POSEiDON

PersOnalized Smart Environments to increase Inclusion of people with DOwn's syndrome

Deliverable D4.3

Interactive Table

Call:	FP7-ICT-2013-10
Objective:	ICT-2013.5.3 ICT for smart and personalised inclusion
Contractual delivery date:	31.10.2016 (M36)
Actual delivery date:	31.10.2016 (M36)
Version:	Final
Editor:	Silvia Rus (FhG)
Contributors:	Sebastian Zander-Walz (FhG)
Reviewers:	Dean Kramer (MU)
	Lars Thomas Boye (Tellu)
Dissemination level:	Public
Number of pages:	34

Contents

1 Executive Summary

The deliverable D4.3 – Interactive Table is the third deliverable of Work Package 4 - User Interaction and associated to Task 4.4 – Capacitive interaction table creating in the scope of WP4 an innovative interaction device, which is unobtrusively integrated in a regular couch table. The final version extends the previous version by rearranging some chapters and by adding Chapters 9 and 10.

After a short introduction into the topic, the role of the interactive table in the POSEIDON project will be described. In the following chapters the suitable requirements for the interactive table selected from the requirements analysis in D2.1 – Report on requirements will be presented. Subsequently the iterative development will be presented.

In Chapter 4 we present prototype I, which is the prototype which is integrated into a regular couch table. We describe setup, algorithms, interaction scheme and present the results of the performed evaluation.

In Chapter 5 we present the planning phase of the design of the mobile interactive table. The system architecture of the implemented prototype for pilot 1 is presented in Chapter 6, followed by a description of the data processing. Here we describe the development changes since the last iteration. Having as a starting point the two system architectures, we motivate the smaller set of interaction patterns available for the pilot 1 prototype. The migration challenges between the two system architectures, from a peripheral device to an alone standing interaction device are described. These are various incompatibilities of libraries. We had to finally implement them ourselves. Because of this additional workload not all interaction patterns are available. The position of the hand in the air, on the table and taps on the table can be recognized.

The application design of the Moneyhandling Training is presented in Chapter 7. We shortly describe how the concept for the Moneyhandling App has been developed and how this relates to the requirements, scenarios and personas. Chapter 8 presents the detailed description of the communication protocol, the API of the interactive table.

Chapter 9 describes interactive table prototype for pilot 2. We treat the feedback provided by D6.3 Results of Pilot 1 which address the Interactive table and the Moneyhandling application. Point by point we are discussing the suggestions and propose future improvements which are implemented in pilot 2 and the final prototype. The improvements regarding the software and the hardware of the interactive table are presented in the subsequent sections while other improvements are described in other deliverables.

In Chapter 10 we state the final status of the interactive table and point to software repositories for the interactive table.

2 Introduction

The goal of the Poseidon project is to enable people with Down Syndrome to be more autonomous and independent in their daily life. This is aimed to be achieved by developing assisting technology designed especially for the needs of people with Down Syndrome, thus driven by a user centred approach. Following this goal the interactive table is intended to be specialized for the primary users but also usable by the secondary users.

During the last decade, multi-touch tables and touch screens have become widely popular. Their growing popularity is especially driven by their user friendly nature and by a solid usability. For this type of devices, the user interaction relies on gestures on a touchscreen.

Moreover, there is also a class of touch-free devices which support free-air interaction either of the whole body or of hands and fingers. This detaches the user from the device and thus provides more open forms of interaction. Different categories of sensors, such as visible light cameras, infrared systems, or capacitive proximity sensors are used to track the position of the user's hands or fingers. A classification of this position over time allows inferring gestures.

Our proposed solution is an interactive table that uses a mix of capacitive proximity sensors for detecting the free-air gestures, while an acoustic approach using microphones is used to detect events on the surface of the table.

3 The interactive table and the POSEIDON system

3.1 Role of the interactive table in the POSEIDON system

The aim of the POSEIDON project is to create smart environments which can intelligently, and tactfully, provide support to people with Down Syndrome at different stages of daily life. The interactive table will achieve this being part of a technology infrastructure layer composed of a Virtual Reality (VR) kit and a smartphone or tablet for mobility.

The Interactive Table will allow working on a bigger surface. With the power of a computer it is possible to create material and gain experiences useful for school and for work, as well as to support entertainment. A dimension to be investigated is how it supports or complements the VR system with gestures or selections. Based on capacitive technology it allows the tracking of two hands to allow intuitive passive interaction. This auxiliary input device will allow an immersive interaction with the POSEIDON platform.

The static devices (Interactive Table and Virtual Reality Set) will be used at specific locations, for example at home, school or work, whilst the users will have access to the inclusion services everywhere and all the time through the mobile device. Notice the main users are people with Down Syndrome, but their family, school teachers, employers, bus drivers, and other people interacting with them will also be able to use the static and mobile devices with different interfaces and benefits.

3.2 Requirements of the interactive table

In D2.1 Report on requirements, the needs of the people with Down Syndrome, e.g. what they want and how they perceive the effects of the new technology for their life, have been assessed in multiple ways such as:

- Online questionnaires for secondary users to assess the requirements of people with Down Syndrome.
- Face-to-face interviews with people with Down Syndrome.
- Personas and scenarios to illustrate what impact POSEIDON might have on the life of people with Down Syndrome.

Concerning entertainment technology, the results of the requirement analysis show that more than 50 percent of the people with Down Syndrome are familiar with entertainment electronics such as CD- and DVD-players, TV, cameras, game controllers, mp3-players and radios. In the case of information technology usage, more than 50 percent of people with Down Syndrome are familiar with mobile- and smartphones and more than three quarters of them use laptops/PCs and tablets. People with DS need more assistance when using information technology devices than in the case of entertainment devices. The portability, ease of use and multi-functionality of tablets can bring completely new experiences for people with Down Syndrome.

These findings support the observations made at the first user group workshop where the participants tried out the touch and free-air interaction with tablet, Wii and GestIC (a free-air gestures device in the size of a touchpad, developed by microchip[1]). The results of this workshop are presented in D6.2 Feedback from user group workshops. These outline that the free-air interaction is no problem once the controls are simple and easy to learn and once the purpose of the application is clear. The touch

[1] http://www.microchip.com/pagehandler/en_us/technology/gestic

interaction with the tablets posed no problem as such, once the purpose of the application was understood.

In D2.1 Report on requirements, technical requirements are formulated in addition to the user requirements. The requirements have been formulated for the hardware, software and for the framework serving as a foundation. The technical requirements are classified into functional, non-functional, hardware and design requirements. The requirements for the interactive table are listed in Table 1. They cover functional and hardware requirements, which are planned to be achieved in the second iteration of the interactive table prototype. The overall priority of all requirements is high.

Table. 1 Technical requirements of the interactive table

Label	Requirement category	Requirement	Proto-type	Priority
Fun30	Functional	Interactive table should combine methods for arm tracking and touch recognition	2	7
H12	Hardware	Interactive table should allow control of applications with coarse gestures	2	5
H13	Hardware	Interactive table should be integrated in a typical living room table	2	5
H14	Hardware	Interactive table will use capacitive sensing kits, microphone for acoustic tracking and small PC for processing	2	5
H15	Hardware	Interactive table & Virtual reality kit shall use a common API	2	5

3.3 Development timeline

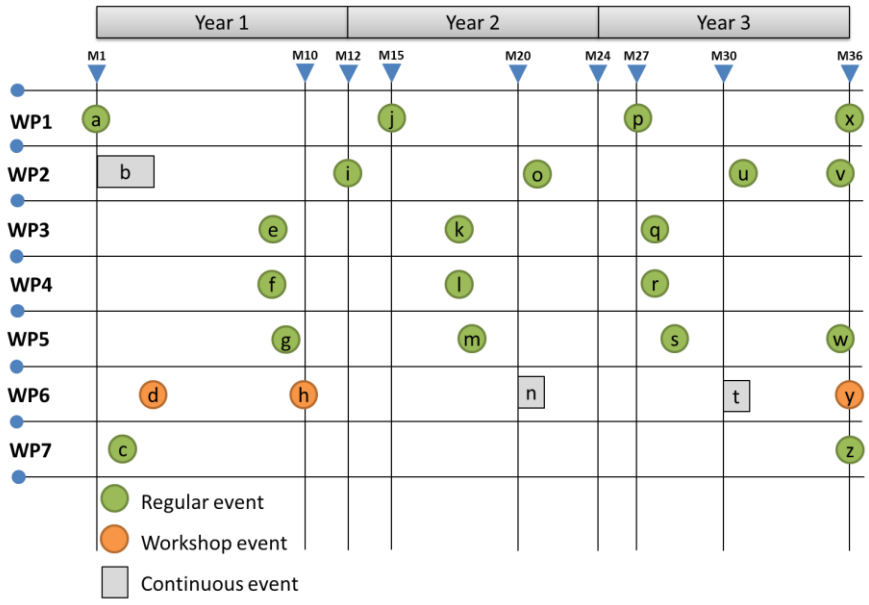

Figure 1. Project and work package milestones and events

In Figure 1 the milestones and events of the POSEIDON project for each work package are outlined. The interactive table is developed fully in WP4 and integrated in WP5. WP4 presents three major events. Event f) represents the final development point for prototype I of the interaction table. This event is followed by the second user group workshop where a first evaluation of the interactive table is planned. In event i) the results of the second user group workshop are analyzed and used as input for the further development of the interactive table. Events m) and s) represent two more iterations of the interactive table which are going to be tested in the two pilot events n) and t). Finally, the framework in which the interactive table is integrated will be published towards the end of the POSEIDON project in event w).

