ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Κατασκευή Actuated Tangibles και Αλληλεπίδραση με Επιφάνειες Αφής

Διπλωματική Εργασία

Χαρίσης Παπαχαρίσης

Επιτηρητές:
Ignacio Aedo, Καθηγητής, Universidad Carlos III de Madrid (UC3M)
Στέφανος Κόλλιας, Καθηγητής, Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ)
Andrea Bellucci, Ερευνητής, Universidad Carlos III de Madrid (UC3M)
Κώστας Καρπούζης, Ερευνητής, Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ)

Αθήνα, Δεκέμβριος 2014

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
Faculty of Electrical and Computer Engineering
Department of Information Technology and Computer Science

Construction of Actuated Tangibles and Interaction with Interactive Tabletops

Thesis Dissertation

Charisis Papacharisis

Supervisors:
Ignacio Aedo, Full Professor, Universidad Carlos III de Madrid (UC3M)
Stefanos Kollias, Full Professor, National Technical University of Athens (NTUA)
Andrea Bellucci, Researcher, Universidad Carlos III de Madrid (UC3M)
Kostas Karpouzis, Researcher, National Technical University of Athens (NTUA)

Athens, December 2014

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Κατασκευή Actuated Tangibles και Αλληλεπίδραση με Επιφάνειες Αφής

Διπλωματική Εργασία

Χαρίσης Παπαχαρίσης

Επιτηρητές:
Ignacio Aedo, Καθηγητής, Universidad Carlos III de Madrid (UC3M)
Στέφανος Κόλλιας, Καθηγητής, Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ)
Andrea Bellucci, Ερευνητής, Universidad Carlos III de Madrid (UC3M)
Κώστας Καρπούζης, Ερευνητής, Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ)

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την.....

....

Αθήνα, Δεκέμβριος 2014

.....

Χαρίσης Παπαχαρίσης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Η/Υ Ε.Μ.Π.

Copyright © Χαρίσης Παπαχαρίσης
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

It is prohibited to copy, save and/or distribute this work or parts of it, for commercial purposes. It is permitted to reproduce, save and/or distribute for non-commercial purposes, educational or research ones, provided that the source will be mentioned and this message will be maintained. Questions regarding the use of the dissertation for commercial purposes must be addressed to the author.

The statements and conclusions included in this document express the author’s perspective and they do not represent the official viewpoints of the National Technical University of Athens.

Ευχαριστίες

Η εργασία εκπόνηθηκε σε συνεργασία του τομέα Τεχνολογίας Πληροφορικής και Υπολογιστών, της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ του Εθνικού Μετσόβιου Πολυτεχνείου, και του DEI lab του τμήματος Πληροφορικής (Computer Science department) του Πανεπιστημίου Carlos III της Μαδρίτης (Universidad Carlos III de Madrid), στα πλαίσια του προγράμματος ανταλλαγής Erasmus στο οποίο συμμετείχα κατά το ακαδημαϊκό έτος 2013-2014.

Θα ήθελα λοιπόν να ευχαριστήσω τον υπεύθυνο καθηγητή του εν λόγω εργαστηρίου Ignacio Aedo, για την ανάθεση διπλωματικής στη συγκεκριμένη έδρα.
Ιδιαίτερες ευχαριστίες για τον επιβλέποντα για τη συγκεκριμένη διπλωματική Andrea Bellucci, για την πολύτιμη καθοδήγηση και ενθάρρυνση σε όλη τη διάρκειά εκπόνησης της εργασίας, όπως και μετά την περάτωσή της.
Επιπλέον, ευχαριστώ θερμά τους επιτηρητές μου στο Εθνικό Μετσόβιο Πολυτεχνείο, κκ. Στέφανο Κόλλια και Κώστα Καρπούζη, που μου έδωσαν την ευκαιρία να πραγματοποιήσω την διπλωματική μου μέσω του εν λόγω προγράμματος, όπως και για την συνολική υποστήριξή τους.
Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την συμπαράστασή τους καθ'όλη τη διάρκεια των σπουδών μου.

Acknowledgements

This work was fulfilled in cooperation between the Information Technology and Computer Science department, of the Electrical and Computer Engineering faculty, of the National Technical University of Athens and the DEI lab of the Computer Science department of the Universidad Carlos III de Madrid, as part of the exchange program Erasmus in which I participated during the academic year 2013-2014.

I would like to thank therefore, the professor in charge of the DEI lab, Ignacio Aedo, for assigning to me a thesis in this laboratory.
Special thanks go to the supervisor of this thesis, Andrea Bellucci, for the precious guidance and motivation throughout the whole project, as well as after its completion.
In addition, I would like to thank my supervisors in the National Technical University of Athens, Stefanos Kollias and Kostas Karpouzis, for offering me the opportunity to carry out my thesis through this program, and for their support in general.
Finally, I would like to thank my family for their encouragement throughout my studies.

Abstract

Αν και η ιδέα της αλληλεπίδρασης με απευθείας φυσική επαφή γίνεται όλο και πιο κοινή χάρη στην ύπαρξη ολοένα και περισσότερων touch devices, η χρήση διαδραστικών tangibles πάνω σε επιφάνειες και οι κοινωνικές τους προεκτάσεις, αποτελεί κάτι σχετικά νέο.

Αντικείμενο της εν λόγω διπλωματικής εργασίας είναι να ερευνήσει την λειτουργία των actuated tangibles και του συνδέσμου που προσφέρουν μεταξύ του «ψηφιακού» και του «φυσικού» κόσμου, καθώς και την χρηστικότητα που εισάγουν. Χάρη στην διαισθητικότητα που χαρακτηρίζει την λειτουργία της αφής, ο χρήστης εξοικειώνεται πολύ γρήγορα με την χρήση τόσο της επιφάνειας όσο και του actuated tangible. Ως εκ τούτου, γίνεται αντιληπτή η ιδιαίτερη σημασία της λειτουργικότητας τους όπως επίσης και οι πολλές δυνατότητες που αυτή προσφέρει.

Η διπλωματική εργασία επικεντρώνεται στην ανάπτυξη ενός actuated tangible όπως επίσης και στην αλληλεπίδραση του με οθόνες αφής, για παράδειγμα: ένα tablet. Εκτός αυτού, ο στόχος μας είναι να ερευνήσουμε τις αξιοσημείωτες προοπτικές που εισάγουν τα actuated tangibles σε διάφορους τομείς, δίνοντας έμφαση στον σχεδιασμό και οργάνωση σε καταστάσες εκτάκτου ανάγκης (emergency cases).

Abstract

Although the concept of interaction through direct physical contact is becoming more commonplace with the growing availability of many touch devices, the use of interactive tangibles on digital tabletops and their social aspects is relatively new.

The goal of this thesis is to examine the function of the actuated tangibles and the interlink they offer between the ‘digital’ and ‘physical’ world, as well as the usability they introduce. Thanks to the intuitiveness that characterizes the sense of touch, the user is getting very quickly familiar with the usage not only of the touch screen, but of the actuated tangible as well. For this reason, the importance of their functionality is easily noticeable, as well as the numerous prospects that it offers.

The dissertation focuses on the development of an actuated tangible as well as its interaction with touch screens, for example: a tablet. In addition, our goal is to explore the remarkable potential that the use of actuated tangibles introduces in various sectors, emphasizing in the field of emergency planification.

Key words: actuated tangibles, interaction, tangible user interfaces, interactive tabletops, TUI, Arduino, TUIC, emergency response, planification

Table of Contents

Chapter One: Introduction

1.1 Motivation

During the last two decades, Tangible User Interfaces (TUIs) have emerged as a new interface type that interlinks the digital and physical worlds. Drawing upon users' knowledge and skills of interaction with the 'real' world, TUIs demonstrate a potential to enhance the way in which we interact with digital information.
Various technological approaches in the area of the next generation's user interfaces (UIs) have been influencing each other, resulting in mixed approaches that combine different ideas or interaction mechanisms.

Tangible user interfaces (TUIs) combine the dynamic qualities -typical of digital information representations- with physical affordances. TUIs, in combination with multi-touch tables, provide passive haptic feedback for hand gestures. This allows people to interact with the input devices in the same way they interact with everyday objects, applying real world skills without the need of training or instructions. The benefits of these user interfaces include the simultaneous reduction of cognitive load placed on users (while they interact with an application) and simplification of the interaction itself.

Specifically, actuated TUIs allow data to be connected to, and represented by, physical objects (e.g. dynamic data can be linked to dynamics of the objects). They also facilitate more engaging, playful interaction and afford the use of movement as an expressive output modality.

Until recently, the coupling between 'tangible' and 'digital' has usually been in one direction; we could change digital information through physical handles, but the digital world had no effect on tangible interface elements. The use of physical motion, however, strongly relates to tangible UI philosophy. The exploration of self-actuation seems to be a natural direction for tangible user interfaces research to take. Indeed, one of the most attractive properties of the digital world is malleability: digital objects are easy to create, modify, replicate, and distribute. Physical objects on the other hand are rigid and static, which limits their utility in tangible UIs. If we could dynamically change physical properties of tangible UI elements: their shape, texture, position, speed of motion, and so on, the design vocabulary of tangible user interfaces would expand tremendously.

Therefore, we decided to construct an actuated tangible (a small vehicle) and study its movement onto a touch screen. By that means, we intended to experience the level of interaction that its self-actuation introduces, and reflect on its potential future use, primarily in the sector of emergency/disaster response.

Our project could be briefly described like this: With a software application created, the user is able to define any straight route on the screen, setting the initial and final points. Afterwards, placing the vehicle on the initial point, the car starts moving towards the final one, where it stops automatically, thanks to the continuous position-feedback that receives from the tablet we used. The user is then able to define a new route.

1.2 Structure

The project consists of three stages that are going to be analyzed more extensively later: 1) the design of the electrical circuit and the total construction of the vehicle, 2) the implementation of the tablet app for the path creation, 3) the wireless communication between the tablet and the car for the provision of the feedback required.

Provided that, the thesis follows the following structure:

In the first chapter, we have made the introduction and we have presented our implementation in general terms.
In the second chapter, we make some references on the history of Tangible User Interfaces and the related existing work.
In the third chapter, we examine the different uses of TUIs as well as the sectors where they can contribute positively. Here we make a further examination of the field of emergency planning, which is the sector that interests us the most for future applications.
In the fourth chapter, we describe the different stages that our project went through (as mentioned above) and the procedure that was followed, in greater detail.
In the fifth chapter, we describe the limitations, strengths and future work possibilities that characterize our implementation.
Finally, in the conclusion, we extract our outcomes and present our ideas and expectations regarding future development of the work presented.

Chapter Two: History of Actuated Tangibles and Related Work

2.1 History of Actuated Tangibles

Actuation

To begin with, there is a crucial question to ask: What is actuation?
Actuation means: 'to put in action, to move'. Therefore, we define actuated interfaces as interfaces in which physical components move in a way that can be detected by the user.

There are many types of actuation, for example:

- Change in spatial position of objects or their parts, e.g. their position, orientation;
- Change in speed of motion of objects or their parts, e.g. speed of rotation, speed of linear motion, direction of motion;
- Change in surface texture of objects or their parts, e.g. visible or perceived by touch;
- Change in force applied to the user, e.g. change in force amplitude, direction, or torque.

In order to take advantage of the benefits of a tabletop TUI not only for input but also for output, the interface has to be bidirectional. Some form of actuation mechanism is required, so that not only the user but also the computer can actively move the tangibles.

In 2003, Koleva [33] conducted an analysis of the types of coupling between the physical and digital and compared this with systems available at the time, revealing an asymmetry. Whereas many systems used physical objects to control digital objects, only few examples of 'tangibles that push back' were found. Most systems supported only one-way communication, either being an ambient display or a data manipulation tool. Although actuation had been a part of the vision of TUIs from the very start, given the technical difficulties, it is only lately emerging as a strong trend.

Gibson in 1960s underlined that giving the users freedom to actively and repeatedly move their hands and fingers while exploring the shape of objects increases the amount of information received through haptic sense. Note that such free exploratory movements are difficult to create with indirect haptic devices (like haptic stylus) since they usually allow for only one point of contact.

From another perspective, Ishii underlined the importance of tailored design of a tangible for a specific function to benefit from natural affordances, at the same time preserving a balance to a more general approach for the reason of reusability. There are several examples in related work how the functionality of a tangible can be customized.

Different Approaches

Throughout the years, different technologies have been influencing each other, resulting in totally new approaches that combine different views or interaction mechanisms. For instance, we can mention Tangible Augmented Reality Interfaces and Tangible Tabletops:

Tangible Augmented Reality (Tangible AR) Interfaces combine tangible input with an augmented reality display or output. The virtual objects are 'attached' to physical objects that the user manipulates. A 3D-visualization of the virtual object is overlaid onto the physical manipulative which is tagged with a visual marker (detectable with computer vision). The digital imagery becomes visible through a display, often in the form of see-through glasses, a magic lens, or an augmented mirror. Such a display typically shows a video image where the digital imagery is inserted at the same location and 3D orientation as the visual marker.

Tangible Tabletop Interaction combines interaction techniques and technologies of interactive multi-touch surfaces and TUIs. Many tangible interfaces use a tabletop surface as base for interaction, embedding the tracking mechanism in the surface. With the advancement in interactive and multi-touch surfaces the terminology has become more specific, tabletop interaction referring predominantly to finger-touch or pen-based interaction. But simultaneously, studies within the research area of interactive surfaces increasingly investigate mixed technologies, typically utilizing a few dedicated tangible input devices and artifacts on a multi-touch table.

Hornecker and Buur [34] suggest the term tangible interaction to describe a field of approaches related to, but broader than TUIs. They argue that, many systems developed within arts and design aimed at creating rich physical interactions, share characteristics with TUIs.

Implementation Technologies

The breadth of technologies, devices, and techniques used for prototyping and implementing TUIs can be bewildering. Thus, we use a number of organizing properties to discuss and compare common TUI implementation technologies. Following, we describe three implementation technologies that are often used in the development of TUIs: RFID, computer vision, and microcontrollers.

RFID

Radio-Frequency Identification (RFID) is a wireless radio-based technology that enables to sense the presence and identity of a tagged object when it is within the range of a tag reader (an antenna). There are generally two types of RFID tags: active RFID tags, which contain a battery and thus can transmit a signal autonomously; and passive RFID tags, which have no battery and require an external source to initiate signal transmission. In general, RFID tags contain a

transponder comprising of an integrated circuit for storing and processing information, and an antenna for receiving and transmitting a signal.
Most RFID-based TUIs employ passive inexpensive RFID tags and hence consist of two parts: a tag reader that is affixed to a computational device and a set of tagged objects. The communication between a tag and a reader only occurs when both are proximate. The actual distance varies based on the size of the antenna and that of the RFID tag, as well as the strength of its field.

Due to the cost of larger antennas, RFID is usually constrained to short distance detection, with objects required to be placed directly on -or swiped past- the reader. Some tag readers are capable of detecting multiple tags simultaneously, or writing small amounts of data to individual tags. Other tag readers are read-only or only capable of detecting a single tag at a time. When a tag is detected, the tag reader passes an ASCII ID string to the computer. The TUI application can then interpret the ID input string, determine its context, and provide feedback. Multiple TUIs are implemented using RFID technology.

Computer Vision

In the context of TUIs, computer vision is often used for spatial, interactive surface applications because it is capable of sensing the position of multiple objects on a 2D surface in real time while providing additional information such as orientation, color, size, shape, etc. Computer vision systems can be characterized as being either of the artificial intelligence variety (where sophisticated algorithms are used for automatically interpreting a picture) or of the tag variety (where the system tracks specifically define fiducial markers that are attached to physical objects).

