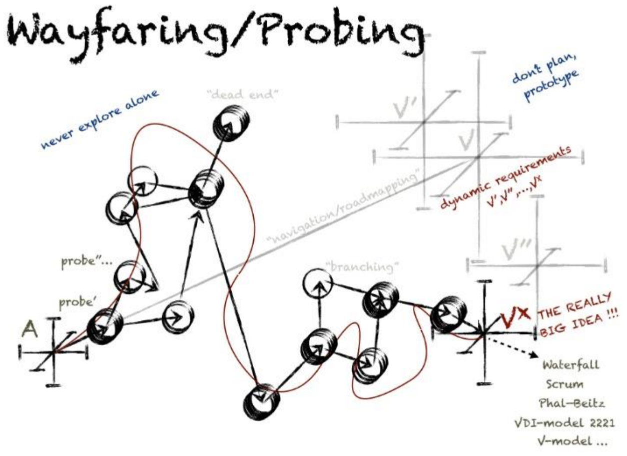


Fig. 1. Wayfaring journey in product development.

Many of our engineering problems are **multidisciplinary** and require interdependent knowledge between disciplines that cannot be covered by individuals or homogeneous teams. Two or more disciplines of the project are interdependent when design changes in one discipline lead to requirement adaptation in at least one other discipline. We argue that including team members or at least domain perspectives from all involved disciplines early in the project helps to reveal desirable and undesirable **interdependencies** already in early decision making phases. Even if actual deliverable input from every member is dispensable early on, the benefit of learning early overcomes the cost of participation. One of the greatest threats in new product development is the fear of failure [12]. According to Snowden [10] safe failing is identified as one of the cornerstones while innovating in the complex domain. The **interlaced knowledge,** developed through sense-making and justification of ideas to the other involved disciplines, is also beneficial when designing within one discipline while having the entire system in mind and thereby knowing when the other disciplines need to be taken into account and their input is needed [13]. This is a skill that can only be learned when combining all involved disciplines from the first day of the project.

The nature of trying out new concepts entails that outcomes cannot be guaranteed and some problems, opportunities and interdependencies are difficult, if not impossible, to foresee. When trying out something never attempted before we can no longer base our assumptions on past experiences and unexpected discoveries can arise. Snowden calls these discoveries **unknown unknowns** because we unknowingly discover something previously unknown [10]. In order to achieve these unexpected discoveries new experiences must be created from **probing ideas**. One of the ideas of probing is therefore to build and test prototypes that create completely new knowledge – knowledge that is impossible to accurately anticipate regardless of what our expectations may be. The concept of probing is depicted in Figure 2. Each probe is a prototype where new knowledge is deductively, inductively and/or abductively created and tested. The vision and requirements are then evolving dynamically until they are locked. The development cycle is executed through different roles of disciplines. Each probe is ideated through divergent thinking where open questions are asked in order to stimulate the creative process followed by convergent thinking, that evaluates and analytically benchmarks the ideas through proof-of-concept prototypes. The interesting interlaced knowledge lies in the boundaries of the different disciplines and presents the potential for serendipity discoveries.

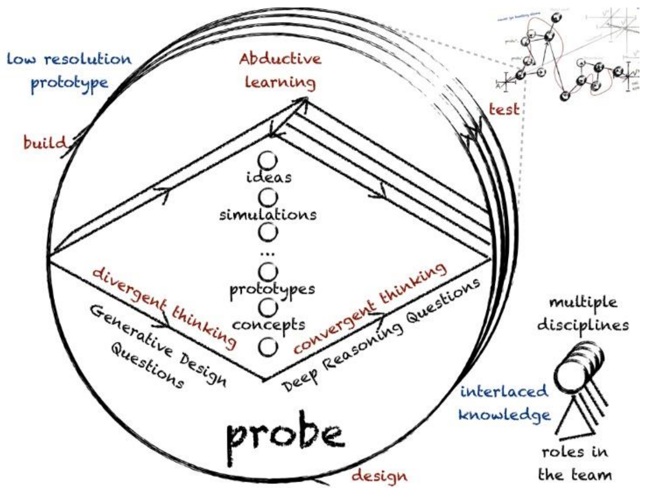


Fig. 2. Probing cycle

To continue with the chess analogy, we do not expect to win if we must plan all our moves (and anticipate the opponent's) in the beginning. However, if allowed to experiment and revert moves a thousand times during the game, it will quickly become a game of probing (or prototyping) multiple moves. Through not following an optimal game strategy, this will eventually lead to overall winning the game in case of a complex game scenario. Because the cost of probing is minimal, it allows us to explore opportunities that are not immediately perceived as profitable. It leads to moves that would normally not be taken, to discoveries that are normally not found, and may potentially lead to surprising and highly innovative ways of winning the game. Therefore, the aim must be to make the probing and the learning of ideas as low-risk (i.e. fast and cheap) as possible in order to create the experience needed to reflect, to understand the outcome, and then **abductively reason** and opportunistically choose the next step [14].

The notion is to put the focus on testing the **most critical functions**, thus leaving the development of the “nice to have” add-ons for later. It is preferable to utilize the resources for discovering the essentials and preferably fail there early. The probing removes uncertainty and an undiscovered problem is revealed before it forces undesired **requirement changes** at a later stage [15]. The testing usually involves building a **low resolution prototype** with the intention to either find the critical function or to build a prototype for user testing in order to avoid developing into an unnecessary direction. Low-resolution prototypes can be anything from cardboard models to Arduino hacks to proof-of-concept prototypes. Often developers have major problems in failing. Low-resolution prototypes in very fast iteration rounds do not resemble the finished object and are thus one way to allow and speed up experimentation. It seems to be inherent to human nature to fear failure, thinking it will cost too much. This can lead to a non-willingness to take risks and make cooperation hard with people from other disciplines. This skill of creative competence [16,12] does not come naturally. This is why changing the mindset into one that favors building prototypes with the option of failing safely before planning is critical while developing new concepts. Hence, despite the natural fear of failing, the mindset should be biased towards building low-resolution prototypes in order to gain experience instead of thinking the idea through and remaining with doubt.

Another finding is to **merge system components** as soon as possible in order to tackle potential integration issues very early on. This follows the same line of thought as aiming to discover unknown interdependencies as early as possible. Whenever a component individually fulfills its critical function, it ought to be integrated with other components to test its critical function in the context of the whole system. So, even when the system can and is divided into modules, integration should be tested while changes to the system are still easily possible. We believe that there is no point in fully developing one component and then risking requirement changes in other components that would endanger the previous development. This requires quasi-**simultaneous prototyping** to ensure that components can be merged. Thus in our context, simultaneous prototyping means understanding and probing ideas from multiple disciplines at the same time.

The **main purpose of probing** is to find solutions to the evolving problem by abductive reasoning and to continuously update the understanding of the problem. While probing different paths for the project one of the most important mindsets is to be opportunistic, to find, recognize and take chances that present themselves. Another benefit is the possibility to abandon disadvantageous concepts, “dead ends”, in an early stage at the lowest cost and involvement possible. All in all, the wayfaring model calls for a bias towards action and learning in action.