4 Prototype I – Interactive Table

In the following chapter the status of the development of the interaction table at the stage of the couch table prototype will be described. At first the basic working principles of capacitive proximity sensing will be explained briefly while the physical setup and the software components will be described in more detail.

4.1 Capacitive sensing methods

The most ubiquitous usage of capacitive sensing technology can be found in touch screens. These use various layers of transparent electrodes or nanowires to detect the mutual capacitance as objects enter the detection area [1]. Figure 2 shows three different projected capacitive sensing methods. The difference between these modes is given by the ability of the capacitive system to sense an object at different distances from the surface. Hence, for touch sensing the sensors are densely distributed and must be close to the surface. For the floating touch mode the sensors are as well densely distributed but have a higher sensitivity such that an object can be detected already at very short distance from the surface. In the third image of Figure 2 the proximity sensing capacitive method is shown. Here the sensors are distributed sparsely but create a stronger electric field that propagates into space in order to detect larger objects which are in proximity of the interactive surface. Achievable distances are up to 30 centimeters and the sensors may be applied below thick non-conductive material. Smartphones and tablets use the first two capacitive sensing methods for touch sensing and floating touch sensing while the interactive table we developed focuses on capacitive proximity sensing.

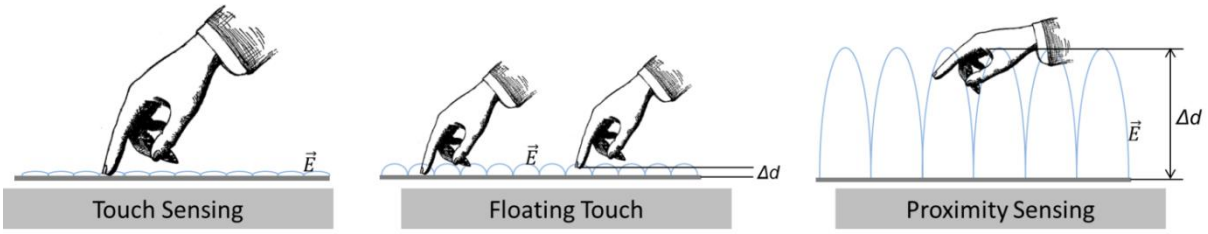

Figure 2. Different projected capacitive sensing methods based on distance

4.2 Hardware setup of couch table prototype

The interactive table has been realized by installing the components underneath the surface of a regular living room table like shown in Figure 3. All devices can be integrated into the table in a way that it is not distinguishable from the not-augmented piece of furniture. The data is visualized on a nearby TV, connected to the table using a HDMI cable.

Figure 3. View of interactive table prototype

Underneath the surface 24 copper electrodes are attached. Each of them is connected to a capacitive proximity sensor, like shown in Figure 4. For the touch event detection a microphone is used and is connected to a miniature PC, see Figure 4.

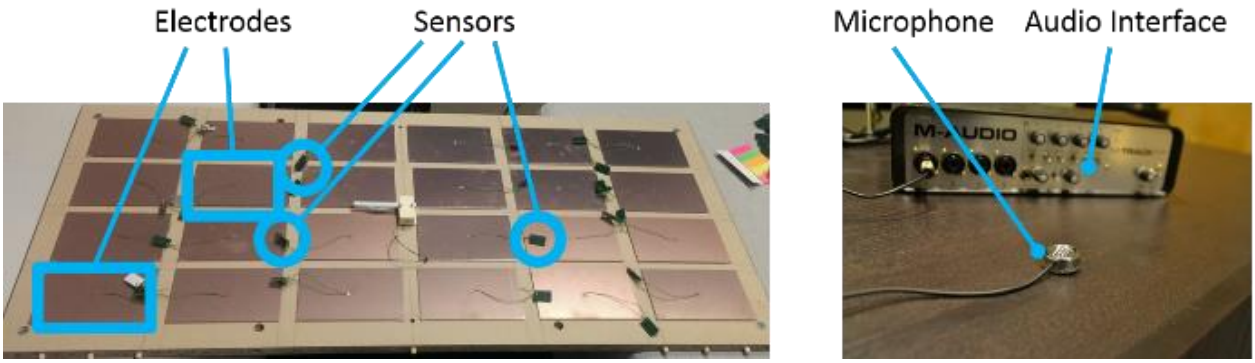

Figure 4. Detail views of the prototype system: left - electrodes and sensors, right - audio interface and contact microphone

The setup is visualized in Figure 5. The system is comprised of 24 electrode & sensor pairs that are connected to three different measurement boards, OpenCapSense boards [2]. The microphone is attached to a USB Audio Interface. Overall there are four different boards connected via USB to a PC that executes and merges the different types of data processing and links it to the software suite.

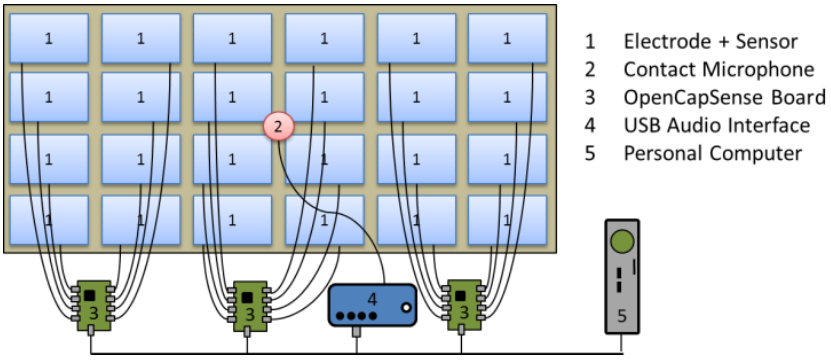

Figure 5. Abstract view below the surface of the prototype including capacitive sensing electrodes and touch detection microphone

The debug software was developed with C# using the .NET 4.5 framework. We are using the Emgu CV library based on OpenCV for image processing and application of the Kalman filter to the determined palm locations. The sound processing is implemented in C++ and Java using a modified version of ChucK for audio sampling and the WEKA framework to apply the machine learning on top. We are using sockets to transfer data between the different modules. The debug application allows a fine control of the various processing steps in both image and audio signal processing.

4.3 User interaction

There are several limitations of free-air gesture interaction systems, related to user fatigue [3][4]. Additionally, the achievable resolution of the area above the surface of the capacitive proximity sensors is considerably lower than on the plane. A combination of these two aspects, an interaction pattern that includes a tactile sensation for improving selection events in graphical user interfaces and a sparse quantization of the interaction space above the surface seems viable. Thus a layer-based

interaction for capacitive proximity sensor devices has been chosen. This consists of a touch layer that can be used to register several different types of touch events and three different distance layers based on the proximity of the hand.

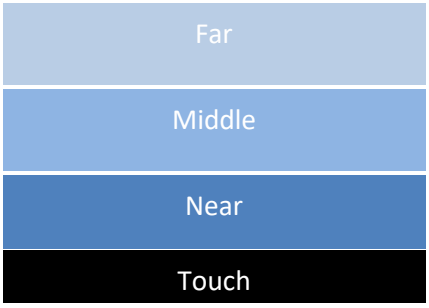

Figure 6. Interaction layers

The different layers in Figure 6 are intended for touch and free-air interaction. The touch layer at the bottom of the figure represents the interaction on the surface of the table. The layers on top are formed by dividing the area above the table in three equally large parts – the near layer, the middle layer and the far layer. Inside each layer a set of different gestures can be executed and recognized while inter-layer changes may trigger additional events. Different interaction types are used for the touch and the free-air gesture layers. Swiping and dwelling gestures are suited for the free-air gesture layers while at the touch layer tapping, double tapping and different swiping patterns can be used. These interaction types may have at each layer a different functionality.

4.3.1 Touch detection

The microphone placed in the center of the interactive table is used to detect touch events on the surface of the table. In Figure 7 the touch events and the according Fast Fourier Transform (FFT) are shown. One can see that the features of the FFT of the knock and finger tap events strongly differ from the ones of the finger and hand swipe events. Impact events show low-frequency peaks on the left and swipe events show constant values on the right. To differentiate between these events two classifiers are used.