Figure 2.1 Photo of the reacTable TUI

Tag-based systems tend to be more robust, more accurate, and computationally cheaper than systems of the artificial intelligence variety. Thus, tag-based computer vision is often used in the development of TUIs. Examples include Urp [21], a tangible user interface for urban planning; the reacTable [22], a tangible electro-acoustic musical instrument, and Tern, a tangible programming language for children.

Performance and reliability of vision-based systems is susceptible to variations in lighting and motion blur. Using color to identify objects can be relatively robust, but limits object recognition to a small number of high contrast colors. A way to improve the robustness and speed of detection is to paint tokens so they reflect infrared light and to employ a camera filter. This will

result in the camera only detecting the painted objects, but reduces the systems' ability to distinguish different objects.

Microcontrollers, Sensors, and Actuators

Microcontrollers act as a gateway between the physical world and the digital world. They are small and inexpensive computers that can be embedded in a physical object or in the physical environment. Microcontrollers receive information from the physical world through sensors, and affect the physical world through actuators.

There is a wide variety of sensors and actuators available to be used in embedded systems. Sensor technology can capture a wide range of physical properties including light intensity, reflection, noise level, motion, acceleration, location, proximity, position, touch, altitude, direction, temperature, gas concentration, and radiation.

Actuators affect the digital world by producing light, sound, motion, or haptic feedback. Microcontrollers may also be connected to RFID readers. Frequently used actuators include LEDs, speakers, motors, and electromagnets.

Many TUI systems are built using embedded microcontrollers. Examples include Posey [35], a poseable hub and strut construction toy; System and Flow Blocks, an educational TUI for simulating system dynamics; or Senspectra [23], a physical modeling toolkit for sensing and visualization of structural strain.

While some of the microcontrollers used for developing TUIs require low-level programming skills, several easy-to-use prototyping platforms are currently available for educational purposes as well as for TUI developers from non-technical backgrounds.

Arduino [14] is an open source physical computing platform based on a simple I/O board and a development environment. Arduino can be used to develop stand-alone interactive devices or can be connected to software running on a computer. The Arduino development environment is a cross-platform Java application that provides a code editor and compiler and is capable of transferring firmware serially to the board. It is also well connected with Processing [15], a development environment aimed at the electronic arts and visual design communities. The Arduino programming language is related to Wiring, a C-like language.

Tool Support for Tangible Interaction

Several toolkits and software libraries have emerged to support the implementation of functional TUI prototypes.

Phidgets [24] provides a set of "plug and play" USB-attached devices (e.g., I/O boards, sensors, and actuators) that are analogous to widgets in graphical user interfaces. For example, Phidets allows any analog sensor to be plugged into its board, as long as it modulates a 5-V signal. Similarly, any on/off switch and other digital I/O devices can be plugged to the board and controlled by a binary value. Phidgets are aimed to support software developers in the

implementation of mechatronic TUI prototypes composed of wired sensors and actuators. Such TUIs are capable of both physical input and physical output. The main advantage of Phidgets is that they are centrally controlled through a conventional computer rather than through a standard microprocessor. Thus, the integration of digital capabilities such as networking, multimedia, and device interoperation becomes easier.

iStuff [25] uses Java to control a set of light-weight wireless physical devices. iStuff is aimed at enabling interaction designers to rapidly prototype applications for an environment called the iRoom. Through an intermediary software called the Patch Panel, interaction designers can define high-level events and dynamically map them to input and output events.

We have already discussed Arduino in the previous section on implementation technologies. It is a toolkit consisting of the Arduino board and programming environment. Different from Phidgets and iStuff, which entail specifically built sensors and actuators that are easily plugged together and are centrally controlled through a conventional computer, Arduino interfaces with standard electronics parts. Arduino thus does not 'black-box' the electronics, but requires physical wiring, circuit building, and soldering.

The major advantage of the tools discussed above is that they lower the threshold for implementing fully functional TUI prototypes by hiding and handling low-level details and events. Hence, they significantly reduce the duration of each design/implementation/test cycle.

Two-handed Interaction

Regarding new design processes and forms of interaction, it is worth mentioning few words about the two-handed interaction:
While early HCI studies on two-handed interaction viewed two handed input as a technique for performing two subtasks in parallel, later studies showed that two-handed interaction provides additional benefits in the context of spatial manipulations and 3D input.

Hinckley found that for 3D input, two-handed interaction can provide additional benefits:
(a) users can effortlessly move their hands relative to one another or relative to a real object, but moving a single hand relative to an abstract 3D space requires a conscious effort;

(b) while one-handed 3D input can be fatiguing, two-handed interaction provides additional support. When hands can rest against one another or against a real object, fatigue can be greatly reduced;

(c) using two hands, a user can express complex spatial relations as a single cognitive chunk. This not only makes the interaction parallel, but also results in an interface which more directly matches the user's task.

Hinckley [36] noted: 'In related experimental work, we have demonstrated that using two hands can provide more than just a time saving over one-handed manipulation. Two hands together provide the user with information which one hand alone cannot. Using two hands can impact

performance at the cognitive level by changing how users think about a task: using both hands helps users to reason about their tasks.'

2.2 Related Work and other Examples

Regarding the related existing work, Brave demonstrated the first distributed actuated tabletop TUI with the PSyBench (1998) [20]. Using two synchronized motorized chessboards, his system could position one object at a time with an electromagnet but could not control the rotation.

With his Actuated Workbench(2002) [8], Pangaro introduced the concept of actuated tangibles used for two-way communication with a computer system. They built an array of electromagnets under the surface of a table to position small pucks fitted with permanent magnets and an LED for optical tracking. However, TUIs with an actuation mechanism built into the table involve high technological effort and are not very flexible in the size of the interaction volume or the mobility of the setup.

Early examples of actuated TUIs include the peek-a-boo surrogates which are rotated by a servo motor, and the Navigational Blocks which are equipped with orientation sensors and electromagnets, and programmed to attract or repel each other to give actuated feedback on whether a particular configuration yields any output for a database search. The Actuated Workbench used magnetic force to move objects on a table. A later incarnation of the same idea is Pico, a TUI that can sense and move small objects on top of a surface. This type of system is described by Poupyrev as self-rearranging displays that consist of multiple parts that rearrange themselves in space, in contrast to Shape displays that directly create 3D physical shape.

Actuated tangibles with autonomous behavior can also be interpreted as robots. Jacobsson presented the see-Puck [27] and the GlowBots [28], a collection of LED displays on autonomously moving robotic platforms that communicate with each other, resulting in a continuously changing LED pattern on the surface and reacting to people picking up and relocating them. Robotic tangibles might extend to an architectural scale: Biloria presented a series of experiments developing real-time interactive spatial prototypes, using pneumatics for actuation e.g., shape-shifting furniture and room units.
As an example of shape-shifting displays, Poupyrev presented Lumen [30], a low resolution 13 X 13 pixel bit-map display where each pixel can individually move up and down. This enables a moving physical shape or texture augmenting a 2D visual display as well as tangible 3D controls that can literally be pressed down. Sprout I/O [29] combines textiles and shape-memory alloys to create a kinetic display from soft textile that can sense touch.
The Linguabytes project provides a nice example of using actuation in a subtle way, almost unnoticeably easing a task. This learning system helps multi-handicapped toddlers with motor impairment. It directs their hand when they place a physical piece on a platform (if the RFID tag is detected, electromagnets pull the piece into place) and has a slider that automatically limits range depending on its position.

Other TUIs employ actuation as vibrotactile feedback on information being detected and sucked into a handheld device. Actuation, being the technical basis for product movement, is investigated also in product design, where movement (or 'behavioral expression') provides a means to increase product expressiveness. Movement can make abstract shapes come seemingly alive, as human perception has an intrinsic tendency for animism.
For example, the Mac laptop's softly and rhythmically blinking LED gives the impression of breathing. Products can also move physically, movement constituting a fourth dimension of design. Product movement can even express emotions through variation in speed, acceleration, type and volume of path, rhythm, and randomness.

Another approach when it comes to actuated tangibles, is the 'Shape display' [7].
Shape displays attempt to create 3D physical shapes directly and some of the shape displays share following common properties:

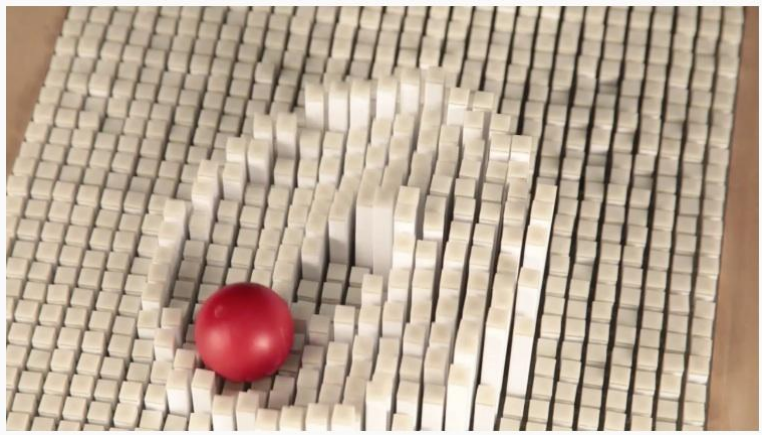

Figure 2.2 Demonstration of the 'Shape display' function

- They display relief-like shapes by physically displacing a surface of the device. This is done either by changing the properties of the materials, e.g. Protrude, Flow or Snoil, or by using mechanically actuation, such as in case of Aegis Surface, FEELEX and etc.

- They combine dynamic shapes with images, e.g. Aegis Surface, Lumen or Feelex. Combining shape with image is important. For example, if our goal is to display a 3D shape, it is natural to assume that the shape's surface would have color and patterns, as in the case of real objects.

That would require image producing capabilities. Based on these observations, the shape displays can be generalized as an extension of traditional bitmapped displays where each pixel has an additional attribute: height.
The actual mechanism of displacement, the shape and the arrangement of the pixels depends on implementation. We call this design approach an 'RGBH graphics', where RGB is a color components and H is a height of a pixel.

Regarding more recent work, Touchbugs [2] are active tangibles that are able to move across surfaces by employing vibrating motors and can communicate with camera-based multi-touch surfaces using infrared LEDs. Touchbugs' embedded inertial sensors and computational capabilities open a new interaction space by providing autonomous capabilities for tangibles that allow goal directed behavior.

Figure 2.3 Touchbugs

They are small tangibles that use directed bristles and vibration motors for actuation (giving them the ability to move independently). Their infrared LEDs allow multiple Touchbugs to be spatially tracked (position and orientation) on optical multi-touch tables and to communicate information about their internal state to the table. Embedded inertial sensors, which capture displacement and orientation, provide rich opportunities for interaction design including direct physical manipulation, as well as symbolic and metaphorical gestures. This novel combination of sensing and actuation capabilities goes beyond simple changes of (virtual) states (e.g. by the use of buttons) offering significantly more potential of expressive interaction.

Apart from the above, a novel class of actuated tabletop tangibles is called magnetic widgets, or Madgets [6]. Besides moving Madgets across the table, the user can take advantage of new actuation dimensions, such as height, force feedback, and power transfer. Furthermore, Madgets are low-cost, easy to prototype, and do not require any built-in electronics or power source for actuation or tracking.
Madgets introduce the following very useful characteristics:

- Force feedback. Beside the inherent haptic feedback of tangible controls, actuation can provide active force feedback as an additional output channel.
- Resistance. By default, moveable parts such as the turning arm of a knob can be rotated freely. Actuation, however, allows us to change the resistance or the perceived friction of an object by adapting the PWM signal. By attracting a slider knob we can make it harder to move.
- Vibration feedback. The tangential actuation can be used to let Madgets vibrate. This can act as an audiohaptic signal that a Madget needs attention, e.g., when a critical value is reached or when a remote user has changed a value. Vibration patterns could also be used to create more complex feedback.

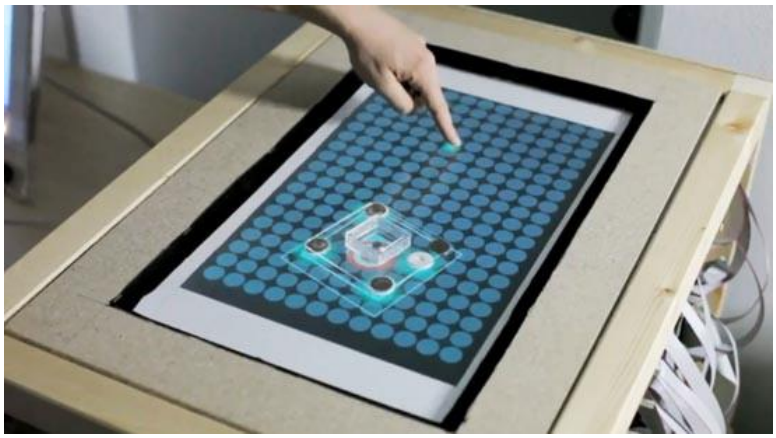

Figure 2.4 Madgets in action

- Inductive energy transfer. Induction is a well understood charging technique for devices ranging from electric tooth brushes

to mobile phones. This basic principle can be applied to transfer power from the table to Madgets in order to support electronics without the need for batteries or cables.

Another actuated tangible is called ACTO [9], and it is principally characterized by the concept of reusability and rapid prototyping.

Its development follows a modular design strategy for actuated tangible user interface objects (ACTOs). Because of the high grade of modularity, typically integral parts like the actuation mechanism or physical configuration can be individually customized and interchanged, allowing reusability for different scenarios and setups. The ACTOs can be equipped with different I/O devices, ranging from simple buttons to complex sensors. The system is also very mobile and can be set up quickly anywhere. The concept targets in minimizing the time spent on the redesign of the system for different experiments.

Chapter Three: Uses of TUIs and Emergency Planification

Here we firstly present the big variety of applications where the actuated tangibles can benefit the users, and afterwards we take a closer look in the domain of emergency response.

3.1 Uses of TUIs

TUIs for Learning

A large number of TUIs can be classified as computer-supported learning tools or environments. There are several underlying reasons for this. First, learning researchers and toy designers have always followed the strategy of augmenting toys to increase their functionality and attractiveness. Second, physical learning environments engage all senses and thereby support the overall development of the child.

A range of projects often combine augmented reality techniques with tangible interface notions. Africano present 'Ely the Explorer' [37], an interactive play system that supports collaborative learning about geography and culture while practicing basic literacy skills. The system mixes touch-screen technology, use of physical knobs to interact with screen content, tangible toys, and RFID-tagged cards. Related to literacy education is WebKit, a system supporting the teaching of rhetorical skills to school children. A more recent development is TUI's supporting learning for children with special needs. Digital construction kits such as Topobo [38] and Lego MindstormsTM are increasingly used within educational robotics specifically for special needs education.

Problem Solving and Planning

Generally speaking, tangible representation is most compelling in spatial or geometric application domains such as urban planning and architecture where the physical arrangement and manipulation of objects has a direct mapping to the represented problem. It has been found that using a TUI can support designers' spatial cognition, reduce cognitive load, and enable more creative immersion in the problem. However, several studies have also demonstrated the benefits of tangible interaction with abstract information tasks.

Urp is a TUI for urban planning that allows users to collaboratively manipulate a series of physical building models and tools upon a surface, in order to perform an analysis of shadows, proximities, reflections, wind, and visual space. While users place and manipulate building

models upon the surface, the interface overlays digital information onto the surface, activating and updating multiple simulations.
Physical Intervention in Computational Optimization (Pico) is a TUI based on a tabletop surface that can track and move small objects on top of it. The position of these physical objects represents and controls application variables. The Pico interface has been used to control an application for optimizing the configuration of cellular telephone network radio towers. While the computer autonomously attempts to optimize the network, moving the objects on the table, the user can constrain their motion with his or her hands, or using other kinds of physical objects (e.g., rubber bands).

