D2.4 - Software package with multimodal interface to kernel of the user-machine-interface

Project acronym:	ALIAS
Project name:	Adaptable Ambient Living Assistant
Strategic objective:	ICT based solutions for Advancement of Social Interaction of Elderly People
Project number:	AAL-2009-2-049
Project duration:	July, 1st 2010 – June, 30th 2013 (36months)
Coordinator:	Prof. Dr. Frank Wallhoff
Partners:	Technische Universität München Technische Universität Ilmenau Metralabs GmbH Cognesys GmbH EURECOM Guger Technologies Fraunhofer Gesellschaft pme Familienservice YOUSE GmbH

D2.4

Version:	1.00
Date:	2013-03-22
Author:	FhG Sven Fischer
Dissemination status:	PU

This project is co-funded by the Ambient Assisted Living (AAL) Joint Programme, by the German BMBF, the French ANR, and the Austrian BMVIT.

Once completed please email to alias-mgt@aal-alias.eu
with a copy to the WP leader.

D2.4	Abstract

The ALIAS robot features several different ways of user interaction. The most obvious method of controlling the robot is via its touch-screen and the graphical user interface (GUI). In addition, there are speech commands via the robot's automated speech recognition (ASR) system and the brain computer interface (BCI). The robot may provide feedback by using its display unit and/or its speech synthesis module.

The combined inputs via all of these user interfaces are joined, interpreted and controlled by the dialogue manger (DM) that represents the kernel of the user-machine-interfaces. The main focus of this report is on describing the user interaction modules and interfaces to the DM.

The report begins with a short overview of the robot's user interaction modalities. It continues with the robot modules, mainly responsible for user interaction (e.g. DM, ASR, and GUI), describing the interfaces between them. Thereafter an overview of some of the robot's function is presented, focussing on multimodal user interactions and internal interfaces between the previously described robot modules.

Dissemination Level of this deliverable (*Source: Alias DoW p21 - p23*)	
PU	Public
Nature of this deliverable (*Source: Alias DoW p21 - p23*)	
P&R	Prototype & Report

Due date of deliverable	2012/02/28
Actual submission date	2012/03/22

Authorisation			
No.	**Action**	**Name/ Company**	**Date**
1	Prepared	Sven Fischer / FHG	2013/03/07
2	Approved by 1st reviewer	Michael Pasdziernik / COG	2013/03/11
3	Approved by 2nd reviewer	Larissa Pieper / PME	2013/03/21
4	Released	Sven Fischer / FHG	2013/03/22

Contents

1 Introduction

The ALIAS robot features several different ways of user interaction. The most obvious method of controlling the robot is via its touch-screen and the graphical user interface (GUI). In addition there are speech commands via the robot's automated speech recognition (ASR) system and the brain computer interface (BCI). The robot may provide feedback by using its display unit and/or its speech synthesis module.

The combined inputs via all of these user interfaces are joined, interpreted and controlled by the dialogue manger (DM) that represents the kernel of the user-machine-interfaces. The main focus of this report is on describing the GUI and ASR components and interfaces to the DM. (A closer view on the DM prototype is provided by Deliverable D3.4, whereas the BCI is the topic of Deliverable D5.1.)

The report begins with a short overview of the robot's user interaction modalities. It continues with the robot modules, mainly responsible for user interaction (e.g. DM, ASR, and GUI), describing the interfaces between them. Thereafter an overview of some of the robot's function is presented, focussing on multimodal user interactions and internal interfaces between the previously described robot modules.

2 User Interaction Modalities

The ALIAS system handles various kinds of user interaction; most obvious the graphical user interface running on its display unit. Additionally the robot may also interact by means of verbal commands or simulating human behaviour, e.g. by looking at its user. In case the user is severely handicapped (e.g. a paralysed stroke patient), the robot may also be controlled by means of a brain-computer-Interface.

This section provides an overview to the robot's user interaction modalities, while focusing on the general working concepts in a "what does it do?"-like perspective. Some of these modules will be reviewed in section 3, again in a more technical way, describing the internal interfaces, structure, and implementation details.

2.1 Touch-Screen

The most obvious way of interacting with the robot is by using its touch-screen. The standard single input touch-screen works just like a public information retrieval terminal, ticket machine, or ATM. The user may navigate through the robots menus and access functions by tapping on the buttons on the screen, cf. Figure 1.

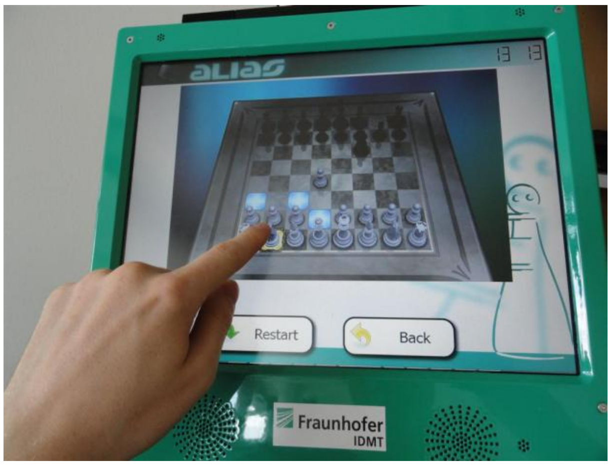


Figure 1: The graphical user interface can be controlled by touch input via the display unit.

If deemed necessary, the robot may also control the GUI on its own, e.g. initiate an emergency call and use the screen to notify the user of its intentions, displaying an abort-button to cancel the process, cf. Figure 2. A close view on the graphical user interface software can be found in section 3.4.

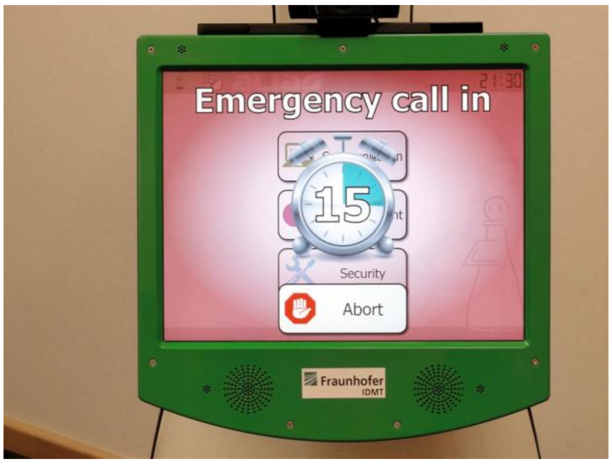


Figure 2: The ALIAS system uses the display unit to notify the user about an eminent emergency call.

2.2 Speech

By means of the integrated ASR system, the robot is able to receive and recognize verbal commands. The user's speech will be recorded by the robot's microphone (cf. Figure 3) and processed by the ASR system, thus converted from acoustical domain to a textual representation that will be interpreted by the DM.

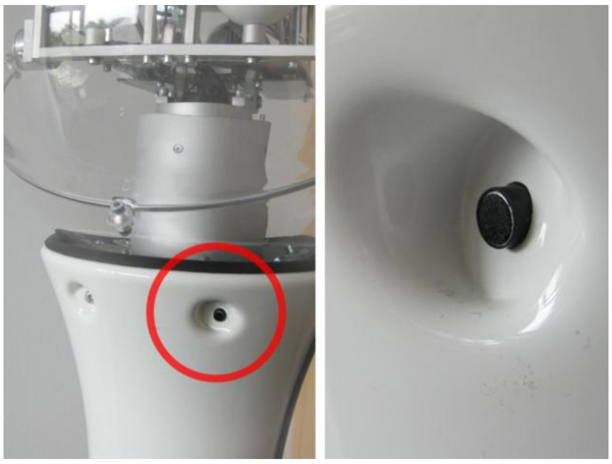


Figure 3: The robot's revised microphone, located at its neck.

The DM is also able to utilize the robot's text-to-speech (TTS) module to intonate verbal responses. Using this speech synthesis feature; the robot is able to engage in verbal user interactions. More detailed descriptions on the ASR and TTS modules can be found in sections 3.2 and 3.3, respectively.

2.3 Brain-Computer-Interface

The brain-computer-interface is a very specialized user interaction modality for users with severe physical handicap. While the robot’s other input modalities require at least some degree of movement, the BCI can also be used by persons who are paralysed, e.g. stroke patients. It operates by putting electrodes on the users head and measuring the brain activity patterns. Figure 4 depicts the brain cap, housing several electrodes. Using such a cap reduces the hassle of putting all electrodes in the correct place and order. (A more detailed explanation of the BCI system is provided by Deliverable D5.1.)

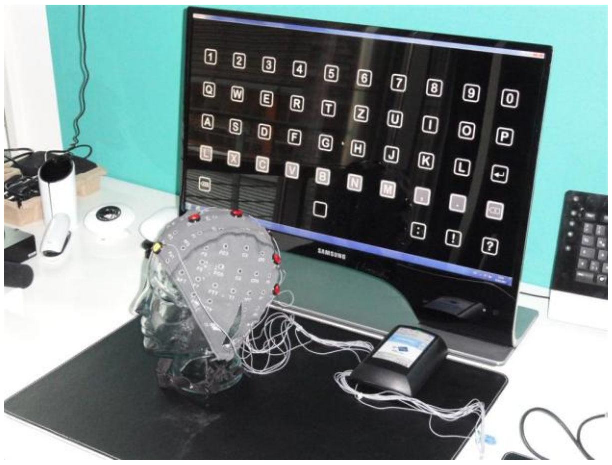
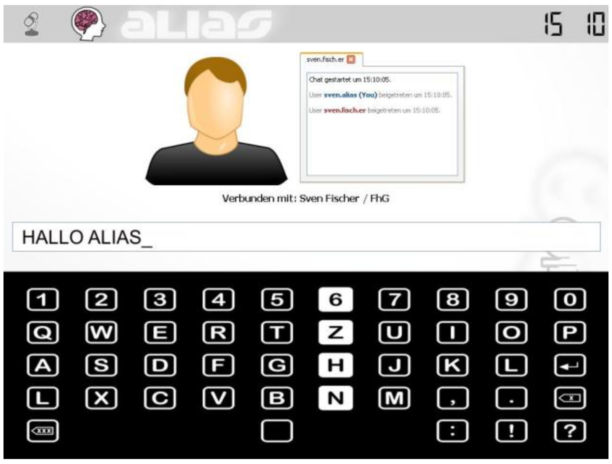


Figure 4: In case the user is unable to use speech or touch inputs, the robot can also be controlled by means of a brain-computer-interface.