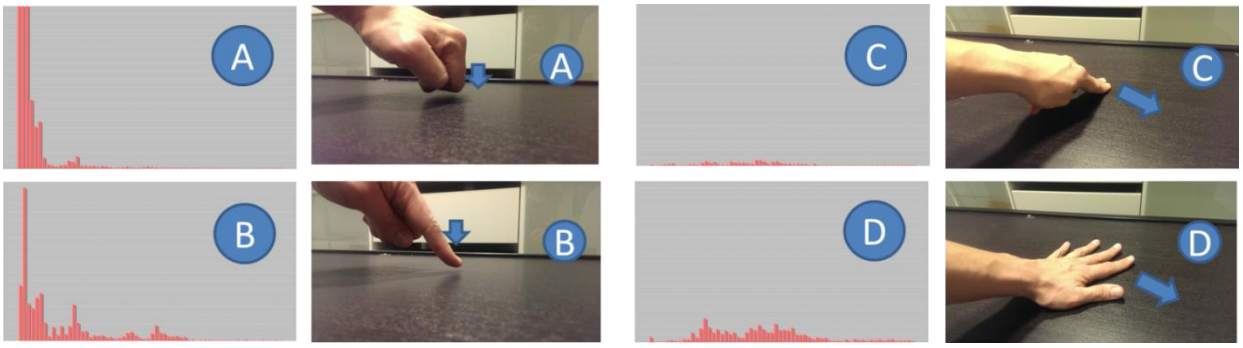

Figure 7. FFTs and photo for a knock event (A), a finger tap (B), a finger swipe (C) and a hand swipe (D)

4.3.2 Free-air object recognition

The sensor data of the 6x4 electrode array can be displayed as a 24 pixel grayscale image, shown in Figure 8. An image from this sensor data can be reconstructed using a selection of image processing methods that allows to successfully detect location and orientation of one or more arms. The process is distinguished into four distinct steps:

1. Creating a grayscale image from the acquired sensor data

Figure 8. Grayscale image from sensor data

2. Apply a feature-preserving image upscaling method

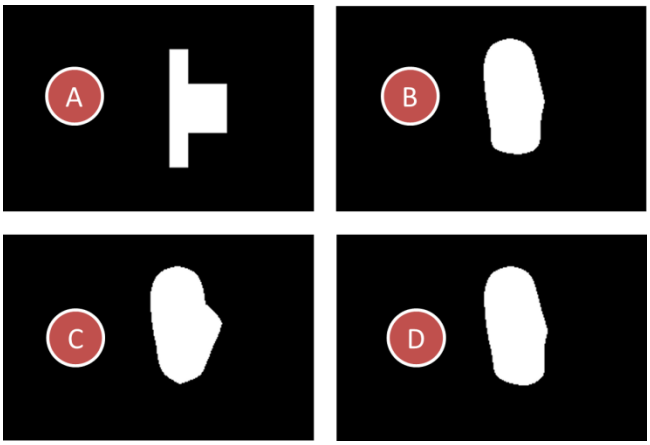

Figure 9. Effect of different upscaling methods on shape

3. Find the contours of the present objects according to pixel values
4. Analyze the image moments of the contour areas and fit human arms

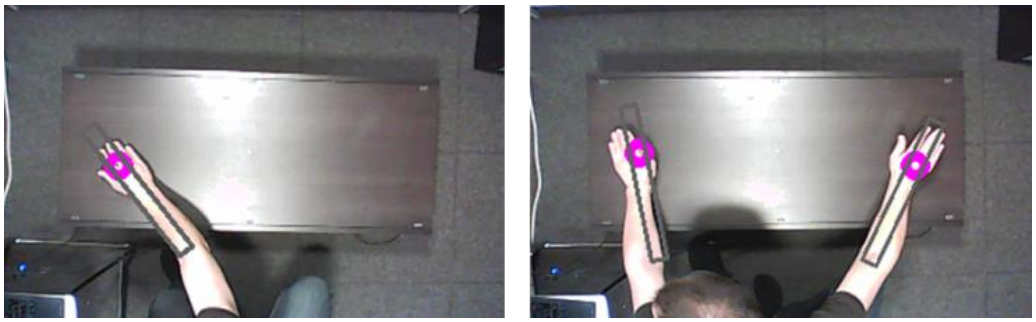

Figure 10. Overhead camera picture of the scene overlaid with live arm and palm reconstruction for one arm (left) and two arms (right)

4.4 Image viewer demonstration application

By using an image viewer, the different interaction types and the according functions can be shown. In Figure 11 the different views of the image viewer are depicted. In the top left image of the figure the grid view of the image viewer is displayed. This view can be reached by either entering the far layer of the interaction area or by changing from the middle to the far layer. At this upper level only the swiping function is available which allows scrolling through all thumbnails. By getting closer to the surface of the table the middle interaction layer is activated. The change from the far to the middle layer leads to the view presented in the top right image of Figure 11 where one enters the zoomed grid view.

At the middle layer the swiping gesture and according functionality is available. Once the near layer of the interaction layers is reached the user can select a picture for full view by selecting it via cursor and dwelling. We can also make use of touch events, e.g. to launch zoomed views or present tooltip

information as shown in the bottom of Figure 11. Additional touch and swipe gestures can be used to further manipulate the images.

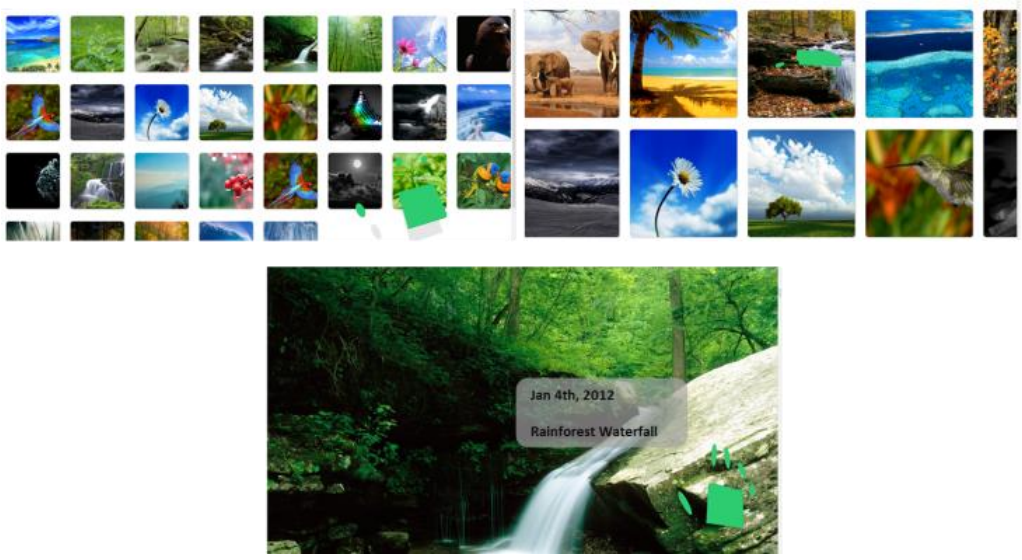

Figure 11. Different views of image viewer. Top left - far layer pan through many thumbnails. Top right - middle layer pan through more detailed thumbnails. Bottom - near layer shows detailed picture with tooltip on tap.

4.5 Performed evaluation

In order to evaluate the interactive table a combined study by 10 users who were invited to test the accuracy of the touch detection and benchmark the interaction speed has been performed. They predominantly had plenty of experience with touch devices. Experience with gesture interaction systems like the Kinect or Leap Motion was less prevalent.

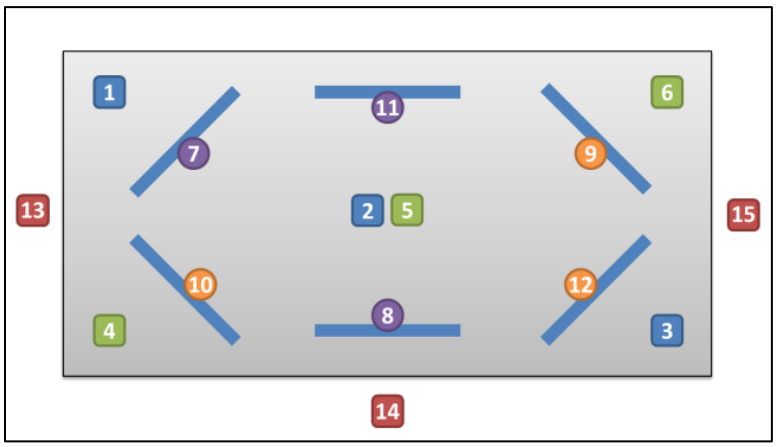

Figure 12. Finger tap (blue), knuckle knock (green), finger swipe (purple), hand swipe (orange) and stomping (red) spots relative to tabletop.

4.5.1 Touch detection accuracy

One of the most interesting aspects of the evaluation of the interactive table is the accuracy of the touch detection. The accuracy is reached using a classification that is only trained by a limited number of users. Six different types of touch events are tested by the different users - finger tap, double finger tap, knuckle knock, double knuckle knock, finger swipe and hand swipe. To get an idea if outside influences can disturb the signal, the users stomp at three different locations in close proximity of the

table. Overall there are 12 different areas on the table that have to be touched in different ways by the users. These are executed three times each. The results are shown in Table 2.

