SEVENTH FRAMEWORK PROGRAMME

AsTeRICS Deliverable D2.2

Updated System Specification and Architecture

UCY

AsTeRICS

European Commission
Information Society and Media

Document Information

Issue Date	**30 September 2011**
Deliverable Number	**D2.2**
WP Number	**WP2 System Specification and Architecture**
Status	**Final**
Dissemination Level	**RE** **PU Public** **PP Restricted to other programme participants (including the Commission Services)** **RE Restricted to a group specified by the consortium (including the Commission Services)** **CO Confidential, only for members of the consortium (including the Commission Services)**

AsTeRICS – Assistive Technology Rapid Integration & Construction Set
Grant Agreement No.247730
ICT-2009.7.2 - Accessible and Assistive ICT
Small or medium-scale focused research project

Disclaimer

Version History

Version	Date	Changed	Author(s)
0.1	27 Jun 11	First draft	UCY
0.2	14 Jul 11	KI-I first round of contribution	KI-I
0.3	05 Sep 11	STARLAB first round of contribution	STARLAB
0.4	05 Sep 11	IMA first round of contribution	IMA
0.5	07 Sep 11	KI-I second round of contribution	KI-I
0.6	07 Sep 11	FHTW first round of contribution	FHTW
0.7	07 Sep 11	UCY first round of contribution	UCY
0.8	07 Sep 11	STARLAB second round of contribution	STARLAB
0.9	08 Sep 11	Integration of partners' input	UCY
0.10	09 Sep 11	HARPO input for native asapy and Sweety	HARPO
0.11	12 Sep 11	New user-driven requirements section	UCY
0.12	12 Sep 11	Added user driven requirements, updates requirement tables, added some text, corrected typos	FHTW
0.13	12 Sep 11	Second round of contribution, updates on 3D models	IMA
0.14	13 Sep 11	KI-I more input and comments	KI-I
0.15	13 Sep 11	Input integration	UCY
0.16	16 Sep 11	STARLAB editing	STARLAB
0.17	17 Sep 11	UPMC input	UPMC
0.18	19 Sep 11	Input integration	UCY
0.19	28 Sep 11	Post peer-review refinement	UCY

Glossary and Declaration of Terms

1 Terms Specific to AsTeRICS

ACS .... **AsTeRICS Configuration Suite**

A graphical software application running on the host PC for AsTeRICS model configuration and monitoring of the runtime system

ARE .... **AsTeRICS Runtime Environment**

The configured system model (which consists of pluggable components and their interconnections) which has been deployed to the execution environment (usually the AsTeRICS embedded computing platform)

ASAPI .... **AsTeRICS Application Programming Interface**

Provides methods to setup and interact with the ARE via a TCP/IP connection. The ASAPI builds the functional framework used by the ACS to generate and deploy the ARE system model and to configure ARE components. Furthermore, the ASAPI can be used to send and retrieve live data to or from the ARE, and to get logging/status from the ARE. The ASAPI can be used by third-party software applications integrate the ARE.

CCM .... **Core Computer Module**

A customized hardware / PCB with high performance low power CPU, battery and battery management and extension interface for CIMs.

EP .... **Embedded Platform (aka AsTeRICS Personal Platform)**

Core Computer Module and dedicated Communication Interface Modules to support connection of sensors and actuators.

PCOM .... **Pluggable Component Module**

Software components of the ARE, which can be grouped into **Signal Sources** (featuring output ports), **Signal Processors** (featuring input and output ports) and **Signal Sinks** (featuring input ports

SVM .... **Smart Vision Module**

A computer vision sensor with dedicated computing module for feature extraction. Transfers images features to the ARE.

2 Other Relevant Technical Terms

BCI	Brain Computer Interface
BNCI	Brain or Neural Computer Interface
FFT	Fast Fourier Transformation
IR	Infrared
JNI	Java Native Interface
OS.....................	Operating System
OSGi	Open Service Gateway initiative
CPU	Central Processing Unit
DSP	Digital Signal Processor
IMU	Inertial Measurement Unit
PCB	Printed Circuit Board
SRAM	Static Random Access Memory
USB	Universal Serial Bus
VPIF	Video Port Interface
LCD	Liquid Crystal Display
UART	Universal Asynchronous Receiver/Transmitter
DAC	Digital to Analog Converter
HID	Human Interface Device
PCA	Principal Component Analysis
CSP	Common Spatial Pattern
OVR	One Versus the Rest
CVT	Canonical Variates Transformation
LDA	Linear Discriminant Analysis
ADC	Analog to Digital Converter
GPIO	General Purpose Input / Output
EMG	Electromyography

EEG	Electroencephalography
EOG	Electrooculography
SVEM	Support Vector Machine
FPR	False Positive Rate
TPR	True Positive Rate
VOG	Video-Oculography
SDK	Software Development Kit
HMI	Human Machine Interface
FIR	Finite Impulse Response
JVM	Java Virtual Machine
FIFO	First In First Out
MVC	Model View Controller
MVVM	Model View Viewmodel
WPF	Windows Presentation Foundation
COTS	Commercial Off-The-Shelf
AT	Assistive Technology

Table of Content

1 Introduction

The purpose of Work Package 2: “System Architecture and Specification” is to define in detail the requirements and characteristics of the Assistive Technology Rapid Integration and Construction Set (AsTeRICS) project. AsTeRICS’ objective is to enable easy-to-use Assistive Technology, based on user requirements and needs. This document updates the status on the System Specification and Architecture of the AsTeRICS project as it was defined in D2.1. Special emphasis is given to the user-driven requirements emerged after the completion of the user-based evaluation of prototype 1. We have also updated existing requirements to reflect new needs and priorities. Furthermore, we provide updates on the state of the art as well as changes in the hardware and software specifications. Finally, we provide updates on the Enoio Biosignal Unit and the Smart Vision Module.

1.1 AsTeRICS System Architecture Overview

The system architecture of AsTeRICS consists of hardware and software components, which provide the flexible Assistive Technology Rapid Integration- and Construction Set.

Hardware Components:

- AsTeRICS Personal Platform – an embedded high performance computing system
- Communication Interface Modules (CIMs), allow attachment of sensors and actuators
- Sensors and actuators, which include:
    - modules designed and manufactured by consortium members as major objectives of the AsTeRICs project (like the Smart Vision Module)
    - extensions of existing hardware of consortium members (like the Starlab Enobio device)
    - integration of commercial off-the-shelf AT devices (like special joysticks, switches, IR-gateway etc.)

Software Components:

The software components consist of individual packages for host computer and for the runtime environment. The software packages dedicated to the host computer are:

- the AsTeRICS Configuration Suite (ACS)
- the AsTeRICS Application Programming Interface (ASAPI)
- applications which are developed or extended in course of the AsTeRICS project
- the BNCI Evaluation Suite by Starlab
- AT-software solutions by SENSORY like the OSKA on-screen keyboard

The software components dedicated to the AsTeRICS Runtime Environment include:

- the middleware architecture for the ARE
- the Pluggable Component Modules (PCOMs):
    - Signal Sources
    - Signal Processors
    - Signal sinks

The architectural framework as a whole provides the functionality of the AsTeRICS system as defined in the Description of Work document [1], approved by the EC on 22-10-2009, which can be summarised as follows:

- Making sensor data from the user available for processing on the platform (interfacing with sensor module-drivers which are part of the operating system),
- Routing the sensor data to and among signal processing and signal shaping elements, as defined by the interconnection of these elements, for the purpose of extraction of relevant information about voluntary user commands. Furthermore, routing this information to actuator elements,
- Controlling actuators for AT purposes (interfacing with actuator module-drivers provided by the operating system),
- Enabling remote access to the embedded platform via TCP/IP (for download and upload of actual configurations, error reporting and display of live data).

1.2 Relationship to other AsTeRICS Deliverables

The deliverable is related to the following AsTeRICS deliverables:

- D2.1 “System Specification and Architecture” [2]: The first version of system specification and architecture for the overall project. D2.1 is the ancestor of this document and provides a first detailed specification of the system requirements and hardware/software specifications. D2.2 will serve as an update of D2.1 indicating any changes or advances from the initial system design and specification.
- D2.4 “Report on the State of the Art” [3]: Outlines the technological basis for the planning of the AsTeRICS architecture and describes relevant hardware and software components available on today’s market. D2.2 includes an updated state of the art section with new technologies emerged since the delivery of D2.4.
- D1.1 “Report on Users, Preferences and Needs” [16]: AsTeRICS follows a user-centric approach and the user needs and requirements are the main input for elicitation of the system requirements. D2.2 updates system specification and design based on feedback from initial tests.
- D1.3 “Technical Specifications” [17]: This document describes the transformation of user requirements into technical requirements and outlines basic use cases for the AsTeRICS system. It thereby defines the capabilities which have to be met by the AsTeRICS architecture and any updates defined in this deliverable.
- D1.5 “Evaluation Report Integrated Prototype” [4]: The results from the first round of user tests that have been reported in D1.5 have been taken into consideration and mapped to new requirements reported in this deliverable.

1.3 Relationship to Description of Work

According to the Description of Work [1], AsTeRICS will provide a flexible and affordable construction set for user driven Assistive Technologies or assistive functionalities. The AsTeRICS architecture is derived from the user’s requirements and needs and combines the flexibility and modularity of the Java programming language and OSGi compositional framework with state-of-the-art software and hardware sensors and actuators. This deliverable provides an update of the initial system specification based on new technologies and needs identified since the delivery of D2.1 and D2.4.

1.4 Structure of This Document

The rest of this document is structured as follows: section 2 provides the updated system requirements. Most notably, section 2.1 lists user-driven requirements as they have emerged after the first round of user tests. Section 3 provides updates on the state of the art since the delivery of D2.1. Among others, we discuss the specifications of the new platform we plan to use for the second prototype, new software technologies for the ACS as well as extra, off-the-shelf sensors and actuators we plan to use. Section 4 provides updates on the hardware specifications for the embedded platform and pluggable components as well as the smart vision module we plan to implement. Section 5 discusses changes and advances in the software specifications of the system and section 6 provides updated information on the BNCI evaluation suite. Finally, section 7 concludes this document with a summary of the updated system specification and architecture.

2 AsTeRICS Requirements Update

In the following sections we revisit hardware and software requirements lists. The status of already defined requirements is updated and new requirements are added to the list. The last column of the table indicates if a requirement is new (N), if it is modified (M) or if it remains unchanged (U).

A particular requirement will be identified by a prefix for the group and a unique number. The requirement description includes the priority of a particular implementation and the expected project month to be delivered. Requirements identified as high priority requirements should be provided as they reflect basic functionalities expected to be delivered by the project as defined in the AsTeRICS Description of Work [1]. Requirements with medium or low priority are either “nice to have” features or extra functionality not directly affecting the project’s goals. The “Prototype” column indicates if the requirement has been already fulfilled (1) or pending for Prototype 2.

2.1 User-driven Requirements

Based on the user tests performed by the end of prototype 1, new requirements have emerged which could not - or only partly - be satisfied by the first system prototype. (The evaluation report can be found in D1.5 “Evaluation Report Integrated Prototype” [4].)

Following the user-centered design methodology of the project, these new user requirements have been transformed into technical requirements for the final system prototype, in a cooperation of the user experts and the technical experts of the AsTeRICS consortium.

Each of the user needs has been mapped to a new technical requirement listed in the tables below, together with the priority stated by the users and an outline of the technical feasibility for the realization of the requirement in course of the final prototype implementation. To provide a reference to the existing requirement tables, the corresponding table numbers have been added to the new requirements (please see the Table column in following tables).The major focus for this consideration was to make the final AsTeRICS prototype as appealing to the end users as possible.