There are several paradigms of utilising BCI systems. The BCI system, used with the ALIAS system is the so-called P300-paradigm. It is centred on detection of the P300 event related potential that is a relatively stable indicator to account for brain activity. The P300 is a pronounced shift in the electric brain potentials that occurs 250 to 500 milliseconds after a certain stimulus has been presented. So the P300 roughly relates to a latency of 300 milliseconds.

In order to be used with the ALIAS robot visual stimuli are provided using the robot's display unit or another auxiliary device like specially mounted LEDs. In case of ALIAS these stimuli are menus and buttons, flashing on the screen. The user will be instructed to focus on the button to be selected, and count the number of flashes, hence generating the cognitive processing that can be detected. The Depending on the temporal relations between provided visual stimuli and measured brain activity; a corresponding user input is generated.



sven.fisch.er

Chat gestartet um 15:10:05.

User **sven.alias (You)** beigetreten um 15:10:05.

User **sven.fisch.er** beigetreten um 15:10:05.

Verbunden mit: Sven Fischer / FhG

HALLO ALIAS_

Figure 5: A mock-up sketch of the robot's chat functionality.[1]

An exemplary mock-up sketch of the BCI, integrated GUI and its chat (i.e. telephone) module is presented in Figure 5.

[1] This is still a work in progress, so there is no final design, yet. Hence this figure is still a mock-up.

2.4 Other Interaction Modalities

There are several additional, more autonomous user interaction modalities that are part of the ALIAS system. Mostly they cover very specialized forms of user-interaction or supplement other modules. These modalities will be presented in this section.

2.4.1 Wii Remote Control

The robot features a Nintendo Wii gaming console that can be controlled by means of a special Wii Remote control, cf. Figure 6. The Wii system utilises several different technologies and enables the user to play games by physical movements of the Wii Remote control, e.g. playing a Tennis game by actually taking a swing with the remote control. The Nintendo Wii has a strong focus on games that encourage physical activities and motivate players to stay fit and healthy. Thus the Wii gaming system is especially popular with people who previously had no interest in playing computer games, like seniors.

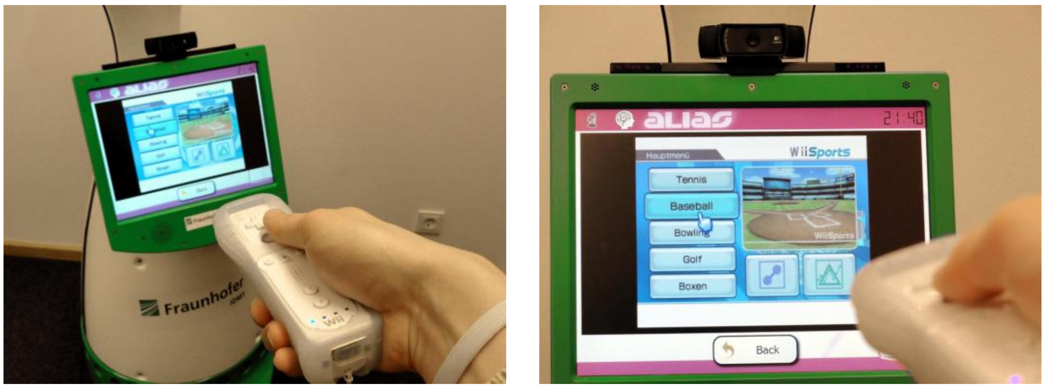


Figure 6: The integrated Wii gaming console module is operated by its proprietary Wii Remote controller. Actual physical movements are reflected in cursor movements on the screen (left.) The Wii Remote control requires a clear line of sight to the sensor array on top of the screen (right.)

The Wii Remote controller system is basically a proprietary third party technology, thus it cannot be fully integrated with the ALIAS robot system. Hence its use is limited to the Wii gaming module only.

2.4.2 Person Detection / Identification

Person detection is relevant in order to coordinate some of the remaining robot functionalities, e.g. robot navigation or eye orientation. The ALIAS system integrates several functions that may be used for detecting and identifying persons in its close proximity.

One way of detecting persons is by finding leg-pairs using the robot's laser scanner. The scanner is able to measure the distance of obstacles within a 270° angle, centred on the robot's front. The angular resolution is 0.5° and measurements could be made up to 30 meters with an error tolerance of 5mm. Such a scan results in a series of distance measurements corresponding to a horizontal cross section of the robot's environment. The scan data will then be clustered into contiguous segments, i.e. separate objects. For every segment additional analysis and classification will be performed. In case two segments in close proximity to each other are both recognized as legs, a person has been detected. However, while this method is very reliable in detecting actual persons, there is also a high probability of mistaking other leg-like objects (e.g. table legs, chairs, or window frames) for a person.

Another method for detecting persons is to use the omni-directional camera that is mounted on top of the robot's head. This camera provides a 360° image of the robot's surroundings. After preprocessing the image, a cascaded classification is applied to detect faces. If found, the location of the detected face may be joined with the laser-scanner approach to get achieve a more robust, accurate person detection. Furthermore, results of the face detection system are forward to a face identification module. By comparing detected faces to a database of known persons (i.e. their faces), the robot may identify its user and locate.

An alternate way of identifying persons is represented by the speaker identification module. In this case the speech commands given to the robot will be used to match the speaker's distinct voice patterns to a database of known persons (i.e. their voices). If the voice pattern is found in the database, the speaker can be identified.

The different techniques, as mentioned above, complement one another and can be merged to a single combined person detection and location system. However, data fusion has to consider properties and limitations of individual techniques under various conditions. For example, the face identification may not be as reliable in low light conditions, while speaker identification doesn't work for persons who remain silent. More detailed descriptions of the person detection and identification modules are presented in Deliverable D3.7.

2.4.3 Robotic Head

The ALIAS robot platform is equipped with a robotic head that may be used to simulate emotional responses or mimic human behaviour. It may be turned by 360° horizontally, and tilted 15° up or 6° down. Its eyes may be turned sideways and its eyelids may be closed (cf. Figure 7). In addition the LED array on top of its eyes may be configured to flash in various patterns.

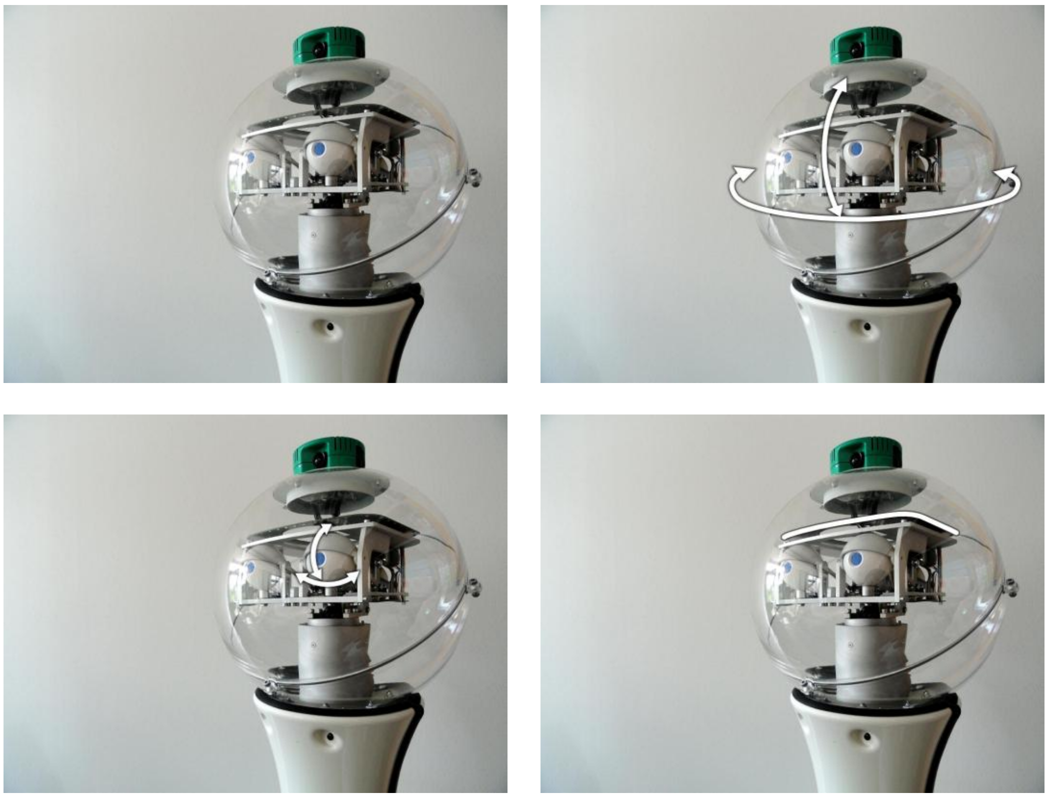


Figure 7: The robotic head (upper left) can be rotated and tilted (upper right). Its eyes can turn sideways and its eyelids can be closed (lower left). On top its eyes an array of LEDs (lower right) may be switched on or made blink in various styles.