Table 2. Results of touch detection for single and double taps (SFT, DFT), knocks (SKK, DKK), finger swipe (FS), hand swipe (HS) and stomp (STO). Noted are the overall samples, errors, no event errors, wrong classification errors and the percentage of correct classification.

	SFT	DFT	SKK	DKK	FS	HS	STO
Samples	90	90	90	90	90	90	90
Errors	1	3	17	36	18	2	22
No event	1	0	0	0	6	0	0
Wrong Class	0	3	17	36	12	2	22
Percentage correct class	98,89	96,67	81,11	60,00	80,00	97,78	75,56

The system was very well capable of recognizing the different taps having a success rate of 96% or more. The results were not as good for knock detection, with only 81% correct classification of single knocks and 60% correct classification of double knocks. However, it should be noted that there was a high variance in results.

4.5.2 Hand tracking and interaction speed

In order to properly identify gestures the recognized paths of the hands are the most important measure. We have added a visualization module to the registered camera image introduced previously that allows us to show the tracks followed by one or more hands. Figure 13 shows several of the paths that were generated this way using single hand tracking. The hand is in this case moving about 10-15 cm above the table surface. While we have not connected the system to a generic path-based gesture recognition system, we can see that the system can create smooth trajectories that can be analyzed further.

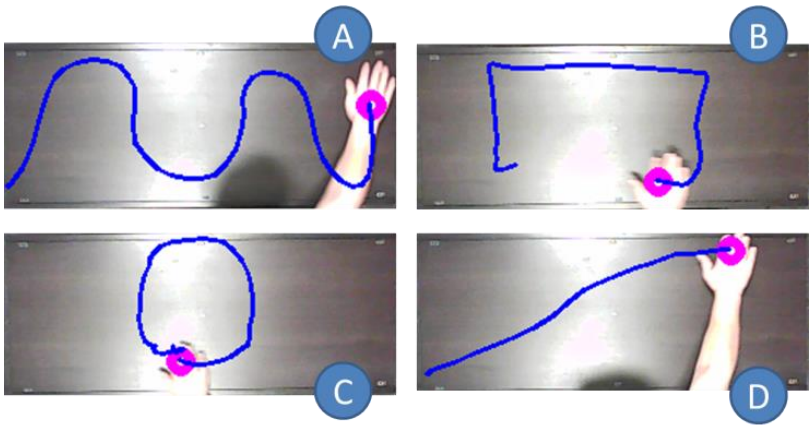

Figure 13. Tracks generated by Kalman filtered palm position, a sine wave (A), a rectangle (B), a circle (C), and a diagonal swipe (D)

A major point of interest for us was to check if users without Down Syndrome could successfully use the layer interaction pattern introduced before and if the option for adding unobtrusive touch detection would have any influence on the interaction speed. For this we used a small game whereas the participants had to put a cursor into a box and perform either a dwell (approximately 300ms) or touch activity for selection. The cursor reflected the current interaction layer by color coding. We counted the time it took to complete a run of 15 boxes.

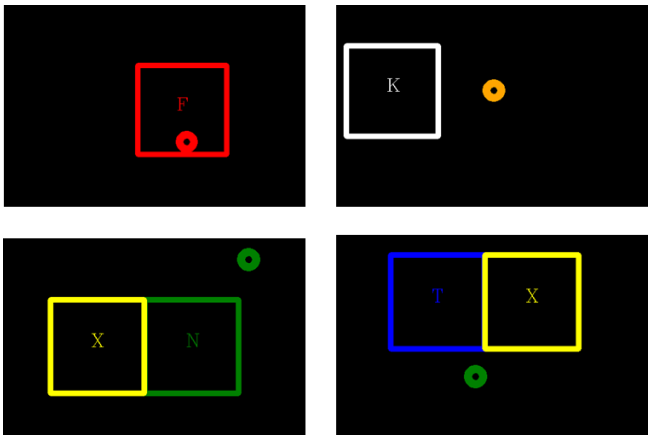

Figure 14. Interaction speed evaluation. Different types of boxes for near layer (N), knock (K), far layer (F), and disturber (X).

Some example boxes are shown in Figure 14. There were six different types referring to dwelling in the three different layers, knock, tap and disturber. Each participant performed four runs. First a training run, then a run with all interaction layers and the two touch events, then a run with dwelling boxes only (without layers), and finally a run with tap boxes only. The dwell and tap runs also included some disturber boxes to slightly increase the challenge. The order of the runs was switched equally between the different participants. Regarding the full interaction run we wanted to know if the user understands the different layers and can achieve a good interaction speed. The main hypothesis we wanted to test was that tapping improves the interaction speed as opposed to dwelling and may reduce the overall interaction times, thus reducing potential fatigue. We expected the tap run to be shorter than the dwell run as selection events can be performed faster.

Table 3. Results for interaction time in the different runs of the interaction speed test

	Full run	Dwell run	Tap run
Average Time	40.12	37.97	33.42
Shortest Run	27.28	32.20	24.29
Longest Run	51.38	48.11	47.72
Standard Deviation	7.22	5.27	7.93

The results are shown in Table 3. Running a paired t-test on dwell and tap run the resulting p value is 0.0071, indicating a high statistical significance that the interaction using taps is faster than the interaction using dwelling. This fits expectations from literature [4]. While this can be countered by reducing the dwell time, the risk of wrongful selection of nearby objects increases significantly.

4.6 Development plan for secondary user workshop and pilot 1

The pilot 1 development phase started after the second end user workshop and lasted until month 18. This month marked the beginning of the first pilot at the end users homes. Possible developments in this next phase were:

- Developing a more easily transportable version of the interactive table.
- Designing applications for the interaction pattern of the table. None of the previous applications used the orientation of the arm.
- The touch tracking showed some weaknesses when used by new persons that were not in the training set. Methods will be investigated that will allow quickly configuring and training the

table for new users. Additionally the robustness of the used method can be increased, e.g. by adding a second microphone that should enable to account for non-uniformity of the surfaces.

- Improvements to better cope with noise caused by unintended body parts in detection range of the interactive table. It is common that the knees are recognized by the table and are recognized as an additional hand interacting with the table. Stopping the recognition of knees, is something that can be detected based on data. It could be compensated by using an extended set of image processing algorithms.

In addition to these points, for the second user group workshop we planned two applications connected to the interactive table. First, we connected Google Earth to the interactive table and used the free-air interaction to navigate to a place. The second application was an obstacle avoiding game.

5 Prototype II - Mobile interactive table

Since the interactive table, also called CapTap, had to be evaluated and used in the user group workshops as well as in the pilot trials, a mobile version of the prototype interactive table has been designed and was ready for the second user group workshop, see Figure 15 and 16.

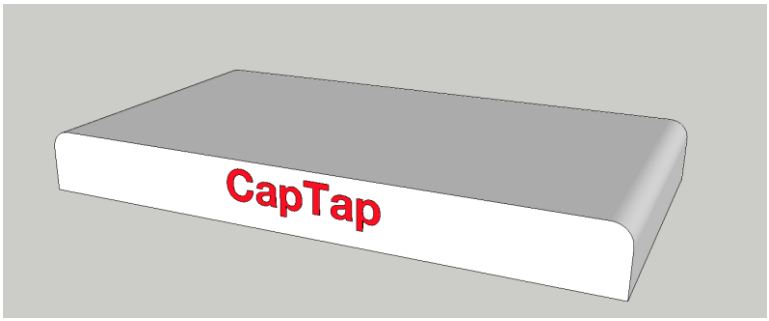

Figure 15. Overview of the design of the mobile interactive table

The mobile version of the interactive table is intended to be placed on top of a surface. It is intended to be smaller (90cm x 50cm) than the implementation of the interactive table in the regular living room couch table (115cm x 60cm). In Figure 16 one can see the 4x6 array of electrodes which will be attached underneath the lid, the three OpenCapSense Boards and the miniPC. In the middle of the electrode array there is a circular space carved out where the microphone will be placed. The connections between the components are not shown for simplicity reasons.

Once the mobile interactive table is placed on a surface, it needs to be plugged in into a power supply and it can be connected through USB to a PC.