Information Visualization

By offering rich multimodal representation and allowing two-handed input, tangible user interfaces hold a potential for enhancing the interaction with visualizations. Several systems illustrate the use of tangible interaction techniques for exploring and manipulating information visualizations.
GeoTUI [32] is a TUI for geophysicists that provides physical props for the definition of cutting planes on a geographical map that is projected upon a surface. The system enables geophysicists to select a cutting plane by manipulating a ruler prop or selection handles upon the projected map.

Tangible Programming

The concept of tangible programming, the use of tangible interaction techniques for constructing computer programs, has been around for almost three decades since Radia Perlman's Slot Machine interface [26] was developed to allow young children to create physical Logo programs.

Several TUIs allow children to teach an electronic toy to move by repeating a set of guiding motions or gestures. Examples include Topobo, Curlybot, and StoryKits [31]. This approach for programming is often referred to as programming by demonstration or, as suggested by Laurel, programming by rehearsal. Many tangible programming systems use physical constraints to form a physical syntax that adheres to the syntax of a programming language. For example, Tern consists of a collection of blocks shaped like jigsaw puzzle pieces, where each piece represents either a command (e.g. repeat) or a variable (e.g. 3).

Entertainment, Play, and Education

Many museum interactives that combine hands-on interaction with digital displays can be interpreted as TUIs. For example, at the Waltz dice game, in the Vienna 'Haus der Music' (Museum of Sound), visitors roll with two dice to select melodic lines for violin and recorder,

from which a short waltz is automatically generated. An exhibition about DNA at the Glasgow Science Museum includes several exhibits that allow visitors to tangibly manipulate DNA strands to understand how different selections effect genes.

Leitner presents a truly mixed reality gaming table that combines real and virtual game pieces. Real objects are tracked by a depth camera and can become obstacles or a ramp in a virtual car race, or real and virtual dominos are connected to tumble into each other.

Music and Performance

Music applications are one of the oldest and most popular areas for TUIs, becoming ubiquitous around the millennium. Music TUIs are either designed for the novice where they provide an intuitive and easily accessible toy, or aim at the professional that appreciates physical expressiveness, legibility, and visibility when performing electronic music in front of an audience.
The Audiopad system [39] allows users to manipulate and mix sound samples by placing tangible tokens onto an augmented surface. New samples can be dragged onto the surface from a menu on the rim.

Biomedical Purposes

Science demands high levels of accuracy, which limits the types of tools available to researchers. In the study of any protein, there are two major questions: what does it do and how does it do that? Until now, tools like protein viewers and articulated models are mostly static, and as such, provide limited scope. The next step in this area is an advanced augmented reality environment that incorporates haptic information in the tangible model.
Actuated tangible interfaces can improve protein study by providing physical handles to manipulate and understand the complex 3d shapes and movements that determine protein function, as Brown and Raffle demonstrated [40].

3.2 Emergency Response Planning

Emergency response planning is a major sector that has attracted the attention of plenty of researchers nowadays.
Emergency response planning is a process that involves many different stakeholders who may communicate concurrently with several channels and exchange different information artifacts.
The planning typically occurs in an emergency operations centre (EOC) and involves personnel both in the room and also in the field. The EOC provides an interesting context for examining the use of tablets, tabletops and actuated tangibles, and their role in facilitating information and communication exchange in an emergency response planning scenario.

Large scale emergencies and disasters highlight the vulnerability of modern society to collapses of infrastructure that is crucial to daily life (e.g. roads, phone service, and electricity). A significant challenge with emergencies also arises from the different types that can occur, from unplanned events like natural disasters, train derailments, and chemical spills, to planned large-scale events like the Olympics and the World Cup.

Emergency response has always been both a challenging case and an experimental base for the emerging technology. In the early 20th century, strategy and emergency response took place around draft tables. Experts worked around these draft tables with strategic information artifacts; their goal was to maintain control in dynamic and critical situations. Later on, computer-supported collaborative work focused on enabling information representation with distributed displays.
Today, we are looking for even more efficient methods, in order to maximize our performance and one very efficient style of interaction is combining tangible user interfaces (TUIs) with tabletops. This is beneficial because such situations require intuitive and direct manipulation of sophisticated information layers.

Emergency response planning is comprised of many important tasks, from detecting and monitoring the emergency to the deployment of resources and communication management. Emergency response planning is also inherently a peripheral process; critical information about an emergency can arrive from numerous sources (e.g. first responders, reporters, or online sources) and information processing and analysis are typically done in parallel with the primary emergency response-planning activity, frequently with interruptions. In any case, accurate and timely information is as crucial as is rapid and coherent coordination among the responding organizations.

Key Emergency Response Management (ERM) functions are navigating through the map (e.g. specific zoom in a certain region while the rest of the map remains unchanged), filtering data, selecting information recipients, searching datasets, drawing time-dependent freeform areas, and assigning tasks (e.g. to fire-fighters, medical personnel, etc). Under time pressure the mouse and keyboard could be insufficient; therefore, intuitive graspable solutions, such as tangible user interfaces (TUIs), are undoubtedly better suited for ERM.

Besides digital pen and touch gestures, physical tokens can be used in disaster planning systems on tabletops. They act as input as well as output, improving the experience of the users. They can change simulation parameters according to their physical position above the tabletop, and provide feedback through their change of position/state/reaction. The manipulation of physical tokens to interact with emergency systems has reduced the learning curve of these systems.

For example, CoTracker [1] is a tangible tabletop system, currently developed, with high potential for ERM teamwork. On an interactive map expert team members can discuss an operational picture using TUIs like bricks, frames (and pens, as well).

Research has been done on how to access and distribute key information in an ERM situation. Rauschert [42] examined how speech and gesture recognition can be coupled with a knowledge-based dialogue management system for storing and retrieving geospatial data. They showed how

a multimodal, multiuser Geographical Information System (GIS) interface benefits collaborative work on large displays. Wigdor pointed out the importance of hosting experts in a shared horizontal workspace and allowing them to work on their particular subtasks without interfering with each other. Another possibility is the CERMIT system [41], where light-emitting tangible devices and mobile phones were used to interact with the tabletop environment.

While we usually assign generic functions to devices-physical tools (like rulers, dials, and pens) and use them in many different cases, we seldom create specialized case-specific devices such as a frame for multilayered data access or a palette for sketching. Ullmer and Ishii proposed that specialized TUIs could offer richer interaction capabilities but at the cost of reduced flexibility. The degree of cognitive support offered by a TUI has been shown to be a function of the TUIs degree of specialization.
While malleable solutions have a lower degree of specialization, tools like frame, palette, ruler, and caliper are more specialized. The inherently complex structure of an ERM case requires rather specialized TUIs. This is why CoTracker employs tailor-made tangibles.

Figure 3.1 ePlan multi-surface

Another interesting new prototype is the prototype is the ePlan Multi-Surface [10], a multi-surface environment for emergency response planning exercises that was designed with domain experts from C4i Consultants Inc.

There are several crucial questions about the approach that should be followed to achieve better functionality: Is it better to have multiple tangible devices, each with a limited number of dedicated functions (that is: more space-multiplexing), or a limited number of devices where users of one device can choose among several functions (that is: more time-multiplexing)? Should user interface designers employ generic or specialized tangible devices? What is a good balance between generic and specialized tangible devices?

In general, it is understood that collaborative use of large interactive surfaces benefits the ERM domain. There are several possible approaches: use of multi-touch surfaces, employment of tangible objects like graspable devices, mechanic tools, clay-like materials, or combinations of these approaches.

Speaking about ERM, we should mention as well, the exceptional work of James Patten. Both his ‘Sensetable’ and ‘Thumbles’ projects, although not exclusively directed to ERM use, can contribute significantly towards that orientation.

Figure 3.2 The Thumbles combine the advantages of a GUI with the intuitiveness that a physical object implies

- Sensetable [16] is a system which tracks the positions of intelligent objects on a tabletop surface, and projects information onto the objects themselves. Sensetable has been applied to situations like business supply chain management, urban planning, interactive visual art, and the performance and composition of electronic music.

- Thumbles [17] are small robots that drive around on a tabletop surface under computer control to create a new type of interface. Users interact by grasping and moving the Thumbles around, and the computer can respond by moving them as well. Thumbles combine the versatility of a graphical user interface with the tactile advantages of interacting with physical objects. The robots rearrange themselves on the table based on what the user wants to do. For example, the Thumbles can represent characters in a video game, or molecules in a chemistry simulation. Thumbles are particularly well suited for these problems, particularly when teams of people must work together to develop a solution.

Chapter Four: Project Description

Here we firstly make a presentation of the electronic part of the project and the process it went through. Then, we describe the programming aspect of our work and in the last sub-chapter we refer at the function of the total application.

4.1 Electronic Part

Regarding the electronic stage of the project, the aim was to create a relatively small and light physical object (car) suitable for examining its movement on the tablet that was provided by the laboratory.
For the completion of this stage, we made use of the following elements:

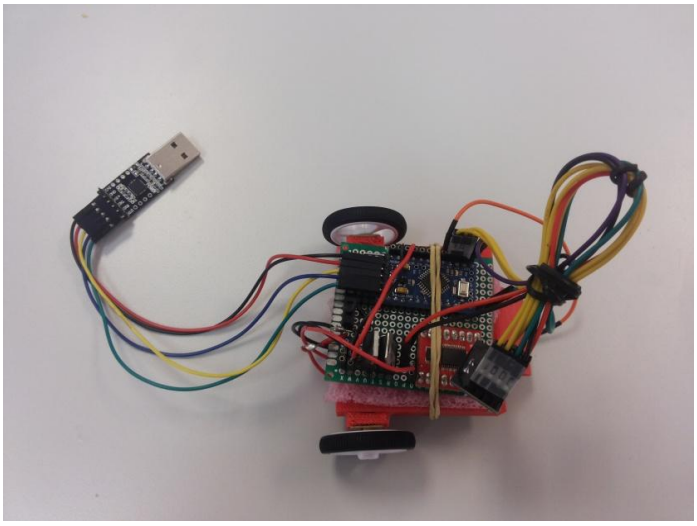

Figure 4.1 View of our actuated tangible from above

-An Arduino Pro Mini microprocessor, particularly the version that operates at 5V. This board was ideal in order to accomplish small dimensions and a low cost implementation.
-An FTDI USB cable, for the uploading of Arduino sketches.

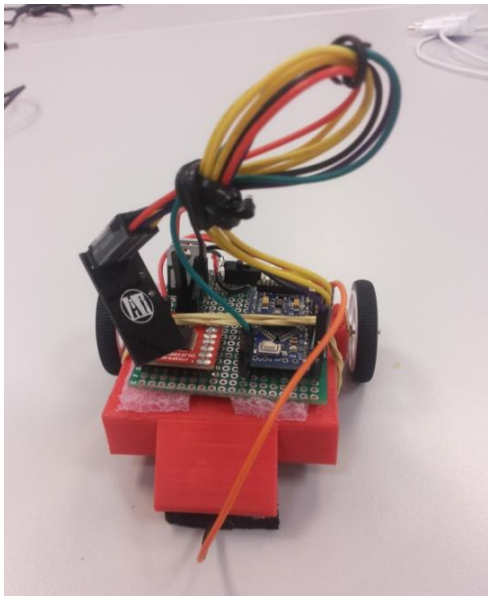

Figure 4.2 Front view of the vehicle

-The car chassis, that was created with 3D printing.
-Two motors with regulators, operating at 6V, mounted at the two back wheels (dimensions 24 x 10 x 12 mm). Thanks to the regulators, the appropriate power could be achieved.
-Two back rubber wheels.
-The TB6612FNG Motor Controller, responsible for the control and regulation of the motion of the back wheels.
-A metallic ‘free’ wheel (diameter: 9.5mm), mounted at the center of the front part of the chassis (replacing the two front wheels)
-The nRF24L01 RF Antenna (2.4GHz) [19], operating at 3.3V, for the transmission of messages between the car and the Arduino Uno, that was playing the role of

our server.
-Two small 12V batteries: One for the power supply of the motors and the other for the supply of the rest of the circuit.
-One 3.3V voltage regulator (LD1117V33), necessary for the powering of the RF antenna.
-One 5V voltage regulator (L805CV), necessary for the powering of the Arduino Pro Mini.
-A two-way switch, connected in series with the battery of the circuit, in order to switch on and off the power.
-Male and female pins and cables for the connections required.

Few words about our main components:

- The Arduino Pro Mini is a microcontroller board based on the ATmega168. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 8 analog inputs, an on-board resonator, a reset button, and holes for mounting pin headers. A six pin header can be connected to an FTDI cable or Sparkfun breakout board to provide USB power and communication to the board. There are two version of the Pro Mini. One runs at 3.3V and 8 MHz, the other at 5V and 16 MHz.(we used the second one).

- The RF antenna is a 2.4 GHz Radio module that is based on the Nordic Semiconductor nRF24L01+ chip. The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. It operates at 3.3V and the usual (limit) distance of its function -quoted by different suppliers- is 100 meters.

- The TB6612FNG motor driver can control up to two DC motors at a constant current of 1.2A (3.2A peak). Two input signals (IN1 and IN2) can be used to control the motor in one of four function modes - CW, CCW, short-brake, and stop. The two motor outputs (A and B) can be separately controlled, the speed of each motor is controlled via a PWM input signal with a frequency up to 100kHz. Logic supply voltage (VCC) can be in the range of 2.7-5.5VDC, while the motor supply (VM) is limited to a maximum voltage of 15VDC.

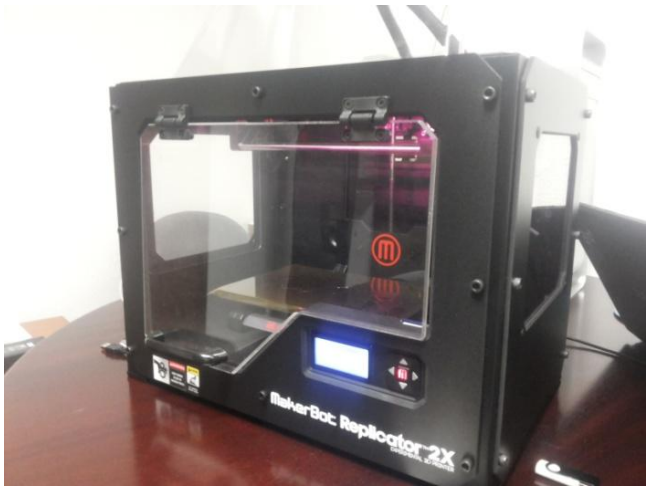

Figure 4.3 The Makerbot Replicator2X 3D printer we used to print our components

With regards to the creation of the chassis, we used the Makerbot Replicator2X 3D printer. The design of the chassis was made with the Google SketchUp 2014 and the printing by using the MakerWare software.

Specifically, we printed two 3D parts for the car:

Firstly, the main body: Mainly a rectangular platform of dimensions 6x6mm (so that the soldered circuit would fit on it) and a slot spacious enough to facilitate the cable connections between the motor controller

pins and the motors. The edges of the platform are extended towards down at a length of 1.5mm with the appropriate openings for the wheel motors. These features are visible in the figure 4.4.
In addition, at the back part of the body we included a thin extension for the mounting of the extra metallic wheel, using two screws, as shown in the pictures presented later.
Secondly, we printed a bumper to accommodate the smooth contact of the car with the surface through capacitive material (specifically: anti-static conductive foam) glued at the bottom part of the bumper. The capacitive material was essential for the constant detection of the car's position, thanks to the capacitive sensing technology of the tablet (explained later in detail).
Generally, we intended to maintain the design as simple as possible focusing on its functionality without adding extra parts that could potentially incommode the future debugging (for example, a car hood).