5. Conclusions of wayfaring

We propose a method suitable for developing new products with a high degree of uncertainty. It is largely based on including all disciplines related to the product from the beginning on and iterative cycles of probing ideas by designing, building and testing prototypes. The intent of this approach is to discover **unknown unknowns** and unexpected interdependencies early in order to minimizing losses due to failure and to spot opportunities and hitherto unknown potentials. Both, the initial problem statement and the targeted project vision remain in flux much longer than usually. The relatively early requirement fixation stage becomes a delayed dynamic requirement evolution process. The decisions to fix the dynamic requirements are made based on gained and tested information, based on learning cycles trough low-resolution prototyping and probing. We believe the headway and learnings, both in terms of breadth and depths have been superior to pre-planned or more traditional process models. We thus invite the community to deploy and test this approach in the early, pre-requirement definition phase and to share their insights.

Acknowledgments. We thank the project owner L. Nilsen, CEO of Metis Productions AS. This research is supported by the Research Council of Norway (RCN) through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

1. Sanchez, R., & Mahoney, J. T. (1996). Modularity, flexibility, and knowledge management in product and organization design. Strategic Management Journal, 17(S2), 63-76
2. Baldwin, C., & Clark, K. (2000). Design Rules: The power of modularity. MIT Press
3. Reime, T. et al., Entertainment Computing--ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29 - October 2, 2015, Proceedings. Springer, 2015.
4. Eisenhardt, K.M. (1989). Building Theories from Case Study Research. The Academy of Management Review, 14(4), 532 – 550
5. Yin, R.K. (2013). Case Study Research: Designs and Methods. SAGE Publications.
6. Burks, A. W. (1946). Peirce’s theory of abduction. Philosophy of Science, 13(4), 301–306.
7. Eris, O. (2004). Effective inquiry for innovative engineering design. Springer Netherlands.
8. Leifer, L. J., & Steinert, M. (2011). Dancing with ambiguity: Causality behavior, design thinking, and triple-loop-learning. Information, Knowledge, Systems Management, 10(1), 151–173.
9. Steinert, M., & Leifer, L. J. (2012). “Finding One”s Way’: Re-Discovering a Hunter-Gatherer Model based on Wayfaring. International Journal of Engineering Education, 28(2), 251.
10. Snowden, D. J., & Boone, M. E. (2007). A Leader’s Framework for Decision Making. Harvard Business Review, 69 - 76
11. Ingold, T. (2007). Lines: a brief history. Routledge.
12. Bandura, A. (1990). Perceived self-efficacy in the exercise of control over AIDS infection. Evaluation and program planning, 13(1), 9-17.
13. Türtscher, P. et al. (2008). Justification and Interlaced Knowledge at ATLAS, CERN. Organization Science, 25(6), 1579 – 1608
14. Schön, D. A. (1983). The reflective practitioner. Basic books New York.
15. Kriesi, C., Steinert, M., Meboldt, M., & Balters, S. 2014. Physiological Data Acquisition for Deeper Insights into Prototyping. DS 81: Proceedings of NordDesign 2014, Espoo, Finland 27-29th August 2014.
16. Kelley, T., & Kelley, D. (2013). Creative confidence: Unleashing the creative potential within us all. Crown Business.

4 Learnings and reflections

The process undergone during this project was incremental as a consequence of the multiple interdependencies between functions and features from different engineering disciplines. According to Snowden, a complex system has traits that make predetermining outcomes impossible (Snowden & Boone, 2007, p. 3). Consequently, it becomes necessary to probe an environment before taking action. As explained in (Gerstenberg et al., 2015), a wayfaring approach was used to probe in an open solution space during the first six weeks; this being a complex system, only specifications that were vague and dynamic could be established in the beginning to allow for the flexible modularity. The learnings thereof were process related and qualitative, where becoming aware of traits, tools, and mindset was perhaps the most personally valuable learning.

4.1 From complex to complicated

After submitting the papers, i.e. reaching the second milestone, the process changed by not being entirely focused on probing new solutions. Instead, focus was on improving the existing system. The main reason for this change of process was that the third milestone required a prototype ready for demonstration, not more probing of concepts. A complete critical function prototype was now in place, and it came with concrete technology constraints (resistors, LightBlue Beans, Processing, etc.). But to make it ready for a live demonstration, improvements on a full system level were needed. This gave the challenge more traits similar to that of a complicated system rather than a complex one. E.g. the development has known unknowns, and relations can be predicted. In Shar's description of the wayfaring model, this seems to correlate to the 'transportation' phase that comes after the 'kill': "transporting is more of a convergent, cognitive skill requiring a sense of geometry; the shortest distance between two goals, where the parameters are known and efficiency is the goal" (Schar, 2011, p. 51). This describes the nature of the third milestone better than the unchartered nature of the wayfaring phase. Note: some probing into other concepts was still conducted during this phase, e.g. using I2C (section 2.5) and new connectors with free orientation (section 2.3). I found the process change was surprisingly educational as well: it forced development of final solutions rather than solutions on which new work had to be done. Rather than remaining on the conceptual or visionary level of the first prototype, it was brought down to earth through a more holistic solution. If it had not been for the third milestone, the code for stabilizing the system may not have been created (section 2.5). This could in turn have made me unaware of challenges that must be tackled when the product specifications are defined. Thusly, a more convergent phase proved to be healthy for the sake of the functionality of the low resolution prototype. In other words: the first prototype confirmed the feasibility of certain solutions and made educated assumptions regarding what the future prototype could be. The third milestone was about converging on a more holistic solution from predefined specifications, i.e. bringing it home. My personal take on this is that the 'transportation' phase may have the ability to uncover unknown unknowns on a deeper level than the divergent phase; it forces the concrete application of solutions found in the divergent phase, and unwanted interdependencies not discovered in our probing cycles will then come to light.

4.2 Input from all domains was not included

After the initial project task had been given and preliminary goals were set, work on the first prototype commenced. During this project, never was there such a close call as when it was discovered that communication between all parts was not required in terms of IP. As described in (Reime et al., 2015), the initial principles regarding identification was based on all parts intercommunicating. This lesson was commented on in (Gerstenberg et al., 2015) as a finding that underlined the importance of including input from all domains from the beginning of the project. Had this simple piece of information not been declared, the following decision making would be based on a totally different knowledge base, and the outcome of this project could have been radically different.

While the project owner supplied us with this the critical information regarding IP, there is an absolute absence of input from gameplay developers and interface design for children. These knowledge domains are capable of making an entire concept void because they are, more than most domains in this thesis, directly tied to the users' needs. Knowingly excluding them can only be justified by clearly stating that the initial goal of the project was not to make a solution ready for user testing, but to make a prototype able to convey the concept to investors. Only later, after significant modification, could the prototype potentially become ready for user testing. This allowed me to greatly simplify the design of the prototype. In the beginning, merging with gameplay was thought to become possible late in the project. However, the 2nd and 3rd milestone did not allow for time to include gameplay or more user friendly designs, leading to finalizing the prototype meant for investors' eyes only.

4.3 Take measures to protect IP

When making a prototype with the purpose of showing off a concept, it may be an advantage to use low resolution solutions for the simple sake of protecting IP. If a prototype containing patentable technology was used for demonstration, investors may actually take this as a sign of higher risk. The same goes for publications that may contain detailed description of the prototype. To avoid this in the future, any prototypes used for demonstration or in published material should either a) not include sensitive technology, b) make the technology invisible to the viewer, or c) demand a declaration of confidentiality.