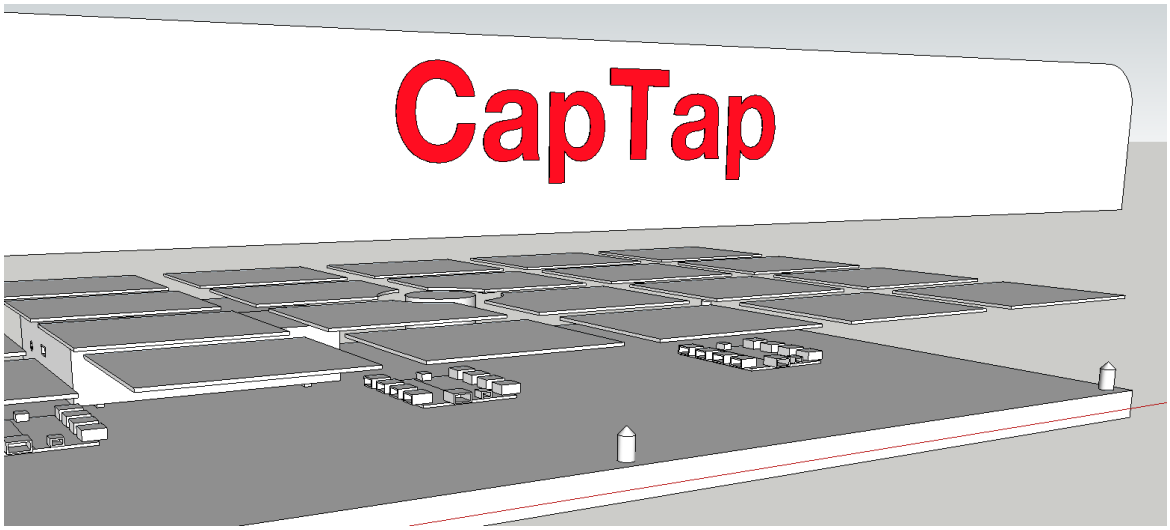

Figure 16. Side view of inner setup of mobile interactive table – electrodes, OpenCapSense boards, miniPC

After this hardware planning phase we have implemented the design in hardware. The according building process is described in deliverable D5.2 – Prototypic systems, Interactive table chapter.

6 Integrated prototype for pilot 1

The main concern in the development of the interactive table for pilot 1 was to further develop the prototype towards a plug and play interaction device, a device built-in into our environment. This means to be able to connect the device to the home network, being triggered from diverse applications. The application running on the laptop or PC would this way be able to get the input through the local network.

6.1 System architecture for second primary user workshop

The system architecture of the first prototype tried out in Mainz at the second primary user workshop looked as shown in Figure 17. It shows the interactive table in front of a monitor. The monitor is connected to the used PC or laptop. The interactive table has a power plug for the active USB hub inside the table and the USB cable which connects the interactive table to the PC or laptop. Through this connection the raw measured data is being transferred to the application. There the data is being processed. Positions of the hand and the sound sensed by the microphone are directly transferred to the PC/laptop. Here the data processing and classification for the microphone is done, as well as the deduction of the gestures from the hand positions detected by the capacitive system. This system has a central processing point, the PC/laptop, while the interactive table is merely a peripheral device. On the central processing point the data processing and the gesture recognition are built into the application. This means that each application would have to compute its own gestures.

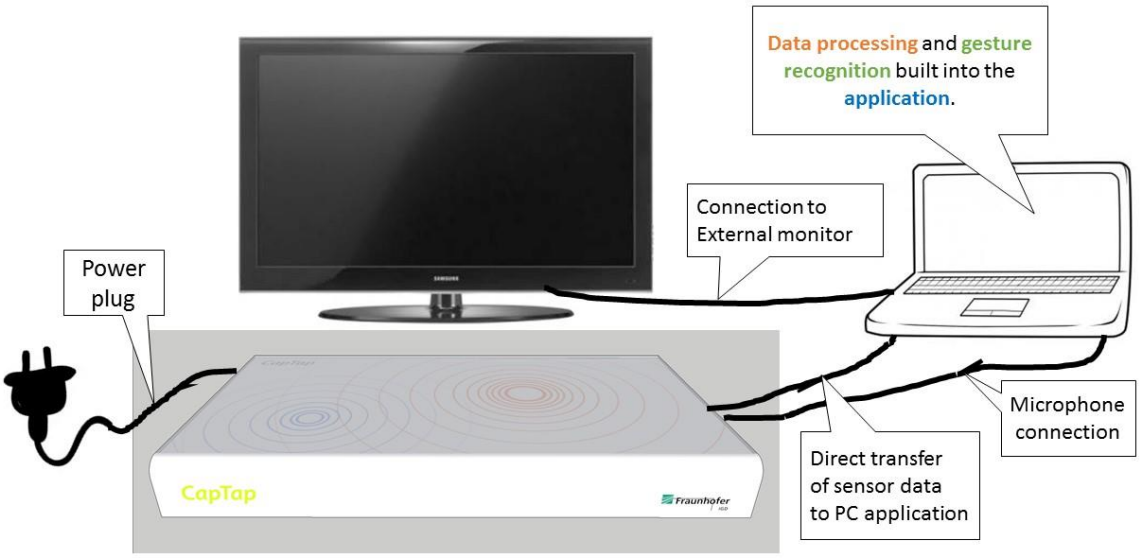

Figure 17 – System architecture of Prototype 1 for user group workshop in Mainz

6.2 System architecture for pilot 1

The aim of the developments for the pilot 1 prototype was to separate this direct connection. Ideally, the data processing and the gesture recognition part would be executed on the table. Through a communication protocol, the application would request what it needs. If it needs the position of a hand, it would request to send these, or directly request to use a given set of gestures, which then would only be sent once recognized.

The system architecture of the prototype for pilot 1 is shown in Figure 18. The interactive table is placed in front of a monitor, which is connected to the laptop/PC. Between laptop/PC and interactive table there is no longer a direct connection. The interactive table has to be connected to local network through an Ethernet connection, and plugged in to a power plug. This power plug and the Ethernet connection are connected inside the table to a mini PC. This mini PC is now receiving the sensor data and microphone data and executes the required processing. It sends the calculated hand position and recognized gestures through the local internet connection to the application on the laptop/PC.

On the laptop/PC runs the actual application. The gesture recognition for the capacitive sensing has to be implemented here, if needed. This means, that if one would like to recognize a swipe in the air, from left to the right, then the received data of the hand position has to be tracked and processed in such a way, that such a gesture would be recognized.

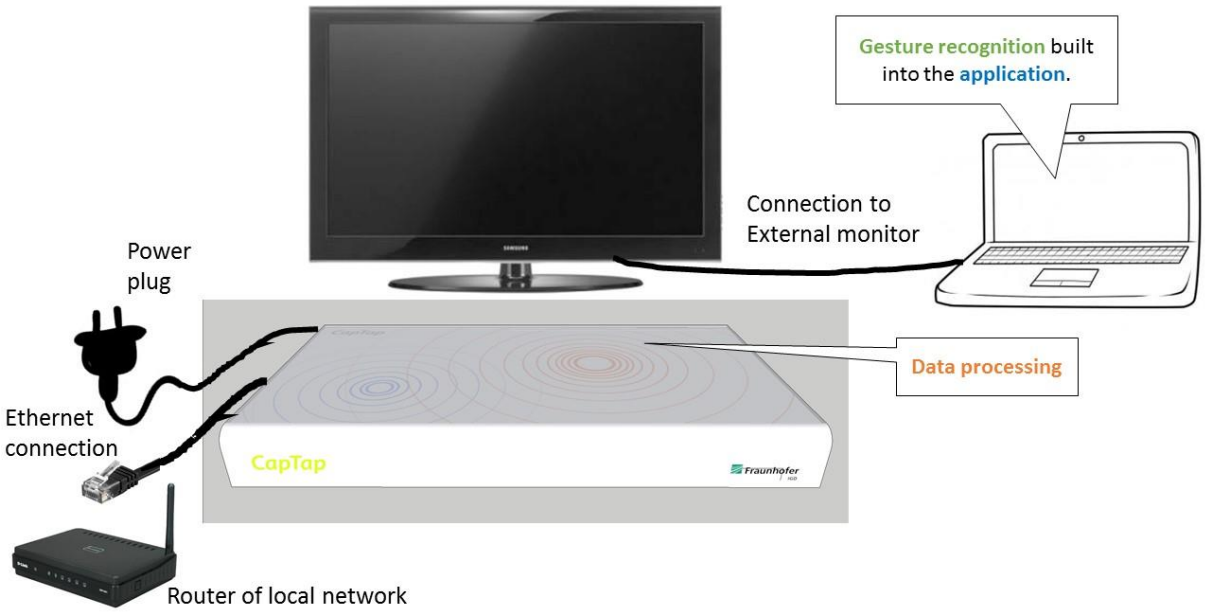

Figure 18 – System architecture of prototype for pilot 1

Taking a look inside the table, like shown in Figure 19, we can see that the mini PC is the central component of the system. It collects the data from the evaluation boards and the microphone.