As for the current prototype, the robotic head will turn and look at faces, once they are detected. In case the navigation module has stopped the robot's movements, to avoid a possible collision, the LED array will start blinking as long the robot is waiting until resuming execution of its previous command. A detailed description of the various emotional responses is presented in Deliverable D3.7.

2.4.4 Navigation

Some functions, like guiding the user to the bathroom, require the robot to move from one location to a certain target position. The robot is able to move on its own, so it may approach its user if it is called or if something else is happening, e.g. an incoming phone call. It may turn in order for its display unit to face its user. In any case, the movements are initiated by the DM, while the actual driving, path finding or avoidance of obstacles is executed autonomously by the navigation module.

The navigation module is located on the embedded Linux PC it has direct access to most of the robot's core systems and hardware modules, like the laser scanner, the ultra sonic-sensors, and the fish-eye camera. All of which are evaluated to estimate the robots current location on a pre-defined map, detect obstacles, and plot a suitable course to the desired target location.

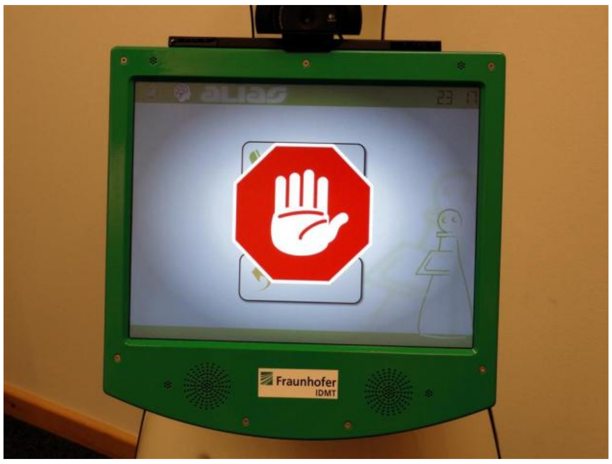


Figure 8: In case the robot is moving, the touch screen unit acts as an emergency stop button.

This, too, is a way of user interaction. In addition, movements of the robot are always accompanied by displaying an emergency stop button on the touch screen, cf. Figure 8. In case something goes wrong the robot can be stopped immediately by tapping on its screen.

3 Software Modules with Interfaces to the Dialogue Manager

This section describes technical details and interface of the software components that are mainly responsible for the user interaction. It covers some modules that have already been mentioned in section 2, though with a more technical focus on the actual implementation and internal interfaces.

Figure 9 provides an overview of the ALIAS system and connections between its separate modules.

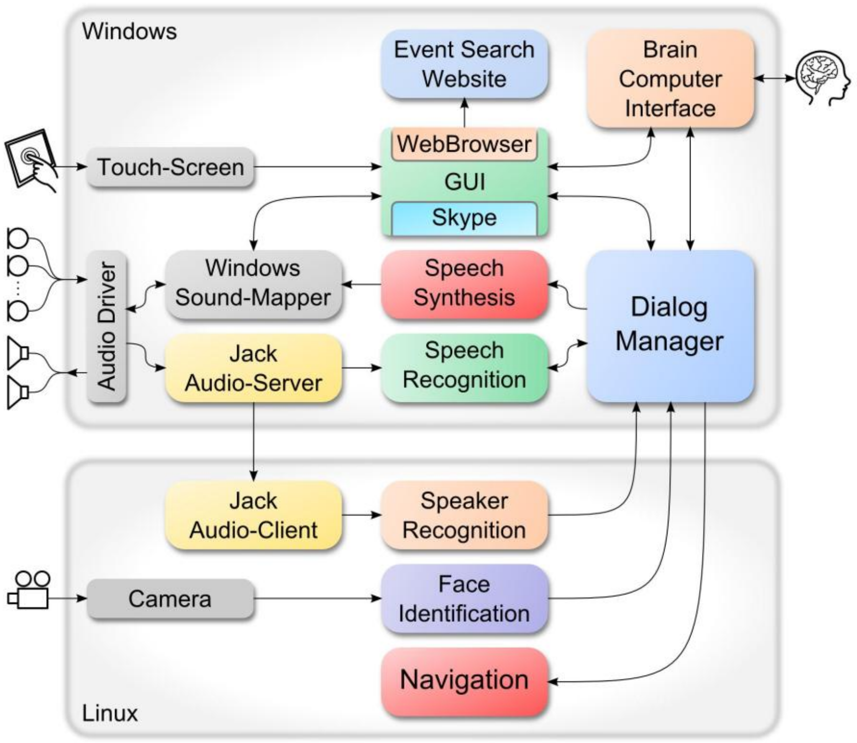


Figure 9: Overview of the ALIAS robot's software modules, distributed on two separate computers. User interaction modalities illustrated on the left and to the right edge.

3.1 Dialogue Manager

The Dialogue Manager represents the kernel of the user-machine-interface. It receives user inputs and other events from the system's various modules. All of which are processed and evaluated. Thereafter, the DM decides on what kind of action the robot is to perform and sends the corresponding commands to the respective modules. The DM prototype has been described in Deliverable D3.4.

The DM implements a mapping from events (e.g. an incoming phone call) and user inputs (e.g. a speech command to pickup the incoming phone call) to result signals (to control the robot, to provide visual and/or verbal feedback to the user). In order to achieve a consistent behaviour in responding to inputs from the whole system, event processing is serialized. For each input received, the DM decides on suitable response behaviour and activates it by sending commands to the respective modules.

The DM consists of two parts: the DM Core operates on a uniform conceptual representation which maps the input events to suitable output actions, whereas the DM Communicator deals with the communication protocols and translates from and to the external data formats. These two parts are implemented as separate applications and processes, communicating via an XML-RPC protocol.

The DM Communicator manages the communication between the DM Core and the other components of the robot. To communicate with the other components, various protocols and data formats are used (see Deliverable D3.7). The Communication-Engine of the DM Communicator uses different adapters to abstract from this diversity by translating different input signals into a uniform conceptual event representation and by translating DM Core action decisions into different outgoing signal formats (cf. Figure 10).

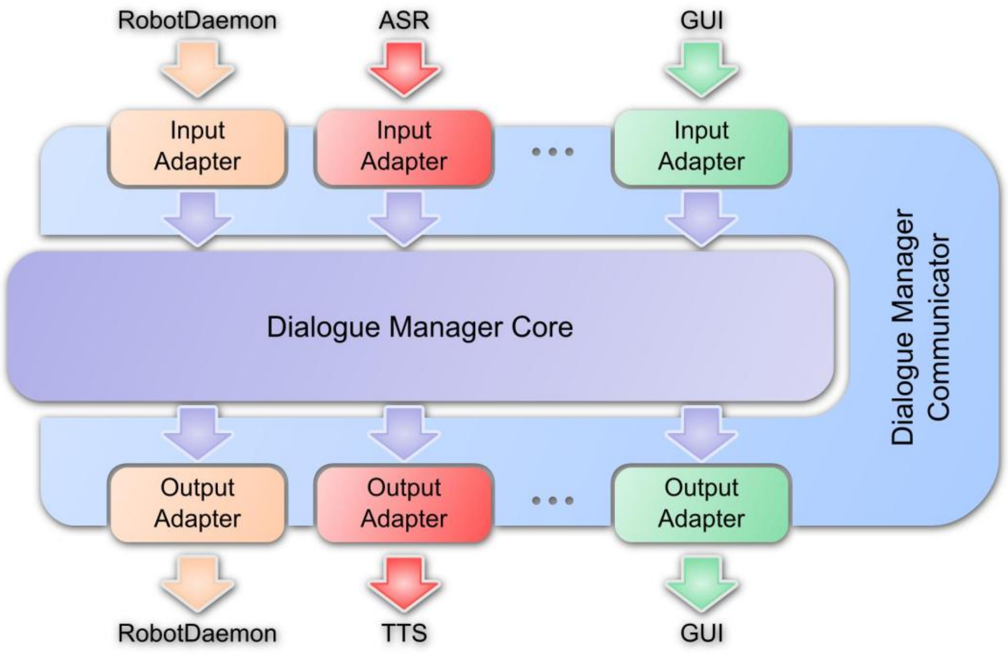


Figure 10: Schematic View of the DM Communicator.

The DM Core encompasses two major components, the NLP-Engine and the Decision Engine. The natural language processing engine (NLP-Engine) translates spoken user input provided as text by the ASR component into events that are then further processed by the decision engine. The Decision-Engine receives events from the NLP-Engine and the DM Communicator and decides on the robot's behaviour by choosing the next action. To choose the appropriate action, the Decision-Engine takes the current situation into account, which is based on the history of past events.

The general DM event loop cycle can be summarized as follows: when an input from other parts of the system (e.g. the modules described in previous sections, or the GUI, or the ASR) arrives at the DM Communicator, it is received and pre-processed by the corresponding input adapter (cf. Figure 10). After the input has been translated to a representation on a conceptual level, it is passed on to the DM Core. There, the input is processed in two phases: first, the understanding phase establishes the semantics of the input. Afterwards, the decision making phase selects a set of appropriate actions. These actions are then returned to the DM Communicator that activates the appropriate output adapters to send the corresponding signal to the destination. In the general case, there is of course no fixed connection between input and output channels: an input arriving via a certain input adapter can generate outputs for any number of output adapters.

To fulfil its tasks, the DM Core uses two knowledge databases. The NLP knowledge base provides general information regarding the world and expert knowledge about the life of elderly people and to the translation of natural language to abstract event entities. The situation model knowledge base stores information needed by the Decision-Engine to compute the new situation and to choose the next action, both with respect to the current situation.

3.2 Automated Speech Recognition

The ASR system operates by processing recorded audio signals, i.e. stripping it down to its relevant features only, and comparing them to a set of previously generated speech models. Doing so, and selecting the most probable speech models respectively, enables the ASR system to deduce the speech phrases that have been pronounced and recorded.