4.4 Detours yielded important results

Throughout the 14 weeks of development, minor detours were taken to explore other options. E.g. I2C for inter component communication, presenting structure by loading premade pictures instead of real time loading of 3D models, and using ring magnets to get a universal self-aligning connector. These detours opened up a larger library of possibilities, some of which can have a real impact on the end product. I found that the detours were not only useful for exploring other technologies in themselves, but also to get a more distanced view of the prototype. Detouring appears to be a part of the wayfaring model where constant search for potential “gold nuggets” can yield solutions that radically improve the original design.

4.5 Use ‘big pictures’ and assumptions when applicable

The process described in the (Gerstenberg et al., 2015) aims to identify interdependencies between domains. This enabled a ‘big picture’ to emerge; a rough, mental sketch of how all major components would interact. Built into this ‘big picture’ was a multitude of assumptions regarding minor facets of each component: it did not include detailed solutions to every aspect, but rather assume that the simplest solution would work or that another would be found with ease. These assumptions were not taken lightly and only when deemed simple enough to not require prototyping or unlikely to include unknown unknowns. An example of this is the first prototype where the ‘big picture’ in was ‘Arduino measures resistors and sends data to Processing’. In this case, Arduino and Processing were the big, critical components. Previous experience with these allowed me to assumed that the sending to and analysis of data in Processing would either be done by direct reading of the string, or by some simple data conversion. The measured resistances are treated as values by Arduino. However, it turned out that Processing reads the data from Arduino as individual characters in a string, and so the data needed to be converted before Processing would interpret the data as actual numbers. Two lines of conversion code were added to the Processing script (see code below), and the system worked as envisioned in the 'big picture'. This assumption would not have been made if I did not feel secure that a solution would be found.

```
float floatVal = Float.parseFloat(SplitMyInts[i]);
intVal = int(floatVal);
```

Another example that did not fare as well was the assumption that data could be sent from a Bean to any computer with a Bluetooth. The assumption was wrong because of OS protocols that I assumed were irrelevant, and it led to the need for a computer with OS X. Nevertheless, I believe such assumptions save considerable amounts of time because they allow for the more critical parts of the system to be prototyped first. Assuming that Processing could read the data sent from a Bean was another assumption that proved correct, but it was only prototyped after it was confirmed that the Bean could run the electrical components. Consequently, another major learning was to always prototype the most important functions first.

4.6 Favor a 'moldable' concept

Use of resistors led to it being possible to highly modify and mold the prototype. Because the microcontroller took simple measurements, it was easy to manipulate those measurements and create solutions that fit, e.g. the thresholds for resistor values. Additionally, it made it possible to make adjustments for future solutions: when improving the code before the third milestone, the use of numbers made it possible to perform mathematical operations on the raw data. E.g., if I had chosen to use letters instead of numbers, then the system would rely completely on perfect signal transmission, and signal manipulation would be impossible. I2C would perhaps be even more moldable because the communication is easily redefined. The lesson is that the solution that opens for multiple options should be favored over one that has only specific uses.

4.7 Interdependencies increase exponentially with addition of domains

Adding the needs just one more domain will create consequences for all the original domains. As explained in 2.6, this is especially clear the case of user interface, which would not necessarily have been a part of the task had this been a "standard" engineering problem. The inclusion of this domain led to a multitude of changes in the other engineering disciplines. This is also where the exclusion of the gameplay domain potentially had its advantage: to establish a way of sending data into a game could potentially have had massive repercussions all the way down to how the resistors are measured. Naturally, including this interface prototype with a game engine would likely require the game itself to be radically changed also, showing the two-way challenge of including knowledge domains.

4.8 Prototype as many things simultaneously as possible

This was one of the key topics of (Gerstenberg et al., 2015) and it is duly added as a major finding. It proved useful to prototype as many things simultaneously as possible when applicable and if effects were discernable. The probing cycle described how interlaced knowledge appears when multiple domains regard one aspect simultaneously, thus allowing both a multitude of solutions and potential pitfalls to be discovered. The perhaps best example of simultaneous prototyping is the prototype described in week four in (Reime et al., 2015): an entire system of modules was integrated, but the functionalities in each module had low resolution. Nevertheless, it not only proved that the principle technologies in each module worked, but it also, and more importantly, proved that they worked in total unison, thus establishing interfaces between modules of a complete system. The magnetic connector prototype in figure 1-c and -d in (Gerstenberg et al., 2015) is an example of simultaneous prototyping that found a pitfall: the header connectors were the original focus of the probe, but I decided to add prototyping of magnets as well. This led to an interesting discovery regarding symmetry of magnetic connectors, and it could potentially have been quite damaging if the two components had not been simultaneously prototyped early in the process.

4.9 Favor experimentation – tacit vs. explicit knowledge

Most of the knowledge about the use and functions of the technologies in the final prototype was acquired through experimentation; doing rather than reading about doing. This was especially useful in the earliest part of the project when I experimented with resistors and programming. Naturally, some background research was eventually needed when Arduino, Processing, and LightBlue Beans were used. However, I found that searching online for specific terms often gave answers that required me to search for more terms to get a complete understanding. In fact, I found myself searching for “can it be done” more often than “how is it done”. This often led to discarding options when assuming them to be too demanding. An alternative noteworthy approach was the use of examples in the libraries of the mentioned programs: opening a script and working with the code yielded fast learning about e.g. loading of 3D models in Processing. In the end, I strongly felt that this approach was effective for fast learning of tacit know-how. The great risk is that great opportunities were missed because I decided to not spend much time reading. As such, I found that this ‘double-edged sword’ certainly triumphed in this prototyping challenge, but it seems clear that its use should be strictly limited. I propose that this “skimming the surface” approach is only suited when a) the problem appears to not require very deep knowledge in its field but b) extracting the tacit knowledge from the explicit is still difficult and c) time is an important factor. In its essence, this suggests favoring experimentation of principles. Examples of this can be found throughout the project: developing the resistor principle; learning to program Arduino, shift registers, and Processing; communication between Arduino and Processing; instead of spray painting, spray into a cup and use a brush.

5 Personal remarks

This master project was centered on developing a radically new product where the starting point only contained a vision and loosely defined functions. As such, I believe it was a good challenge for an engineer who aims to specialize in early stage product development. I found the openness liberating compared to previous projects with set parameters, timeline, deliverables, and milestones. However, I was only working on my own; mentors, project owner, and other students only contributed with input. Therefore, there was never any team dynamics or discussions about options. This poses some uncomfortable questions regarding whether the end results reached its maximum potential or not: did working alone contribute by having less time consuming team dynamics, or did it counteract by not having as much knowledge and ideation power as possible? Naturally, the answer to both is yes. But when comparing with previous projects, I believe that if there had been one or two more team members, at least one more milestone towards achieving patentable technology could have been reached.

Regarding learning outcomes, I found that this way of constantly searching, experimenting, and adapting allowed me to quickly learn what I needed. The problem with this self-governing is that I might have stopped short because I decided for myself what was important to spend time doing. As such, it was a healthy exercise in completing a race where I had to motivate myself to get over some chores and obstacles.