2.1.1 User-driven general requirements

ID	Name	Description	Priority	Feasible	Comments	Table
URGEN1	Wheelchair control	AsTeRICS should be able to control wheelchairs.	High	No	The AsTeRICS Personal Platform does not comply to the necessary medical certifications and high dependability requirements for online wheelchair control equipment. This could be the focus of a separate spin-off project or customisation of a particular model by an interested 3rd party. To evaluate the feasibility of direct wheelchair control using AsTeRICS, software simulators for wheelchair operation can be connected.	Table 1
URGEN2	Speech input	AsTeRICS should have a voice recognition input. It should also recognise sounds from a person who cannot speak at all, but is able to generate a limited number of distinct sounds.	Medium	Yes	Speech recognition integration will part of the final AsTeRICS prototype. For this purpose, an interface to "Dragon Naturally Speaking" will be implemented, which can be used to control various AsTeRICS functionalities via voice commands.	Table 4
URGEN3	Game access	AsTeRICS should interface with the newest computer games.	Medium	Yes	Game control via the universal HID actuator (mouse/keyboard/joystick emulation) is already possible and will be improved in the final prototype. For the second user evaluation study, dedicated models and scenarios will be available.	Table 2 Table 4
URGEN4	Extending the environmental control	AsTeRICS should control not only the computer but also mobile phones, iPads, etc.	Medium	Party	An interface to Android mobile platforms will be developed for the final prototype. Furthermore, alternative environmental control devices like door openers will be integrated into the system.	Table 4
URGEN5	Models for users with perception problems	Some users are not able to use many devices at the same time because of their perception problems. For theses users a special model should be prepared e.g. with Oska running on the controlled PC, not on the MIMO	High	Maybe	A dedicated model will be developed together with users in course of WP-6	Table 4

URGEN6	Needed models for users who can puff only	There should be models created for the Sip and Puff sensor users who are able to puff only.	High	Yes	Models with Puff-only interaction will be developed	
URGEN7	Joystick mode model	The model for using a webcam in joystick mode should be improved because it was useless for most of the users.	Medium	Yes	Adaptations of this model will be provided	
URGEN8	Camera mouse pause.	There should be an option to pause the camera mouse.	Low	Yes	This option can be added to models and/or plugins	Table 4
URGEN9	Better models for camera mouse	Some users have very limited head movements or different head movement capabilities in different directions. There should be models created which enable the user with limited movements to move the mouse pointer on entire screen, where each direction of movement is configured separately.	High	Yes	Adaptations of the facetracker / mouse integration can be provided and evaluated together with users (in course of WP-6)	Table 4
URGEN10	Positioning of the devices	All AsTeRICS sensors and actuators should be positioned better for the user. Appropriate arms, hand-rests, anti-slide pads, should be adapted.	Medium	Yes	Different mounting options will be tested and provided for the final prototype. The “Manfrotto Magic Arm” will be used as universal mounting device. Furthermore, a special bag will be designed which allows to better carry the final prototype hardware or mount it on a wheelchair.	Table 1
URGEN11	Reduction of involuntary movements	AsTeRICS should better support people with reduced fine motor abilites or involuntary movements	High	Maybe	For this taks, new plugins need to be developed which allow adaptive / learning algorithms which can seperate voluntary from involuntary movements. The iGesture library will be evaluated in course of this work for the final prototype	Table 4

2.1.2 User-driven hardware requirements

ID	Name	Description	Priority	Feasible	Comments	Table
URHD1	Reduction of cables.	AsTeRICS has too many cables. This is inconvenient for the users. There should be more wireless connections to the platform. Sensors such as: sip/puff, accelerometer should be connected wirelessly.	High	Yes	This is already planned for PT2 - the CIMs will be integrated with the PC into one compact box and key wireless components will be developed.	Table 1
URHD2	AsTeRICS should	The AsTeRICS platform should be designed to be	High	Yes	PT2 will be a compact box with a battery backup and	Table 1

	be portable	carried with the user.			a carrying bag for it will be designed.	
URHD3	Handle to carry the platform	The platform should have a handle designed to allow a less able user to carry the platform.	Medium	Yes	The customised bag for the final prototype hardware will have a handle and can be mounted and carried easily.	Table 1
URHD4	Shock resistance	The platform should be shock resistant, because the less able user can easily drop it.	Medium	Yes	The compact box for the final prototype is durable and special attention is payed to the internal mechanical construction to make it more resistant. Furthermore, the carrying bag will provide an additional shelter to the platform.	Table 1
URHD5	Water and frost resistance	The platform should be water and frost resistant, if it is used in an outdoor environment.	Medium	Partly	Not the platform itself but the carrying bag will provide certain level of resistance against water and frost.	Table 1
URHD6	D-SUB input	Platform should have a D-SUB input, because it is more common than DVI.	Low	No	D-Sub is an absolete analog video format and is not planned for PT2. Would be paid by additional unwanted space waste and power consumption. However, an exteranl converter can be used.	Table 1

2.1.3 User-driven ACS requirements

ID	Name	Description	Priority	Feasible	Comments	Table
URACS1	ACS should be easier.	ACS is too complicated for secondary users. Some easy configuration methods should be implemented for non-expert secondary users.	High	Partly	Several key values will be adjustable within a special interface (called EasyConfig) on the ARE. It will become possible to design a graphical Interface for the ARE on ACS. Furthermore, the grouping of elements will make the desing of model much easier.	Table 6
URACS2	Automatic backup of models	Automatic model backup is needed for fast cancellation of changes introduced by the user to the model.	Low	Yes	Will be part of PT-2 development	Table 6
URACS3	Proper documentation.	Many users will not be able to use ASC without the proper documentation.	High	Yes	Will be part of PT-2 development	Table 6

2.1.4 User-driven ARE requirements

ID	Name	Description	Priority	Feasible	Comments	Table
URARE1	Alternative method to change	Users should be able to change the current model using their standard sensors.	High	Yes	Alternative methods for switching models will be installed	Table 4

	the current model.					
URARE2	Priority action	AsTeRICS should have some priority actions, for example: to call for help.	Medium	Maybe	It depends on how these actions will be integrated into the system: integrating e.g. a button which calls a certain person via mobile phone will be possible, if these functions are integrated into a model.	Table 4
URARE3	Keep sending IR code	There should be a component (or the IR component should be improved) to keep sending a code for a certain time	Medium	Yes	The IR-plugin can be modified to resend IR codes for a desired timespan.	Table 4
URARE4	Face recognition algorithm should be improved	Sometimes, the algorithm recognizes other objects instead of the user's face. Sometimes the algorithm keeps recognizing the faces of people standing behind the user. The algorithm also has problems with the user face recognition when the user's head is tilted or the user is lying down.	High	Maybe	Alternative face recognition algorithms will be (and have been) explored. However, the downside of these – more stable – algorithms is the recognition speed. However, these more complex algorithms can be used on a high performance computing platform.	Table 4

2.1.5 User-driven sensor- and actuator requirements

ID	Name	Description	Priority	Feasible	Comments	Table
URSA1	Home equipment interface	AsTeRICS should be able to perform actions such as: open and close water tap, open and close window screen, flush the toilet, control a massager or electric bed.	Medium	Partly	A door opener will be integrated into the list of supported devices for the final prototype. If we can integrated other devices will depend on cooperations with 3rd parties. This will be probably a focus of post-project activities.	Table 2
URSA2	Eye control	AsTeRICS should also have eye control input.	High	Yes	The second prototype of the AsTeRICS Computer Vision System will provide a head-mounted iris tracking device.	Table 2 Table 4
URSA3	Nintendo’s Wii	Nintendo’s Wii should be integrated with AsTeRICS	Low	Maybe	This will be done if the project resources permit it (low-priority task)	Table 2 Table 4
URSA4	Alternative sensor to 6 switch mouse	There should be a single sensor (e.g.: joystick) which will be an alternative to 6 switch mice.	Medium	Yes	Joystick with digital switches can be interfaced to the 6-switch mouse model easily	Table 2
URSA5	Discreet sensors	Sensors used outdoors should be discreet and not too visible for other people.	Medium	Partly	The final prototype will have less cables and more wireless interfaces. This will allow to mount several sensors in a more discreet manner.	Table 2
URSA6	Gripper handle	The Gripper tube should be longer, the mouth-piece needs to be redesigned to provide better grip with the teeth.	Low			Table 2
URSA7	Gripper should be lighter for mouth-stick users.	The Gripper is too heavy for mouth-stick users.	Low			Table 2

URSA8	Attach the gripper to a robot-arm	Gripper should be also attached to the robot-arm. The robot-arm should be also controlled by AsTeRICS.	Low			
URSA9	MIMO sensitivity	MIMO sensitivity should be configurable for work with OSKA. MIMO is too sensitive for some users and they sometimes write many letters on one touch. Some other users have to touch MIMO many times to write a letter.	Medium	Maybe	The MIMO display's sensitivity for the resistive touch screen cannot be changed – However, the OSKA application could provide additional feature for key-debouncing or minimum selection time for touchscreen-based interaction	Table 4
URSA10	Larger Screen	AsTeRICS should also have a larger screen to read from a distance or/and from an unusual angle.	Medium	Yes	Due to the flexible system architecture, the ARE can be deployed to different target platforms (netbooks, laptop computers, Windows-7 tablets). Furthermore, alternative computer screens could be connected to the AsTeRICS Personal Platform via DVI connector.	
URSA11	MIMO should have sun protection.	It is hard to see what is on the MIMO in the sun	Medium	Maybe	If the project resources permit, a sun-shield solution for the MIMO screen will be constructed and evaluated.	Table 1

2.1.6 User-driven Enobio requirements

ID	Name	Description	Priority	Feasible	Comments	Table
UREN1	Blink recognition	Enobio needs better recognition of voluntary user blinks. For some users, Enobio frequently takes action on involuntary blinks or other head movements. For others Enobio doesn't recognize voluntary blinks at all.	High	Maybe	The blink detection plugin will be updated to accept custom parameters so the algorithm might show better performance by customizing them.	Table 4
UREN2	Blink recognition should be faster	The delay between the user's blink and its recognition is about 2s. This causes problems in using Enobio as an input for OSKA scanning or games.	Medium	Maybe	The signal filtering, which causes such a delay, might be improved. However there is a trade-off between the filtering and the output performance of the algorithm.	Table 4
UREN3	Improvement of Enobio electrodes activation.	The electrodes didn't activate for some users.	High	Maybe	There will be a new enobio headband and new electrode front end that will have the pellet more accessible so the contac will be improved.	Table 2
UREN4	Alternative mounting for communicate module.	Many of the users have headrests which cause difficulties in mounting the Enobio communication module in the headband on the back of the head.	Medium	Yes	There will be a new cap where the position of the Enobio could be changed to the top of the head.	Table 2
UREN5	Easy Enobio setting.	Some users with minor disability should be able to put Enobio on the head themselves.	Low	Maybe	To put the new Enobio cap will be slightly easier than the current headband.	Table 2

2.1.7 User-driven Oska requirements

ID	Name	Description	Priority	Feasible	Comments	Table
UROS1	Native Keyboards	All OSKA keyboards should be fully localised (including control keys)	High	Yes	Local partners can use the Oska Keyboard Editor to localise keyboards	
UROS2	Alternative to Scanning	Scanning is too slow for some users. An alternative method for one-switch key selection should be found.	Medium	Maybe	It is the technique of scanning that is too slow or that the scanning needs to be quicker. For an alternative technique look at the One Switch Mouse. If the scanning needs to be quicker then change the speed in the settings editor.	
UROS3	Different keyboards	Different keyboards should be prepared for different users' needs. E.g.: with less keys on the keyboard, bigger keys, with larger distance between keys.	High	Yes	The Oska Keyboard Editor can be used to make keyboards with less keys. Oska is being changed to allow bigger keys and to allow more distance between the keys.	Table 4
UROS4	Scanning frame	Scanning frames should be more visible. Maybe there should be also the option for colour inversion.	High	Yes	Two more styles of scanning frames have been implemented (including inversion)	Table 4
UROS5	Scanning cancel	There should be possibility to cancel a wrong selected column or row.	High	Maybe	Technique to be discussed.	Table 4
UROS6	Prediction	OSKA should have a prediction option.	Medium	Yes	This is being implemented	Table 4