Creating an ASR device requires different processing steps. Figure 11 illustrates the structure of such a system. A very important step is to collect a large amount of speech data from many different speakers. For the ASR device used on the ALIAS platform, Fraunhofer recorded approximately 40 different speakers, who each spoke several command utterances. This speech data was used to train the acoustic models for the recogniser.

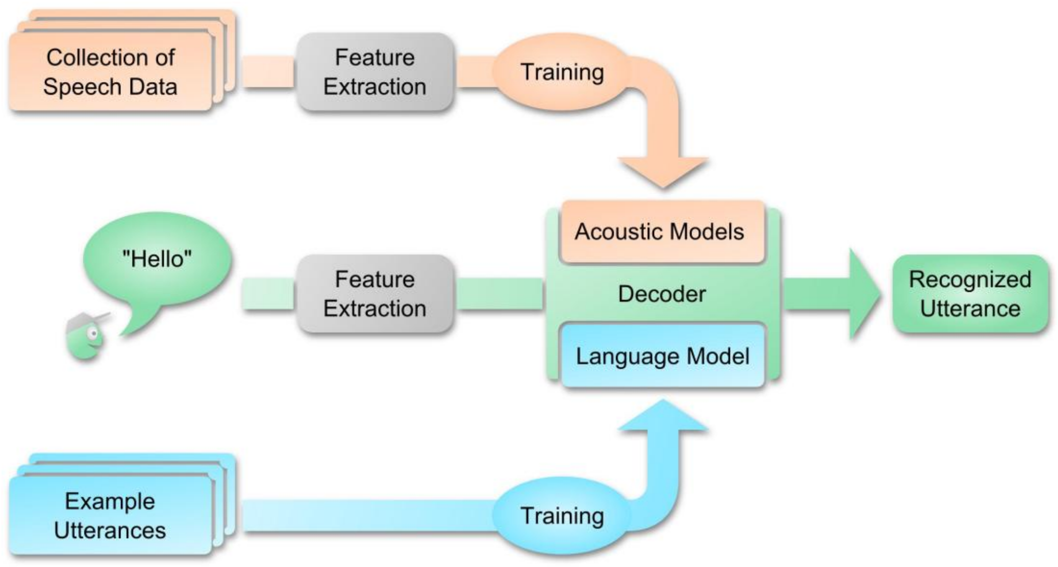


Figure 11: Schematic of the technical design of the ASR device. Acoustic models and the language model are trained once only, thereafter they're used by the decoder to convert pronounced utterances (e.g. "Hello") to recognized utterances.

The current prototype system uses triphone-based Hidden-Markov Models (HMM) for acoustic models. Compared to old-style whole-word HMMs, triphone HMMs are more suitable to accommodate the increased vocabulary size of the ASR system, which comprises about 200 words. This is still not close to true large-vocabulary ASR, which would mean that several thousand words could be classified. That is why the system uses so-called out-of-vocabulary (OOV) models to identify unknown words. Acquisition of additional speech recordings of many speakers (up to 100 or even more) of different age and gender and annotation thereof is still an ongoing process.

In addition to acoustic models a proper speech recognition system also needs a language model. The language model provides "grammatical" information on the utterances that are directed to the ASR system. These models describe interrelations between trained HMMs and their correspondence to a representation of the ASR vocabulary.

With the acoustic models and a valid language model the speech recognition device is able to operate. The user pronounces a command utterance that is recorded by a microphone. ASR features are extracted from the speech input and transferred to the classification algorithm, where the content of speech is decoded. The output of that process is the recognized utterances.

The actual implementation of the ALIAS robot's ASR system is a combination of two differently tuned speech recognition systems. In addition, the speech decoding process operates through probabilities. So there is not one single solution, but a set of several alternatives with varying probabilities. To enable the DM with a broader heuristic, multiple alternatives are provided for interpretation, for both systems.

Communication between ASR module and DM is based on HTTP. ASR results are UTF-8 encoded, formulated in a JSON array structure (cf. Figure 12,) and send via a HTTP POST request. The JSON array contains several alternatives, which, in turn, are arrays of tuples of words and their corresponding probability.

```
[
[["hallo", 1.0],["alle", 0.934]],
[["hallo", 1.0],["du", 0.76],["auch", 0.86]],
[["hallo", 1.0],["OOV", 0.23]]
]
```

Figure 12: ASR recognition, formulated as JSON array. It contains 3 alternatives with 2, 3, and 2 words, whereas the last word of the 3rd alternative was not included in the vocabulary.

3.3 Speech Synthesis

Speech synthesis is the process of artificially creating a human-like voice signal. Speech synthesis is usually coupled with text input, in order to supply the speech phrases, which are to be synthesised. Such systems are commonly called text-to-speech (TTS) systems. A proper TTS system incorporates languages models to deduce the correct pronunciations for written text. Depending on the quality of the TTS system, the resulting speech signal sounds synthetic or preferably as natural as possible. However, the users may even be more comfortable with a distinctly synthetic voice. After all, they're listening to a computer system. In case of the ALIAS system, the built-in Windows Speech API (SAPI) is used. It supports all three required languages (German, English, and French) and has been integrated in a TTS server module that can be accessed by several TTS clients. Clients and server communicate via XML-RPC. The communication protocol is designed to send two strings in a struct. The first string "command" controls the speech server (e.g. language selection, stop current TTS, etc.). The second string "argument" is the text to be spoken. The speech server is running on the windows machine (Mac Mini). A further client was realized on the Linux side. This Linux client is coupled with the Angel Script of the Robot Daemon, thus arbitrary sentences can be spoken. The DM entertains a direct connection to the TTS server, without the need for another client.

3.4 Graphical User Interface

This section provides an overview of the graphical user interface structure. The GUI consists of a series of menus containing few large buttons, each of which leading to another menu or starting an application, i.e. an integrated software module. The GUI's main menu is depicted in Figure 13; it contains three buttons, leading to the communications sub-menu, the entertainment sub-menu, and the security sub-menu.

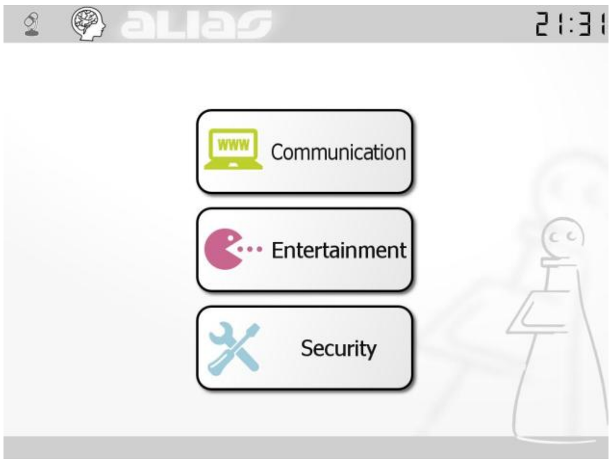


Figure 13: The GUI's main menu.

The GUI has been specially designed for elderly users. Most of its buttons are rather large, use large high contrast fonts and black frames. Buttons will flash, if clicked. Menus will be animated and slide off the screen on transitions (cf. Figure 14). Such visual effects aren't merely gimmicks, but they're used to direct the user's attention to relevant parts of the interface and to provide sufficient visual feedback on received user inputs.

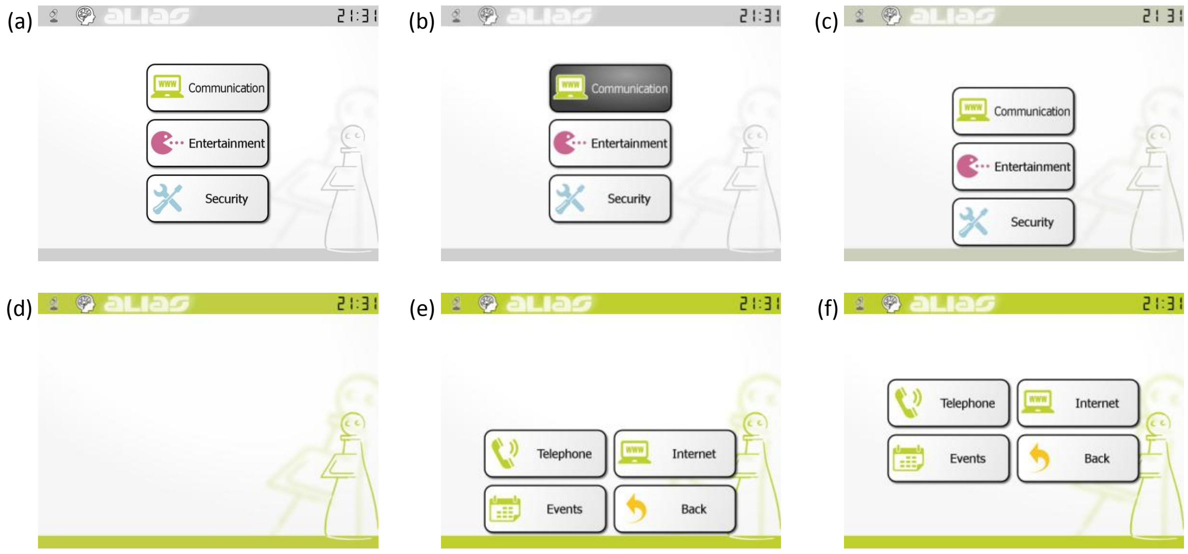


Figure 14: Menu switch animation. Beginning with the main menu (a) the button for the communication menu is pushed (b). This causes the current menu to slide off the screen (c), while the background colour changes to match the target menu (d). The communication menu slides on the screen (e) and becomes active (f).