Regarding the designing process of the electrical layout, we had to deal with few trial-and-error scenarios.
In particular, in the beginning we used nano-motors (of dimensions 6x15mm) trying to minimize the total size of the vehicle as much as possible. After implementing this choice, we noticed that the car, although quite light, did not move easily. The reason was that this type of motors is operating at 12000 rpm, resulting at many revolutions per minute but, at the same time, at less torque. Therefore, we decided to opt for motors with reducers, in order to achieve better torque, which proved to be more efficient. The final motors are operating at 3000rpm offering a reduction rate 10:1.

In addition, since the idea was to keep the size limited, we intended to use lithium coin cells (Cr 2032) for the powering of the circuit. Finally, the change of our motors resulted inevitably at higher power requirements. As a consequence, we chose two 12V cylindrical batteries (of 33mAh capacity), of fairly small size. One of them was responsible for the powering of the two motors and the other one for the supply of the rest of our circuit.

Figure 4.4 Our car parts (platform and bumper) after printing

In the beginning, we tried to create the two back wheels using 3D printing, but we faced some practical problems. In concrete, the rolling surface of the wheels was not equally smooth and the hole required for the mounting of the motors either was not printed exactly in the middle, or was bigger than expected. It turned out that the 3D printing process was not able to support such detailed manufacturing, when it was coming to such small dimensions.

Here you can see the 3D Sketchup designs for the platform and the bumper, as well as their preview in the Makerware environment, just before printing:

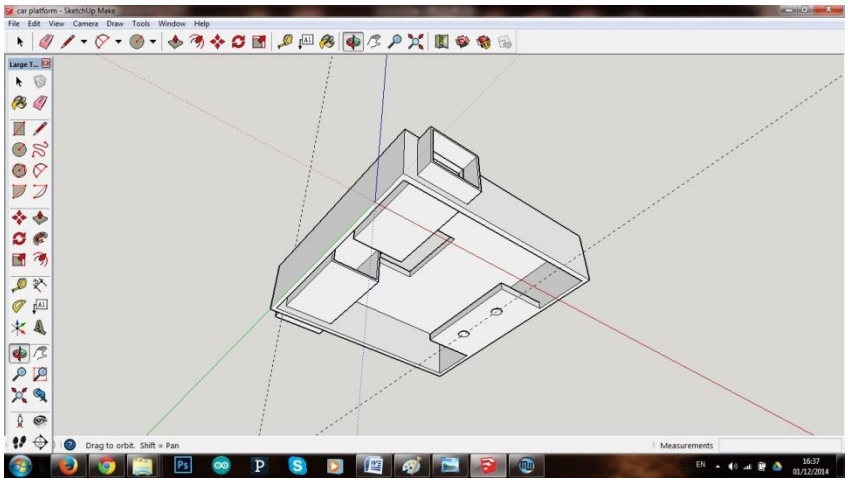

Figure 4.5 3D sketch of the platform

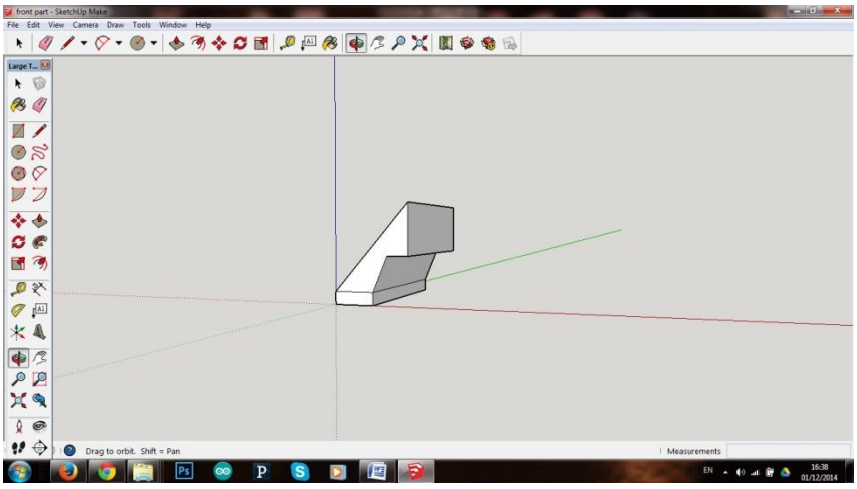

Figure 4.6 3D sketch of the bumper

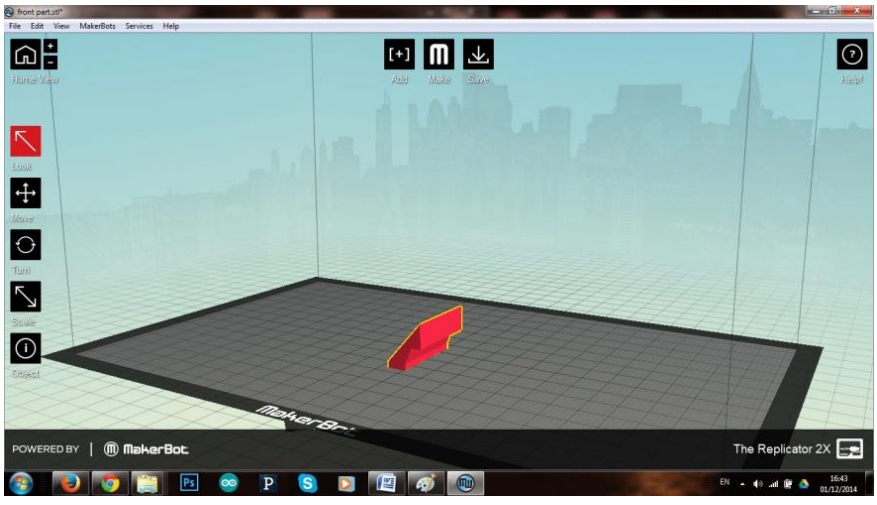

Figure 4.7 Makeware preview of the bumper

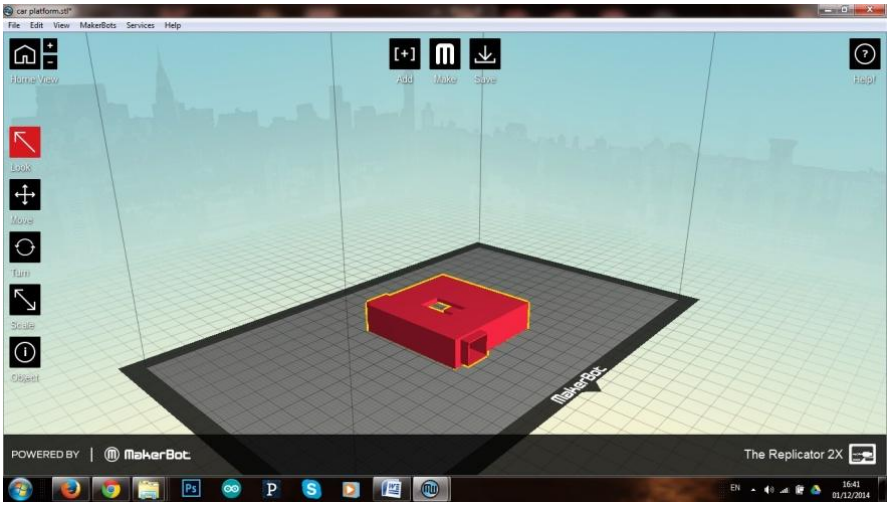

Figure 4.8 Makeware preview of the platform

Here you can see the pin layout of the motor controller (viewed from below) that briefly explains the function of each one of the pins.

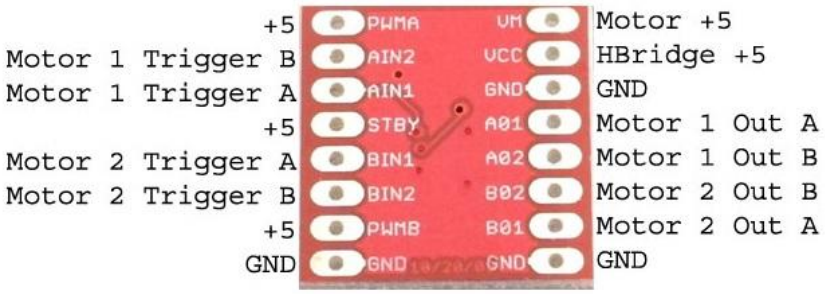

Figure 4.9 Pin layout of the motor controller TB6612FNG

The circuit we designed and followed during the soldering process is the following one: Here you can see the PCB layout of the electrical circuit, viewed from above.

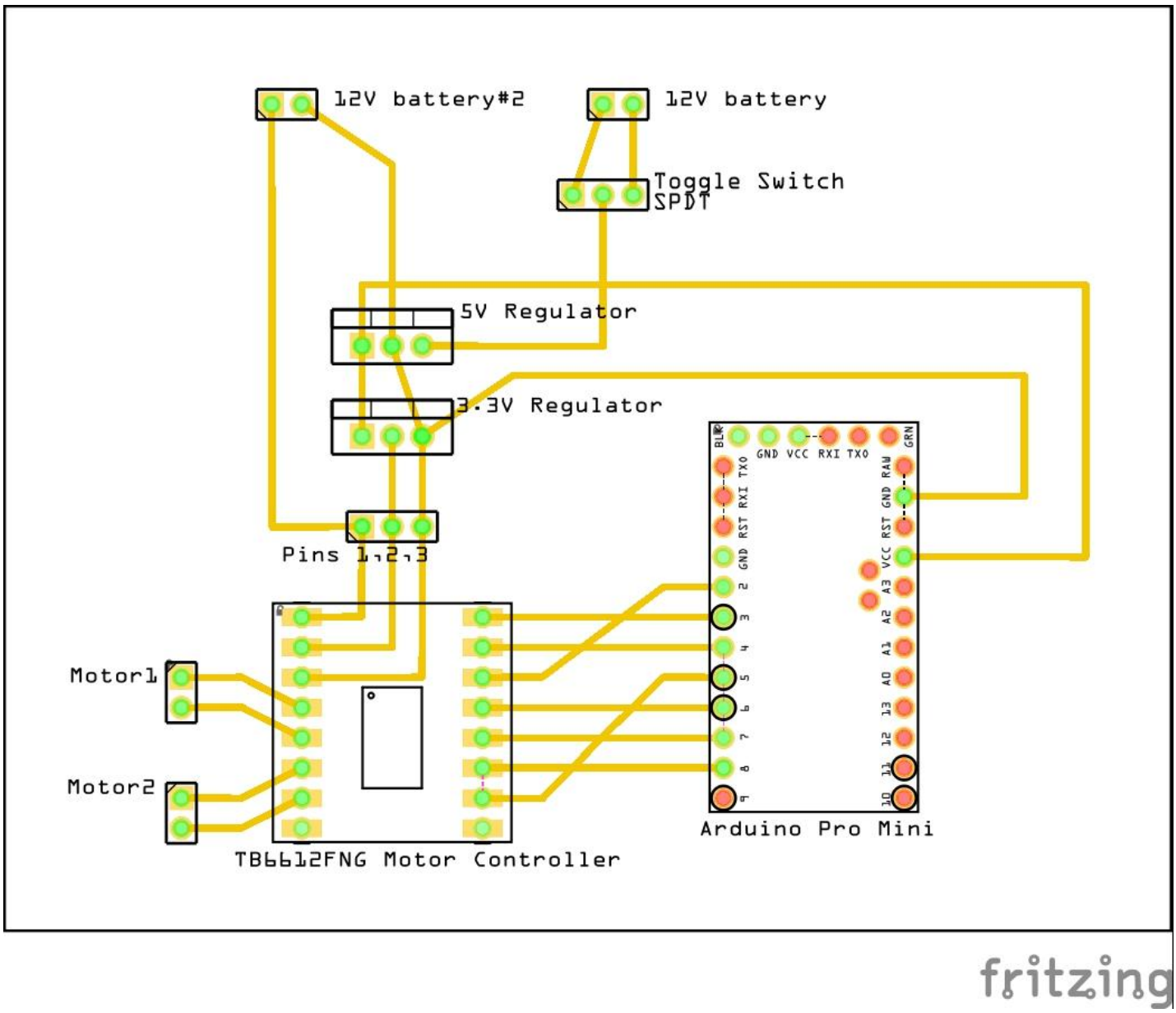

Figure 4.10 PCB of the electrical circuit (designed with the Fritzing program)

Some comments on the previous layout:

The supply consists of two 12V batteries. The first battery leads to a toggle switch offering the ability to power off and on the circuit, as required. Then we have placed a 5V regulator to produce the necessary 5V potential, crucial for the operation of the Arduino Pro Mini. The exit of that regulator is supplied to the Input pin of a second regulator that regulates its Output at 3.3V. Subsequently, we produce the supply needed for the RF antenna (and the motor controller), since this component operates at 3.3V.

The other 12V battery is connected directly to the appropriate pin (VM) of the TB6612FNG motor controller that corresponds to the powering of the two motors. These motors are attached to the A01, A02 and B01, B02 pins of the controller, respectively.

With regards to the Arduino Pro Mini, we soldered its pins (2-8) with the appropriate controller pins (AIN1, PWMA, AIN2, PWMB, STBY, BIN1, BIN2 respectively) to achieve the connections required, as indicated in this scheme:

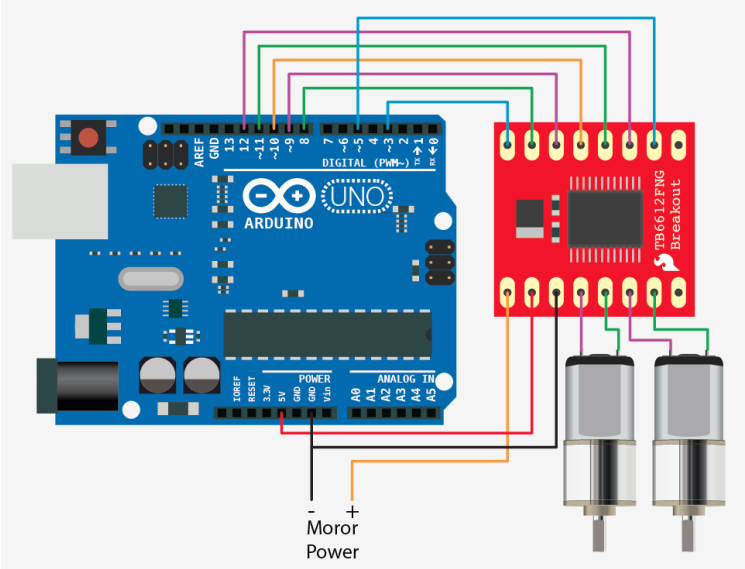

Figure 4.11 Indicated connections between Arduino and motor controller

Last but not least, the pins '2' (3.3V) and '3' (GND) of the above Fritzing layout, as well as the pins 9-13 of the Arduino are attached to the RF module enabling the transfer of data to and from the server. Below you can see the correspondence between the RF and the Arduino pins.

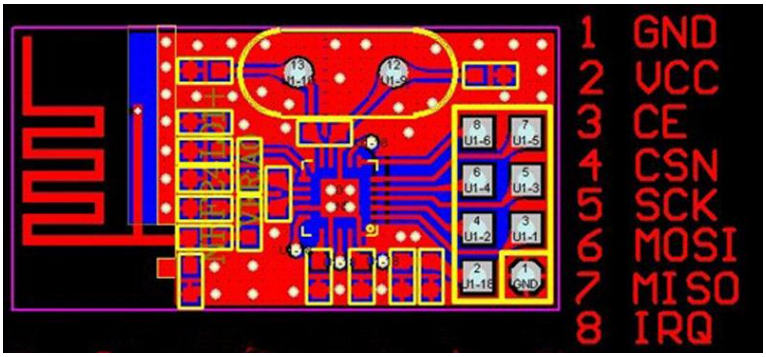

Figure 4.12 Pin allocation of the RF antenna

Signal	RF Module	Arduino pin for RF24 Library
GND	1	GND
VCC	2	3.3V
CE	3	9
CSN	4	10
SCK	5	13
MOSI	6	11
MISO	7	12

Capacitive Sensing Technology

It is worth focusing a little on how the capacitive sensing technology functions, since we take advantage of its qualities throughout the execution of our project.