2.2 Hardware Requirement Updates

Nr.	Requirement	Description	Priority	Proto type	Status
HW1	Size and weight	The Embedded Platform (EP) hardware consists of CORE Computer Module (CCM) and pluggable small Computer Interface Modules (CIM). The CIMs can be plugged into the CCM and can be combined as needed. The CIMs shall support standalone function.	H	2	Modified
HW2	Low power consumption	The hardware components have to operate with low power requirements and/or offer power management functions for acceptable battery lifetime.	M	1, 2	Unchanged
HW3	Temperature Range	The components shall have 0 to 60 ˚C temperature range.	H	1, 2	Modified
HW4	Performance	The CCM of the embedded platform has to be powerful enough to support the AsTeRICS Runtime Environment. The embeded platform shall be based on Intel Atom Z510 or Z530 or newer processor	H	2	Modified
HW5	Hot-Pluggability	A connection or disconnection of a component should not affect or interrupt the functionality of the rest of the system. An automatic detection of a newly connected or disconnected peripheral is desired.	L	1, 2	Unchanged
HW6	Affordable price	The whole system has to be affordable at acceptable cost, compared to other available AT products with similar or less functionalities.	H	2	Unchanged
HW7	Portability	The system can be battery powered and is usable without a connection to the mains power supply for at least 20 minutes (TBD).	H	2	Modified
HW8	Platform Interfaces	The EP supports the hardware interfaces which are necessary for connecting sensors and actuators in sufficient amount (Bluetooth, 4 x USB, ZigBee, WiFi, Ethernet).	H	1, 2	Modified
HW9	UART interface	The EP supports integration with UART interface by external USB to UART interface.	M	1, 2	Modified
HW10	LCD interface	The EP supports integration with an LCD display with touchscreen.	H	2	Unchanged Done
HW11	Sufficient Memory	A large non volatile memory is essential to hold operating system, ARE components and system model configurations (at least 4GB). A large RAM is needed for the runtime system to work efficiently. (at least 512MB, depending on the Operating System).	H	1, 2	Unchanged
HW12	Operating System	For low latency processing of parallel sensor values (multiprocessing), TCP/IP stack, memory management, JAVA Virtual Machine, Windows or Linux based OS solution should be supported.	M	2	Unchanged
HW13	Box	The prototype box shall allow RF transmission from inside	H	2	New

Table 1: Hardware requirements for the AsTeRICS embedded platform

2.2.1 Updated Requirements for AsTeRICS Pluggable Hardware Components

The table below lists the hardware requirements for pluggable components, such as sensors, processors and actuators that the AsTeRICS system is expected to support by default.

Nr.	Requirement	Description	Priority	Proto type	Status
Hardware Requirements for pluggable sensors					
HSEN1	Generic switches (GPIO CIM)	The AsTeRICS system supports connectivity to at least 5 generic switches via a dedicated module (GPIO-CIM, 3,5mm jacks, digital input).	H	1,2	Unchanged
HSEN2	Sweety!	The AsTeRICS system should support Sweety! Switches via Bluetooth connectivity.	L	2	Unchanged
HSEN3	USB HID devices	The AsTeRICS system supports USB mice, keyboards and joysticks via USB connectivity.	H	1, 2	Unchanged
HSEN4	9 DOF Razor IMU	The AsTeRICS system supports the 9 DOF Razor Inertial Measurement Unit.	H	1, 2	Unchanged Done
HSEN5	Strain Gauge	The AsTeRICS system supports Strain Gauge connectivity (via ADC-CIM).	H	1, 2	Unchanged Done
HSEN6	Touchscreen	The AsTeRICS system supports Touchscreen connectivity (e.g. via USB).	H	2	Unchanged Done
HSEN7	Enobio	The AsTeRICS system supports connectivity with the Enobio System (wired or via ZigBee).	H	1, 2	Unchanged
HSEN8	Enobio	The AsTeRICS system supports wireless connectivity with the Enobio System	H	2	New
HSEN9	Enobio	Enobio receiver shall be physically integrated in the prototype box	H	2	New
Hardware Requirements for pluggable actuators					
HACT1	Mobile Phone Interface	The AsTeRICS system supports integration with Mobile Phone Interface via Bluetooth wireless link	H	2	Unchanged
HACT2	LC-Display Interface	The AsTeRICS system should support integration with an LCD interface (e.g a USB-pluggable LCD with touchscreen)	H	1, 2	Unchanged Done
HACT3	IR Connectivity	The AsTeRICS system should support infrared connectivity by interfacing to a suitable IR gateway	H	2	Unchanged Done
HACT4	Digital to Analog Converter (DAC-CIM)	The AsTeRICS system provides a Digital-to-Anolog Converter module (DAC-CIM) with 4 channels (minimum) between 0 and to 24 Volt.	M	2	Modified
HACT4	Digital to Analog Converter (DAC-CIM)	The AsTeRICS system provides a Digital-to-Anolog Converter module (DAC-CIM) with 5 channels (minimum) between 0 and to 25 Volt. At least 2 channels provide high power up to 5 Watts	M	2	Unchanged
HACT5	HID Mouse Actuator	The AsTeRICS system supports mouse emulation for a PC via the USB HID device class	H	1, 2	Unchanged Done
HACT6	HID Keyboard Actuator	The AsTeRICS system supports keyboard emulation for a PC via the USB HID device class	M	2	Unchanged Done
HACT7	HID Joystick Actuator	The AsTeRICS system supports Joystick emulation for a PC via the USB HID device class	H	2	Unchanged Done
HACT8	KNX	The AsTeRICS System supports integration with the KNX homeautomation standard via the KNX-IP gateway	H	2	Modified

Hardware Requirements for Non-Classical PC-User Interfaces					
HNCPUI1	3D accelerometers	The AsTeRICS system supports connectivity with 3D accelerometers	H	2	Unchanged Done
HNCPUI2	Webcam as Face Mouse	The AsTeRICS system should support connectivity with webcams that serve as face mice	L	2	Unchanged Done
HNCPUI3	Webcam for other input modalities (e.g. colour tracking)	The AsTeRICS system should support connectivity with webcams that serve as other input modalities	L	2	Unchanged
HNCPUI4	External Touchpad/keypad (used in novel ways, e.g. in “Joystick” mode)	The AsTeRICS system supports connectivity with Touchpad / Keypad that can be used in novel ways for PC-User interfacing	H	2	Unchanged
HNCPUI5	GamePads/ Joysticks	The AsTeRICS system supports connectivity with GamePads/Joysticks	M	2	Unchanged
HNCPUI6	Mobile phone with touch screen	The AsTeRICS system supports connectivity with Mobile Phones with touch screens	M	2	Unchanged

Table 2: Hardware requirements for the AsTeRICS pluggable components

2.2.2 Updated Hardware Requirements for the AsTeRICS Smart Vision Module

The table below lists the requirements expected to be provided by the Smart Vision Module which is going to be developed during the project.

Nr.	**Requirement**	**Description**	**Priority**	**Proto type**	**New (Y/N)**
SVM1	Eye camera	The Smart Vision Module should acquire images from an eye camera	H	2	Modified
SVM2	Scene camera	The Smart Vision Module should acquire images from a scene camera	H	2	Modified
SVM3	Inertial Measurement Unit	The Smart Vision Module should acquire data from an IMU and process them	L	2	Modified Done
SVM4	Synchronization interface	The Smart Vision Module should synchronize the data of the eye and scene cameras and the IMU	L	---	Dropped
SVM5	Head-mounted support	The head-mounted system of the Smart Vision Module should be lightweight and intrusiveness should be minimized	H	2	Modified

Table 3: Hardware requirements for the AsTeRICS Smart Vision Module

2.3 Software Requirement Updates

2.3.1 Updated Software Requirements for Pluggable Component Modules

The table below lists the software requirements for pluggable components (software plugins), such as sensor-plugins, processor-plugins and actuator-plugins that the AsTeRICS system is expected to support by default. Each hardware requirement in Section 2.2.1 should have a counterpart software requirement in the table below. In addition requirements for purely software components are listed below.

Nr.	Requirement	Description	Priority	Proto type	Status
Software Requirements for sensor plugins					
SSEN1	GPIO/ Generic switches plugin	A software plugin exists which makes the state of the external generic switches available on its output ports.	H	1,2	Unchanged Done
SSEN2	Sweety! plugin	A software plugin exists which makes the state of the Sweety! – buttons (connected via Bluetooth) available on its output ports.	L	2	Unchanged
SSEN3	USB HID class support	The operating system of the AsTeRICS Embedded Platform supports USB devices classes and HID host functionality to connect mice, keyboards and joysticks.	H	1, 2	Unchanged Done
SSEN4	9 DOF Razor IMU plugin	A software plugin exists which provides data from the 9 DOF Razor Inertial Measurement Unit.	H	1, 2	Unchanged Done
SSEN5	ADC/Strain Gauge plugin	A software plugin exists which provides ADC data from the ADC CIM (provide e.g. Strain Gauge data).	H	1, 2	Unchanged Done
SSEN6	Touch-screen	The AsTeRICS system and/or the operating system of the EP supports touchscreen connectivity.	H	2	Unchanged Done
SSEN7	Enobio	A software plugin exists, which supports channel data readout of the Enobio system.	H	1, 2	Unchanged Done
SSEN8	FTDI chip driver	In case a USB2.0 connection to the Enobio system is used, the FTDI driver has to be available in the operating system.	H	1,2	Unchanged Done
SSEN9	IEEE 802.15.4 antenna driver	In case a built-in wireless receiver is used, its software driver shall be available for the Enobio software control.	M	2	Unchanged Done
SSEN10	Serial port driver	The operating system allows access to the serial port / virtual COM port to provide wired connections to UART/RS232 devices.	M	1,2	Unchanged Done
Software Requirements for actuator plugins					
SACT1	Mobile Phone Interface plugin	A software plugin exists which supports integration of one or more mobile phones (send SMS, call a number).	H	2	Unchanged
SACT2	LC-Display Interface support	The AsTeRICS system and/or the operating system of the EP supports integration of a portable LCD module with touchscreen.	H	1, 2	Unchanged Done

SACT3	IR Connectivity plugin	A software plugin exists which supports interfacing with a suitable infrared gateway.	H	2	Unchanged Done
SACT4	Digital to Analog Converter plugin	A software plugin exists which can control the Digital-to-Analogue converter module (DAC CIM).	M	2	Unchanged Done
SACT5	HID Mouse Actuator plugin	A software plugin exists which supports mouse emulation for the PC via the HID mouse actuator.	H	1, 2	Unchanged Done
SACT6	HID Keyboard Actuator plugin	A software plugin exists which supports keyboard emulation for the PC via the HID keyboard actuator.	M	2	Unchanged Done
SACT7	HID Joystick Actuator plugin	A software plugin exists which supports joystick emulation for the PC via the HID joystick actuator.	H	2	Unchanged Done
SACT8	KNX	The AsTeRICS System supports integration with KNX via the KNX-IP gateway.	H	1, 2	Modified
Software Requirements for signal processing plugins					
SPROC1	Filtering EMG signal: 8-500 Hz	Frequency filtering to be applied to ENOBIO channels shall be implemented.	H	1	Unchanged
SPROC2	Filtering Alpha band and Mu Rhythm: 8-12Hz	Frequency filtering to be applied to ENOBIO channels shall be implemented.	H	1	Unchanged
SPROC3	Filtering Beta band: 12-30 Hz	Frequency filtering to be applied to ENOBIO channels shall be implemented.	H	1	Unchanged
SPROC4	High pass filter with frequency cut at 1 Hz	Frequency filtering to be applied to ENOBIO channels shall be implemented.	H	1	Unchanged
SPROC5	Laplacian filter analysis	The feasibility and usefulness of Spatial Laplacian filter for the reduced number of ENOBIO channels shall be analyzed.	L	2	Unchanged
SPROC6	FFT computation	Computation of Power Spectrum Density through FFT.	H	1	Unchanged
SPROC7	Epoch cutting	Epoch cutting.	H	1	Unchanged
SPROC8	Epoch Averaging	Epoch Averaging.	H	1	Unchanged
SPROC9	Linear Transformation	Linear Transformation (matrix product).	H	1	Unchanged
		Transformation matrix will be defined offline (this will allow online ICA, PCA, CSP, linear inverse solution, weighted mapping, CAR referencing, simple Laplacian, etc…).	H	2	Modified
SPROC10	Threshold	Application of a threshold to transform continuous output into a binary output.	H	1	Unchanged
SPROC11	Derivative	Derivative of some selected channels.	H	1	Unchanged
SPROC12	Decimation	Decimation of some selected channels.	H	1	Unchanged
SPROC13	Dissimilarity	Dissimilarity.	H	1	Unchanged
SPROC14	PCA projection	PCA projection (projection matrix can be pre-computed).	M	2	Modified
SPROC15	CSP and OVR	CSP and OVR.	L	2	Unchanged
SPROC16	CVT	CVT.	L	2	Unchanged
SPROC17	SVEM	SVEM (Support Vector Machine).	M	2	Unchanged
SPROC18	LDA	LDA.	M	2	Unchanged
SPROC19	Fuzzy C-means	Fuzzy C-means.	M	2	Unchanged