The GUI makes a clear distinction between menus and application modules, though both are supposed to look quite similar on the screen. Menus provide access to sub-menus and integrated software modules i.e. "applications," using a tree-like menu structure, which is defined by a configuration file. Every menu is identified and accessed by a unique ID.

Application modules (e.g. like the Solitaire game) are also identified and accessed by means of unique IDs, but once an application has been started, there is no common structure. Every application implements its own layout, buttons, features, and remote-control capabilities, so some features are available after the application has been started, only.

Different techniques have been applied to integrate a selection of application into a common user interface. Each integrated component is perceived as a separate program or an application to be executed within the GUI. This can be achieved by pushing the corresponding button on the robot's touch-screen or remotely by the DM, using the GUI's external interface. The same interface is also used to provide events (e.g. menu switches or incoming phone calls) to the DM.

DM and GUI modules communicate via network interface using the User Datagram Protocol (UDP) in order to exchange data packages, containing text messages. These packages are composed according to Extensible Markup Language (XML) guidelines, but neglect the explicit declaration of a Document Type Definition (DTD). For character encoding the common 8-bit Universal Character Set Transformation Format (UTF-8) is used, without specification of a byte order marks (BOM). This allows for the messages to contain non-English special characters.

There are three types of packages: “command,” “signal,” and “request.” Commands are issued in order to trigger the GUI or the DM to perform a specific task, e.g. switching a menu, or driving the robot to another location. Signals are notifications used to propagate recent status updates, e.g. the successful execution of a previously issued command. Requests are commands in need for verification. They are sent to the Dialogue Manager to be acknowledged and returned as a command, if the Dialogue Manager deems this to be okay.

The following text will present the three data package types in closer detail. Package structures will be visualized using the DTD specifications, as they might precede the related package. These DTDs are templates for the related packages, though they're excluded from the actual transmission, saving some bandwidth. Every menu and every integrated application has a unique identification string that will be used within the data packages in order to address them.

Command Package

Commands are, as the name suggests, commands to be executed by the GUI, e.g. switching a menu or starting a game application. Each command package is composed according to the DTD template as presented in Figure 15.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE command [
<!ELEMENT command (arg*)>
<!ATTLIST module CDATA #REQUIRED
          id     CDATA #REQUIRED
          value  CDATA "">
<!ELEMENT arg CDATA>
<!ATTLIST arg
          name  CDATA #REQUIRED>
]>
```

Figure 15: Document type description for the command XML structure. (To be excluded from the actual transmission.)

The *command* attributes *module* and *id* are mandatory for every command. Whereas *module* contains the command category identification string, i.e. the module to which the commands belongs. IDs “menu” and “app” are reserved for switching to a specific menu or for starting an (integrated) application. The *module* attribute is also used for addressing a specific application; in that case *module* corresponds to the application's identification string. Basically each application module may have its very own set of commands and signals (see below.)

The second mandatory attribute *id* represents the actual command's identification string whose actual meaning depends on the provided *module* attribute. In case *module* is "menu", the *id* attribute contains the ID of the menu to be shown on the screen. With *module* set to "app" the *id* attribute contains the identification string of the application to be executed. Otherwise, in case *module* contains an application ID, the meaning of the *id* attribute is up to the specified application.

The third *command* attribute *value* is optional. Its meaning is depending on the provided combination of *module* and *id* attributes. For commands with *module* "menu" or "app", there is no *value*. (If specified anyway, it'll be ignored.)

Optional child nodes *arg* may be used to provide the command with additional arguments or data. The *arg* node's attribute *name* represents the name of the provided argument, while the node body contains the actual argument. These arguments can be used to send messages containing special characters like ":", while the three main attributes may not.

Signal Package

The signal type packages are used to acknowledge received commands, propagate user inputs, or to provide status updates, e.g. an incoming phone call or temporary ASR suppression. Signals may be a direct or asynchronous response to a received command, they may be affiliated to a specific module or occur at random, e.g. an incoming phone call. The actual package structure is similar to the *command* package; it is composed according to the DTD template as presented in Figure 16.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE signal [
<!ELEMENT signal (arg*)>
<!ATTLIST module CDATA #REQUIRED
          id     CDATA #REQUIRED
          value  CDATA "">
<!ELEMENT arg CDATA>
<!ATTLIST arg
          name  CDATA #REQUIRED>
]>
```

Figure 16: Document type description for the signal XML structure. (To be excluded from the actual transmission.)

The attributes *module* and *id* are used in the same manner as presented in the command package description, to affiliate the signal to a certain module and command. In most cases the *value* attribute is used to store the current status, which is either "ok" or "error"; however this depends on the provided *module* and *id*. After all the value attribute is optional and may be empty or missing as well.

As with the command package, optional child nodes *arg* may be used to provide the additional arguments or data. The *arg* node's attribute *name* represents the name of the provided argument, while the node body contains the actual argument.

Request Package

The *request* package is exactly the same as the *command*, with one exception. The root node is named *request* instead of *command*. In case the GUI needs to execute a command itself, it generates a request package and sends it to the DM. Doing so enables the DM to asses the requested and issues an appropriate command to the GUI, or ignores the request if deemed inappropriate. However, this feature is rarely used with the current ALIAS prototype. The request package's DTD is presented in Figure 17.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE request [
<!ELEMENT request (arg*)>
<!ATTLIST module CDATA #REQUIRED
          id     CDATA #REQUIRED
          value  CDATA "">
<!ELEMENT arg CDATA>
<!ATTLIST arg
          name  CDATA #REQUIRED>
]>
```

Figure 17: Document type description for the request XML structure. (To be excluded from the actual transmission.)

Example

A short exemplary communication between Dialogue Manger module and the robot's GUI is presented in Figure 18; it demonstrates the initiation of a phone call using the public switched telephone network (PSTN.)

```
DM → GUI     <command module="app" id="id_skype"/>
GUI → DM     <signal module="app" id="id_skype" value="ok"/>
GUI → DM     <signal module="id_skype" id="start" value="ok"/>
DM → GUI     <command module="id_skype" id="call_by_number" value="+49123..."/>
GUI → DM     <signal module="id_skype" id="call_by_number" value="ok"/>

... phone conversation ...

... phone call disconnected by pushing the hang-up button ...

GUI → DM     <signal module="id_skype" id="call_disconnect" value="ok"/>
DM → GUI     <command module="menu" id="id_mainmenu"/>
GUI → DM     <signal module="id_skype" id="close" value="ok"/>
GUI → DM     <signal module="menu" id="id_mainmenu" value="ok"/>
```

Figure 18: Exemplary communication between the Dialogue Manger (DM) module and the robot's GUI. The DM starts the Skype module and initiates a phone call to the specified phone number. The call is disconnected via the GUI thereafter the DM decides to switch the GUI back to the main menu.

Note that in the example the GUI could have sent a request for the desired "disconnect" command to the Dialogue Manager that in turn could have provided it. Or the Dialogue Manager could discard the GUI's request for various reasons. In this example, there is no point in preventing the user from disconnecting the call via the GUI, so the GUI doesn't bother the Dialogue Manager with a request. Instead, the Dialogue Manager will be kept informed by the GUI's status signals.

A more detailed description of the interface between GUI and DM is provided Deliverable D2.2. Although explicit specification of the "close" command has been removed, it will be triggered automatically in case the related application module exits.

3.5 Brain-Computer-Interface

The BCI is one of the more exceptional user interface modules. It is comprised of several distributed components. The main BCI software runs on a separate BCI computer, which is linked to the ALIAS robot via wireless network (i.e. “WiFi”). The BCI computer is connected to the brain cap including electrodes that are placed on the scalp of the robot’s user. Visual stimuli are provided on the robot’s touch-screen, using a BCI overlay that has been integrated with the GUI.

The DM signals the GUI to display the BCI overlay; thereafter it prepares the BCI mask to be displayed. Using a WiFi connection and the UPD network protocol, the BCI mask is transmitted to the BCI computer. The BCI mask is interpreted and the BCI system configured accordingly. The BCI computer uses the WiFi connection to the BCI overlay in order to display a processed version of the BCI mask, originally provided by the DM. The communication channel between BCI computer and BCI overlay remains active in order to synchronise BCI signal processing and stimuli display. Connections between these modules are illustrated in Figure 19.

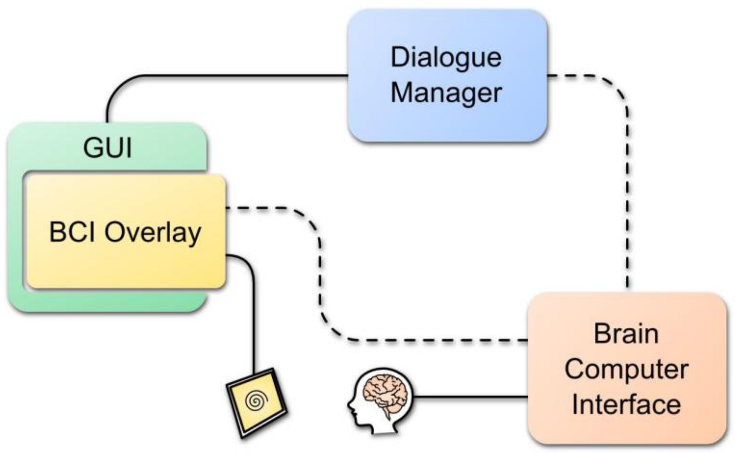


Figure 19: BCI system is composed of separate modules, communicating via internal network connections (solid lines) and WiFi (dashed lines). All connections are bidirectional.

In case the BCI has recognized a user input within the measured brain potentials, this input is transmitted to the DM. The DM interprets the input and reacts accordingly, changes a menu, runs an application module, generates new BCI mask and the whole process starts anew.

For communication between DM and BCI computer, UTF-8 encoded XML structured text messages are used. An exemplary BCI mask, as it would be provided by the DM, is presented in Figure 20. It includes specifications of the network connection to the BCI computer and the dimensions of the grid layout. It also defines the symbols to be displayed (in this case plain text) and the corresponding commands that would be transmitted to the DM on activation.