Firstly, the capacitive sensing technology is based on the capacitive coupling, that is the transfer of energy within an electrical network by means of the capacitance between circuit nodes. Many types of sensors use capacitive sensing, including sensors to detect and measure proximity, position or displacement, humidity, fluid level, and acceleration. Human interface devices based on capacitive sensing, such as trackpads, can replace the computer mouse. In general, it is acclaimed that capacitive touch screens are more responsive than resistive touch screens, but less accurate. There are two major types of capacitive touch technology: surface capacitive and projected capacitive.

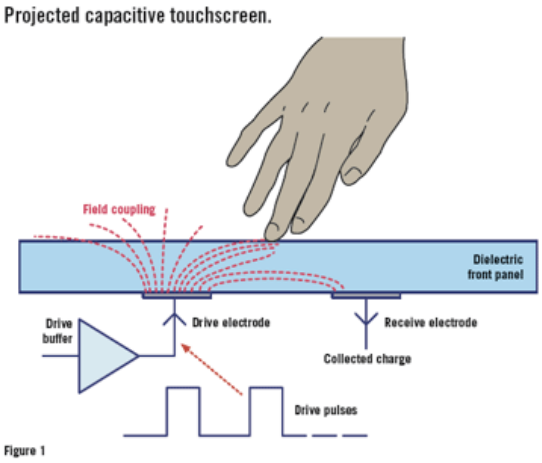

Figure 4.13 Projected capacitive touch screen

The surface capacitive touch panel is coated with conductive layer on one side of the insulator, and small voltage is applied to the layer. Once a conductor, such as human finger, touches the other side of insulator, a capacitor is formed.

By means of measuring the change of capacitance from the four corners of the panel, the panel's controller can determine the location of the touch. Currently, multi-touch devices are generally made by projected capacitive technology (PCT).

Single conductive layers of X-Y grid or two separate, orthogonal conductive layers are etched on projected capacitive touch panels. The multi-touch controller of PCT sense changes at each point along the grid. In other words, every point on the grid generates its own signal and relays multi-touch points to the system. For instance, SmartSkin used capacitive sensing and a mesh-shaped antenna to detect multiple hand positions and object's shapes.

Moreover, Diamondtouch developed at Mitsubishi Electric Research Laboratories, is another interactive table system based on capacitive sensing and supports the ability to distinguish among multiple users.

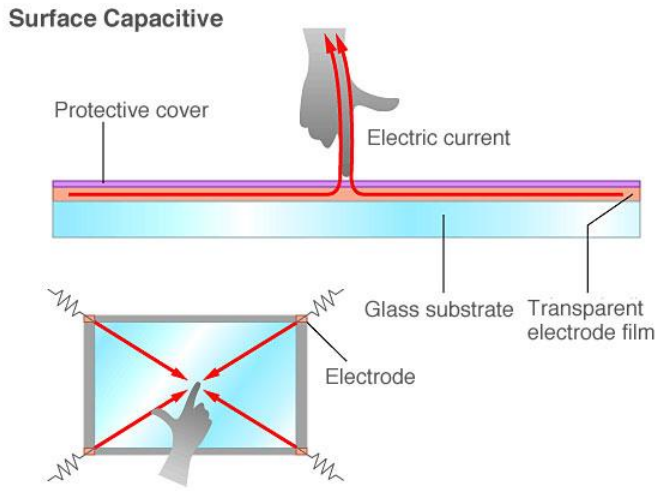

Figure 4.14 Surface capacitive touch screen

In our case, we take advantage of the

surface capacitance of the capacitive tablet screen. In this basic technology, only one side of the insulator is coated with conductive material. A small voltage is applied to this layer, resulting in a uniform electrostatic field.
When a conductor, such as a human finger, touches the uncoated surface, a capacitor is dynamically formed. Because of the sheet resistance of the surface, each corner is measured to have a different effective capacitance. The sensor's controller can determine the location of the touch indirectly from the change in the capacitance as measured from the four corners of the panel: the larger the change in capacitance, the closer the touch is to that corner.

So, our conductive foam is pinned to one analog Arduino pin and when it touches the surface, a capacitor is formed, respectively. Therefore, as the car moves across the screen, the exact location of the vehicle is easily tracked. In the same time, the distance between its current location and the final point is dynamically recalculated.
What happens is that the car sends continuously a message to the Arduino Uno regarding the pixels remaining as it approaches the final target. Finally, when the distance remaining is less than a prearranged limit (in our case, we opted for 100 pixels) the car (or more specifically the Pro Mini attached in the car construction) receives the order from our server to stop the movement and pauses the motors' motion. The exact way of communication between the tablet and the car is explained later.

Apart from the above, since we are presenting the electronic aspect of the project, we are entitled to mention some information regarding the TUIC technology, since we will comment on that in the 'future work' section.

TUIC Technology

We have to mention that TUIC [4] is based in the capacitive sensing, which is used in our case. TUIC is a technology that enables tangible interaction on capacitive multi-touch devices, such as iPads, iPhones, and 3M's multi-touch displays, without requiring any hardware modifications. TUIC simulates finger touches on capacitive displays using passive materials and active modulation circuits embedded inside tangible objects, and can be used with multi-touch gestures simultaneously.
TUIC consists of three approaches to sense and track objects: spatial, frequency, and hybrid (spatial plus frequency), which enables tangible interaction on unmodified capacitive multi-touch panels. TUIC uses passive materials and active modulation circuits to simulate multi-touch gestures. These multi-point patterns and gestures are designed to be easily distinguishable from human gestures, and to encode object IDs. TUIC tags can be embedded inside tangible objects to sense the objects' identification, movement, and rotation.

The frequency approach, called TUIC-f, utilizes the fast response time supported by capacitive touch sensing. It encodes data in the time domain by simulating finger touches at the same location at various frequencies. There are two advantages of an active frequency tag:

Firstly, only a single touch point is required to encode data, enabling more tags to be used simultaneously. Also, it is possible to build a tag with a smaller footprint. Secondly, a tag can

change its frequency dynamically and the corresponding object ID or state. This enables the tag to represent a button or a dial, supporting the types of tangible interaction in Sensetable and SLAP, for example.
There are several limitations to frequency tags. The first is the delay in sensing object IDs because several cycles may need to be observed. Second, fast movement causes a second touch point to be registered at a different location, and is difficult to distinguish from a human gesture.

Another very useful way to take advantage of the TUIC approach, is the creation of authentication keys.
In general, users encounter two problems while keying the PINs or passwords on mobile devices such as iPhones or iPads. First is pressing the wrong keys on the virtual keyboards. Second, entering passwords in public space, like a bus or elevator, potentially exposes the passwords to bystanders. We could use TUIC tags as authentication keys to replace PINs and passwords.
In this scenario, users can carry these tags, say fastened to a key-ring, and simply place the tags on a device's display for authentication. In addition, the key assures contact-based, secure authentication that prevents remote attacks.

4.2 Programming Part

About the server part we had to use:

- An Arduino Uno, operating at 5V, connected serially (with USB cable) with our PC.
- Another nRF24L01 RF Antenna (2.4GHz), operating at 3.3V, for the transmission of messages between the car and the Arduino Uno.

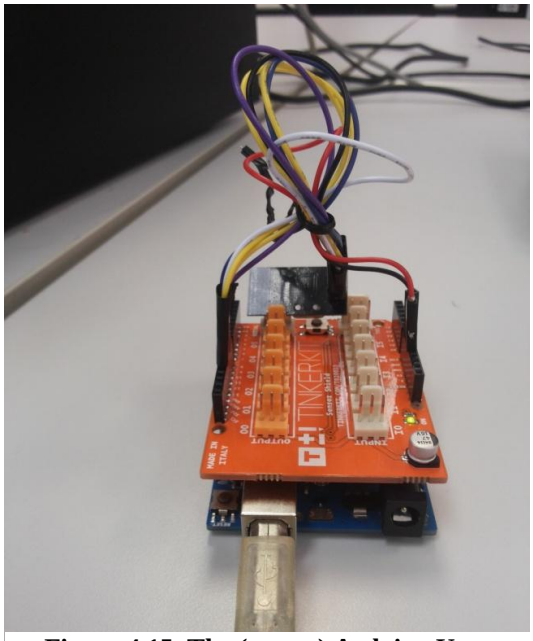

Figure 4.15 The (server) Arduino Uno, connected with the RF antenna

- A computer, serially connected to the Uno, necessary to run the appropriate server sketches.

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connects to a computer with a USB cable. Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. It operates at 5 volts.

Now, regarding the programming languages used: The programs of the vehicle (Pro Mini) and the server (specifically, the Uno part) are written in Arduino- Code, which is basically a C/C++ variant. The open-source Arduino environment is

very user-friendly and facilitates the writing of programs and quick uploading to the i/o board. The environment is written in Java and based on Processing, avr-gcc, and other open source software. A developer can specify the pins used for the components and their nature (input or output, analog, digital or PWM). Apparently, we had to import the necessary libraries for the use of the RF antenna (<nRF24L01.h>, <RF24.h>) and the serial communication of the Arduinos with peripheral devices (<SPI.h>).

For the server part and the tablet interface, we used the Processing IDE. Processing is an open source programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities. The language builds on the Java language, using a simplified syntax and graphics programming model.
In addition, Processing communicates very well with the Arduino IDE, which made it a very useful choice for our project.

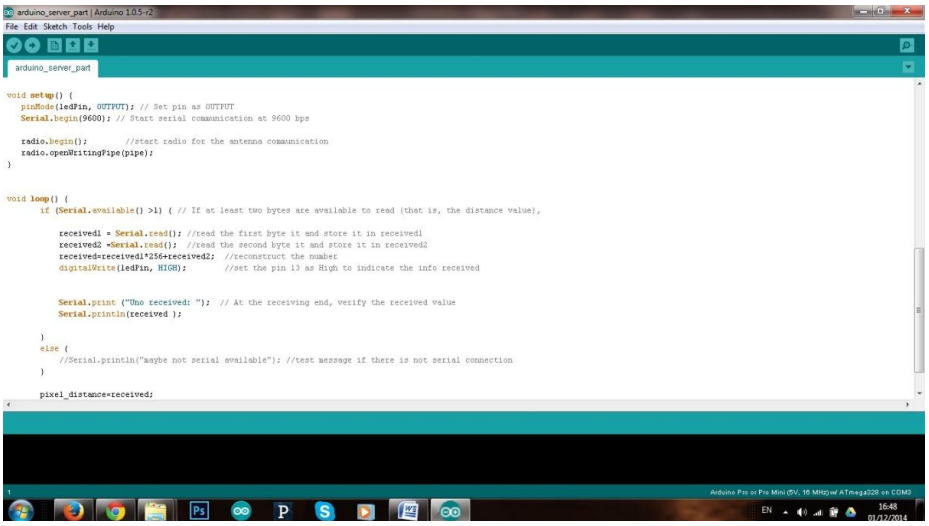

Figure 4.16 View of the Arduino IDE

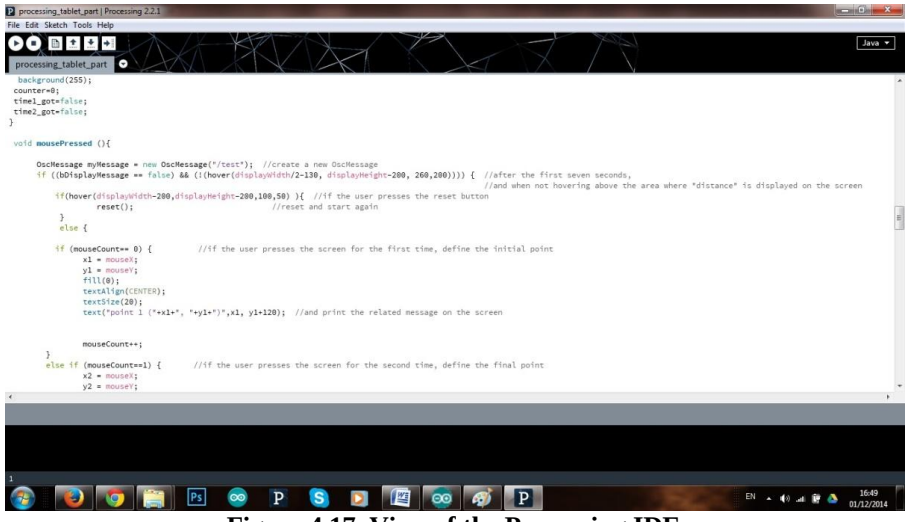

Figure 4.17 View of the Processing IDE

For the wireless communication between the two Processing scripts (that is, the tablet part and the server part) we used a server-client socket connection.

In few words, normally a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

Figure 4.18 The server listens for clients' requests

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

If anything goes wrong with the connection, for example the host is not there or is listening on a different port, an exception is thrown.

Figure 4.19 Server accepting the client's request and establishing a connection

In terms of libraries, we made use of the oscP5, a library written by Andreas Schlegel [18] for the programming environment Processing. OSC is the acronym for Open Sound Control, a network protocol developed at CNMAT (Center for New Music and Audio Technology), UC Berkeley.

The following schematic represents the wireless communication accomplished between the actuated tangible and the tablet, through the server.

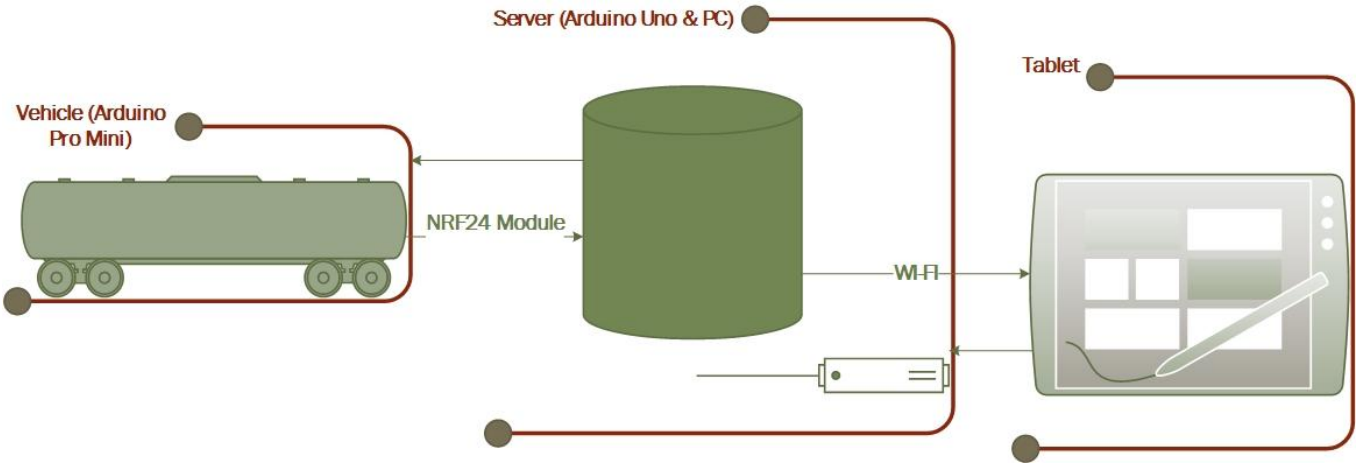

Figure 4.20 A schematic of the Wi-Fi communication between the actuated tangible and the tablet (made with Microsoft Visio)

In the Appendix you can check the code of the four sketches we used (Arduino-car part, Arduino-server part, Processing-server part, Processing-tablet part).