SPROC20	Cross Correlation	Correlation function for template matching.	H	1	Unchanged
Software Requirements for BNCI signal processing plugins					
SBPROC1	Teeth grinding detection	The output will be a numerical value within a range.	H	2	Unchanged
SBPROC2	Frown detection	Frown detection to be analyzed. The output will be a numerical value within a range.	L	2	Modified
SBPROC3	Blink detection	Blink detection. The output will be binary.	H	1	Unchanged
SBPROC4	Double blink detection	Double blink detection to be analyzed.	H	1	Unchanged
SBPROC5	Horizontal eye movement's detection	Horizontal eye movement detection. The output will be single events (right/left).	L	2	Modified
SBPROC6	Winks detection	Winks detection to be analyzed.	L	2	Modified
SBPROC7	Mouth movement	Mouth movement detection to be analyzed.	L	2	Modified
SBPROC8	Emotional content	Emotional content alpha asymetry to be analyzed.	L	2	Unchanged
SBPROC9	Motor imagery	Methodologies for motor imagery interface shall be implemented	H	2	Modified
SBPROC10	Detection of language thinking	Detection of language thinking to be analyzed for binary.	L	2	Unchanged
SBPROC11	EMG detection	General methodology for muscle movement detection shall be implemented (subsumes SBPROC7-9)	M	2	New
SBPROC12	VEP methodologies	Methodologies for controlling OSKA through VEP shall be implemented	M	2	New
Software Requirements for AsTeRICS Smart Vision Module					
SSVM1	Eye detection	An eye detection algorithm should be proposed/modified and implemented	H	1,2	Unchanged
SSVM2	Gaze estimation	A gaze estimation algorithm should be proposed/modified and implemented	H	1,2	Unchanged

Table 4: Software requirements for the AsTeRICS pluggable components

The Requirements from SPROC1 to SPROC20 are extracted from the SotA review on EEG, EOG and EMG based assistive technologies and Report on Pattern Recognition Technologies for BNCI. The purpose of the signal processing functionality set to be integrated in the ARE is online process of different physiological signals to build assistive technologies.

2.3.2 Updated Software Requirements for the AsTeRICS Runtime Environment

The AsTeRICS Runtime Environment provides the execution framework for the pluggable components and supports the ASAPI communications. For more details on the updated ARE specification please see section 5.1. The table below lists the updated requirements for the ARE.

Nr.	**Requirement**	**Description**	**Priority**	**Proto type**	**Status**
ARE1	Support sensor plugins	The ARE supports Sensing modules which provide information via output ports.	H	1,2	Unchanged Done

		For low level sensors that need OS/driver communication we should use JNI for communicating data to ARE.			
ARE2	Support processor plugins	The ARE supports processing elements which transform information from input ports and provide results on output ports.	H	1, 2	Unchanged Done
ARE3	Support actuator plugins	The ARE supports actuator elements, using information from input ports that might be connected to the output port of a sensor or processor plugin. May interface to operating system / driver layer via JNI.	H	1, 2	Unchanged Done
ARE4	Dynamic deploy, activation, deactivation and removal of components	ARE supports dynamic deploy, activation, deactivation and removal of elements. ARE should be build upon OSGi / Java to enable this feature.	H	1, 2	Unchanged Done
ARE5	Remote access to the platform	The ARE supports remote access to the platform via a communication network.	H	1, 2	Unchanged Done
ARE6	Feedback and error reporting	The ARE supports querying deployed components status and error reporting.	M	2	Unchanged Done
ARE7	Load / store configuration on platform	The ARE supports loading an existing configuration system model and storing a new or changed configuration system model from/to a hard disk.	M	2	Unchanged Done
ARE8	Computing load can be monitored	The ARE should be capable of monitoring the system computing load.	L	2	Unchanged
~~ARE9~~	~~Direct manipulation of pluggable components~~	~~The ARE allows setting and manipulating element properties directly (not through model submission) through a communication network or other direct interaction with the ARE (buttons, touch screen, etc).~~	~~M~~	~~---~~	Dropped
Requirements for signal connections and data flow within the ARE					
ARE10	Data types on Input / Output ports	Each Input and Output port attached on a component has a specific data type. Supported data types include boolean, int, longint, real (float), int-vector, float-vector, string, state; while new data types for new elements should be possible.	H	1, 2	Unchanged Done
ARE11	Restricted connections	Connections (channels) can only be established from one output port to one or many input ports of the same type. Only matching connections can be established, also it is not possible to connect more than one signal to an input port.	H	1, 2	Unchanged Done
ARE12	Metadata on ports	Each port can be connected with some type of metadata about its capabilities, data type, signal bandwidth, block size, timestamp, etc.	H	1, 2	Unchanged
ARE13	Data flow	Data can be transferred via channels (from output to input ports) as single values or in blocks (data chunks) if deemed necessary.	H	1, 2	Unchanged
~~ARE14~~	~~Channel update rate~~	~~The update rate of a channel is defined by the metadata of the output port. This rate can be fixed or changed according to the output port activity.~~	~~M~~	~~2~~	Dropped No need for channel update rate
ARE15	~~Synchronous and~~ Asynchronous events reaction	Components are able to listen to Synchronous or Asynchronous events (through ARE) and react accordingly.	H	1, 2	Modified (only asynchrono

					us events)

Table 5: Software requirements for the AsTeRICS Runtime Environment (ARE)

2.3.3 Updated Requirements for the AsTeRICS Configuration Suite

The AsTeRICS Configuration Suite (ACS) is mainly used to graphically design the layout of the system, as a network of interconnected components. ACS should also support the deployment of additional components, such as the design of an easy configuration user interface at the ARE. Using this easy configuration interface, selected properties can be adapted at the ARE during runtime. A full specification of the ACS is given in Section 5.2

Nr.	Requirement	Description	Priority	Proto type	Status
ACS1	Graphically design system model	ACS enables graphical design of the system model on a host PC as a network of components. Arrangement of the graphical elements as well as properties configurations should be enabled via mouse / keyboard actions.	H	1, 2	Unchanged Done
ACS2	Graphically interconnect elements	The ACS GUI allows for connecting two components through their output/input ports. Drawing of connections (channels) from output to input port defines data flow, channel and port parameters should be adjustable.	H	1, 2	Unchanged Done
ACS3	User-friendly graphical user interface	The Configuration Suite provides a user friendly and accessible graphical user interface. It is important that usage of the ACS is as easy and accessible as possible.	H	2	Unchanged
ACS4	Support connection with the ARE	The ACS is able to connect to the ARE the ASAPI. The connection should be used to download or upload the system model, changing plugin parameters and logging reports.	H	1, 2	Unchanged Done
ACS5	Components can be grouped and ungrouped in the graphical display	To handle complexity of larger designs, components can be grouped and ungrouped in the graphical display.	H	2	Unchanged
ACS6	Edit properties	Display and edit properties and metadata information on components and ports.	H	1	Unchanged Done
ACS7	Easy configuration user interface	Design an user interface out of predefined components. This interface will be running on the ARE, setting up some selected properties during runtime	M	2	New
ACS8	Internationalisation	Beside the UI of the ACS itseltf, all port and property names should support defferent languages	L	2	New

Table 6: Requirements of the AsTeRICS Configuration Suite

2.3.4 Updated Requirements for the AsTeRICS Application Programming Interface (ASAPI)

The AsTeRICS Application Programming Interface (ASAPI) provides a well defined method for software applications like the AsTeRICS Configurations Suite (ACS) or third party applications to configure, monitor and control the AsTeRICS Runtime Environment (ARE). Section 5.3 provides detailed specification of ASAPI. The table below collects the requirements for the AsTeRICS application programming interface (ASAPI).

Nr.	Requirement	Description	Priority	Proto type	Status
ASAPI1	Detection of ARE	ASAPI provides methods for detecting an instance of ARE server.	H	1, 2	Unchanged
ASAPI2	Connection with ARE	ASAPI provides methods for establishing a connection with a detected ARE server.	H	1, 2	Unchanged Done
ASAPI3	Handle communication errors	ASAPI provides methods for handling ARE communication errors and breakdowns.	M	1, 2	Unchanged
ASAPI4	Query installed plugins	ASAPI provides methods for querying plugins installed in ARE directly (i.e, without retrieving the whole system model). Queries should return plugins' parameters and properties.	H	1, 2	Unchanged Done
ASAPI5	Install plugin	ASAPI provides methods to install plugins to the ARE.	H	1, 2	Unchanged Done
ASAPI6	Create plugin instance	ASAPI provides methods for creating new instance of installed plugin.	M	1, 2	Unchanged Done
ASAPI7	Ports connection	ASAPI provides methods for interconnecting installed plugins.	H	1, 2	Unchanged Done
ASAPI8	Run/Stop plugins	ASAPI provides methods for starting or stopping individual plugins.	M	1, 2	Unchanged Done
ASAPI9	Retrieve system model	ASAPI provides methods for retrieving the system configuration model in a serializable format.	H	1, 2	Unchanged Done
ASAPI10	Deploy system model	ASAPI provides methods for deploying a new configuration system model.	H	1, 2	Unchanged Done
ASAPI11	Run/Stop deployed model	ASAPI provides methods for initiating or terminating the execution of deployed model.	H	1, 2	Unchanged Done
ASAPI12	Retrieve logging information	ASAPI provides functions for retrieving logging. and status information from the runtime environment.	M	1, 2	Unchanged Done
ASAPI13	Exchange live data	ASAPI provides methods for exchanging live data with connected ARE.	L	2	Modified So far no need for this, changed to Low priority
ASAPI14	Native Interface	Certain functions are provided natively (in C#) for PC AT developers via the Native ASAPI (e.g. for making mobile phones or special sensors available).	M	1,2	Unchanged

Table 7: AsTeRICS Application Programming Interface (ASAPI) requirements

2.4 BNCI Evaluation Suite Requirement Updates

The table below lists the requirements for the BNCI Evaluation Suite. The requirements table is divided into *functional* and *other* requirements. They present priorities and feasibility of the requirements as established.