```
<?xml version="1.0" encoding="utf-8"?>
<AppList xmlns:xsi ="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="XMLSchema_UDPinterface_v3.xsd">
  <ControlGroup CGName="ALIAS_BCI_CONTROL">
    <CGAddress>
      <IPAddress>192.168.100.85</IPAddress> <!-- set apropriate ip address -->
      <Port>22346</Port> <!-- define appropriate port see manual -->
    </CGAddress>
    <MaskSize>
      <NoLines>3</NoLines>
      <NoCols>4</NoCols>
    </MaskSize>
    <DispSymbol><Text>ALIAS</Text></DispSymbol>
    <SamplingRate>-1</SamplingRate>
    <Application AppName="AudioBooks">
      <AppID>id_audiobooks</AppID>
      <!-- Audio Book #1: 20.000 Meilen unter dem Meer -->
      <SingleCommand CommandName="AB1">
        <Instruction>playback_start</Instruction>
        <ICONPosition>[1,1]</ICONPosition>
        <DispSymbol><Text>20.000 Meilen</Text></DispSymbol>
        <CommType>multi</CommType>
        <CommandParameter>
          <Parameter type="value">id_20000_meilen</Parameter>
        </CommandParameter>
      </SingleCommand>
  <!–- Audio Books #2 to #7 are omitted in this example. -->
      <!-- Audio Book #8: Tom Sawyer -->
      <SingleCommand CommandName="AB8">
        <Instruction>playback_start</Instruction>
        <ICONPosition>[2,4]</ICONPosition>
        <DispSymbol><Text>Tom Sawyer</Text></DispSymbol>
        <CommType>multi</CommType>
        <CommandParameter>
          <Parameter type="value">id_tom_sawyer</Parameter>
        </CommandParameter>
      </SingleCommand>
      <!-- Stop Command -->
      <SingleCommand CommandName="PlaybackStop">
        <Instruction>playback_stop</Instruction>
        <ICONPosition>[3,2]</ICONPosition>
        <DispSymbol><Text>Stop</Text></DispSymbol>
        <CommType>multi</CommType>
      </SingleCommand>
      <!-- Exit Command -->
      <SingleCommand CommandName="BackMainMenu">
        <Instruction>id_mainmenu</Instruction>
        <ICONPosition>[3,3]</ICONPosition>
        <DispSymbol><Text>Back</Text></DispSymbol>
        <CommType>multi</CommType>
      </SingleCommand>
    </Application>
  </ControlGroup>
</AppList>
```

Figure 20: Exemplary BCI mask of the audio book module (for brevity audio books #2 to #7 have been omitted)

4 Robot Functions

This section presents the robots actual functionality that has been realized by means of the previously described modules. Most of this functionality has been included with at least one of several proposed use-case scenarios like the self-experience scenario.

Most of the features that are integrated with the GUI are accessible to the DM via the GUI's external network interface, meaning these functions can be controlled by the DM in case a corresponding speech command or BCI input is received.

4.1 Menu System

The user is able to navigate through the robot's menu by means of pushing buttons on the touch-screen, by means of verbal commands or the BCI. The menu provides access to sub-menus and the robot's application modules, cf .Figure 21. Every menu, sub-menu and application has a unique ID that can be used by the DM in order to display it on the screen or gain access to its functions.

Speech commands like "Zeig mir deine Spiele." (engl. "Show me your games.") may be used to cause the DM to show up a specific menu (in this case the games menu) on the screen. While the GUI notifies the DM on every menu switch, thus the DM may interpret speech commands in a different context, depending on the current screen contents.

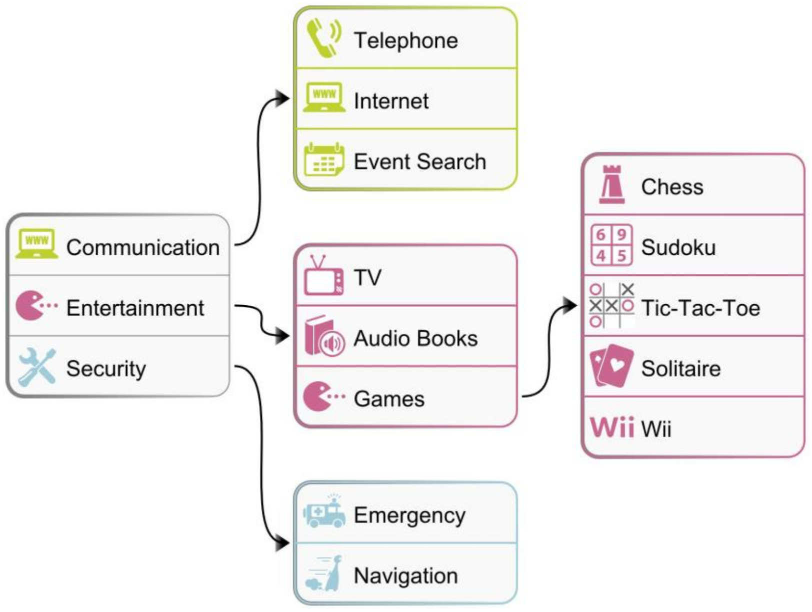


Figure 21: The ALIAS robot's hierarchical menu structure. Menus are organised in three main groups communication, entertainment, and security; indicated by different colors.

The menu structure has been evaluated as part of a bachelor thesis at Fraunhofer. Although all test subjects consider the usability of the hierarchical menu structure "as is" quite high, about two thirds slightly preferred planar menus, e.g. like the iOS main screen. However, planar menu styles that have been evaluated were criticised because their buttons had to be very small in order to fit on the screen. In short: users consider the current menu style adequate.

4.2 Status Bar

The GUI includes a status bar (cf. Figure 22) that is used to display the current status of the ASR system (active or inactive), the BCI input system (active or inactive), and a clock. The latter is autonomous and displays the current system time. The former two display the status as they are told from the DM. they also act as on/off switches, this sending the status change request to the DM, but do not perform any action on their own. Activity of several modules (e.g. TV) may cause the DM to disable the ASR system temporarily.

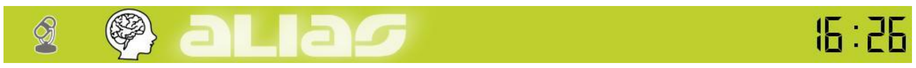


Figure 22: The GUI's status bar. Located on the left hand side are status indicators/switches for ASR and BCI systems. On the right hand side it contains a clock, displaying the current system time.

4.3 Telephone