Lastly, a note regarding the control of the motors by the controller: Each motor has three control pins: two for direction, and one for speed.
When one direction pin is HIGH and the other is LOW, the motor will spin towards one direction. If we 'flip' the HIGH-LOW values of the two pins controlling direction, the motor will start moving to the opposite direction. On the other hand, if we set the direction pins both HIGH or both LOW, the motor stops.
The PWM pin allows you to 'analogWrite' to this pin to control the speed of that one motor. Using 'analogWrite(0)' the motor stops. On the contrary, using analogWrite (255) it will go full speed. Setting any analog value between 0 and 255 can control respectively the speed of the motor. The above points can be easily seen in the 'Car part Arduino' sketch in the Appendix.

4.3 The Project

To start with, our system consists of:
-the tangible object (the small car we designed and constructed)
-the tablet
-the server, namely a computer with an Arduino Uno serially connected to it, by a USB cable.

In the beginning, the user is prompted by a text message on the screen of the tablet to define the path after a few seconds. Then, he can select any two points of the surface as the initial and final ones.
That way, a straight route is immediately created, as indicated at the same time on the screen with the visible creation of a path.

Afterwards, the user places the vehicle on the initial point -or within a small distance from it, predefined by a specific radius of tolerance. The car should face the final point.
After its placement on the monitor, the car starts moving towards the final point, where it finally stops automatically. After this trial, the user can press the "Reset" touch button that appears at the lower right corner of the screen, redefining once again another path of different length.
And so on.

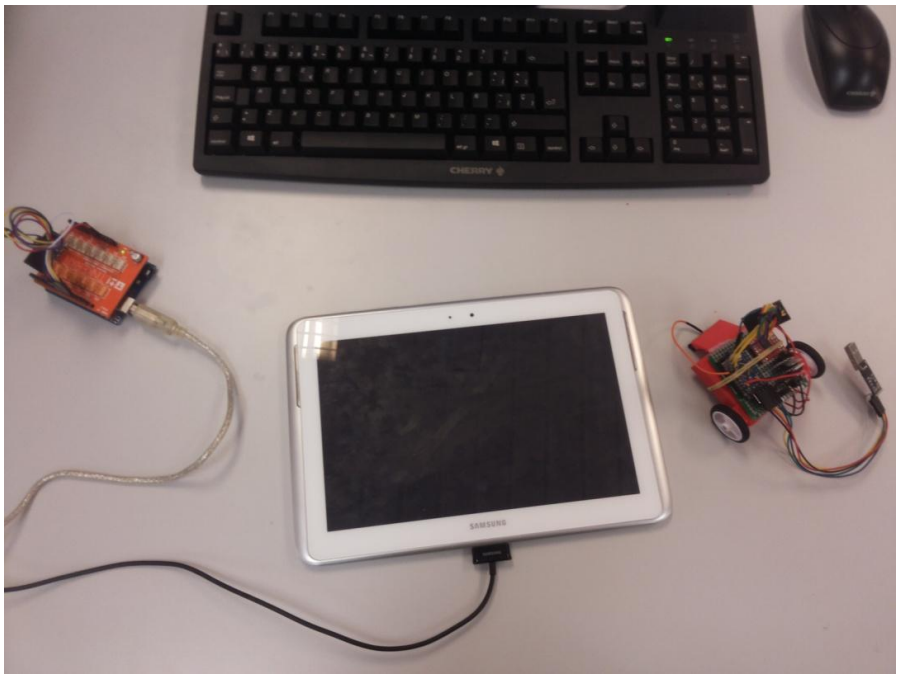

Figure 4.21 Our system consists of the tangible, the tablet and the server

Our application works like this:

The tablet constantly calculates the distance between the car and the final point, in terms of pixels. Of course, the initial value of this (decreasing) distance is the distance between the initial and final point of the path -defined by the user.
That distance is transmitted wirelessly to the computer, through OSC messages.

Then, the computer sends that info to the serially connected Arduino Uno, which functions as the interface of the server with the vehicle. The Uno transmits the distance left to the Pro Mini, thanks to the RF module.

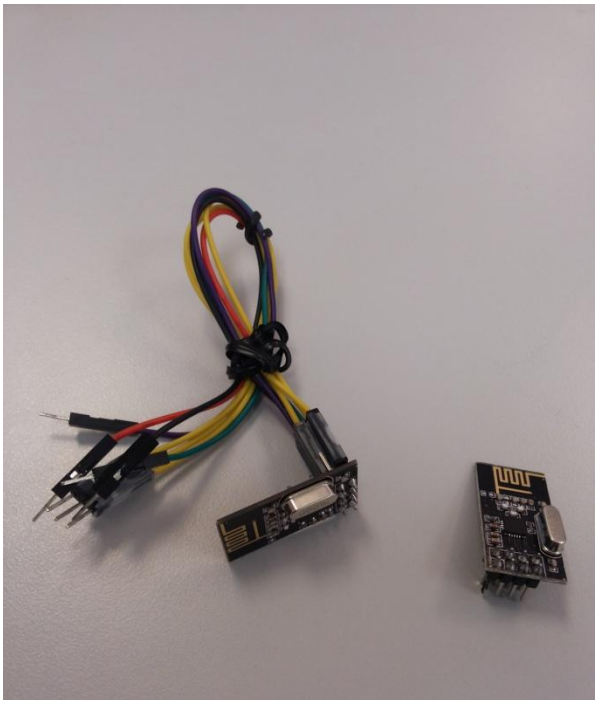

Figure 4.22 For the communication between the Pro Mini and the Uno, each of them uses an RF module

When it comes to the car, the Pro Mini controls the motor controller and subsequently, the movement (or not) of the vehicle. Apparently, as the car keeps moving, the distance to the target is being reduced continuously and this info is still being transferred to the Pro Mini, owing to the server. When that value gets lower than a predefined threshold (in our case, 100 pixels) that implies that the car has practically reached the target. Therefore, the microprocessor (Pro Mini) stops the motion of the motors.

Afterwards, when the user sets another path, the new "total distance" surpasses again the threshold and the car starts moving, until it reaches finally the new target.

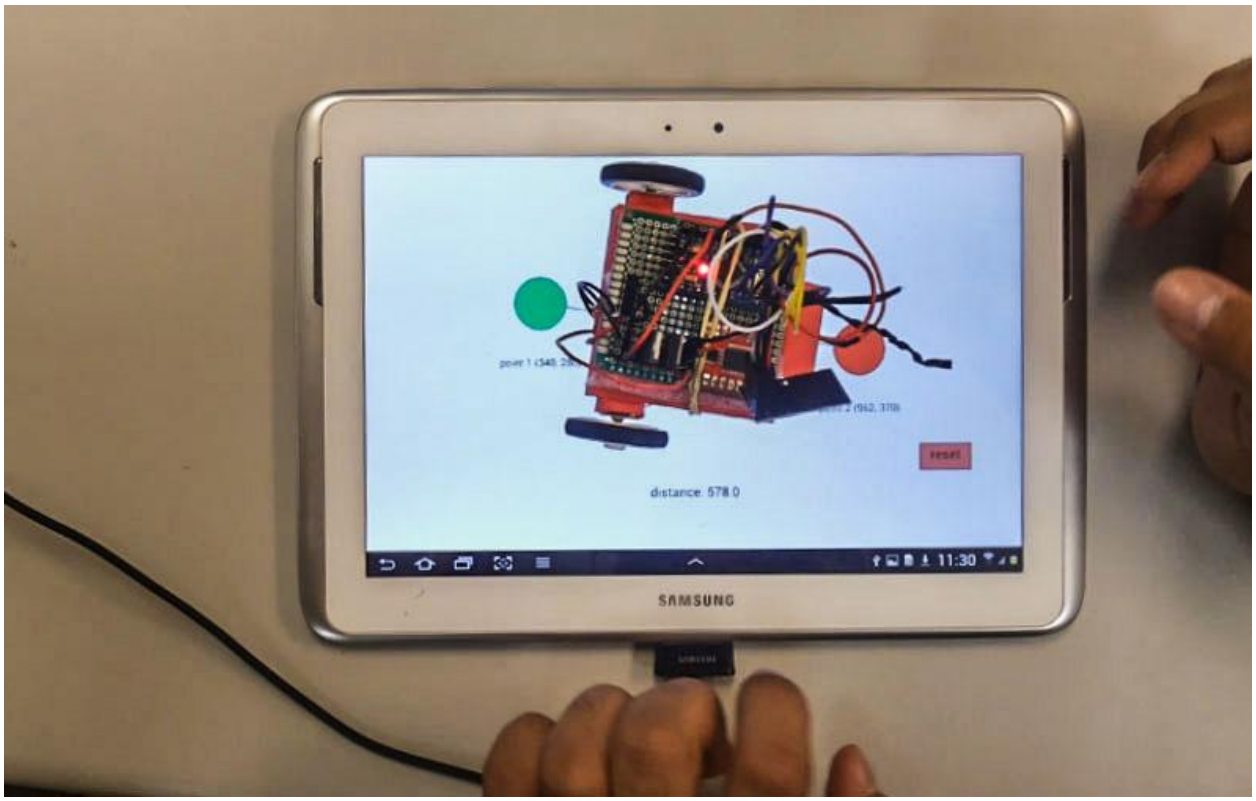

Figure 4.23 Our actuated tangible in the middle of its itinerary (from the initial-green point to the final-red one).

Chapter 5: Limitations, Strengths and Future Work

5.1 Limitations

There are specific limitations that characterize our system:

- Until now, our implementation examines the simple case of a straight line path, where the car is facing towards the target. Provided that, it is understood that if the vehicle is facing towards an irrelevant direction, it firstly has to turn towards the final point.

- The tangible cannot rotate around its center, or move backwards and therefore can get stuck at the corners of a table.

- On the long run, we would be interested to develop an emergency response application using more than one objects simultaneously. But the problem appearing here is that, the number of tangibles that can be used at the same time is constrained by the number of touch points and the size of the screen.

- Due to factors such as irregular surfaces and slight differences in the friction all over the tabletop, the car is unlikely to move by default in a straight line. Consequently, a feedback control loop is required to stabilize the motion for these situations.

- Inevitably, we face a power supply constrain. Since our intention was to reduce the dimensions of the project as much as possible, we had to reach a compromise when it comes to the size (and battery life) of the batteries used. It is noticed that our 12V batteries do not last very long. The solution we can apply here is to use rechargeable ones.

- One of the biggest challenges for TUIs is scalability. Successful applications for small problems or data sets often do not scale up to complex problems involving many parameters and large data sets. Although our project is adequate for this fairly small representation, we could face difficulties in larger representations.

- Poupyrev pointed out that while digital objects are malleable, easy to create, modify, replicate, and distribute, physical objects are rigid and static. As physical objects are not mutable, the system cannot transform these into different objects or alter their physical properties (e.g. change their color). Specificness of objects, while promoting learnability, conflicts with abstraction and versatility. That is the case in our situation, as well, where the car has a totally defined shape and context, lacking abstraction.

5.2 Strengths

- The tangible object is small and light, avoiding impediment or fatigue of the user. In addition, it works wirelessly, being charged autonomously (thanks to its batteries). That means that it does not require cable connection to keep operating.

- The project can be executed/demonstrated in practically any flat surface (with the precondition that the surface is capacitive).

- Active tangibles (in our case, the car) can provide a great extent of flexibility. For example, the included inexpensive microprocessor (Pro Mini) could allow the integration of a wide range of components (like extra sensors/ accelerometers) in the future, if we choose to modify the project.

Furthermore, we should mention some additional strengths/advantages that apply generally when it comes to actuated tangibles -consequently, in our very project, as well:

- For a flexible TUI, scalability and actuation are important factors. An actuation mechanism that is integrated *into the table* would mean that, in case of extension of the working volume (or even: in case of the change of the display technology) we would face an immense effort. In our situation, including the actuation mechanism into the tangible allows costs to scale with the number of tangibles and stay almost unrelated to the size of the table. In fact it would be preferable, if the actuation mechanism would be interchangeable as well and no special surface would be required.

- Tangible artifacts can act as resources for action in several ways. They allow a very wide range of actions and can be manipulated independently of the system ('offline', or for other purposes). This makes them resources for physical manipulation, for example in making use of epistemic action.
  Tangibles support referential, social, and contextually oriented action, making it easy to make references by pointing, touching, or performing publicly visible actions. They can be a shareable resource. Tangibles can also be resources for perception and sensory experience, allowing for bodily engagement.

- Space-Multiplexing and Directness of Interaction: In tangible interfaces that employ multiple interaction objects, input

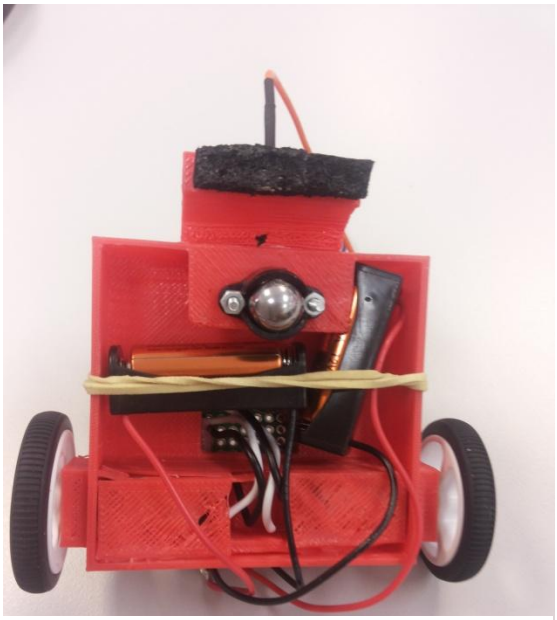

Figure 5.1 View of our car from below. We can observe the presence of the two batteries and the 'free wheel' that replaces two conventional front ones.

is space-multiplexed. Different physical objects represent different functions or different data entities. This enables the system designer to take advantage of shape, size, and position of the physical controller to increase functionality and decrease complexity of interaction. Consequently, in our case we can introduce new objects with different characteristics that would be associated with different acts.

- Tangible interfaces can be interpreted as a specific implementation of the original notion of Ubiquitous Computing. In the sense that, Ubiquitous Computing is aimed at allowing users to remain situated in the real world, and at retaining the primacy of the physical world. Yet, while embedded in context, the design goal for tangible interfaces is not the invisibility of the interface, but rather the physicality of the interface.

- Our physical body and the physical objects with which we interact play a central role in shaping our understanding of the world. Children learn abstract concepts through bodily engagement with tangible manipulatives. Professionals such as designers, architects, and engineers often use physical artifacts to reason about complex problems. One of the strengths of TUIs compared to traditional user interfaces is that they enhance this connection of body and cognition by facilitating tangible thinking- thinking through bodily actions, physical manipulation, and tangible representations.

5.3 Future Work

- Our solution takes into account the case of the straight line. Given that, it is understood that a solution should be implemented for more complex paths and random trajectories. For this, we could use a gyroscope (or an accelerometer) to sense its deviation from the initial orientation. The gyroscope measures the angular velocity of the device around three perpendicular axes.

- Our current tangible is not well suited for tilted surfaces. Therefore, we also want to integrate a more sophisticated tracking solution. Including the use of accelerometer and a gyroscope to sensor the orientation would result to a large data set transmitted, a wide range of additional features in fact available. In addition, these sensors allow the recognition of simple motions (like tapping on the device) or gestures such as shaking, which could be used as a direct input or captured for later analysis.

- The device could also have the ability to detect if it is laid on its back or if a user is holding it (using motion classification). In this case, the vibrating motors can be used as an output to provide variable haptic feedback or to attract the attention of the user.

- Since the surface used is capacitive, we should take advantage of the TUIC approach (as mentioned in the 5.1 section). Specifically speaking, the idea is to send a tone from the car (essentially, the Pro Mini) to the screen at a specific frequency. To achieve this we

could use a Tone library of the Arduino IDE, or send PWM from a pin. As a result, the car would be recognized by this 'unique' kind of identity –its frequency- enabling consequently the use of different objects on the same screen.
Each one of them would be individually recognized by our system. Consequently, we could potentially modify the app so that the screen provides additional info to the user regarding each object when it is placed (like its name, purpose, etc) offering a much more vivid experience for him.