Nr.	Description	Feasi-bility	Priority	Proto=type	Status
BNCI Evaluation Suite Functional Requirements					
BNCI1	BNCI Evaluation Suite shall not work in real-time.	H	H	1, 2	Unchanged

BNCI2	Recalling parameters for real-time on-line processing might be allowed.	M	M	2	Unchanged
BNCI3	The input file data formats will be the standard ones used by the acquisition hardware.	H	H	1	Unchanged
BNCI4	Data formats used by BIOSIG toolbox [15] shall be accessed from the Evaluation Suite.	H	H	1	Unchanged
BNCI5	Starlab data cube (SDC) format shall be used.	H	H	1, 2	Unchanged
BNCI6	Classification functionalities of the BIOSIG toolbox shall be interfaced from the Evaluation Suite.	H	H	1	Unchanged
BNCI7	Temporal windowing of trial sequences shall be implemented.	H	H	1	Unchanged
BNCI8	Temporal 8-order Chebyshev Type I band-pass filtering of trial sequences shall be implemented.	H	H	1	Unchanged
BNCI9	Different projection techniques for dimensionality reduction shall be implemented.	H	L	2	Modified
BNCI10	A decision tree methodology might be implemented.	M	L	2	Modified
BNCI11	Re-sampling training functionalities might be implemented.	H	L	2	Unchanged
BNCI12	Data fusion operators shall be implemented for integration in classifier ensemble methodologies.	H	H	1	Unchanged
BNCI13	Genetic algorithms might be implemented for system optimization.	H	M	1	Unchanged
BNCI14	A fuzzy control algorithm shall be implemented.	H	L	2	Modified
BNCI15	Feature extraction based on Bereitschaft potential shall be implemented	H	L	2	Modified
BNCI16	PSD estimation based on multitaper approaches shall be implemented.	H	L	2	Modified
BNCI17	An approach based on mutual information ranking shall be implemented for feature selection.	M	L	2	Modified
BNCI18	SVEM shall be implemented.	H	H	1	Unchanged
BNCI19	Linear proximal SVEM shall be implemented.	M	H	1	Unchanged
BNCI20	A logistic regression algorithm shall be implemented.	M	L	1	Unchanged
BNCI21	Linear discriminant analysis shall be implemented.	H	H	1	Unchanged
BNCI22	Temporal decimation of trial sequences shall be implemented.	H	H	1	Unchanged
BNCI23	Temporal averaging of trial sequences shall be implemented.	H	H	1	Unchanged

BNCI24	Performance evaluation shall be implemented based on TPR and CA measures.	H	H	1	Unchanged
BNCI25	Wavelet transformation of a temporal sequence shall be implemented.	M	H	1	Unchanged
BNCI26	A procedure for analysis of variance (ANOVA) shall be implemented.	H	M	1	Unchanged
BNCI27	At least one performance measures for BCI shall be implemented, e.g. kappa, TPR/FPR, ITR.	H	H	1	Unchanged
BNCI28	Recall parameters of a particular framework might be saved on disk for posterior on-line processing.	L	M	2	Unchanged
BNCI29	An application based on P300 for image browsing shall be implemented.	M	M	2	Modified
BNCI30	P300 classical approach shall be implemented.	H	M	2	Modified
BNCI31	Classical BCI based on motor imagery might be implemented.	H	H	2	Modified
BNCI32	An approach for single-trial source reconstruction for BCI might be implemented.	M	L	2	Modified
BNCI33	StarEEGlab toolkit user interface shall apply.	H	H	1	Unchanged
BNCI34	Interface through Visual Evoked Potential (VEP) shall be analysed	M	M	2	New
BNCI35	Methodology for Spatial Filter extraction to be used in VEP shall be implemented	M	H	2	New
BNCI36	Notch filter to be used in VEP shall be implemented	H	H	2	New
BNCI37	Eigenvalue decomposition for VEP shall be implemented	H	H	2	New
BNCI Evaluation Suite Other Requirements					
BNCI34	BNCI Evaluation Suite shall be implemented in two phases. Prototype 1 (PT1) to be delivered till 30/04/2011.	H	H	1	Unchanged
BNCI35	BNCI Evaluation Suite shall be implemented in two phases. Final Prorotype (FP) to be delivered till 31/05/2012.	H	H	2	Unchanged
BNCI36	Starlab shall integrate the Evaluation Suite functionalities within the StarEEGlab toolkit.	M	H	2	Unchanged

Table 8: BNCI Evaluation Suite requirements

3 State of the Art Analysis

3.1 Analysis of Hardware Platform

For the second prototype, the decision has been made to replace the Kontron pITX-SP board with a nanoETX form factor solution. This will allow more design freedom, smaller form factor and lower power consumption for the final AsTeRICS platform (compared to the first prototype), while keeping software compatibility and reuse of the existing PT-1 software framework at a maximum level.

A further advantage is also possibility of upgrade in future to a newer embedded computer module of the same form factor (COM Express compatible size, 84 x 55 mm, Pin-out type 1).

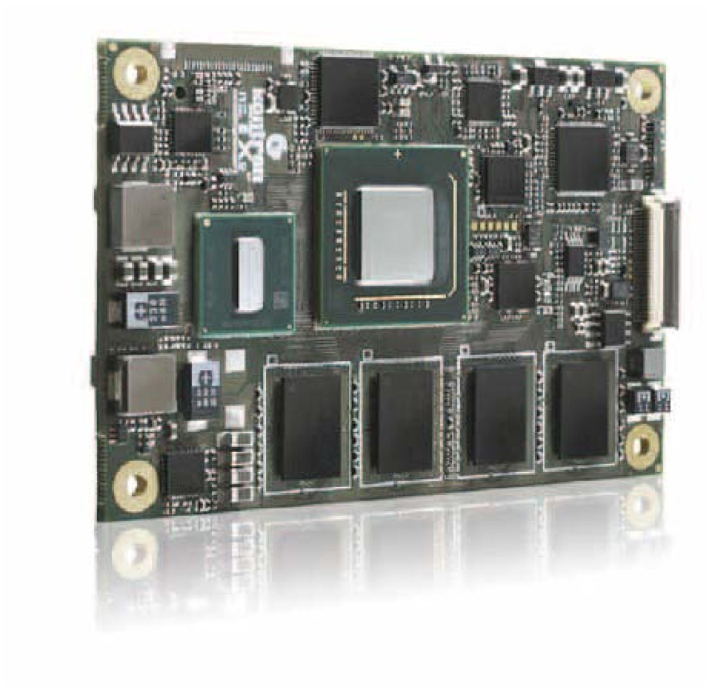

Figure 1: The Kontron NanoETX express Computer-On-Module (COM)

Feature	**Description**
Processor	Intel Atom Z510 (1.1 GHz) or Z530 (1.6 GHz) with 24 kB data and 32 kB instruction L1 cache and 256/512 kB L2 cache
Chipset	Intel US15W (Poulsbo) - 400/533 MHz Front Side Bus (FSB), One DDR2-400 / DDR2-533 unbuffered DDR-SDRAM (SODIMM form factor) up to 2 GB
Graphic controller	Integrated Intel GMA500 graphic controller with dual independent display support, supports Ultra DMA (UDMA5), onchip Video Graphics Array (VGA), hardware acceleration of following video decode standards: H.264, MPEG2, MPEG4, VC1 and WMV9
Audio controller	Integrated Intel® High Definition audio controller (HD audio), digital interface
USB support	Onchip Universal Serial Bus: 8 ports are capable to handle USB 1.1 (UHCI) and USB 2.0 (EHCI), one port alternatively supports USB client functionality as a peripheral mass storage volume or RNDIS device
Ethernet-support	Gigabit LAN (PCI Express): Intel 82574L, full duplex operation at 10/100/1000 Mbps
Mass storage support	1 Serial-ATA (PCI Express) onboard SSD flash drive (up to 8GB)
Other Interfaces	Two PCI Express ports (x1 lanes) Low Voltage Differential Signaling (LVDS) flatpanel interface supports single clock Digital I/O (CPLD): four inputs and four outputs, +3.3V signal level
Temperature monitoring	One onchip thermal sensor and one remote temperature sensor (CPU), SMBus Winbond W83L771W
BIOS	AMI Bios, 1 MB Flash BIOS
Realtime Clock	Supported, requires external battery

Form Factor	55mm x 84mm, Fully COM Express Type 1 compatible
Power consumption	Typ. Idle 2,8W – 3,2W @ 12 V

Table 9: Features of the Kontron nanoETX-SP single board computer

The Kontron nanoETXexpress Computer-On-Module does not provide standard connectors for common interfaces (eg. USB, SATA). It also does not provide analogue audio and DVI. Therefore dedicated AsTeRICS baseboard will be developed. The baseboard will provide standard interfaces (eg. USB, LAN, analog audio interface, DVI), additional Solid State Disc for data storage, miniPCIexpress slot as well as interfaces for AsTeRICS Communication Interface Modules. The idea is shown on **Figure 2** below. The Solid State Disc uses the newest mSATA interface [9] (see **Figure 3**); the mSATA connector is designed especially for applications where a small form factor is desired, such as portable electronics devices.

The use of new Kontron Computer-on-Module (nanoETXexpress) and use of the new miniature mSATA SSD will enable the achievement of small proportions of AsTeRICS Embedded Platform while using standard components.

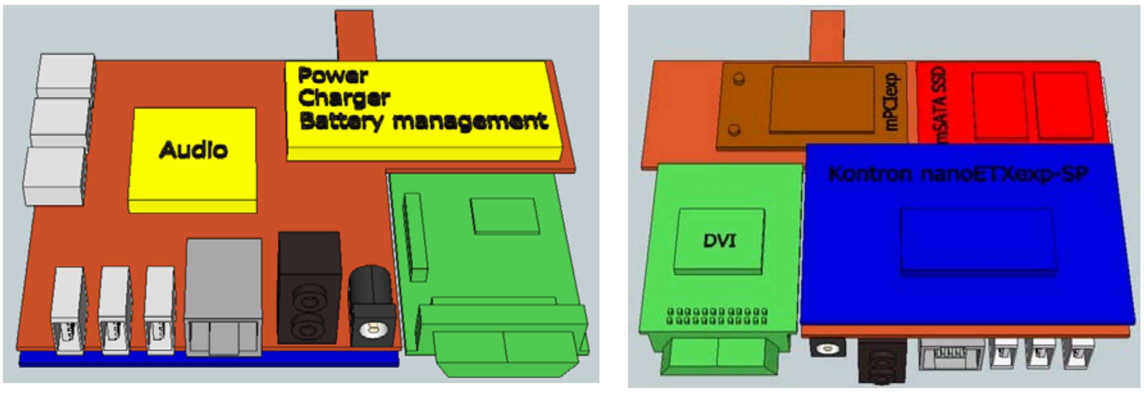

Figure 2: Baseboard of AsTeRICS Core Computer Module; top and bottom view

Figure 3: Solid State Disc with the newest mSATA interface [9]

3.2 Analysis of Software Technologies

3.2.1 Software Technologies of the ACS

As the ACS is built on the Microsoft .NET framework version 3.5, a state of the art analysis regarding the framework and the used libraries has been made. Nearly 18 month after the introduction of .NET 4.0, it is now quite common as standard .NET framework. Following the fact of increased performance[1] and less resource usage, the decision has been made to change form version 3. 5 to version 4.0.

Internal, third party libraries (as the serialisation framework Thrift, used for ASAPI), are also updated to the current release versions. These versions, released during the implementation phase of PT1, includes bug-fixes and performance updates. A detailed list of all used libraries can be found at the documentation of the ACS.

3.2.2 Operating System for the Embedded Platform

In course of the deployment of the first prototype software images to the hardware platforms of various partners, it became clear that Windows-XP (and also the embedded derivate) is not a good strategy for the final system prototype because of the following reasons:

- It was difficult to obtain XP licenses for some partners
- The customer- and technical support for Win XP is limited (e.g. 2015 also for the embedded versions)

A short market analysis revealed that an update to Windows Embedded Standard-7 will have several advantages:

- Deployment of runtime images including drivers and 3rd party software
- No problems with volume licenses
- Improved toolkit and target platform analyser
- Extended technical- and customer support

Because of these reasons, Windows Embedded Standard-7 will be chosen as operating system for the final AsTeRICS platform. For an evaluation of Windows Embedded Standard-7 on the AsTeRICS hardware and the porting of OSGi/Java framework please refer to D4.8 [7].