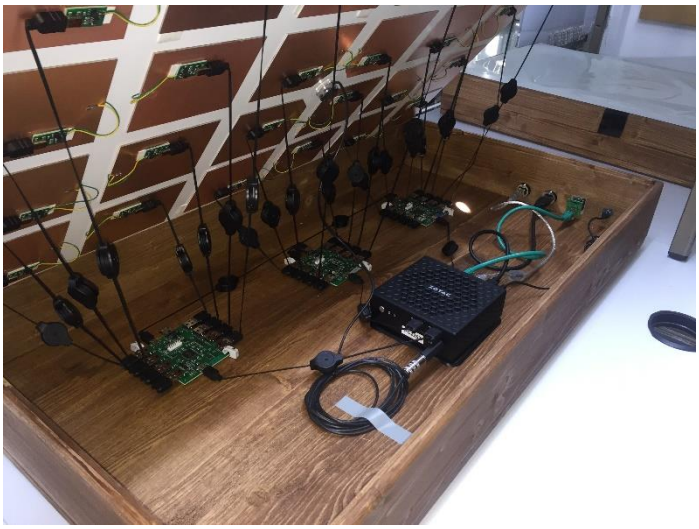

Figure 19 – inner components of the interactive table - electrodes, sensors, evaluation boards, microphone, mini PC and the connection to the outer part of the table

In Figure 19 a different version of the mobile interactive table is shown, than in the images up to now. We have built seven additional prototypes of this form for the pilot 1 trials. Details regarding the building process were published on the POSEIDON website and are included in D5.2 - Prototypic systems on the Poseidon concept chapter 7. Details on how to start the interactive table and how to check the basic functionality are described in D4.5 - HCI user and developer manuals.

6.3 Data processing software of the interactive table

Like mentioned in the previous section, the prototype for pilot 1 has a more restricted set of interaction capabilities like the prototype for the second user workshop. The reason for this development is explained in the following. However, it is important to mention, that for the further development building the previous features in is still an important goal.

First of all, the choice for this specific mini PC, ZOTAC ZBOX CI320 NANO PLUS, depended on two factors: the audio touch recognition requires an actively amplified microphone interface. The Zotac nano plus supports this active amplification on a separate input connector for microphones. Additionally, it needed to have enough USB ports, so that the additional USB hub could be avoided. Once we found out, that the microphone is supported, and that it has 6 USB ports, this was the suitable choice.

Most embedded computer systems in the world run on Linux systems nowadays. The final version of the CapTap should be an integrated system that connects to other systems as an input device. Therefore, we decided to migrate the existing software to a Linux based systems and equipped our mini PCs with a recent Ubuntu distribution (Ubuntu 14.04.2 LTS).

During the migration we observed the following problems. Some of them have been resolved, some of them have not been resolved yet.

The result is a java application with dependencies of different libraries. Most important, the OpenCV library including the java bindings, which is used to compute the data from the sensors using image processing techniques such as the Kalman-filter. However, the Kalman-filter did not work in combination with the java-bindings. Another solution had to be found, finally leading to

implementing and setting the right parameters of the Kalman-filter using another library, the JKalman.

ChucK is a programming tool for building complex audio synthesis programs. It has been used in the previous prototype. ZeroMQ (ZMQ) should have been the library which assures the communication with ChucK via TCP/IP. Unfortunately it was not possible to embed it, receiving errors from the compiler. Furthermore, the version of ChucK embedded in the Ubuntu distribution was to old and could not calculate fast fourier transforms. So a newer version of ChucK had to be compiled, however in none of the versions worked the output using TCP/IP. In the end, we had to implement the functionality on our own.

For the audio processing, due to the several problems with incompatibilities already mentioned, the functionality had to be reduced to a simple tap detection. It currently works using a threshold which can be set by the application. The microphone input of the different mini PCs are affected differently by noise. Some of them are very noisy. Due to this, the sound signal cannot be directly filtered, first the data has to be sampled, windowed, then a fast fourier transform calculated, highpass filtered in the frequency spectrum and from this result the Root Mean Square (RMS) calculated. The detection is done by using a timeout and thresholding the calculated output. The threshold level can be set by the application and can be adapted to the hardware. The detection is robust, however, when there is too much noise, very soft interactions like a swipe on the surface of the table are not recognized. Double taps could be recognized by the application, looking at the temporal distribution of single detected taps. As an upside to the rewriting of the code, the performance has improved.

7 Moneyhandling application for interactive table

To show the capabilities of the interactive table and to cover an area of needed support mentioned in D2.1 - Report on requirements, money handling, we have developed a money handling training application. Its goal is to train the users to get to know money and to pay using the right coins. Figure 20 is extracted from D2.1- Report on requirements and shows the results of the questionnaires regarding money handling. These results and the description of personas and scenarios like Jennifer (She struggles to understand money and shopping) guide us in the further development of the application.

Here we mention just a small part from a scenario in D2.1, where Jennifer is involved.

"Jennifer is not confident with money and numbers. Her parents buy almost everything for her and not very experienced with handling money. She brings her own lunch that she makes with her mother to college, which means she misses out when her friends go to the cafeteria and buy lunches together. When her mum gives her money for a snack, she tends to drop it or forget to take it with her. She is also too shy to go shopping with her friends at college at weekends, which she would really like to do, because she does not want her friends to know she struggles with prices and counting.

Jennifer needs help getting used to handling her own money and going shopping. Whilst her parents will play a key role in supporting this, gradually taking her through how to pay for things and shop more independently, the app could reinforce this support. Jennifer could use prompts to remember to bring money with her to buy food. If she buys particular items at the cafeteria, the app could remind her how much these cost and what coins she needs to use to pay for them. She could also use the app to help her step by step with shopping, reminding her to go to the till and what notes and coins mean. The app should be visual, since Jennifer finds number intimidating."

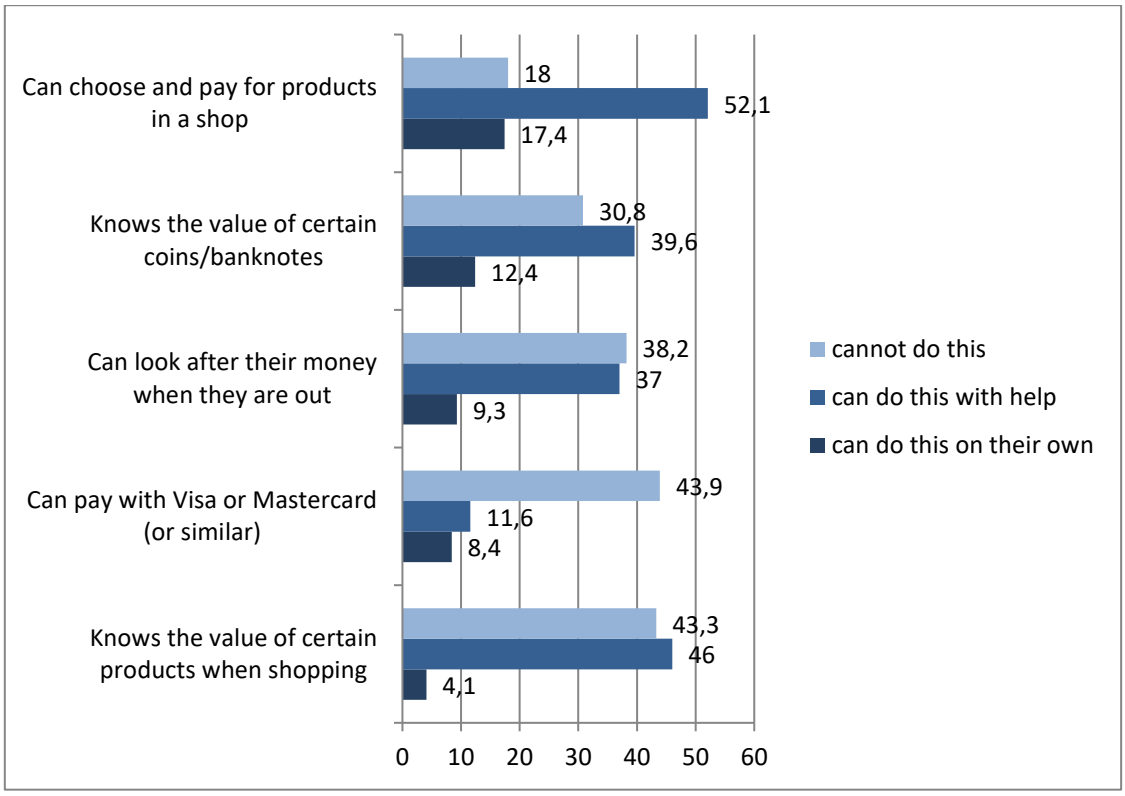

Figure 20 - From D2.1 Everyday life competencies; handling money

The current Moneyhandling App is described in more detail in D4.5 – HCI user and developer manuals, chapters 5 and 11. D4.1 – Interface strategy describes the design of the application in Chapter 9.3, 10.7 and 10.8. An overview of the functionality is given in D5.2 - Prototypic systems. However, the general concept has been developed having the learning phases in mind for people with learning disabilities. An extract of the learning phases for handling money is described in **Feil! Fant ikke referansekilden.**4.1 Interface strategy, Chapter 9.3.

Aiming to use the capabilities of the CapTap, we had the idea of using real money on the surface of the table. The first paper prototypes are also found in **Feil! Fant ikke referansekilden.**4.1 Interface strategy, Chapter 9.3. However, as a first step we are going to create an overlay where the coins and bills are displayed.

Looking at Jennifer's needs, the table covers the need to reinforce knowing the coins and bills and adding them up to pay the exact right amount. Having the chance to train this at home creates the opportunity to train it alone, without the pressure in the real situation.