- In the paper 'Introducing Robots to Interactive Tabletops' [5] it is mentioned that when placing actuated TUIs or robots on an interactive tabletop, we expect them to be able to engage and attract attention beyond what is possible by visual aspects of the tabletop. However, robots can be viewed as unique entities, affording social attributes that are not demonstrated by actuated TUIs. Robots beyond their physicality and form can provide a sense of agency, a sense of being, and requiring enhanced awareness from their user, in ways that are not that remote from being aware of another human user.

These characteristics allow the robots to take on different social roles, i.e. story teller, a companion, assistant, tool or just an attractive and engaging toy. These abilities of the robot to become a social partner in an interaction scenario can dramatically enhance the interaction experience on digital tabletops. An example of this, is a recently proposed work from researchers in the Department of Computer Science, University of Calgary: the 'Spidey'.
In our case, we could expand the application constructing and putting to use more anthropomorphic actuated tangibles, or introducing robot-like devices as well. The concept would be to create this feeling of familiarity for the user and enhance his experience.

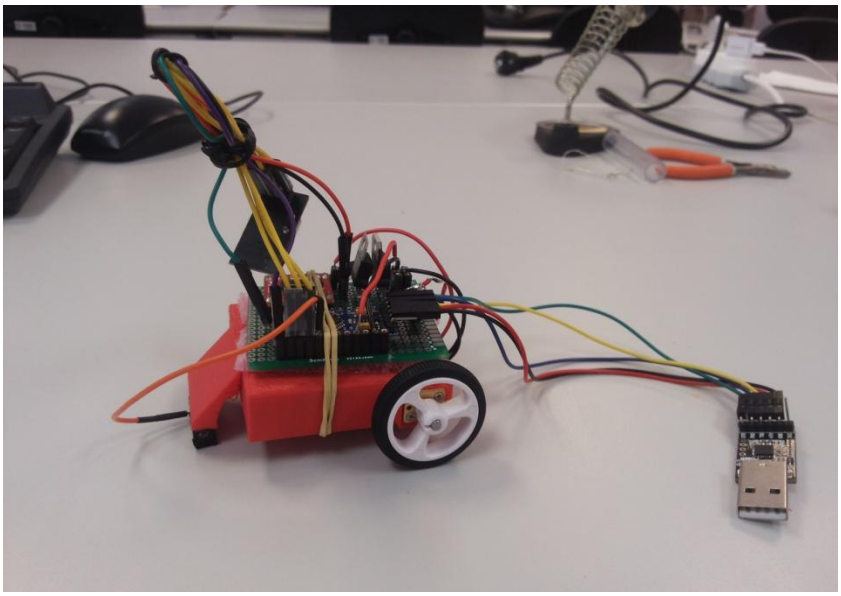

Figure 5.2 Side view of our car from below. Here it is connected with the FTDI cable that enables the loading of sketches from the computer.

- As we have mentioned, we would like to concentrate more in potential use of our application in ERM-focused applications. To start with, an emergency response plan clearly varies depending on the situation. In particular, there are different parameters we should take in notice, concerning the environment (natural, social, etc) and the infrastructure related to each case. Anyway, we are accepting the hypothesis that in the

majority of the cases, the Response Management functions remain quite similar (e.g. as mentioned before: navigating through the map, filtering data, selecting information recipients, drawing time-dependent freeform areas, assigning tasks).
Ideally, in order to create an adequate Emergency Response application, we might have to model the majority of the factors that intervene (like citizens, buildings, vehicles, medical/fire services, etc) making the planning more 'tangible' and intuitive.
Under this scope, the idea presented previously, regarding the introduction of 'robots' to interactive tabletops [5], seems very appealing.
In our project, we made a step towards the modeling of the 'vehicles', but we still need to surpass our limitations to improve its functionality.
Hopefully, in the future we could incorporate our approach into a broader project that would take into account more complex conditions. Meaning an ERM application that will include the models of different entities as well (that is, the other stakeholders of the emergency scenario, as stated before).

In general, the philosophy behind the design of such an application needs to answer to the controversial debate: Should user interfaces utilize generic or specialized tangible devices? What is the advisable equilibrium between generic and specialized ones?
We propose that, when it comes to future research in the ERM field, we should focus on developing a system with few generic devices but with a high degree of specialization. In that way, we will limit the number of devices to avoid complicating the system. Subsequently, although the system will require some cognitive effort from the users in the beginning, it will make (on the long run) our application less time-consuming and more efficient under the stressful conditions of an emergency case, thanks to this specialization.

Chapter 6: Conclusion

In the last decade TUI research has become an established research area in HCI, as attested by the growing body of work and the number of venues dedicated for TUI research. Seeking to provide seamless interfaces between people, digital information, and the physical environments, TUI research shows a potential to enhance the way in which people interact with and leverage digital information.
However, TUI research is still in its infancy. Extensive research is required to better understand the implications of interlinking the physical and digital worlds, to design tangible interaction techniques for complex real-world application domains, and to develop technologies that bridge the digital and physical worlds.

Actuation seems to be the next frontier in tangible user interfaces. There is a lot of space for further development. New emerging technologies, will potentially allow creating efficient and inexpensive actuated tangible interfaces in the future that can be used in communication, information, presentation, emergency planning and other applications. Developing such applications would perhaps require stepping outside of the boundaries of the classic tangible UI domain and combining expertise from robotics, haptic interfaces, design and architecture.

In this paper, we intended to explore the usability of the actuated tangibles and the level of interaction that they introduce. To acquire a first-hand experience, we decided to design and construct an actuated tangible object ourselves. Consequently, blending the fields of electronics and computers we developed our physical object (vehicle) and an appropriate tablet application testing its functionality.

In particular, the user is asked to define a path on the tablet selecting any two points of the surface as the initial and final ones. That way, he is able to designate any straight-line trajectory. Afterwards, the user places the vehicle on the initial point and the car starts moving towards the final point, where it stops automatically. Then, he can press the 'Reset' touch button available, redefining once again another path of a different length.
The application is based on the wireless communication of the tablet with the Arduino Pro Mini microprocessor that is responsible for the motion of the tangible. The messages are transmitted through our server that consists of a computer and an Arduino Uno.

Furthermore, we referred at the history of tangible interaction that led to today's evolution and examined the existing implementation technologies. In addition, we presented the previous work in the field of Tangible User Interfaces, commenting on a variety of TUIs with completely different philosophy, function and target (e.g. Pangaro's 'Workbench', Shape displays, Touchbugs, Madgets). The type of actuation can also vary significantly (e.g. vibrotactile, change of orientation, use of the third dimension).

In addition, we emphasized on the numerous uses of TUIs (like problem solving, entertainment, learning) and took a closer look at the field of emergency planning and its principal functions and aspects. Then, we analyzed our application in greater detail and we explored the limitations

and strengths that lay among its characteristics. Finally, we commented on the future work and possible improvements of our implementation, and we stressed out the usefulness of tangible interaction in emergency case management (ERM), aiming to canalize our project towards that direction.

Chapter 7: References

[1] Kunz, A., Yantaç, A. E., Alavi, A., Woźniak, P., Landgren, J., Sárosi, Z., & Fjeld, M., (2013, June), Tangible Tabletops for Emergency Response: An Exploratory Study

[2] Diana Nowacka, Karim Ladha, Nils Y. Hammerla, Daniel Jackson, Cassim Ladha, Enrico Rukzio, Patrick Olivier, (2013, May), Touchbugs: Actuated Tangibles on Multi-Touch Tables

[3] Shaer, O. & Hornecker, E., (2010), Tangible User Interfaces Past: Present, and Future directions

[4] Neng-Hao Yu, Li-Wei Chan, Seng-Yong Lau, Sung-Sheng Tsai, I-Chun Hsiao, Dian-Je Tsai, Lung-Pan Cheng, Fang-I Hsiao, Mike Y. Chen, Polly Huang, Yi-Ping Hung, (2011), TUIC: Enabling Tangible Interaction on Capacitive Multi-touch Display

[5] Sowmya Somanath, Mario Costa Sousa, Ehud Sharlin, (2013, February), Beyond Actuated Tangibles: Introducing Robots to Interactive Tabletops

[6] Malte Weiss, Florian Schwarz, Simon Jakubowski, Jan Borchers, (2010, October), Madgets: Actuating Widgets on Interactive Tabletops

[7] Ivan Poupyrev, Tatsushi Nashida, Makoto Okabe, (2007), Actuation and Tangible User Interfaces: the Vaucanson Duck, Robots, and Shape Displays

[8] Gian Antonio Pangaro, (1997), The Actuated Workbench: 2D Actuation in Tabletop Tangible Interfaces

[9] Emanuel Vonach, Georg Gerstweiler, Hannes Kaufmann, (2014, November), ACTO: A Modular Actuated Tangible User Interface Object

[10] Apoorve Chokshi, Teddy Seyed, Francisco Marinho Rodrigues, Frank Maurer, (2014, November), ePlan Multi-Surface: A Multi-Surface Environment for Emergency Response Planning Exercises

[11] Eckard Riedenklau, Thomas Hermann, Helge Ritter, (2014, February), An Integrated Multi- Modal Actuated Tangible User Interface for Distributed Collaborative Planning

[12] Andreas Kunz, Asim Evren Yantaç, Ali Alavi, Paweł Woźniak, Jonas Landgren, Morten Fjeld, Zoltán Sárosi, (June 2013), Tangible Tabletops for Emergency Response: An Exploratory Study

[13] Wikipedia web page, <http://en.wikipedia.org/wiki/Main_Page>

[14] Arduino web page, <http://www.arduino.cc/>

[15] Processing web page, <https://www.processing.org/>

[16] James Patten, The Sensetable project, <http://www.jamespatten.com/sensetable.php>

[17] James Patten, Thumbles project, <http://www.pattenstudio.com/thumbles/>

[18] Sending and Receiving OSC Data Using Processing, Codasign web page, <http://learning.codasign.com/index.php?title=Sending_and_Receiving_OSC_Data_Using_Processing>

[19] nRF24L01 2.4GHz How-To, Arduino Info web page, <http://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo>

[20] Scott Brave, Colyn Bulthaup and Professor Hiroshi Ishii, (1998), PSyBench, Tangible Media Group web page, <http://tangible.media.mit.edu/project/psybench/>

[21] John Underkoffler, Hiroshi Ishii, (1999), Urp: a luminous-tangible workbench for urban planning and design

[22] ReacTable web page, < http://www.reactable.com/>

[23] Vincent LeClerc, Amanda Parkes, Hiroshi Ishii, (2007), Senspectra: a computationally augmented physical modeling toolkit for sensing and visualization of structural strain

[24] Phidgets web page < http://www.phidgets.com/>

[25] Rafael Ballagas, Meredith Ringel, Maureen Stone, Jan Borchers, (2003), iStuff: A physical user interface toolkit for ubiquitous computing environments

[26] Leonel Morgado, Maria Cruz, Ken Kahn, (2006), Radia Perlman: A pioneer of young children computer programming

[27] Jacobsson, Bodin, Holmquist, (2008), The See-Puck: A Platform for Exploring Human-Robots Relationships

[28] Jacobsson, Fernaeus, Holmquist, (2008), Glowbots: Designing and Implementing Engaging Human Robot interaction

[29] Coelho, Maes, (2008), Sprout I/O: A textually rich interface

[30] Poupyrev, Nashida, Mauyama, Rekimoto, Yamaji, (2004), Lumen: interactive visual and shape display for calm computing

[31] Andrew Sears,Julie A. Jacko, (2003) Human-Computer Interaction: Design Issues, Solutions, and Applications

[32] Couture, Riviere, Reuter, (2008), GeoTUI: A Tangible User Interface for Geoscience

[33] Boriana Koleva, Steve Benford, Kher Hui Ng, Tom Rodden, (2003), A Framework for Tangible User Interfaces

[34] Hornecker, Buur, (2006), Getting a Grip on Tangible Interaction: A Framework on Physical Space and Social Interaction

[35] Michael Philetus Weller, (2008), Posey: Instrumenting a poseable hub and strut construction toy

[36] Ken Hinckley, (2002), Input Technologies and Techniques

[37] Diana Africano, (2003), Ely: the Explorer- Interactive Play system

[38] Hayes Solos Raffle, Hiroshi Ishii (2004, May), Topobo: A 3-D Constructive Assembly System with Kinetic Memory

[39] J. Patten, B. Recht, and H. Ishii, (2002), Audiopad: A tag-based interface for musical performance

[40] Ashlie Brown, Hayes Raffle, (2009, April), Opportunities for Actuated Tangible Interfaces to Improve Protein Study

[41] Tommaso Piazza, Hannes Heller, Morten Fjeld, (2009), CERMIT: Co-located and Remote Collaborative System for Emergency Response Management

[42] Rauschert, Agrawal, Sharma, Fuhrmann, Brewer, MacEachren, (2002), Designing a human-centered, multimodal GIS interface to support emergency management

Chapter 8: Appendix

Arduino Server Part:

```
 //CONNECTIONS for nRF24L01 Modules:
 // 1 - GND
 // 2 - VCC 3.3V !!! NOT 5V
 // 3 - CE to Arduino pin 9
 // 4 - CSN to Arduino pin 10
 // 5 - SCK to Arduino pin 13
 // 6 - MOSI to Arduino pin 11
 // 7 - MISO to Arduino pin 12
 // 8 - UNUSED



/*-----( Import needed libraries )-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

/*-----( Declare Constants and Pin Numbers )-----*/
#define CE_PIN   9
#define CSN_PIN 10

// NOTE: the "LL" at the end of the constant is "LongLong" type
const uint64_t pipe = 0xE8E8F0F0E1LL; // Define the transmit pipe

/*-----( Declare objects )-----*/
RF24 radio(CE_PIN, CSN_PIN); // Create a Radio

/*-----( Declare Variables )-----*/
int received; // Data received from the serial port
int received1;
int received2;
int ledPin = 13; // Set the pin to digital I/O 13
int pixel_distance;


void setup() {
  pinMode(ledPin, OUTPUT); // Set pin as OUTPUT
  Serial.begin(9600); // Start serial communication at 9600 bps

  radio.begin();       //start radio for the antenna communication
  radio.openWritingPipe(pipe);
}


void loop() {
     if (Serial.available() >1) {
          // If at least two bytes are available to read (that is, the distance value),
```

```
        received1 = Serial.read(); //read the first byte it and store it in received1
        received2 =Serial.read();  //read the second byte it and store it in received2
        received=received1*256+received2;  //reconstruct the number
        digitalWrite(ledPin, HIGH);
              //set the pin 13 as High to indicate the info received


        Serial.print ("Uno received: ");
              // At the receiving end, verify the received value
        Serial.println(received );

    }
    else {
       //Serial.println("maybe not serial available");
              //test message if there is not serial connection
    }

    pixel_distance=received;
    radio.write(&pixel_distance, sizeof(int));
          //send the distance to the car, through the Antenna RF24
}
```