The telephone module enables the user to make video phone calls using the Skype network. This also includes sending and receiving of chat messages and calls to the regular public phone network (without video). The telephone module may be activated via the robot's touch-screen, or remotely via the GUI's external network interface, hence by speech command or BCI. The module consist of several different sub-menus that reflect the current state, i.e. displaying a video image in case a video call has been established, or a keyboard in case a chat session has been initiated (cf. Figure 23).

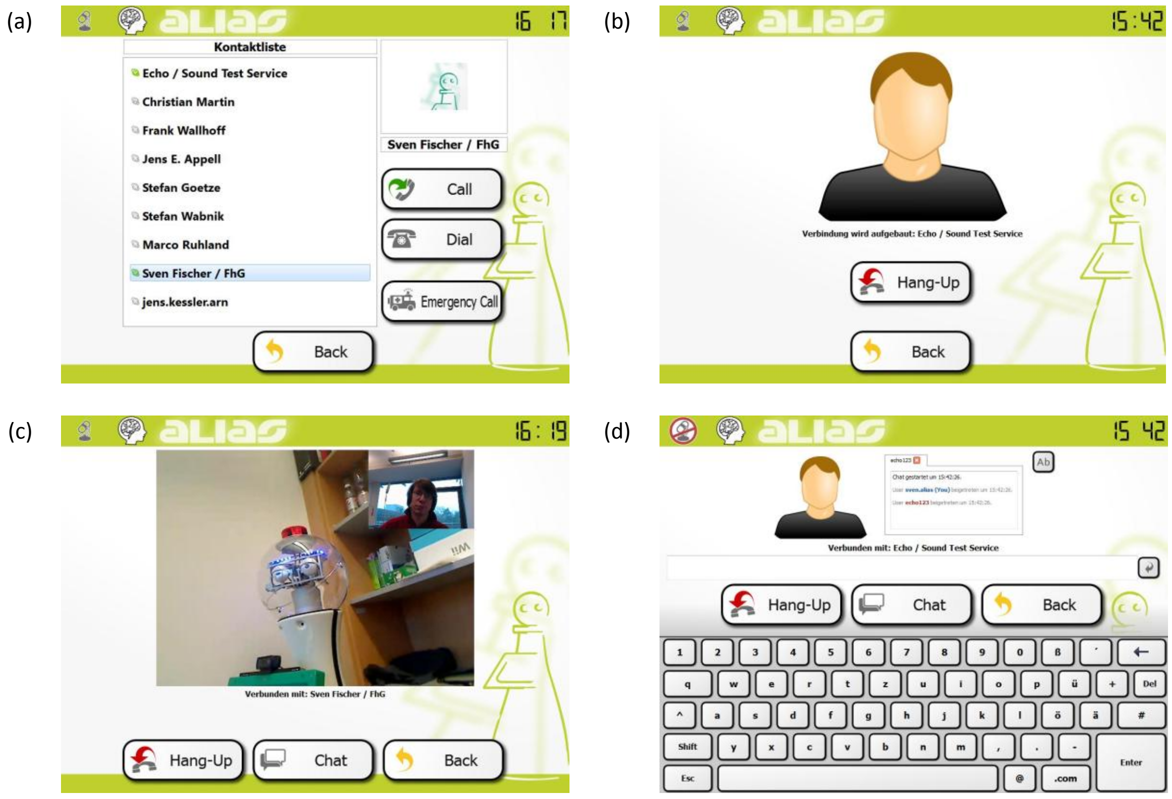


Figure 23: The telephone module in various states: (a) the main screen showing a list of contacts, (b) a call is beeing established, (c) an active video call, and (d) an active chat session (without video.)

All functions of the telephone module are accessible via the touch-screen menu structure. By means of speech commands like "Ruf Bob an." (engl. "Call Bob."), a contact may be called directly. The DM will receive a notification on every command that is executed, regardless of how it is activated. In case the BCI mode is activated, all controls of the telephone module will replaced by matching BCI masks.

The telephone module also features an “emergency” button that initiates the emergency call procedure, see Section 4.9.

4.4 Internet / Event Search

The ALIAS system features an integrated web browser that enables the user to access the Internet (cf. Figure 24). The web browser may be started by tapping the related button on the robot’s touch-screen, or remotely by the DM via the GUI’s network interface, i.e. by speech command or BCI.

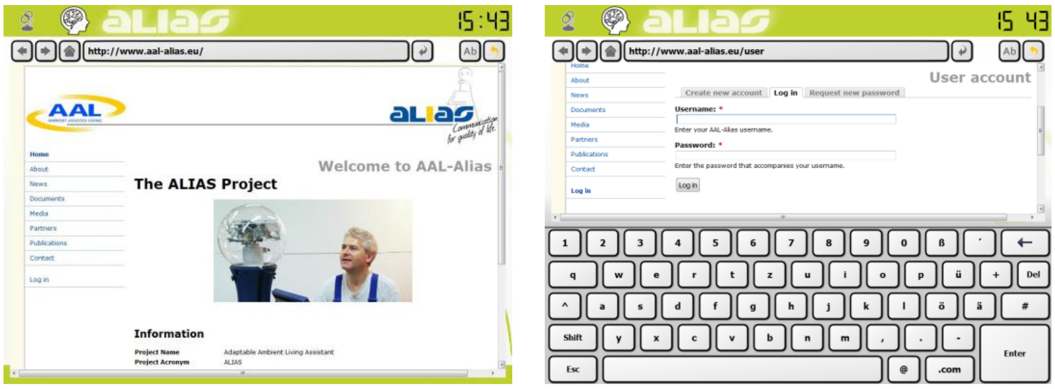


Figure 24: The integrated web-browser: normal view (left hand side) and with keyboard for text input (right hand side).

The web-browser supports basic functions like accessing a web page by typing a URL, loading the start page, and navigating backward and forward through the browsing history. All these functions can be accessed by the DM as well, whereas the DM will be notified every time the currently displayed website is changed.

The browser supports secure connections and enables the user to logon to web-based e-mail services, participate in online-shopping, online-banking, or social networking.

The event search module is very similar to the web-browser module, though both modules are completely independent and have different IDs. The event search module also implements a web-browser to access the server-based EventMedia search engine via the internet. Navigation is implemented exclusively via the EventMedia website, so this module doesn’t include any navigation controls (cf. Figure 25). The event search engine runs autonomous without direct connection to the DM.

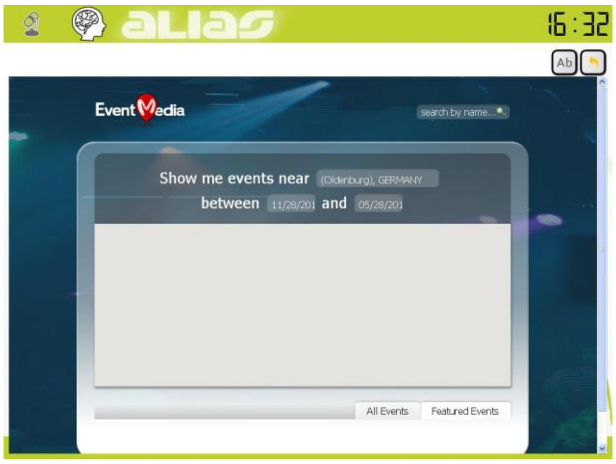

aLIaS
16:32
EventMedia
search by name...
Show me events near (Oldenburg), GERMANY
between 11/28/201 and 05/28/201
All Events
Featured Events

Figure 25: The EventMedia website, displayed in the GUI's event search module.

4.5 TV

The TV module (cf. Figure 26) allows the user to use the robot's screen for watching TV, provided the reception is good enough, though. The TV module may be activated via the touch-screen or by the DM. The current channel may be switched by pushing the related button on the screen, or by the DM via the GUI's network interface.

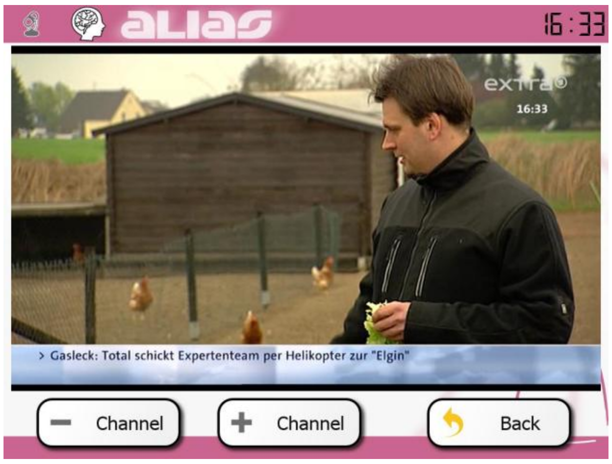

aLIaS
16:33
extra
16:33
> Gasleck: Total schickt Expertenteam per Helikopter zur "Elgin"
Channel
Channel
Back

Figure 26: The integrated TV module. The user may switch the channel by pushing the related buttons.

4.6 Audio Books

The ALIAS system offers a selection of audio books to the user (cf. Figure 27). The module may be activated via the touch-screen or by the DM. In order to start playback, the user may tap on the cover of the audio book that should be played. This functionality is also available to the DM; hence playback may also be started and stopped using the BCI.

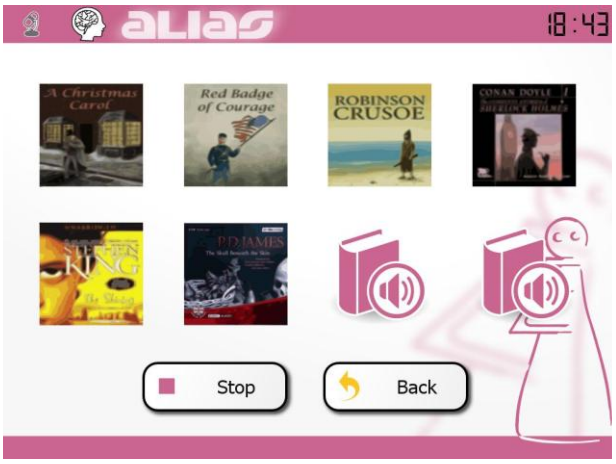


Figure 27: The audio book selection. On this screen there are 6 audio books available for playback, while 2 symbols (lower right) are still unassigned. Playback may bestarted by tapping on the audio book cover.

4.7 Games

For entertainment and for keeping the user mentally active, a selection of games has been integrated with the ALIAS system. At this point, the games may be activated via the DM (e.g. in response to speech commands or BCI input) or the touch-screen, but played via the touch screen, only. For example, the verbal command “Zeige mir deine Spiele.” (engl. “Show me your games.”) causes the DM to notify the GUI to display the games menu on the screen, cf. Figure 28.

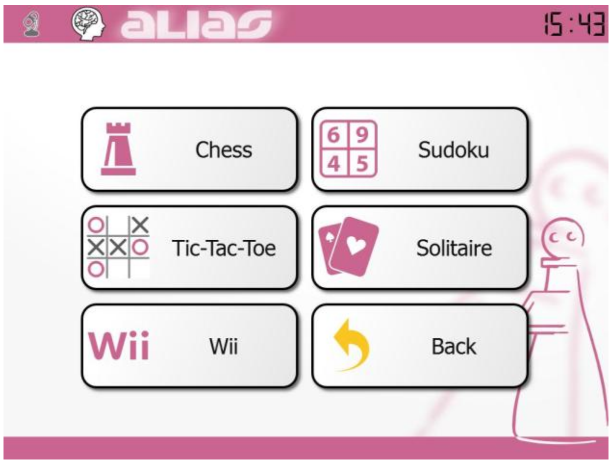


Figure 28: The games menu, providing access to the robot's games selection.

The games selection includes the Chess and Solitaire games, as they are provided by the robot’s Windows 7 operating system. It also features build-in implementations of Sudoku and Tic-Tac-Toe. These games are illustrated in Figure 29. In order to play a game, the user may use the touch-screen to navigate through the menu system, or run the desired game directly by providing a verbal command, e.g. “Starte Schach.” (engl. “Run Chess.”). (The Wii gaming console will be addressed in the following section.)