3.3 Analysis of Sensors and Actuators (off-the-shelf)

This section lists sensors and actuators, which are not listed in the first state of the art analysis. Main target group are wireless components on different wireless technologies, as the many wires are criticised during the user tests [4]

3.3.1 Big Beamer plus Receiver

[1] http://msdn.microsoft.com/en-us/library/ms171868.aspx

The Big Beamer[2] is quite similar to the Jelly Bean switches, but using wireless (radio) technology. Multiple beamers can be used at the same time in the same room without any interferences. The receiver routes the switching signal to a standard 3,5mm Jack plug.

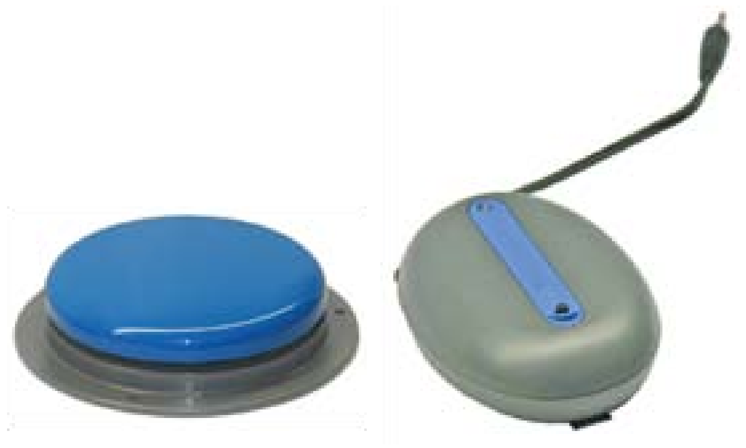

Figure 4: Big Beamer and Receiver

3.3.2 Blue2 Bluetooth Switch

The Blue2 Bluetooth Switch[3] has two built in switches and plugs for the connection of additional switches. The device sends the switching input to an included software, using the Bluetooth wireless communication.

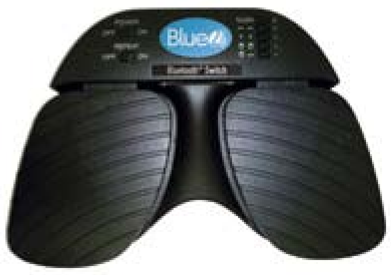

Figure 5: The Blue2 Bluetooth Switch

3.3.3 Swifty and Beam Bundle

In the Swifty and Beam Bundle[4], a switching box is combined with a USB receiver, where the communication will be done wirelessly via infrared. Up to three switched can be connected to the Beam switch box.

[2] http://www.ablenetinc.com/Assistive-Technology/Switches/Jelly-Beamer-and-Big-Beamer-plus-Receivers

[3] http://www.ablenetinc.com/Assistive-Technology/Switches/Blue2%E2%84%A2-Bluetooth%C2%AE-Switch

[4] http://www.orin.com/access/wst/

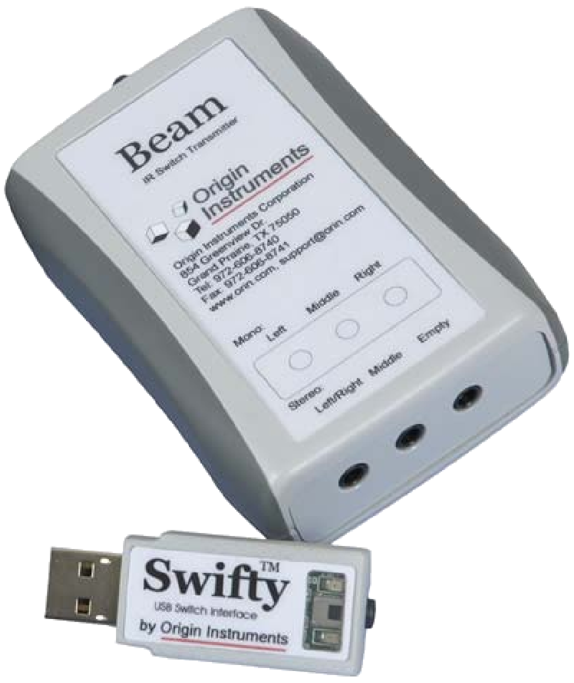

Figure 6: The Swifty and Beam Bundle

3.3.4 SCATIR Switch

The Self-Calibrating Auditory Tone Infrared (SCATIR) Switch[5] is a momentary-contact optical switch with auditory feedback that works by detecting a beam of reflected pulsed infrared light. The SCATIR Switch can be controlled with an eye-blink, eyebrow movement, finger movement, head movement, and facial muscle movement. Optional, two different mounting devices for the sensor are available: one option is an eyeglass mount, the second a gooseneck mount.

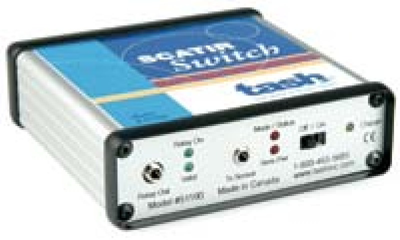

Figure 7: SCATIR Switch

3.3.5 The wireless home automation system FS20[6]

Conrad Electronic offers an affordable (might be called cheap) wireless home automation system called “FS20”. This system uses wireless sensors which collect data like temperature, humidity and send it to central controllers. These controllers process the collected samples and trigger specified actions like turn on the lights or trigger other

5 http://www.ablenetinc.com/Assistive-Technology/Switches/SCATIR-Switch

6 http://www.conrad.at/ce/de/category/SHOP_AREA_17199/Funkschaltsystem-Conrad-FS20

actuators.
The wireless communication between the sensors, controllers and actuators is based on a proprietary protocol which sends data in the SRD (short Range devices) Band on 868.35 MHz, which can be used for free. Conrad also offers controllers, which are able to connect to a wireless area network, to create a connection between the whole "FS20" system and other WIFI enabled devices. WIFI operates on the 2.4 GHz band which is part of the ISM Band (industrial, scientific and medical band) which can also be used freely worldwide.

An example for an actuator is the wireless power outlet FS20ST[7] and the extension lead FS20[8], see Figure 8. These two devices allow the user to turn on and off electrical devices attached to it. It also gives you the possibility to program time delays before the desired action will take place.

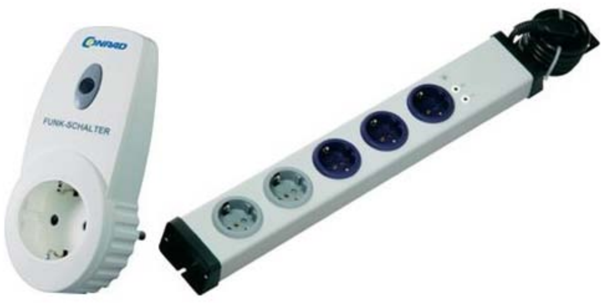

Figure 8: FS20 Components

The interesting part for the AsTeRICS project is to develop a connection to the controllers and actuators of the FS20 system. There are two possibilities to achieve this goal.

The first would be to simulate the 868.35 MHz signals used by the controllers to interact with the actuators. A disadvantage of this approach would be that an additional transmitter which operates in this frequency band and a custom control unit will be needed.

The second method would be to establish a connection to the controllers over WIFI which will be included in the next prototype of the ARE. This will allow direct communication between the AsTeRICS system and the controllers of the FS20 home automation system and furthermore provide the link to control the actuators, without additional hardware needed.

3.3.6 Sweety!

Sweety[9] is a wireless switch interface. It can handle up to five switches. It uses a Bluetooth interface to connect to the PC. Using the Sweety software the user can set actions for each switch, for instance to simulate keyboard key press or to simulate a mouse button click.

[7] http://www.conrad.at/ce/de/product/623004/Funk-Schaltsteckdose-FS20-ST2/2806020&ref=list

[8] http://www.conrad.at/ce/de/product/646446/Funk-Schaltsteckdosenleiste-FS20/2806020&ref=list

[9] http://www.isable.eu/isableshop/shopexd.asp?id=16&bc=no

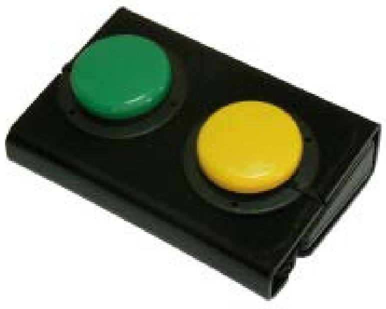

Figure 9: Sweety!

4 Updated Hardware Specification and Architecture

The usual hardware configuration of the main system consists of the Embedded Computing Platform and connected sensors and actuators via standard integrated interfaces or via special Communication Interface Modules (CIMs). Thanks to the high modularity, the system can be adapted to the user's abilities and needs and mounted as a desktop or portable system. The key components of the HW platform will be designed to support both desktop and portable variants as much as possible.

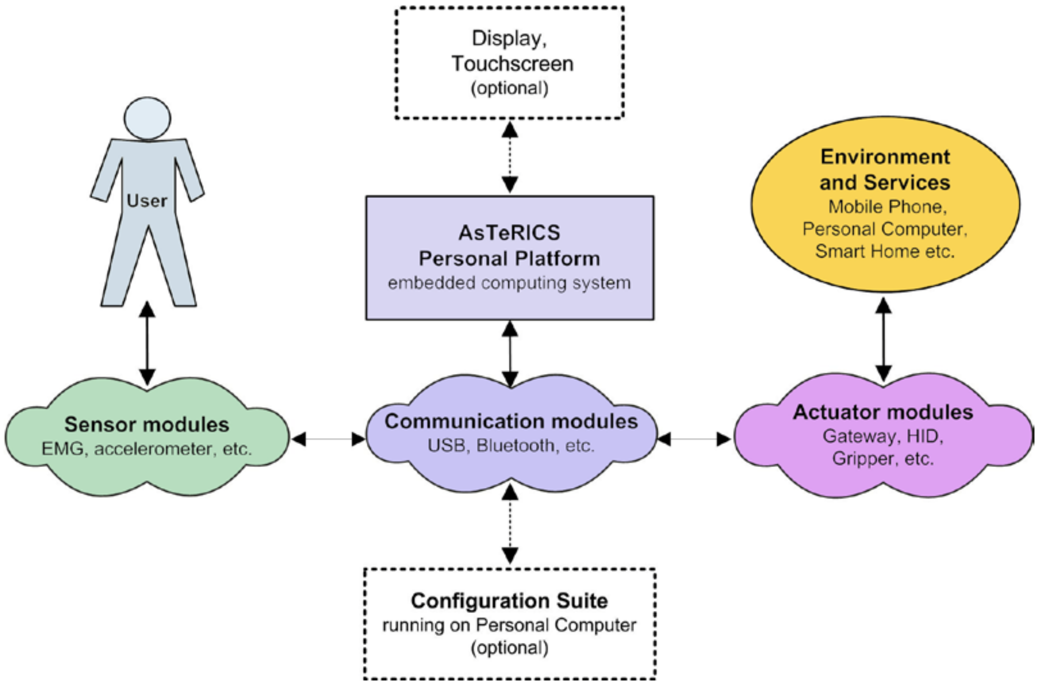
Display, Touchscreen (optional)
User
AsTeRICS Personal Platform embedded computing system
Environment and Services Mobile Phone, Personal Computer, Smart Home etc.
Sensor modules EMG, accelerometer, etc.
Communication modules USB, Bluetooth, etc.
Actuator modules Gateway, HID, Gripper, etc.
Configuration Suite running on Personal Computer (optional)

Figure 10: Concept of the modular Assistive Technology system

Following chapters describes updates in main parts of the AsTeRICS Embedded Platform – the Core Computer Module (CCM), Communication Interface Modules (CIM), Sensors and Actuators, Smart Vision Module and Enobio Biosignal Unit.