Arduino Car Part:

```
 //CONNECTIONS for nRF24L01 Modules:
 // 1 - GND
 // 2 - VCC 3.3V !!! NOT 5V
 // 3 - CE to Arduino pin 9
 // 4 - CSN to Arduino pin 10
 // 5 - SCK to Arduino pin 13
 // 6 - MOSI to Arduino pin 11
 // 7 - MISO to Arduino pin 12
 // 8 - UNUSED



/*-----( Import needed libraries )-----*/
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

/*-----( Declare Constants and Pin Numbers )-----*/
#define CE_PIN   9
#define CSN_PIN 10

// NOTE: the "LL" at the end of the constant is "LongLong" type
const uint64_t pipe = 0xE8E8F0F0E1LL; // Define the transmit pipe
```

```
int STBY = 6; //standby

//Motor A
int PWMA = 3; //Speed control
int AIN1 = 2; //Direction
int AIN2 = 4; //Direction

//Motor B
int PWMB = 5; //Speed control
int BIN1 = 7; //Direction
int BIN2 = 8; //Direction

/*-----( Declare objects )-----*/
RF24 radio(CE_PIN, CSN_PIN);    // Create a Radio

/*-----( Declare Variables )-----*/
int pixel_dist=0;


void setup() {
  //Serial.begin(9600);
  //Serial.println("Nrf24L01 Receiver Starting");
  radio.begin();                  //start radio for the antenna communication
  radio.openReadingPipe(1,pipe);
  radio.startListening();
}


void loop() {

  if ( radio.available()) {   // Read the data payload until we've received everything
     bool done = false;
     if (!done) {
          done = radio.read(&pixel_dist, sizeof(int));    // Fetch the data payload
          //Serial.print("Car received distance:  ");
          //Serial.println(pixel_dist);
     }
   }
   else  {
        //Serial.println("No radio available ");
              //test message if there is not serial connection
   }

   if (pixel_dist>100) {
               //if the distance from the target is more than 100px, keep moving
         move(1,20, 0);     //motor 1, speed=20, right
         move(2,16, 1);     //motor 2, speed=16, right
   }
   else {
         stop();           //otherwise, stop
   }

}
```

```
void move(int motor, int speed, int direction){
  //Move specific motor at speed and direction
  //motor: 0 for B 1 for A
  //speed: 0 is off, and 255 is full speed
  //direction: 0 clockwise, 1 counter-clockwise

  digitalWrite(STBY, HIGH); //disable standby

  boolean inPin1 = LOW;
  boolean inPin2 = HIGH;

  if(direction == 1){
    inPin1 = HIGH;
    inPin2 = LOW;
  }

  if(motor == 1){
    digitalWrite(AIN1, inPin1);
    digitalWrite(AIN2, inPin2);
    analogWrite(PWMA, speed);
  }
  else {
    digitalWrite(BIN1, inPin1);
    digitalWrite(BIN2, inPin2);
    analogWrite(PWMB, speed);
  }
}


void stop(){
  //enable standby
  digitalWrite(STBY, LOW);
}
```

Processing Server Part:

```
//import libraries
import oscP5.*;
import netP5.*;
import processing.serial.*;

//declaration of variables
Serial myPort;
OscP5 oscP5;
NetAddress myRemoteLocation;

int counter=0;
```

```
int sendvalue;
String val;



void setup() {
   size(400,400);
   String portName = Serial.list()[3];
            //change the list number to a 1 or 2 etc. to match the right port of your PC

   myPort = new Serial(this, portName, 9600);
  //serial communication with Arduino Uno, using the Port #portName. Baud rate: 9600 bps

   myPort.bufferUntil('\n');

   oscP5 = new OscP5(this,5001);
  // initialize oscP5 object, telling it to listen for incoming messages at port 5001
  //in the meantime, in the server sketch: "myRemoteLocation = new netAddress     ("163.117.137.206",5001)"
}

void draw() {

}


void serialEvent( Serial myPort) {
   val = myPort.readStringUntil('\n');  //put the incoming data into a String -
       //the '\n' is our end delimiter indicating the //end of a complete packet

   if (val != null) {       //make sure our data isn't empty before continuing
        val = trim(val);    //trim whitespace and formatting characters (like carriage return)
        println(val);
   }
}



void oscEvent(OscMessage theOscMessage)  {
            //Whenever an OSC message is received, it is passed to the oscEvent() function.
    counter++;
   // get the first value as an integer
   String firstValue = theOscMessage.get(0).stringValue();

   // get the second value as a float
   float secondValue = theOscMessage.get(1).floatValue();


   if (counter>=2) {                    //If we have received the distance value
      println("OSC Message Received: ");       // print out the message
      println("My value is: " + secondValue);

      sendvalue= round(secondValue);
      println("I send to Uno: " + sendvalue);

      delay(25);
      myPort.write(sendvalue/256);
```

```
      myPort.write(sendvalue%256);
            //pass value to Arduino Uno, serially (split into two bytes)
  }
}

//the delay() of the Processing API is normally a bad practise, because it halts the //whole program. for this reason, this is a self-made
//function, that serves as an //alternative, mainly based on the "millis()" function
void delay(int del)
{
  int time = millis();
  while(millis() - time <= del);  //delay of "del" milliseconds
}
```

Processing Tablet Part:

```
//import libraries
import oscP5.*;
import netP5.*;

//declaration of variables
OscP5 oscP5;
NetAddress myRemoteLocation;


int [] posX=new int[2];
int [] posY=new int[2];
int counter=0;
boolean bDisplayMessage=true;
boolean check_flag=false;
float radius=100.0;

float init_distance;
float distance;
float distance_from_line;
int x1, y1, x2, y2, x3, y3;
int mouseCount = 0;
boolean been_on_first=false;
float init_time;
float final_time;
float total_time;
boolean time1_got=false;
boolean time2_got=false;


PShape s;

void setup() {

  oscP5 = new OscP5(this,5002);
          // initialize oscP5 object, telling it to listen for incoming messages at port 5002
  myRemoteLocation = new NetAddress("163.117.137.206",5001);
```

```
          // set the remote location to be the localhost on port 5001

 size(displayWidth, displayHeight);  //set the size at Full Screen mode
 textFont(createFont("Arial",30));
 reset();              //call the reset function (reset variables and background colour)

}

void draw() {

 if (bDisplayMessage==true) {      //a message appears, prompting the user to define a
    background(180, 120, 150);    //path in a few seconds
    fill(0);
    textAlign(CENTER, CENTER);
    text("Please define path in",displayWidth/2, displayHeight/2);
    text(7- millis()/1000 + " seconds", displayWidth/2,displayHeight/2+50);


    if (millis() >7000)
    {
        bDisplayMessage = false;
            // After 7 seconds, stop displaying the message, thus proceed with path definition
        background(255);
    }
 }

 if (counter==2) {       //if both initial and final point are set, draw path.
   drawPoints();
 }

 if (check_flag==true) {    //if the path has been drawn
      if (mouseOverCircle(posX[0], posY[0], radius)) {
          fill(0, 255, 0);
          been_on_first=true;          //boolean variable to guarantee that the car //has passed from the initial point in the beginning

          if (time1_got==false) {       //if the car passes by the initial point
                init_time=millis();    //save the "beginning time", for future use
                time1_got=true;
          }
      }
      else {
          fill(255, 0, 0);
      }

      ellipse(posX[0], posY[0], radius, radius);
            //make a circle around the initial point

     if   (mouseOverCircle(posX[1], posY[1], radius)) {
          //if the car passes by the final point
           if  (been_on_first==true) {
              fill(218,112,240);
              textAlign(CENTER);
              textSize(20);
              text("Target reached! ",x2, y2+90);   //announce that we reached the //target

              if (time2_got==false) {       //if the car reaches the target
```

```
                final_time=millis();     //save the time, for future use
                time2_got=true;
            }

            total_time= final_time- init_time;    //calculation of total time of //the itinerary
            fill(150,112,240);
            textAlign(CENTER);
            textSize(25);
            text("Initial distance: "+ init_distance+" px",displayWidth-150,displayHeight-100);  //display in px the total distance

            fill(150,112,240);
            textAlign(CENTER);
            textSize(25);
            text("Total time: "+ total_time +" ms",displayWidth-150,displayHeight-60);    //display in ms the total time spent
        }

      fill(0, 255, 0);
   }
   else {
      fill(255, 0, 0);
   }
  ellipse(posX[1], posY[1], radius, radius);     //make a circle around the final point

}

          //Reset Button
 if (bDisplayMessage==false) {            //if the first seven seconds have passed
    if(hover(displayWidth-200,displayHeight-200,100,50) ) {  //if the user hovers his //finger above the "reset" box
      rectMode(CORNER);                 //make the "reset" box green
      fill(120, 255, 120);
      rect(displayWidth-200,displayHeight-200,100,50);
      fill(0);
      textSize(26);
      textAlign(CENTER, CENTER);
      text("reset",displayWidth-150,displayHeight-180);
    }
    else {                          //if not, make the "reset" box red
      strokeWeight(1);
      rectMode(CORNER);
      fill(255, 120, 120);
      rect(displayWidth-200,displayHeight-200,100,50);
      fill(0);
      textSize(26);
      textAlign(CENTER, CENTER);
      text("reset",displayWidth-150,displayHeight-180);
   }
 }

}


void reset() {         //reset the variables and the background color
 bDisplayMessage=true;
 check_flag=false;
 been_on_first=false;
 mouseCount = 0;
```

```
 background(255);
 counter=0;
 time1_got=false;
 time2_got=false;
}

 void mousePressed (){

    OscMessage myMessage = new OscMessage("/test");  //create a new OscMessage
    if ((bDisplayMessage == false) && (!(hover(displayWidth/2-130, displayHeight-200, 260,200)))) {  //after the first seven seconds,
                                                                                                      //and when not hovering above the area where "distance" is displayed
on the screen
       if(hover(displayWidth-200,displayHeight-200,100,50) ){  //if the user presses //the reset button
             reset();                  //reset and start again
        }
        else {

       if (mouseCount== 0) {          //if the user presses the screen for the first //time, define the initial point
            x1 = mouseX;
            y1 = mouseY;
            fill(0);
            textAlign(CENTER);
            textSize(20);
            text("point 1 ("+x1+", "+y1+")",x1, y1+120);  //and print the related //message on the screen


            mouseCount++;
      }
      else if (mouseCount==1) {      //if the user presses the screen for the second //time, define the final point
            x2 = mouseX;
            y2 = mouseY;
            mouseCount++;

            //additional information
            fill(0);
            textAlign(CENTER);
            textSize(20);
            text("point 2 ("+x2+", "+y2+")",x2, y2+120);   //and print the related //message on the screen
            distance = calculateDist(x1, y1, x2, y2);      //calculate and store the //initial distance, for future use
            init_distance=distance;


            fill(0);
            textAlign(CENTER);
            textSize(25);
            text("distance: "+distance, displayWidth/2, displayHeight-100);
                  //print the distance

            myMessage.add("distance: ");        //add the string "distance: " to the //osc message
            myMessage.add(distance);            // add a float to the osc message

            oscP5.send(myMessage, myRemoteLocation);     // send the message

          }
           else {
                distance_from_line=  getDistance( posX[0], posY[0],posX[1], posY[1], mouseX, mouseY );
```

```
//calculate the position of the car with regards to the path defined

                if  (distance_from_line<100) {
//if the car is moving inbetween an acceptable (small) distance from the line (maximum 100px)
                    rectMode(CENTER);
//clear the previous distance-from-target appearing on the screen
                    noStroke();
                    fill(255);
                    rect(displayWidth/2, displayHeight-100, 200,40);
                    stroke(0);

                    x3 = mouseX;
                    y3 = mouseY;
                    distance = calculateDist(x3, y3, x2, y2);
//calculate the new distance-from-target of the car
                    fill(0);
                    textAlign(CENTER);
                    textSize(25);
                    text("distance: "+distance, displayWidth/2, displayHeight-100);   //print the new distance-from-target on the screen
                  }
            }

           myMessage.add("distance: "); //add the string "distance: " to the osc //message
           myMessage.add(distance); // add a float to the osc message

           oscP5.send(myMessage, myRemoteLocation);     // send the message


            if (counter<2) {   //if one of the two points (initial or final) is being defined
                ellipseMode(CENTER);
                fill(100);
                ellipse(mouseX,mouseY, radius, radius);  //create a circle around //it with a specific radius
                posX[counter]=mouseX;           //and save its coordinates
                posY[counter]=mouseY;
            }
            counter++;      //increase the touch counter
        }
    }
}


 void mouseDragged() {

       if (mouseCount==2) {  //if both initial and final point have been received

            OscMessage myMessage = new OscMessage("/test");  //create a new //OscMessage

            distance_from_line=  getDistance( posX[0], posY[0],posX[1], posY[1], mouseX, mouseY );  //calculate the position of the
car with regards to the path defined

            if   (distance_from_line<100) {     //if the car is moving inbetween an //acceptable (small) distance from the line (maximum
100px)

               rectMode(CENTER);              //clear the previous distance-from-//target appearing on the screen

               noStroke();
```

```
                fill(255);
                rect(displayWidth/2, displayHeight-100, 200,40);
                stroke(0);

                x3 = mouseX;                //calculate the new distance-from-//target of the car
                y3 = mouseY;
                distance = calculateDist(x3, y3, x2, y2);
                fill(0);
                textAlign(CENTER);
                textSize(25);
                text("distance: "+distance, displayWidth/2, displayHeight-100);  //print the new distance-from-target on the screen
              }

              myMessage.add("distance: ");  // add the string "distance: " to the osc //message
              myMessage.add(distance); // add a float to the osc message

             oscP5.send(myMessage, myRemoteLocation);  //send the message to the //laptop with WiFi

        }

}



/* the "drawPoints" function draws the straight line from point (posX[0], posY[0]) to point (posX[1], posY[1]) */
  void drawPoints() {
     beginShape(LINES);
     for (int i = 0; i < 2; i++) {
        vertex(posX[i], posY[i]);
     }
     endShape();
     check_flag=true;  //this boolean variable gets true when the line is finally drawn
  }



/* the "mouseOverCircle" function returns "True" if the mouse (or equally, the finger touching the screen)
 *is hovering above the circle that has-> Coordinates of center:(x,y), Radius: radius */

   boolean mouseOverCircle(int x, int y, float radius) {
  return (dist(mouseX, mouseY, x, y) < radius);
}



/* the "calculateDist" function returns the distance (in pixels) between the points (x1,y1), (x2,y2) */

float calculateDist(int x1, int y1, int x2, int y2) {
  float b = sq(x2-x1) + sq(y2-y1);
  float a = floor(sqrt(b));
  return a;
}



/* the "hover" function returns "true" if the mouse (or equally, the finger touching the screen)
```

```
*is hovering above the rectangle that has: Coordinates of the down-left corner (x,y), Width  w, Height  h */

boolean hover(int x, int y, int w, int h) {
    if(mouseX >= x && mouseX <= x+w && mouseY >= y && mouseY <= y+h) {
       return true;
    }
    else {
       return false;
    }
 }



/* The "getDistance" function calculates the point A on the line (x1,y1) -> (x2,y2) that is closest to the point (x,y)
 * Specifically, the result is a PVector- where result.x and result.y are the coordinates of that point A on the line.
 * The result.z variable contains the Distance from (x,y) to the line, which is finally returned (in pixels) . */

float getDistance( int x1, int y1, int  x2, int  y2, int  x, int  y ) {
   PVector result = new PVector();

   float dx = x2 - x1;
   float dy = y2 - y1;
   float d = sqrt( dx*dx + dy*dy );
   float ca = dx/d;     // cosine
   float sa = dy/d;     // sine

   float mX = (-x1+x)*ca + (-y1+y)*sa;

   if( mX <= 0 ) {
     result.x = x1;
     result.y = y1;
   }
   else if( mX >= d ) {
     result.x = x2;
     result.y = y2;
   }
   else {
     result.x = x1 + mX*ca;
     result.y = y1 + mX*sa;
   }

   dx = x - result.x;
   dy = y - result.y;
   result.z = sqrt( dx*dx + dy*dy );

   return result.z;   //the result returned is the distance from point (x,y) to the line
}


//the delay() of the Processing API is normally a bad practice, because it halts the        //whole program. For this reason, this is a self-made function, that serves as an   //alternative, mainly based on the "millis()" function

void delay(int del)
{
  int time = millis();
  while(millis() - time <= del);   }  //delay of "del" milliseconds
```