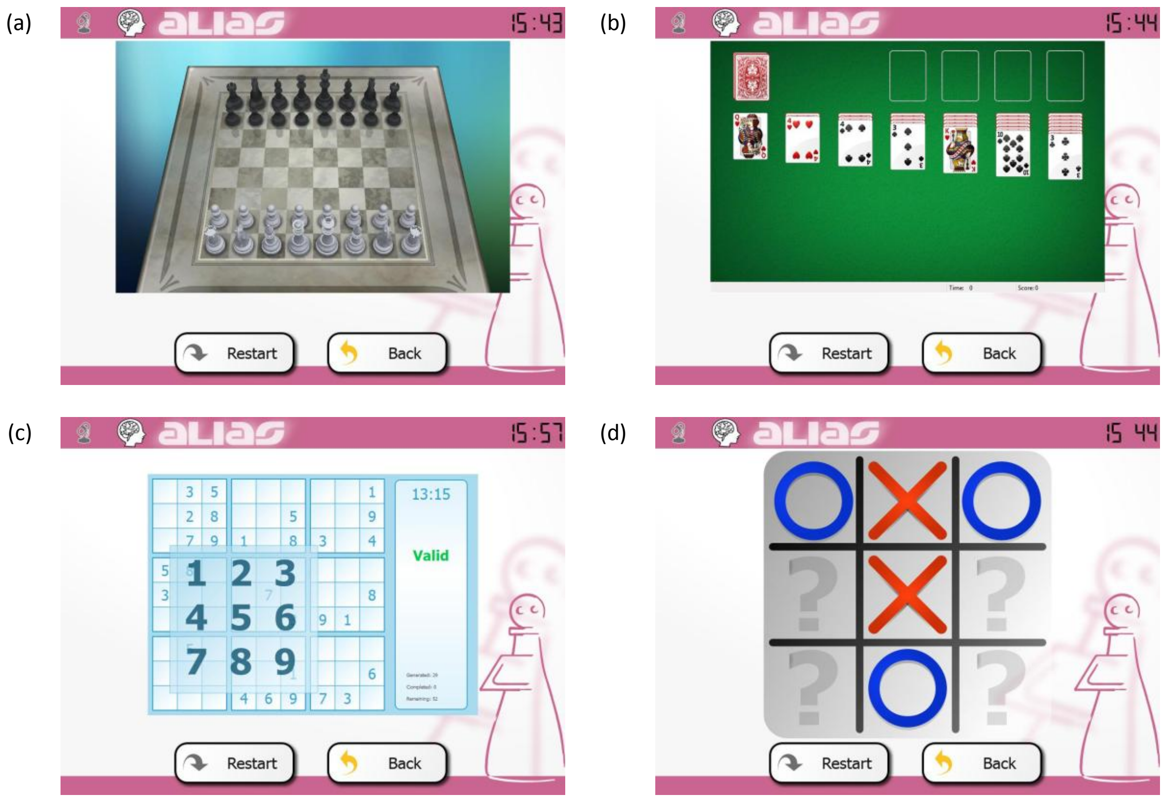


Figure 29: Integrated games: Chess (a), Solitaire (b), Sudoku (c), and Tic-Tac-Toe (d).

4.8 Wii

The ALIAS system includes a Wii gaming console that has been integrated with the GUI (cf. Figure 30). Since the Wii is a proprietary third-party system, it runs autonomously and lacks a proper interface to the DM. Games are controlled by means of the Wii Remote (cf. Section 2.4.1), while the ALIAS screen acts as a mere display.

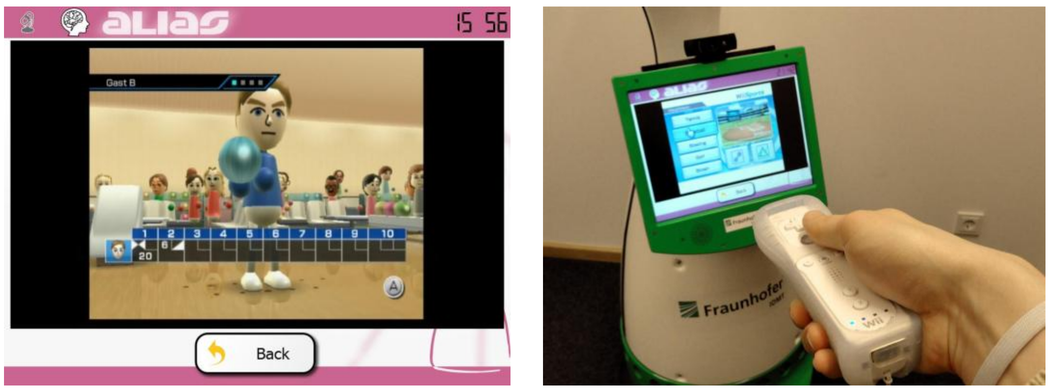


Figure 30: The Wii gaming console, integrated with the GUI (left). Games are played with the special Wii Remote controller (right).

The Wii module can be activated via the touch screen, or by the DM using the GUI's network interface. However, in order to play the games the user has to push the buttons on the Wii Remote controller.

4.9 Emergency Call

In case the emergency call procedure is triggered, the robot will establish a video call connection to an emergency operator. In addition, the robot's navigation control is transferred to the emergency operator for the duration of the call. Due to this reason the operator may not only talk to the user, but drive the robot around and use its cameras to asses the current situation. This is a sensitive feature that may have impact on the user’s privacy or impose an unnecessary drain on resources, if triggered by mistake. To avoid false alarms, the emergency call is preceded by a 20 second countdown (cf. Figure 31), providing the user with the ability to cancel the process before any action is taken.

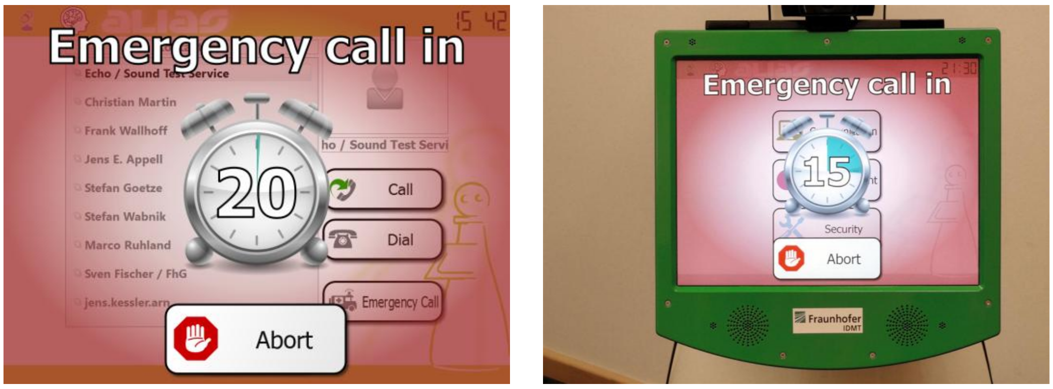


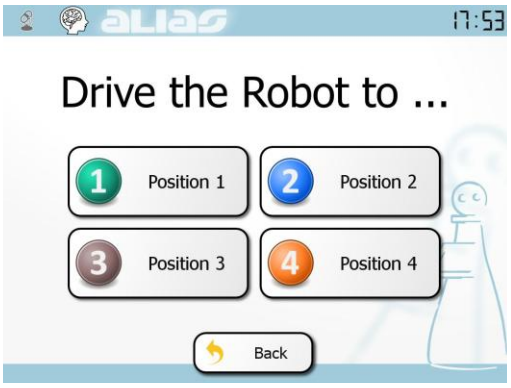
Figure 31: The emergency call procedure is preceded by a 20 second countdown to provide the user with the ability to cancel the process on false alarms.

The emergency call procedure may be triggered by several actions: The user may push the emergency button on the security menu or within the telephone module, or the user may simply call for help verbally. Another option would be that the DM may decide to trigger the procedure for other reasons (e.g. physiological monitoring). In any case, the robot will respond by voice command and wait for the countdown to run down before actually calling the emergency operator.

4.10 Navigation

The navigation module, i.e. the robots ability to move is a vital part of the ALIAS system, since it enables the robot to move from one location to another. The robot reacts on voice commands like “Geh bitte zur Seite.” (engl. “Step aside, please.”) or “Führ mich zum Bad” (engl. “Guide me to the bathroom.“) and starts moving. This involves several of the systems modules: the ASR to receive the speech command, the person detection to determine the location of the user, the DM to interpret inputs and decide what to actions to perform, and the navigation module to plot pathways and actually drive the robot to its new target location, avoiding collisions with the user.

Robot movements may also be triggered via the touch-screen. The GUI’s navigation module offers four configurable buttons that may be used to send the robot to a selection of pre-defined locations, e.g. bathroom, or kitchen, whereas every location corresponds to a set of coordinates on the robot’s internal map. The navigation menu is depicted in Figure 32. This also includes the emergency stop screen that is displayed for as long the robot is moving. Tapping on the screen while the robot is moving causes it to stop immediately.



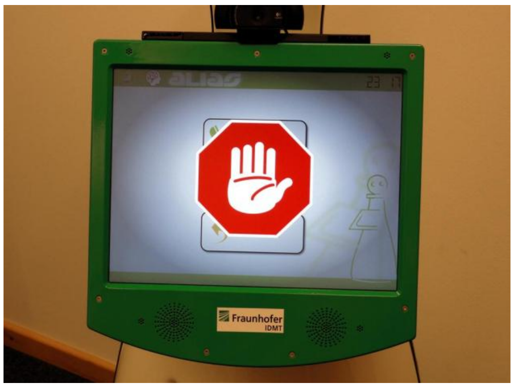


Figure 32: The Navigation module (left) allows the user to send the robot to some pre-defined locations. While the robot is moving, the screen will be replaced with an emergency stop button.

5 Summary and Conclusion

The ALIAS system is comprised of various different modules, enabling user interaction via speech, touch-screen, or BCI. Several other functions like person detection and identification or the robotic head can be utilized in order to simulate a more human-like behaviour. While some systems, e.g. the Wii gaming console, are mostly autonomous, most user inputs are evaluated and processed by the Dialogue Manager that acts as the kernel of the user-machine-interface. It combines inputs and events from the robot’s various systems via several different interfaces and converts them to an internal representation that is then processed.

The centralized processing of user inputs enables the robot’s function to be controlled using several different user input modalities. For example, if the user decides to make a phone call this can be achieved by pushing a button on the touch-screen, providing a speech command, or via the Brain-Computer-Interface. The phone call may be started by speech command and terminated by via the touch-screen. Thus the ALIAS system is able to handle multimodal user inputs.