4.1 Computer Core Module

A Kontron Single Board Computer with Intel Atom CPU will be used. Design will use experience and results from the PT-1 Computer Core Module. A decision has been taken to use Kontron nanoETXexpress-SP single board computer.

The Slot system was selected as best candidate for the PT-2 design. CCM shall have 3 slots for CIM expansion. However, CCM shall be in two variants:

- Full CCM variant with the expansion slots for future expansion and
- Light CCM variant without the expansion slots as a small compact option.

Electrical and mechanical design shall take this into account. CCM shell have fixture for mounting to eg. chair frame.

CCM shall have the following interfaces:

- DVI port
- LAN port
- 3x USB ports on the backside
- 3x USB ports on the side (for CIM expansion Backplane)
- Analog audio (see 3.1.4.3)
- Integrated AT Peripheral interfaces

A preliminary design vision is shown on Figure 7 and 8. The CCM casing is made from aluminium profiles and plates. Top plate shall be made from plastic.

CIMs are inserted in a "CIM container", which is attached to the main CCM module. This forms the Full CCM variant. The CIM itself shall be in plastic case. Preliminary CIM size is 110 x 35 x 15 mm.

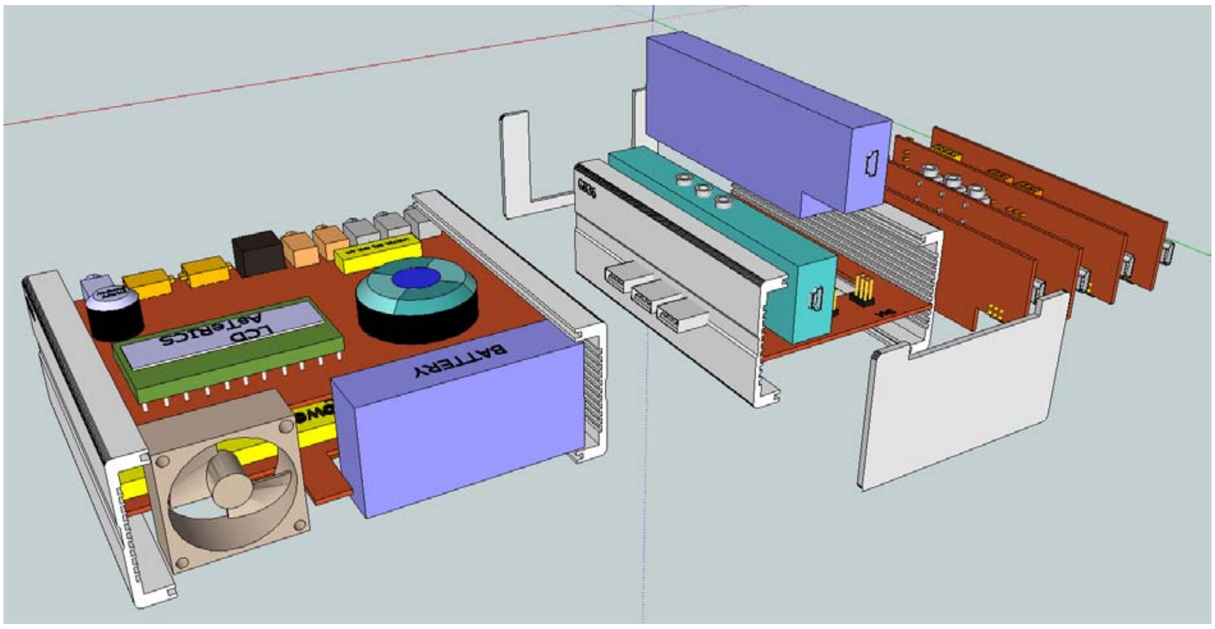

Figure 11: Preliminary design information, front – top CCM view

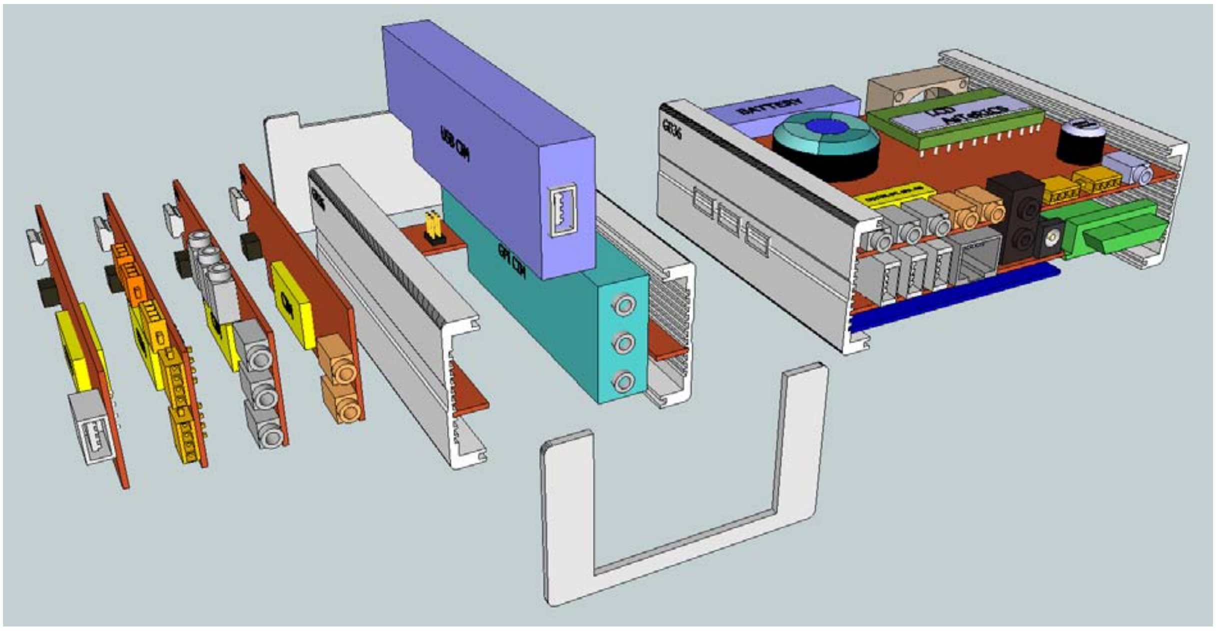

Figure 12: Preliminary design information, rear –top CCM view

4.1.1 Integrated AT Peripheral Interfaces

It is desirable, that certain types of sensors and actuators can interface the PT-2 AsTeRICS Embeded Platform directly, without a CIM module plugged in. Following sections will describe the peripherals integrated on CCM.

4.1.1.1 General Purpose Inputs (CCM-GPI)

The AsTeRICS Core Computer Module shall provide the following inputs for sensors:

- 3 x GPI
- Parameters:
  - 0 to 3V input range (threshold=1.65V)
  - Programmable 3 V pull-up
  - 3.5 mm Jack connector
  - Programmable Wake up function for CoreCIM (via bitmask / CIM feature)

4.1.1.2 General Purpose Outputs (CCM-GPO)

The AsTeRICS Core Computer Module shall thr provide following outputs for actuators:

- 2 x GPO
- Parameters:
  - Open Collector
  - Approx. 500 mA current limit

  - o 3-pin connector; (GND, OC and PWR (+5V) signals, 3V configurable pull-up)

4.1.1.3 Analog inputs (CCM-ADC)

The Input Analog to Digital Core Computer peripheral shall have:

- 2 channel universal (voltage/resistive) input
- Voltage input range 0 – 5 V
- Resistive input range 10 Ohm - 1.5 MOhm
- 2 channel current source for sensor excitation
- 2.5 mm stereo Jacks connectors (GND, input voltage, low current 5V supply voltage for sensors)
- Standalone functionality with USB interface for further external expansion
- Two-color LED indication (Controlled by MCU but override option from ARE)

4.1.2 Power Supply Options

The AsTeRICS Embeded Platform will support the following power supply sources:

- AC/DC (230 VAC to 7,2 – 16 VDC) supply adapter is a nominal power supply source
- CCM contains an internal accumulator for approx. 20 min. operation; this accumulator is recharged when an external power source is connected; status of the battery shall be monitored.
- An external battery or accumulator (7.2 V Li-Pol, 12 V Pb or other with rated performance) can be connected as power supply source; this battery is not charged by CCM and the status may not be monitored.

4.1.3 User interface

4.1.3.1 Power Switching

The AsTeRICS Embeded Platform shall have a Power ON/OFF switch which also acts as Power Stand-by Button. A 3,5mm jack plug connects parallel to the Power ON/OFF switch for external connection of a button.

A short press of the power switch turns the system on or wakes it up from standby mode respectively.

The Power OFF function is triggered by holding the button 6 seconds. It completely disconnects all internal circuitry from internal battery or external power sources.

The Power Stand-by function is triggered by holding the button for 2 seconds and activates power reducing mode. In power reducing mode, the EP functions as following:

- Kontron is in *standby/hibernate* state
- Internal battery is charged when external source is connected
- All USB ports are OFF (switch off by CCM)
- Internal CCM-GPI stays switched on (to enable wakup via connected buttons)
- CCM-GPO are OFF
- *LCD is ON, but backlight is OFF*

4.1.3.2 EP control

The EP shall have an LED display and Navigation keys, similar to PT-1 EP.

4.1.3.3 Audio

The EP shall have an internal microphone and speaker (both mono) and a possibility to connect external stereo headphone and / or external mono microphone similar to PT-1 EP.

4.1.4 CIM Extension Ports

The full variant of CCM shall have 3 slots for extending a function by CIM pluggable module. The slots shall have USB based interface. Power lines of the USB communication ports can be controlled by Kontron Single board computer and remotely switched ON or OFF (all ports at once).

4.2 Communication Interface Modules Updates

4.2.1 GPI CIM

General Purpose Inputs (GPI) shall have:

- channel input (3 at back, 3 at side of the CIM)
- 0 -3 V with programmable 3 V input pull-up resistor (each input), 1.65V threshold voltage
- 3.5 mm mono Jacks
- Standalone functionality with USB interface for further external expansion
- Two-color LED indication (Controlled by MCU but override option from ARE)

- Optional ZigBee wireless variant (if development time is sufficient)

4.2.2 GPO CIM

General Purpose Outputs (GPO) shall have:

- 2 x relay output *(Single Pole, Double Throw connection) 2,5mm 3-pin connector*
- 3x Open Collector (OC) outputs with configurable 3V pull-up, 5V and 12V fixed-voltage; 2,5mm 4-pin connector; GND, OC and PWR (+5V, 12V) signals
- Standalone functionality with USB interface for further external expansion
- Two-color LED indication (Controlled by MCU but override option from ARE)
- Optional ZigBee variant (if development time is sufficient) of the GPO to control room lighting in existing installations; this variant has only one output

4.2.3 ADC CIM

The Input Analog to Digital Conversion CIM shall have:

- 2 channel universal (voltage/resistive) input
- Voltage input range 0 – 5 V
- Resistive input range 10 Ohm - 1.5 MOhm
- 2 channel current source for sensor excitation
- 2.5 mm stereo Jacks connectors (GND, input voltage, low current 5V supply voltage for sensors)
- Standalone functionality with USB interface for further external expansion
- Two-color LED indication (Controlled by MCU but override option from ARE)

4.2.4 D-USB CIM

The Dummy USB CIM enables to connect an external device to USB port in unused slot.

4.2.5 Wireless Accelerometer CIM

Optional, decision taken on available resources and priority defined by user requirements from wired variant tests. Battery supplied.

4.2.6 ZigBee CIM

The ZigBee CIM shall act as a coordinator for ZigBee peripherals developed in AsTeRICS project (Wireless GPI / GPO / Accelerometer). This module does not act as ZigBee receiver for Enobio CIM.

4.2.7 Enobio CIM

The Enobio CIM shall act as transceiver for the Enobio biosignal unit. It shall have the same performance as the Enobio transceiver used in prototype one. Refer to chapter 4.5 for more details.