8 Communication protocol: Interactive table and PC application

In order to use any application with the interactive table, the application has to understand what the table is sending as input as well as use simple commands to set parameters of the table. In the following we describe the communication protocol for application and interactive table.

The CapTap is connected to the laptop/PC application through the connection to the local network. In Chapter 8.1 we attach the documentation for this communication protocol.

The laptop/PC opens a TCP/IP socket to the predefined port 1234. Through this connection the application sends commands and requests to the interactive table. Here, an exemplary command to set the threshold:

Command:	**SET_THRESHOLD**
Description:	Sets the threshold value, used to calculate the user's hand positions.
Syntax:	[SET_THRESHOLD, thresholdValue]\n
Example:	[SET_THRESHOLD, 10]\n

The first line depicts the name of the command which functionality is described in the second line. A general syntax definition is given in line three with an according example in line 4.

8.1 How to communicate with CapTap server software

There are two ways to communicate with the interactive table server:

- Commands
- data telegrams.

8.1.1 Commands syntax

Commands are used to control the server's behavior. They are sent as Strings of the following pattern: *[CommandName ,arg1, arg2,....]\n*. The arguments may be optional, depending on the command. In the following we are presenting the syntax of all the available commands: SETPORTS, START, STOP, RESET, EN_SEND_IMG, EN_SEND_GEST, SET_THRESHOLD, SET_ERODE, GET_PORTS, GET_MAX,SET_AUDIOTHRESH, SET_AUDIOTIMEOUT.

Command:	**SETPORTS**
Description:	Causes the server to set the serial port names to the names given in the argument list.
Syntax:	[SETPORTS,leftPortName,middlePortName,rightPortName]\n
Example:	[SETPORTS,/dev/ttyUSB0,/dev/ttyUSB1,/dev/ttyUSB2]\n

Command:	**START**
Description:	Causes the server to start listening to the serial ports, requires SETPORTS first!
Syntax:	[START]\n
Example:	[START]\n

Command:	**STOP**
Description:	Causes the server to stop listening to the serial ports.
Syntax:	[STOP]\n
Example:	[STOP]\n

Command:	**RESET**
Description:	Causes a reset of the current sensor values.
Syntax:	[RESET]\n
Example:	[RESET]\n

Command:	**EN_SEND_IMG**
Description:	Enables/disables sending the image visualizing the sensor values and processed images.
Syntax:	[EN_SEND_IMG,{true/false}]\n
Example:	[EN_SEND_IMG,true]\n

Command:	**EN_SEND_GEST**
Description:	Enables/disables sending the found gestures.
Syntax:	[EN_SEND_GEST,{true/false}]\n
Example:	[EN_SEND_GEST, false]\n

Command:	**SET_THRESHOLD**
Description:	Sets the threshold value, used to calculate the user's hand positions.
Syntax:	[SET_THRESHOLD,thresholdValue]\n
Example:	[SET_THRESHOLD,10]\n

Command:	SET_ERODE
Description:	Sets the erode value, used to calculate the user's hand positions.
Syntax:	[SET_ERODE, erodeValue]\n
Example:	[SET_ERODE,5]\n

Command:	**GET_PORTS**
Description:	Causes the server to send a data telegram of the type PORTS (see Section 8.1.2)
Syntax:	[GET_PORTS]\n
Example:	[GET_PORTS]\n

Command:	**GET_MAX**
Description:	Causes the server to send a data telegram of the type MAX (see Section 8.1.2)
Syntax:	[GET_MAX]\n
Example:	[GET_MAX]\n

Command:	**SET_AUDIOTHRESH**
Description:	Sets the RMS Threshold for knock detection.
Syntax:	[SET_AUDIOTHRESH, thresholdValue]\n
Example:	[SET_AUDIOTHRESH, 20000]\n

Command:	**SET_AUDIOTIMEOUT**
Description:	Sets the timeout between two knocks in [ms].
Syntax:	[SET_AUDIOTIMEOUT, timems]\n
Example:	[SET_AUDIOTIMEOUT, 400]\n

8.1.2 Data telegram syntax

Data telegrams are used to transfer data, which serves as input to the application. They are sent as Strings of the following pattern: *[TelegramName ,arg1, arg2,....]\n*. The arguments may be optional, depending on the telegram type. In the following we are presenting the syntax of all the available telegrams: POS, IMG, GEST, PORTS, ERROR, MAX, KNOCK.

Telegram:	**POS**
Description:	Contains the positions of the user's hands. If a hand is not detected, the values are -1. Valid hand positions are greater than zero, with a maximum at xmax (ymax).
Syntax:	[POS,leftX,leftY,leftZ,rightX,rightY,rightZ]\n
Example:	[POS,25,250,1,-1,-1,-1]\n

Telegram:	**IMG**
Description:	Telegram containing the visualization image. The size is the image size in bytes, the "\n" is followed by size bytes, containing the image data.
Syntax:	[IMG,size(bin)]\nbin
Example:	[img,360000]\nbyte[360000]

Telegram:	**GEST**
Description:	Contains a recognized gesture.
Syntax:	[GEST,{SWIPE_LEFT, SWIPE_RIGHT, SWIPE_UP, SWIPE_DOWN}]\n
Example:	[GEST,SWIPE_UP]\n

Telegram:	**PORTS**
Description:	Contains a list of available serial port names.
Syntax:	[PORTS,port1,port2,...]\n
Example:	[PORTS,/dev/ttyUSB0,/dev/ttyUSB1,/dev/ttyUSB2]\n

Telegram:	**ERROR**
Description:	Something went wrong.
Syntax:	[ERROR,message]\n
Example:	[ERROR,Couldn't open port]\n

Telegram:	**MAX**
Description:	Contains the maximum x and y value, (coordinates on the table).

Syntax:	[MAX,xmax,ymax]\n
Example:	[MAX,400,300]\n

Telegram:	**KNOCK**
Description:	Indicates a detected knock and the corresponding rms (root mean square) value.
Syntax:	[KNOCK,rmsValue]\n
Example:	[KNOCK,43000.231651]\n

9 Integrated prototype for pilot 2

9.1 Feedback from pilot 1

D6. 3 Results of Pilot 1 presents the main points of feedback brought to us. We list those items which concern the interactive table and address how we planned to implement these for pilot 2 and the final prototype:

1. It is hard to use and reach the buttons at the other end of the table. If one gets too close to the table the buttons don't respond anymore – false reaction. Sometimes the users think that it is their fault and get frustrated by the application.
   → Implement algorithms supporting forearms touching the table, but activating the intended buttons.
2. The CapTap is big and bulky, and too big for some of the PUs rooms.
   → The table is intended to built-in into furniture and is intended to be bigger than usual interaction devices. The size of the table can be adjusted by re-building the table. However, there is a limit to which size makes sense.
3. There should be a "getting-money-back" function to increase the potential of learning successes
   → This is connected to the difficulty level to which the game was intended. For some pilot users the game exceeded their level and for others it was already too easy. Nonetheless, most could see the benefit in training money. For the future iteration we will offer the option to get the change displayed.
4. It would be good to have a colour grouping of the money on the overlay
   → We will address this and design a new layout in cooperation with the DSAs.
5. Technical problems when initializing the CapTap, which was sometimes tricky and led to usage problems.
   → The software start-up on the table should work reliably. We intend to test this and to write some very good user manual and create a good instruction video.
6. To increase the transfer of knowledge, the money needs to have the correct size. The arrangement of coins and bills on the surface of the CapTap was mentioned as being not perfect more than once.
   → The size of the coins on the overlay will be in relation to each other.
7. The table did not always do what was expected. Sometimes the money did not appear when knocking on the money motive.
   → The reason for this was that for many PUs it was quite difficult to reach the buttons at the outer top of the table. Trying to reach them, most of the users got to close to the table and activated the front area with their bodies. We address this issue by sensing the closeness of the body and add an additional command, signal which will alert the user to step back.
8. But a red hand is too much negative feedback for some of the PUs. "too much" and "too little" are concepts which not all people with Down's syndrome can understand. There also should be no negative feedback when the PU paid too much. Instead they could be told: «You paid more than necessary, guess how much money you get back.»
   → Adapt feedback messages.
9. A shopping basket could be added in addition

→ We are planning to be able to adjust the shopping basket with own products and use them as training items one by one.

10. Therefore more levels have to be implemented. Maybe it would be possible to integrate also a subtracting option in order to make the game more advanced. Displaying the value of the money chosen in written, not only with pictures of coins and bills would also be an advantage.
    → We are aiming to implement the option to show the sum of the selected money. This leads to different difficulty levels of the training application.
11. The control with gestures is not always simple
    → We will add possibilities to choose between the preferred interaction model (hovering, knocking, touching).