Retrospectively, I find it hard to evaluate my own performance when there is nothing to compare with. But I feel satisfied with the positive response given by project owner, mentors, and other students. I look forward to applying the culminated experience of my five year education and this master project in a real world setting.

References

Gerstenberg, A., Sjöman, H., Reime, T., Abrahamsson, P., & Steinert, M. (2015). A Simultaneous, Multidisciplinary Development and Design Journey - Reflections on Prototyping. Presented at the 2015 14th International Conference of Entertainment Computing (ICEC), Trondheim, Norway.

Reime, T., Sjöman, H., Gerstenberg, A., Abrahamsson, P., & Steinert, M. (2015). Bridging Tangible and Virtual Interaction: Rapid Prototyping of a Gaming Idea. Presented at the 2015 14th International Conference of Entertainment Computing (ICEC), Trondheim, Norway.

Sanchez, R., & Mahoney, J. T. (1996). Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal*, *17*(SUPPL. WINTER), 63–76.

Sapounidis, T., Demetriadis, S., & Stamelos, I. (2014). *Evaluating children performance with graphical and tangible robot programming tools*.

Schar, M. F. (2011). *Pivot Thinking and the Differential Sharing of Information Within New Product Development Teams*. Stanford University.

Smith, A. C. (2014). Rock Garden programming: Programming in the physical world. In *2014 Fourth International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)* (pp. 430–434). http://doi.org/10.1109/DICTAP.2014.6821725

Snowden, D. J., & Boone, M. E. (2007). A leader's framework for decision making. A leader's framework for decision making. *Harvard Business Review*, *85*(11), 68–76, 149.

Steinert, M., & Leifer, L. J. (2012). “Finding one”s way’: Re-discovering a hunter-gatherer model based on wayfaring. *International Journal of Engineering Education*, *28*(2), 251–252.

Strawhacker, A., & Bers, M. U. (2014). *“I want my robot to look for food”: Comparing Kindergartner’s programming comprehension using tangible, graphic, and hybrid user interfaces*.

Wang, D., Wang, T., & Liu, Z. (2014). A Tangible Programming Tool for Children to Cultivate Computational Thinking. *The Scientific World Journal*, *2014*, e428080. http://doi.org/10.1155/2014/428080

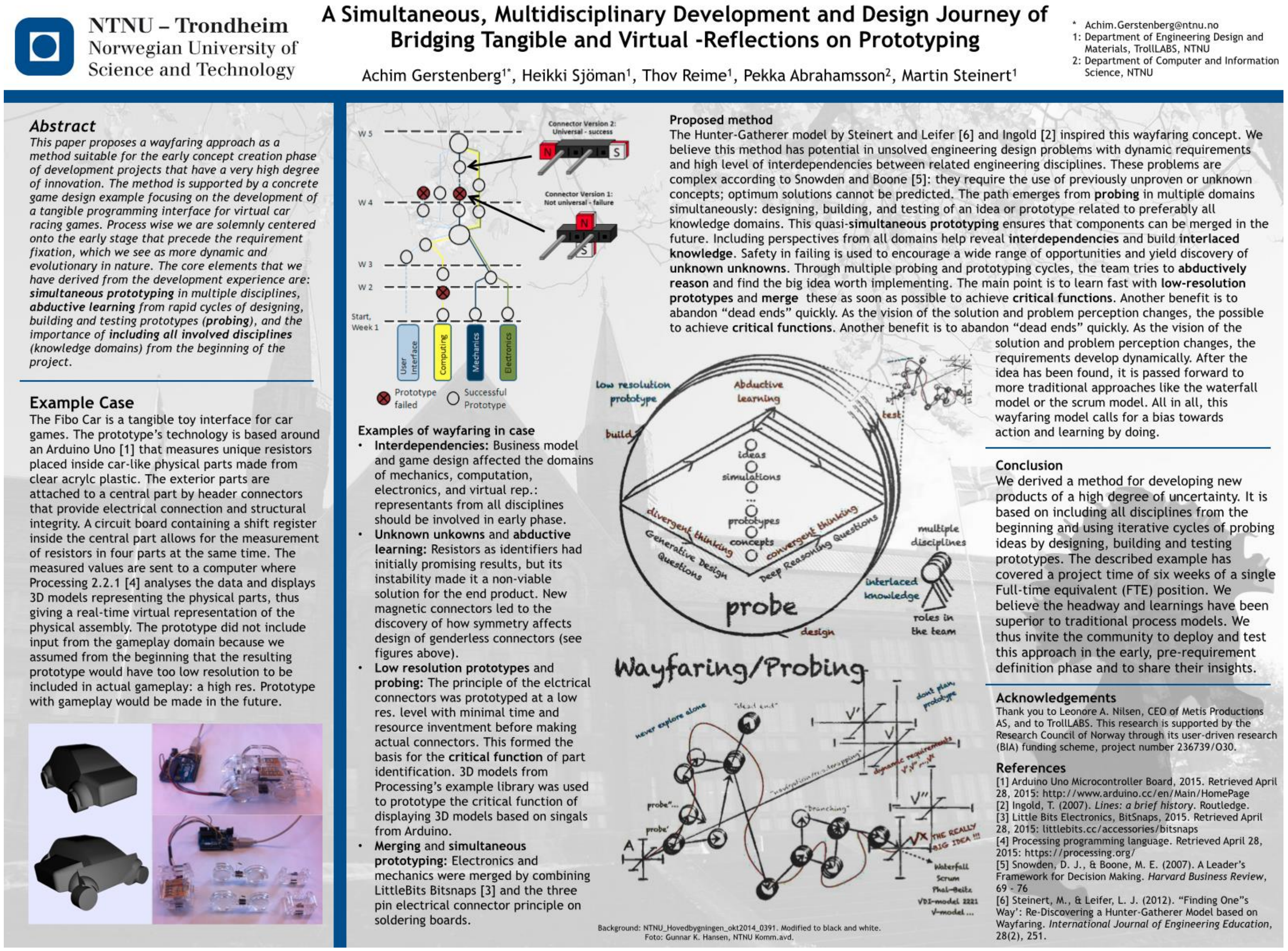
Wang, D., Zhang, C., & Wang, H. (2011). T-Maze: A tangible programming tool for children (pp. 127–135). Presented at the Proceedings of IDC 2011 - 10th International Conference on Interaction Design and Children. http://doi.org/10.1145/1999030.1999045

Appendix A: Uploaded Material

Various digital material have been uploaded to this Google Drive folder for easy access to videos, 3D models, sketches, code scripts, etc. Due to the size of the 3D models and various material, the size of the compressed folder exceeded the size available for uploading on DAIM. Therefore, a Google Drive folder is provided containing all material in one location.

https://drive.google.com/folderview?id=0B50n3O86Q7UGfnEyandpdDR3QUpaTFk1ZHVw
YmZVU2VqbmVFVmtPaW1MZDFVNC1YS3FTalE&usp=sharing

Appendix B: Poster



NTNU – Trondheim
Norwegian University of Science and Technology

A Simultaneous, Multidisciplinary Development and Design Journey of Bridging Tangible and Virtual -Reflections on Prototyping

Achim Gerstenberg[1*], Heikki Sjöman[1], Thov Reime[1], Pekka Abrahamsson[2], Martin Steinert[1]

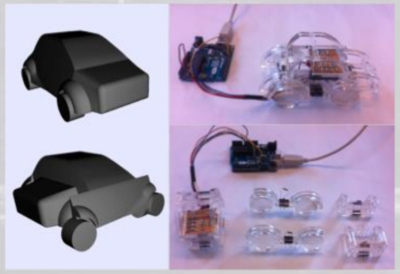
* Achim.Gerstenberg@ntnu.no
1: Department of Engineering Design and Materials, TrollLABS, NTNU
2: Department of Computer and Information Science, NTNU

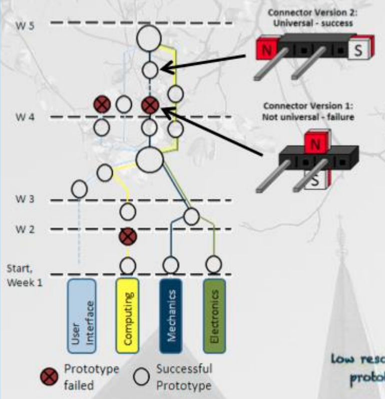
Abstract

This paper proposes a wayfaring approach as a method suitable for the early concept creation phase of development projects that have a very high degree of innovation. The method is supported by a concrete game design example focusing on the development of a tangible programming interface for virtual car racing games. Process wise we are solemnly centered onto the early stage that precede the requirement fixation, which we see as more dynamic and evolutionary in nature. The core elements that we have derived from the development experience are: ***simultaneous prototyping*** *in multiple disciplines,* ***abductive learning*** *from rapid cycles of designing, building and testing prototypes (probing), and the importance of* ***including all involved disciplines*** *(knowledge domains) from the beginning of the project.*

Example Case

The Fibo Car is a tangible toy interface for car games. The prototype's technology is based around an Arduino Uno [1] that measures unique resistors placed inside car-like physical parts made from clear acrylc plastic. The exterior parts are attached to a central part by header connectors that provide electrical connection and structural integrity. A circuit board containing a shift register inside the central part allows for the measurement of resistors in four parts at the same time. The measured values are sent to a computer where Processing 2.2.1 [4] analyses the data and displays 3D models representing the physical parts, thus giving a real-time virtual representation of the physical assembly. The prototype did not include input from the gameplay domain because we assumed from the beginning that the resulting prototype would have too low resolution to be included in actual gameplay: a high res. Prototype with gameplay would be made in the future.



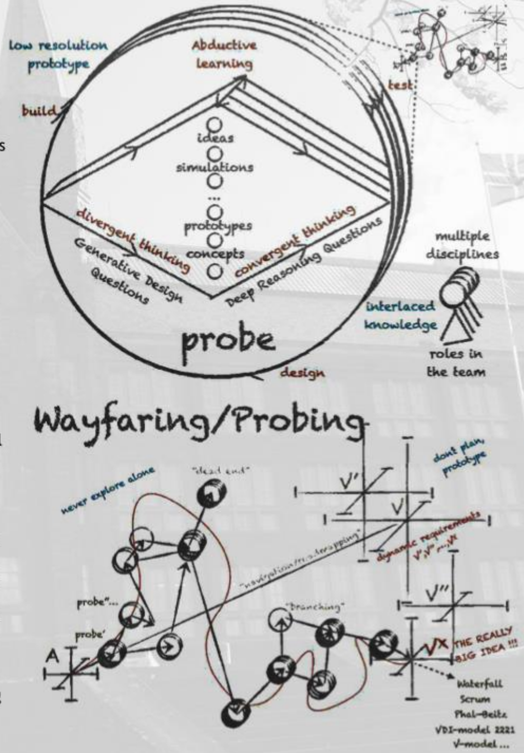


Examples of wayfaring in case

- **Interdependencies:** Business model and game design affected the domains of mechanics, computation, electronics, and virtual rep.: representants from all disciplines should be involved in early phase.
- **Unknown unkowns** and **abductive learning:** Resistors as identifiers had initially promising results, but its instability made it a non-viable solution for the end product. New magnetic connectors led to the discovery of how symmetry affects design of genderless connectors (see figures above).
- **Low resolution prototypes** and **probing:** The principle of the elctrical connectors was prototyped at a low res. level with minimal time and resource inventment before making actual connectors. This formed the basis for the **critical function** of part identification. 3D models from Processing's example library was used to prototype the critical function of displaying 3D models based on singals from Arduino.
- **Merging** and **simultaneous prototyping:** Electronics and mechanics were merged by combining LittleBits Bitsnaps [3] and the three pin electrical connector principle on soldering boards.

Proposed method

The Hunter-Gatherer model by Steinert and Leifer [6] and Ingold [2] inspired this wayfaring concept. We believe this method has potential in unsolved engineering design problems with dynamic requirements and high level of interdependencies between related engineering disciplines. These problems are complex according to Snowden and Boone [5]: they require the use of previously unproven or unknown concepts; optimum solutions cannot be predicted. The path emerges from **probing** in multiple domains simultaneously: designing, building, and testing of an idea or prototype related to preferably all knowledge domains. This quasi-**simultaneous prototyping** ensures that components can be merged in the future. Including perspectives from all domains help reveal **interdependencies** and build **interlaced knowledge**. Safety in failing is used to encourage a wide range of opportunities and yield discovery of **unknown unknowns**. Through multiple probing and prototyping cycles, the team tries to **abductively reason** and find the big idea worth implementing. The main point is to learn fast with **low-resolution prototypes** and **merge** these as soon as possible to achieve **critical functions**. Another benefit is to abandon "dead ends" quickly. As the vision of the solution and problem perception changes, the possible to achieve **critical functions**. Another benefit is to abandon "dead ends" quickly. As the vision of the solution and problem perception changes, the requirements develop dynamically. After the idea has been found, it is passed forward to more traditional approaches like the waterfall model or the scrum model. All in all, this wayfaring model calls for a bias towards action and learning by doing.



Background: NTNU_Hovedbygningen_okt2014_0391. Modified to black and white.
Foto: Gunnar K. Hansen, NTNU Komm.avd.

Conclusion

We derived a method for developing new products of a high degree of uncertainty. It is based on including all disciplines from the beginning and using iterative cycles of probing ideas by designing, building and testing prototypes. The described example has covered a project time of six weeks of a single Full-time equivalent (FTE) position. We believe the headway and learnings have been superior to traditional process models. We thus invite the community to deploy and test this approach in the early, pre-requirement definition phase and to share their insights.

Acknowledgements

Thank you to Leonore A. Nilsen, CEO of Metis Productions AS, and to TrollLABS. This research is supported by the Research Council of Norway through its user-driven research (BIA) funding scheme, project number 236739/O30.