9.2 Development for pilot 2

We can group the feedback from pilot 1 into different categories. We address the different categories in the following subsections and chapters of other deliverables:

- Algorithmic improvements of the interactive table: Feedback 1, 11, 7 is addressed in Chapter 9.3
- Improvement of the Moneyhandling training software interface: Feedback 3, 8, 9, 10 is addressed in D4.1 – Interface strategy, Chapter 10.7, 10.8 and D4.5 HCI user and developer manuals, Chapter 10
- Improvement of Moneyhandling training overlay: Feedback 4, 6 is addressed in D4.1 – Interface strategy, Chapter 10.7
- Hardware implementation of interactive table: 2, 5 - in Chapter 9.4

9.3 Algorithmic improvements of the interactive table

The CapTap firmware is responsible for handling the sensors and serving the calculations done on the sensor-data. It is able to serve different kinds of data:

- The position of a left and a right hand.
- The sensor image
- If a body was detected

For simplicity, a user is assumed at the lower side of the CapTap which is using up to two hands to interact with the CapTap. When the user leans against the CapTap or moves its body too close, this interferes with the hand signals. To prevent detecting the body as a hand, the CapTap tries to guess if a body moved too close to it and serves this data then to the clients. How body detection works is explained in the section Body-Detection.

Hand detection is done by thresholding over the z-axis in the data. Recently a fixed threshold has been used, but this was dropped in favor of the Re-Thresholding technique (Explained in its section).

Aside from these two major improvements, the computational performance and the reliability has been improved.

9.3.1 Body-Detection

For simplicity, a user is assumed at the lower side of the CapTap which is using up to two hands to interact with the CapTap. When the user leans against the CapTap or moves its body too close, this interferes with the hand signals. To prevent detecting the body as a hand, the CapTap tries to guess if a body moved too close to it and serves this data to the clients.

Previously an area has been chosen which was assumed as the source of body signals (On the lower side of the CapTap, fairly high). Everything in this cubic-like area was assumed as a body signal.

Another technique which was used was the time-variance. When an object stayed at the same location for a long time it was assumed as more likely to be a body.

In recent development the body-detection area was shaped differently. It depends now on a function which determines the edge between a body signal and a normal signal. Also the time-variance is used to influence the location and size of the body area.

The function is given below, where *var* is the time-variance.

$$t = \frac{x - 175}{100} + \frac{50 - y}{255 * var * 2}$$

With this the body threshold is adapted more interactively, which lead to much better results in the body-detection performance.

9.3.2 Re-Thresholding

Hand detection is done by thresholding over the z-axis in the data. Recently a fixed threshold has been used, but this was dropped in favor of the Re-Thresholding technique.

The capacitive sensors usually sense small values when the CapTap is empty. To eliminate this noise the data has been thresholded. Also the thresholding reduces the complexity of the data by assuming some data-points as irrelevant in the pre-processing step and therefore reduces the complexity of latter problems.

The advantage of thresholding is, that it is computationally very effective. The disadvantage of losing data doesn't play out too badly because of the constraints we chose. On the one hand, we are trying to detect a body and are able to remove its influence. On the other hand we're searching for up to two hands. Therefore we are only interested in the (up to) two highest, distinct peaks in the data. This also resolves the problem of the influence of the arms in the data.

By choosing a good threshold, optimally, two blobs appear in the data. By assuming that these two blobs actually resemble the searched hands, we can calculate their center and get an estimation for the location of the hand.

Simple thresholding by choosing a fixed value is constrained to one scenario. It might work very well for knocking and tapping at the CapTap, but not for in flight recognition, because of the set threshold. By choosing more optimal values, depending on the situation, latter computation steps can perform much better.

Optimally, thresholding would choose the threshold in a way, that only the exact contour of the relevant points is left (In our case the hands).

An advanced Re-Thresholding technique could also use the information of previous data-frames to easily compute a good threshold.

Instead in the CapTap a simple improvement has been implemented, which plays out very well. Instead of using one simple threshold two thresholds are used. Because thresholding is so cheap, and the advantage is so high, it doesn't matter that it is sometimes performed two times.

One threshold is chosen very high, so it filters direct contact to the CapTap. This catches most of the surface-interaction with the CapTap. When the result of this thresholding is empty, a higher threshold is chosen which detects the remaining interaction cases.

In lab testing the difference between normal thresholding and re-thresholding wasn't very high, but in practice, where body signals and arms low below the surface are common problems this improved the performance notably.

9.4 Improvements of hardware implementation of interactive table

The feedback points from pilot 1 address the size of the table and the difficulty to start the interactive table. The size of the table could be improved by making the whole table surface small. That wold imply the total rework of the table. Thus this improvement was not feasible for pilot 2. The bulkiness of the table was another point where improvements where whished. This could be solved by integrating the electronics from inside the table and the electrodes underneath the surface of the table into a regular couch table. This point was also not addressed, since for the sake of the pilot a decision in the consortium was taken to have mobile prototypes, not real tables. The background for this decision was that many pilot participants didn't want to change their furniture or didn't have space for a new piece of furniture.

Regarding the improvement of starting the interactive table, most problems were triggered by the instability of the electronic components inside the table. They were very susceptible to power fluctuations and have not been tested in long-term trials. We identified that on the used evaluation boards, the OpenCapSense boards, there is a chip, an FTDI chip, which connects the boards to the internal PC in the CapTap. It is responsible for transferring the data. It was crashing in non-deterministic manner, hence the problems in pilot 1. For improving this, a redesign of the electrical components would be needed, triggering a total rework of the table.

One other difficulty which made the start of the interactive table with the Moneyhandling Training difficult was connecting the table to the PC, creating a local network. For some home-PCs of the pilot this meant to disconnect them from their local network. For laptops with WiFi we could still access the internet. Changing the system architecture slightly, by attaching an antenna to the CapTap-PC, the interactive table is now able to communicate wirelessly through the local network with the primary user's PC, thus improving the setup and usage process. Figure 21 shows the adapted system architecture.

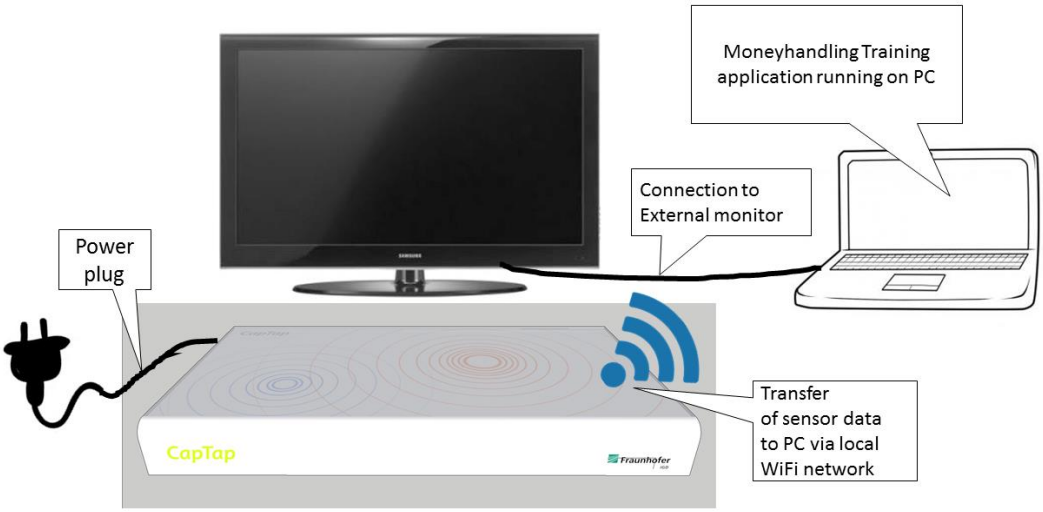

Figure 21 - System architecture for pilot 2

10 Final integrated prototype

The final version of the interactive table is the same as the one after the implementation of the feedback from pilot 1. More efforts were used to integrate the software with the other components of the POSEIDON system and to improve the application. As suggested in the review further work has been re-routed towards additional applications like the Shopping App and the Healthy Eating app.

The final versions of the software for the CapTap and the Moneyhandling Training application for the interactive table are available on the POSEIDON Website:

http://www.poseidon-project.org/developers/code/

11 References

[1] G. Barrett and R. Omote, “Projected-Capacitive Touch Technology,” *Inf. Disp. (1975).*, pp. 16–21, 2010.

[2] T. Grosse-Puppendahl, Y. Berghoefer, A. Braun, R. Wimmer, and A. Kuijper, “OpenCapSense : A Rapid Prototyping Toolkit for Pervasive Interaction Using Capacitive Sensing,” in *IEEE International Conference on Pervasive Computing and Communications*, 2013.

[3] S. Lenman, L. Bretzner, and B. Thuresson, “Using marking menus to develop command sets for computer vision based hand gesture interfaces,” *Proc. Second Nord. Conf. Human-computer Interact. - Nord. ’02*, p. 239, 2002.

[4] L. Sambrooks and B. Wilkinson, “Comparison of gestural, touch and mouse interaction with Fitts’ law,” *Proc. 25th Aust. Comput. Interact. Conf. Augment. Appl. Innov. Collab.*, no. 1992, pp. 119–122, 2013.