References

[1] Arduino Uno Microcontroller Board, 2015. Retrieved April 28, 2015: http://www.arduino.cc/en/Main/HomePage
[2] Ingold, T. (2007). *Lines: a brief history*. Routledge.
[3] Little Bits Electronics, BitSnaps, 2015. Retrieved April 28, 2015: littlebits.cc/accessories/bitsnaps
[4] Processing programming language. Retrieved April 28, 2015: https://processing.org/
[5] Snowden, D. J., & Boone, M. E. (2007). A Leader's Framework for Decision Making. *Harvard Business Review*, 69 - 76
[6] Steinert, M., & Leifer, L. J. (2012). "Finding One"s Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. *International Journal of Engineering Education*, 28(2), 251.

Appendix C: Processing code

```
import processing.serial.*;
Serial StrFromArd;
String myInts;
float floatVal;
int intVal;
String[] SplitMyInts;
int[] Base = new int[4];
int i;

PShape Cupe;
PShape Engine;
PShape WheelsLeft;
PShape WheelsLeftMon;
PShape WheelsRight;
PShape WheelsRightMon;
PShape FuelTank;
PShape F1Front;
PShape F1Rear;
PShape F1Right;
PShape F1Left;


float ry;

public void setup() {
  size(1500, 1000, P3D);
  background(0);
  lights();

  //String portName = Serial.list()[2];
  StrFromArd = new Serial (this, "/dev/cu.LightBlue-Bean", 57600);
  StrFromArd.clear();
  StrFromArd.bufferUntil('\n');

  Cupe = loadShape("Cupe.obj");
  Engine = loadShape("Engine.obj");
  WheelsLeft = loadShape("WheelsLeft.obj");
  WheelsLeftMon = loadShape("WheelsLeftMon.obj");
  WheelsRight = loadShape("WheelsRight.obj");
  WheelsRightMon = loadShape("WheelsRightMon.obj");
  FuelTank = loadShape("FuelTank.obj");
  F1Front = loadShape("F1Front.obj");
  F1Rear = loadShape("F1Rear.obj");
  F1Left = loadShape("F1WheelsLeft.obj");
  F1Right = loadShape("F1WheelsRight.obj");

  i = 0;
}


public void draw() {
  background(230,230,255);
  pointLight(255, 255, 255, width/2, height/2, 200);
  translate(width/2, height-500, -400);
  scale(-0.5,0.5,0.5);
  rotateX(0.35*PI);
  rotateZ(ry);
  ry += 0.015;
```

```
    pushMatrix();
    scale(120);
    shape(Cupe);
    popMatrix();


  while ( StrFromArd.available() > 0) {
    myInts = StrFromArd.readStringUntil('\n');

    if (myInts != null) {
      //Splits myInts into array of strings
      String[] SplitMyInts = split(myInts, ";");

      for (int i = 0; i <= 3; i++) {
        //Converts the array of strings into array of ints
        float floatVal = Float.parseFloat(SplitMyInts[i]);
        intVal = int(floatVal);
        Base[i] = intVal;
      } //for i
    } //if
    println(Base);
  } //while

  // CONVERSIONS
  for (i = 0; i <= 3; i++) {

  // If Base[i] oposite to FuelTank is 247 or 329, change it to 170.
  //Also, if Base[i] oposite to FuelTank is 164, change it to 145.
  if (Base[i] == 2) {
           if (i == 0) {
              if (Base[2] == 247 || Base[2] == 329) { Base[2] = 170; }
              if (Base[2] == 164 || Base[2] == 197) { Base[2] = 145; }
           }
      else if ( i == 1) {
              if (Base[3] == 247 || Base[3] == 329) { Base[3] = 170; }
              if (Base[3] == 164 || Base[3] == 197) { Base[3] = 145; }
           }
      else if ( i == 2) {
              if (Base[0] == 247 || Base[0] == 329) { Base[0] = 170; }
              if (Base[0] == 164 || Base[0] == 197) { Base[0] = 145; }
           }
      else if ( i == 3) {
              if (Base[1] == 247 || Base[1] == 329) { Base[1] = 170; }
              if (Base[1] == 164 || Base[1] == 197) { Base[1] = 145; }
           }
  }

  // If fuel tank is connected and a neighbor is F1 Right,
  // make the other neighbor automatically F1 Left.
      if (Base[0] == 2) {
             if (Base[1] == 495) { Base[3] = 991; }
        else if (Base[3] == 495) { Base[1] = 991; }
      }
      if (Base[1] == 2) {
             if (Base[2] == 495) { Base[0] = 991; }
        else if (Base[0] == 495) { Base[2] = 991; }
      }
      if (Base[2] == 2) {
             if (Base[3] == 495) { Base[1] = 991; }
        else if (Base[1] == 495) { Base[3] = 991; }
      }
      if (Base[3] == 2) {
             if (Base[0] == 495) { Base[1] = 991; }
        else if (Base[1] == 495) { Base[0] = 991; }
      }

  // If the left wheel is connected and a neighbour is F1 wheel,
  // make the F1 Wheel into F1 Rear.
      if (Base[0] >= 22 && Base[0] <= 55) {
        if      (Base[1] == 247) { Base[1] = 164; }
        else if (Base[3] == 247) { Base[3] = 164; }
      }
      if (Base[1] >= 22 && Base[1] <= 55) {
        if      (Base[0] == 247) { Base[0] = 164; }
        else if (Base[2] == 247) { Base[2] = 164; }
      }
      if (Base[2] >= 22 && Base[2] <= 55) {
        if      (Base[1] == 247) { Base[1] = 164; }
        else if (Base[3] == 247) { Base[3] = 164; }
      }
      if (Base[3] >= 22 && Base[3] <= 55) {
        if      (Base[0] == 247) { Base[0] = 164; }
        else if (Base[2] == 247) { Base[2] = 164; }
      }

  // If F1 Front and Rear have a S Wheel between, add 5 to get monster.
  if (Base[0] >= 100 && Base[0] <= 200 || Base[2] >55 && Base[2] <= 200) {
           if (Base[1] >= 32 && Base[1] <= 35 ) {
             Base[1] = 50; }
      else if (Base[3] >= 22 && Base[3] <= 55 ) {
             Base[3] = 50; }
  }
  if (Base[1] >= 100 && Base[1] <= 200 || Base[3] >55 && Base[3] <= 200) {
           if (Base[1] >= 32 && Base[1] <= 35 ) {
             Base[1] = 50; }
      else if (Base[3] >= 32 && Base[3] <= 35 ) {
             Base[3] = 50; }
  }


  } // for i

  //PLACING OBJECTS
  for (i = 0; i <= 3; i++) {
    pushMatrix();
    scale(120);

    // Orients to correct position.
    rotateZ(i*0.5*PI);
    if (i == 0 || i == 2) { translate(0,5,0); }
    else                  { translate(0,3,0); }


  // THRESHOLDS
    if (Base[i] == 0) {}
    else if (Base[i] < 4) { shape(FuelTank); }
    else if (Base[i] < 10) { shape(WheelsRight); }
    else if (Base[i] < 22) { shape(WheelsRightMon); }
    else if (Base[i] < 38) { shape(WheelsLeft); }
    else if (Base[i] < 55) { shape(WheelsLeftMon); }
    else if (Base[i] < 100) { shape(Engine); }
    else if (Base[i] < 150) { shape(F1Front); }
    else if (Base[i] < 200) { shape(F1Rear); }
    else if (Base[i] < 500) { shape(F1Right); }
    else if (Base[i] > 500) { shape(F1Left); }

  /*
    if (i == 2) { shape(FuelTank); }

  if (ry < 1.2*PI && ry > 0) {
    if (i == 0) { shape(Engine); }
    if (i == 1) { shape(WheelsLeft); }
    if (i == 3) { shape(WheelsRight); }
  }
  if (ry > 1.2*PI && ry < 1.7*PI ) {
    if (i == 0) { shape(Engine); }
    if (i == 1) { shape(WheelsLeftMon); }
    if (i == 3) { shape(WheelsRight); }
  }
  if (ry > 1.7*PI && ry < 2.1*PI) {
    if (i == 0) { shape(EngineBumper); }
    if (i == 1) { shape(WheelsLeftMon); }
    if (i == 3) { shape(WheelsRight); }
  }
  if (ry > 2.1*PI) {
    if (i == 0) { shape(EngineBumper); }
    if (i == 1) { shape(WheelsLeftMon); }
    if (i == 3) { shape(WheelsRightMon); }
  }
  if (ry > 4*PI) { ry = 0; }
  */

    popMatrix();
  //Done placing objects

  } //for i

  }  // draw
```

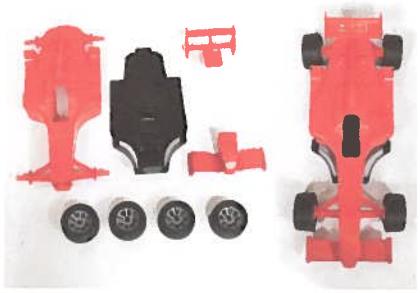
Appendix D: Arduino Code

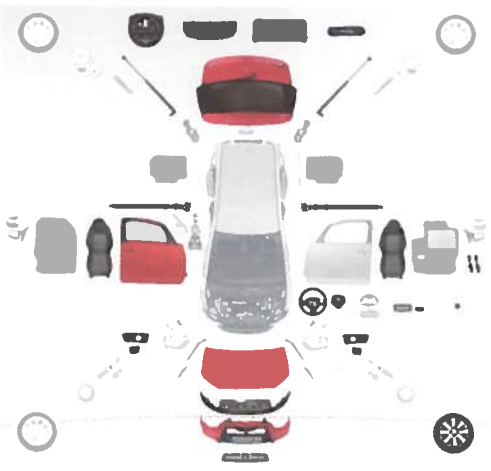
```
int DS_pin = 1;
int STCP_pin = 2;
int SHCP_pin = 3;
int analogPin = 0;

float sensorValue = 0;
int myInts[4];
int oldInts1[4];
int oldInts2[4];
int newInts[4];
float Vin;
float Vout = 0;
float Rref = 970;
float R;

void setup()
{
Vin = Bean.getBatteryVoltage();
Serial.begin(57600);
pinMode(DS_pin,OUTPUT);
pinMode(STCP_pin,OUTPUT);
pinMode(SHCP_pin,OUTPUT);
pinMode(analogPin, INPUT);
writereg();
}

boolean registers[4];

void writereg()
{
digitalWrite(STCP_pin, LOW);
for (int i = 3; i>=0; i--)
{
digitalWrite(SHCP_pin, LOW);
digitalWrite(DS_pin, registers[i] );
digitalWrite(SHCP_pin, HIGH);
}
digitalWrite(STCP_pin, HIGH);
}


void loop()
{
    for (int i = 0; i<=3; i++)
    {
    delay(200);
    registers[i] = LOW;
    writereg();
    sensorValue = analogRead(analogPin);
    registers[i] = HIGH;
    writereg();

    Vout = (Vin * sensorValue) / 1023;     // Convert Vout to volts
    R = Rref * (0.001 / ((Vin / Vout) - 1));   // Formula to calculate
tested resistor's value

    oldInts1[i] = oldInts2[i];
    oldInts2[i] = newInts[i];
    newInts[i] = R;
    if (oldInts1[i] ==  oldInts2[i]) {
      if (oldInts2[i] == newInts[i]) {
         myInts[i] = newInts[i];
      }
    }

    } //for i

Serial.print(myInts[0]); Serial.print(";");
Serial.print(myInts[1]); Serial.print(";");
Serial.print(myInts[2]); Serial.print(";");
Serial.println(myInts[3]);

/*
Serial.print(myInts[0]); Serial.print(";");
Serial.print(myInts[1]); Serial.print(";");
Serial.print(myInts[2]); Serial.print(";");
Serial.println(myInts[3]); //Serial.println(Vin);
*/

/*
for (int i = 0; i <=3; i++) {
  if (myInts[i] == 0 ) { Serial.print("Nothing"); }
  else if (myInts[i] < 4) { Serial.print("S Fuel Tank"); }
  else if (myInts[i] < 10) { Serial.print("S WL small"); }
  else if (myInts[i] < 22) { Serial.print("S WL mon"); }
  else if (myInts[i] < 38) { Serial.print("S WR mon"); }
  else if (myInts[i] < 55) { Serial.print("S WR small"); }
  else if (myInts[i] < 100) { Serial.print("S Engine"); }
  else if (myInts[i] < 150) { Serial.print("F1 Engine"); }
  else if (myInts[i] < 200) { Serial.print("F1 Rear"); }
  else if (myInts[i] < 500) { Serial.print("F1 R"); }
  else if (myInts[i] > 500) { Serial.print("F1 L"); }

  if (i == 3) { Serial.println(); }
  else if (i != 3) {Serial.print(" ; "); }
}
*/

/*
Serial.print("          ");
  for (int i = 0; i<=3; i++)
  {
   if (myInts[i] == 0)       {Serial.print("Nothing "); }
   else if (myInts[i] < 7)   {Serial.print("Wheel "); }
   else if (myInts[i] < 12)  {Serial.print("Transmission "); }
   else if (myInts[i] < 25)  {Serial.print("Gear-box ");}
   else if (myInts[i] >= 25) {Serial.print("Engine ");}

   if (i == 0) {Serial.println(); Serial.println("            |");}
   if (i == 1) {Serial.print(" - Gear-box - ");}
   if (i == 2) {Serial.println(); Serial.println("            |");
                Serial.print("          ");}
//   if (i != 3) {Serial.print("| "); }

  }
  Serial.println(); Serial.println();
*/


}
```

Appendix E: Initial Product Vision from Project Owner

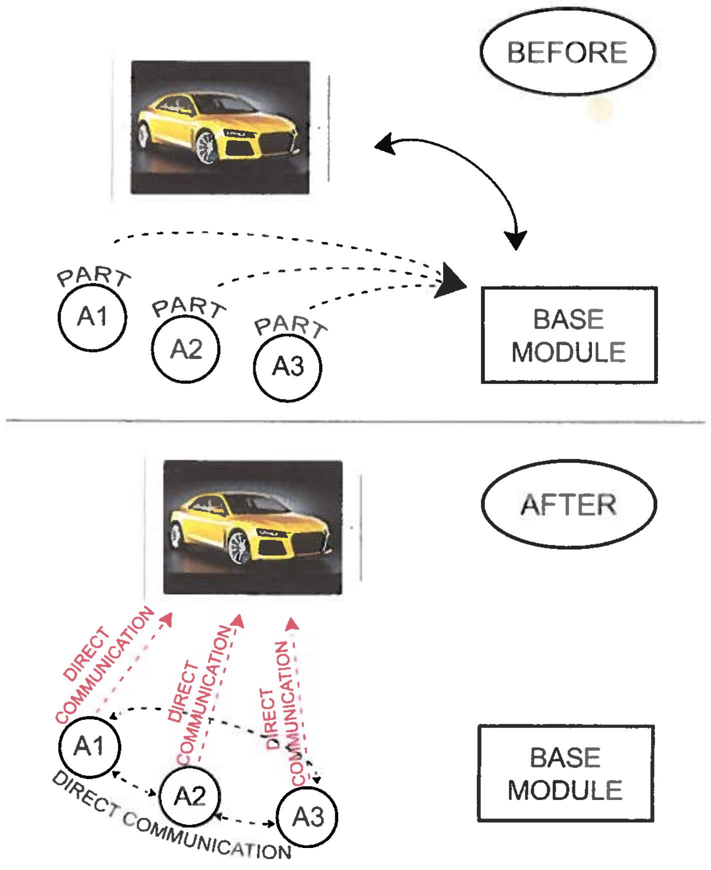
The following pages are scanned from the originals given by the project owner during the first meeting with project participants.

BILDELENE KOMMER I EN PAKKE SOM BARNA KAN SETTE SAMMEN

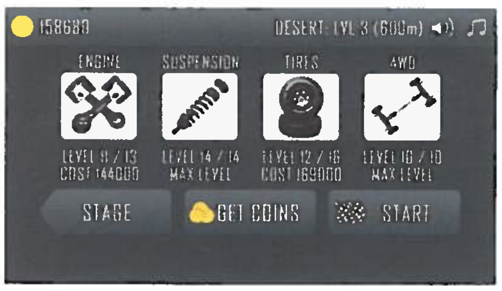




HVER DEL INNEHOLDER EN CHIP SOM KOMMUNISERER MED HVERANDRE UTEN AT ALLE DELER MÅ KOMMUNISERE MED EN BESTEMT DEL, FOR EKSEMPEL CHASSIS. PATENTSTYRET HAR SVART POSITIVT PÅ PATENTERBARHET.

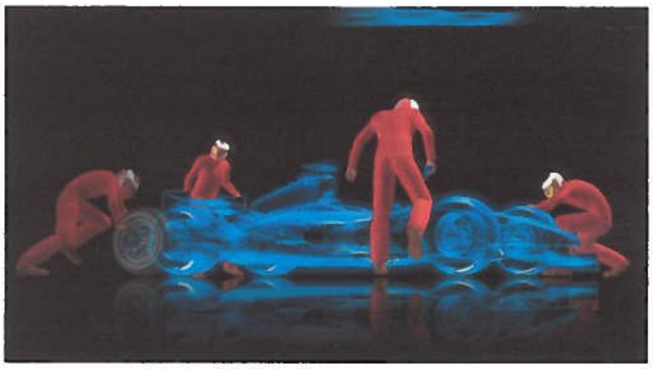


I APP'EN DESIGNES BILEN VIDERE OG DERETTER KJØRES DEN VIRTUELLE BILEN GJENNOM EN VINDTUNNEL HVOR MAN TESTER AERODYNAMIKK FORSTÅELSEN. HAR MAN IKKE FORSTÅTT AERODYNAMIKKEN, VIL IKKE BILEN VÆRE KJØRBAR OG MÅ TESTES PÅ NYTT TIL MAN FÅR EN BIL SOM FUNGERER BRA. HER LÆRES LIFT, WEIGHT, TRUST OG DRAG.

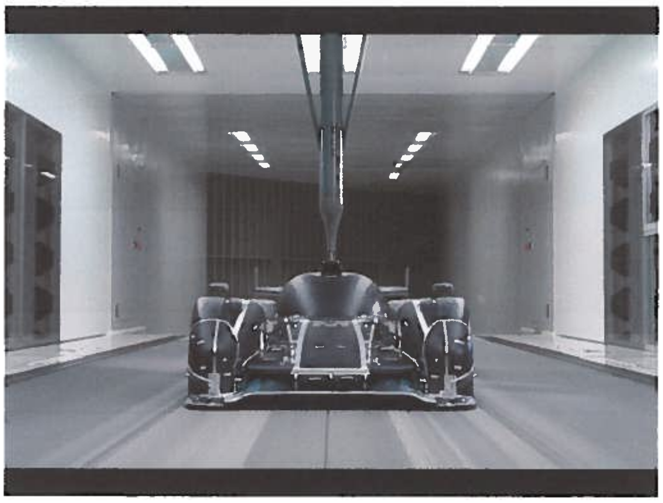


I STEDET FOR MEKANISK MEMORISERING LÆRES SOLIDE TALLKUNNSKAPER; SANNSYNLIGHETSBEREGNING, STRATEGIER I HODEREGNING, POSISJONSSYSTEMER OG KOMBINATORIKK. PROGRAMVAREN REGISTRERER HVILKE OPPGAVER BARNET FINNER VANSKELIG ELLER LETTE SLIK AT ALGORITMENE ER OPTIMALE OG EMOSJONELT STRESS SOM BEGRENSER LÆRINGSPROSESSEN FJERNES.

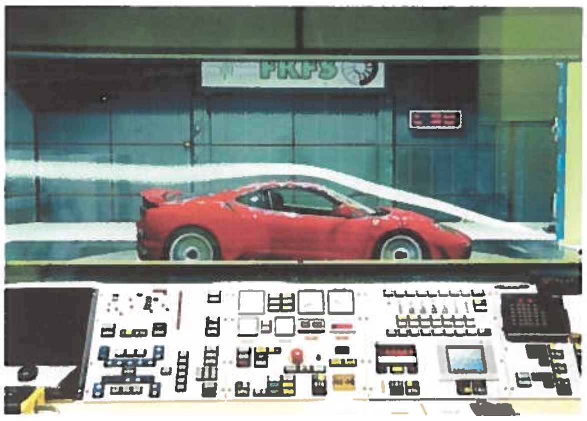
HVIS LANDSKAPET ENDRER SEG KAN MAN VELGE Å BYGGE OM BILEN UNDERVEIS FOR Å TILPASSE SEG OMGIVELSENE.



APP'EN HAR OGSÅ EN AUGMENTED REALITY DEL HVOR BARNET KAN TA MED DEN VIRTUELLE BILEN UTENDØRS OG KONKURRERE MED ANDRE



MAN KAN KJØPE FLERE DELER, BYTTE DELER MED VENNER, OG SAMLE PÅ FERDIGE BILER MAN ER FORNØYD MED.



NÅR MAN HAR KONSTRUERT EN BIL MAN LIKER, DRAR MAN PÅ VIRTUELLE EVENTYR. MAN BLIR EN JAMES BOND PÅ HEMMELIG OPPDRAG. HVER "MISSION IMPOSSIBLE" INNEHOLDER ARITMETIKKOPPGAVER SLIK AT MAN BLIR GOD I MATEMATIKK UTEN AT DET OPPLEVES SOM KJEDELIG.