4.3 Custom Sensor and Actuator Module Updates

There are no updates of the custom sensors and actuators at the moment. The developed and used modules will anymore be used for PT2. New possible modules will be developed, included and tested together with some users (and their special requirements). In particular we plan to develop the following modules:

4.3.1 HID actuator

In course of the second AsTeRICS prototype implementation, the Universal HID actuator module for mouse-, keyboard- and joystick emulation will be improved and equipped with a wireless interface. The problem of the first prototype of this actuator module was that a cable connected from the HID actuator to the AsTeRICS embedded platform was needed, which limited the mobility of the embedded platform with respect to a stationary PC or laptop. The wireless solution will be based on a bluetooth connection. Additionally, an enclosure for the HID actuator dongle will be developed using rapid prototyping technologies.

4.3.2 Sip / Puff Sensor

The Sip / Puff Sensor will be improved, and better mounting options for fixing the sensor at the user’s head will be developed. As reported in the user evaluations, the stationary fixation of the sip / puff sensor is a problem, especially if this sensor is used for clicking in the webcamera-based mouse control applications (where the user moved his/her head). A head-mounted solution for the sip/puff switch will considerably improve this situation.

4.4 Smart Vision Module Updates

This section presents an overview of the plan towards the realization of the Head-Mounted (HM) prototype for gaze driven applications and the plan and current status of the remote camera system plugins, which together represent the computer vision support to the AsTeRICS system in other documents referred as the SVM.

The remote head-tracker for mouse emulation is a joint development effort shared between UPMC and FHTW.

An overview of the head-mounted system is discussed in Section 4.4.1, while Section 4.4.3 contains an excerpt from D4.3 describing the remote system plugins. Further details can be found in the referenced deliverable.

4.4.1 Head-Mounted (HM)

In brief, the roadmap schedules a **HM-v1** prototype to be realized rapidly in order to start working on experimental algorithmic aspects of the calibration routine.

The second and last prototype namely the **HM-v2** will be the finalisation of the previous design. If the mechanical design is too complex for the UPMC Rapid Prototyping tool, the UPMC will rent the access to the professional rapid prototyping tool of a SME specialized in industrial design.

Different camera models will be used for the v1 and v2 version, this is due to the fact that the selected devices are not standard USB cameras but normally used as embedded cameras for mobile phones.

The HM gaze tracking system is not based on the use of near-infrared technology (IR-A, 780-1400nm) that is popular in commercial solutions. The reason being that epidemiological data on IR-A band LED long period exposure do not exist and explicit guidelines have not been yet addressed in any current IR safety standards; potential hazards with are still remaining an open question[10]. Furthermore, head-mounted gaze trackers based on natural light are becoming of high interest in research due to the challenge of being exposed to varying light conditions.

As a reminder the HM system has been introduced in the AsTeRICS to offer an alternative solution for the targeted assistive functionality of mouse emulation, opening new possibilities for on-screen interaction.

The target HM gaze tracker will be a 3 camera device: one pointing at the scene and two pointing at the eyes.

Therefore, its two independent boom arms, left and right, will support two cameras allowing adjusting the camera’s distance and orientation (6 degrees of freedom) with respect to the end user’s eyes.

The forward-looking scene camera will be centrally located on the forehead in a position and orientation that will maximize the overlap between the fields of view of the camera and the subject wearing the HM system.

Eye cameras will be placed in a position which will allow the detection of the eye movements while minimizing the occlusion the visual field of the subject.

4.4.1.1 HM-v1

The HM-v1 is also crucial for assessing the effectiveness of the actual design of the support frame and the placement of the sensors.

The design and the physical realisation of the HM-v1 will be carried out with the help of a 3D modelling software (SolidWorks) and a rapid prototyping machine available at the UPMC (Figure13).

[10]Mulvey, F., Villanueva, A., Sliney, D., Lange, R., Cotmore, S., Donegan, M. (2008), Exploration of safety issues in Eyetracking. Communication by Gaze Interaction (COGAIN), IST-2003-511598: D 5.4.

Figure13 – Stratasys Dimension BST 3D printing machine with Fused Deposit Modeling technology (FDM).

The HM-v1 will be equipped with off-the-shelf small size USB cameras (depicted in Figure14, from Videology Inc.).

The selected cameras(20K15XUSB) have a small form and weight factors and do not need custom software driver for the acquisition thus simplifying the realisation of a simple yet useful prototype which will allow to:

- perform raw-data acquisition of combined views of the scene and the eyes for the first algorithmic assessment of the gaze estimation procedure;
- evaluate the relative placement of the cameras on the supporting frame;
- evaluate the designed shape of the supporting frame.

Figure14 - The smallsize USB colorcamerafromVideology Inc.

Figure15 shows the investigated shape of HM device v1. The mechanical support is adjustable with straps on the back of the head. The evaluation of the final shape with real cameras will guide the finally selected support.

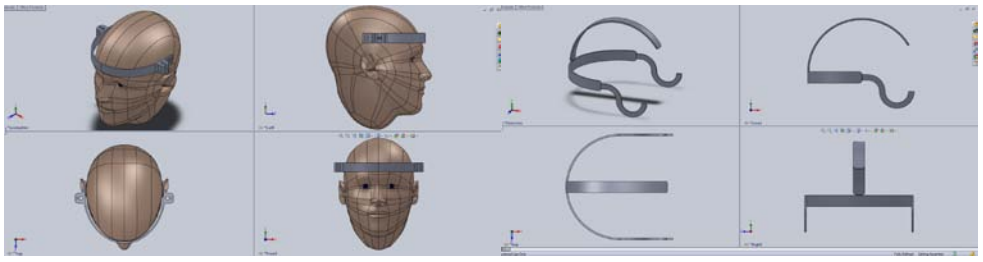

Figure15 - Two different drafts for the head-mounted support.

4.4.1.2 HM-v2

The image acquisition candidates for the final prototype have been initially identified to be the micro cameras produced by OmniVision (OV). Unfortunately the OV cameras should have been placed at least at a distance of 40 centimeters to get a focused image of the eyes thus making them incompatible with a wearable system.

An alternative solution is being investigated: the Ximea company offers excellent micro USB cameras[11] with a complete software support (drivers and standard API) that satisfy the needs for a quality, small-sized and short focus distance camera. A major advantage of Ximea products is that they do not need any additional hardware or software adaption differently from the OV cameras.

4.4.2 Time Schedule

The time schedule is tentative as we are strongly dependent of camera providers located outside of Europe.

4.4.2.1 HM-v1

- September 2011 - January 2012 : build HM-v1, acquire datasets (from the 3 cameras) and implement a gaze estimation procedure.

4.4.2.2 HM-v2

- February - July 2012 : build HM-v2 and adapt the gaze estimation procedure of HM-v1 for HM-v2.

4.4.3 Remote System

The Smart Vision Module (SVM) provides a Computer Vision based sensor for the AsTeRICS Personal Platform. As described in the System Specification and Architecture (D2.1) and in the risk register update for the 4th quarterly project report (12/2010), the SVM will be implemented in two different versions for the AsTeRICS system prototype-1 (where a remote head tracking system based on webcam image acquisition is desired) and the system prototype-2 (where a head-mounted iris tracking application will be developed).

The main advantage of the remote system is that no devices or sensors have to be mounted on the subject’s body, which may increase the comfort for the users, especially in long-term use. The main advantage of the head-mounted system is that eye movements can be measured with high accuracy and that eye-tracking and gaze-estimation applications become possible.

This two-stage approach offers optimal flexibility in the sense of the Assistive Technology Construction Set, as both strategies for camera-based computer interaction will be available at the end of the AsTeRICS project. Furthermore, this approach addresses the problem of the missing hardware budget after the dropout of the initial hardware partner CEDO, causing less possible effort on the hardware side.

[11] http://www.ximea.com/en/products-news/micro-usb-camera-genicam-gentl

In the following, the accomplished SVM implementation for the remote head tracking system will be described in more detail. For the remote SVM, two different approaches have been evaluated and implemented and are now available as sensor plugins for the ARE:

- A head-tracking approach optimized for speed in a low-performance computing environment, based upon a Haar-Classifier Cascade for face detection and an Optical Flow algorithm for feature tracking.
- A head-tracking approach, which offers high face tracking stability, based on Active Shape Models.
- A head-tracking approach, which offers high face tracking stability, based on Active Shape Models.

Both approaches have pros and cons, depending on the current situation and the available computing power, the optimal vision sensor plugin can be chosen.

4.4.3.1 Head Tracking via Haar-like Feature Detection and Optical Flow

The “webcamera” sensor plugin of the ARE offers a head-tracking algorithm which calculates the actual movement of the user’s nose and chin from a remote camera image, captured by a high-quality web camera. The nose- and chin movement are provided at the plugin’s output ports. This data can be used by other plugins for mouse replacement or selection in an on-screen keyboard grid. By computing relative movements of nose and chin via dedicated ARE-plugins, voluntary movements of the chin can be detected and utilized e.g. for selection in an on-screen keyboard scenario or for mouse clicking.

The computer vision approach is based upon a combination of a face-detection algorithm (Haar-Classifier Cascade) and a feature tracking algorithm (Lukas Kanade Optical Flow Tracking), as described in [10]. The advantage of this combination is a low average computational load, because the face detection - which involves complex calculations and matching algorithms - is only performed at initialization and re-initialization phases (i.e. when the face location has been lost), and the lightweight LK-tracking can be performed in about 99,5% of the image frames after a face has been detected. This allows efficient head tracking even on a low performance / low power computing system as represented by the AsTeRICS Personal Platform with Atom CPU.

4.4.3.2 Head Tracking using Active Shape Models

Active Shape Models (ASM) have been introduced by T. Cootes and colleagues in [11] to devise a method for locating an object inside an image based on its projected 2D shape. The ASM were presented as an improvement of the Active Contour Models [12] (also known as snakes), which are formulated as an energy minimisation problem. Snakes fails at capturing the peculiar modes of variation in a category of shapes. The key idea behind the ASM is that an instant of the searched shape cannot deform much differently from the model learned from a training set of properly annotated examples. The training dataset is a collection of face pictures and related landmarks coordinates depicting local salient features (in Figure 16: Labelled points on the training set.Figure 16 two examples of labelled faces).

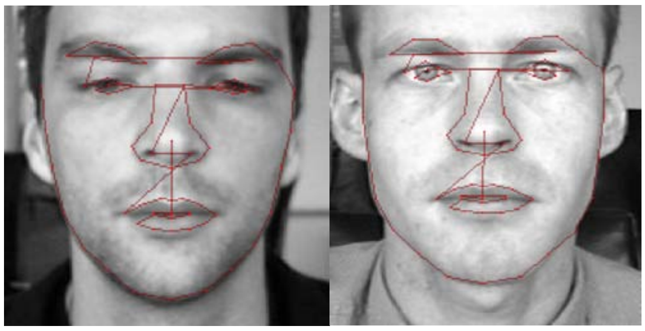

Figure 16: Labelled points on the training set.

All the shapes in the training set are first aligned to each other by minimising the average distance between each other hence giving shape the so-called point distribution cloud at each landmark position. The covariance matrix S computed from all aligned points catches such behaviour and all the pairwise correlations between landmarks. The Principal Component Analysis (PCA) decomposition of S allows finding the principal modes that govern the landmarks displacements. In brief the PCA applied to a covariance matrix computed on a set of observations in R^N finds the subspace of R^N explaining most of the variability in the data. The PCA implicitly assumes a Gaussian as the source of observations.

Two sub-models describe an ASM: the **shape** and the **profile** models.

The **shape** model describes the set of allowable/plausible spatial deformation applicable to the mean/neutral shape in order to match the current face instance.

When constraining the variation of the parameters that govern the shape model to an interval that depends on the expected statistical variability it is possible to generate shapes that are plausible, in the sense that they are likely to belong to the modelled class of shapes.

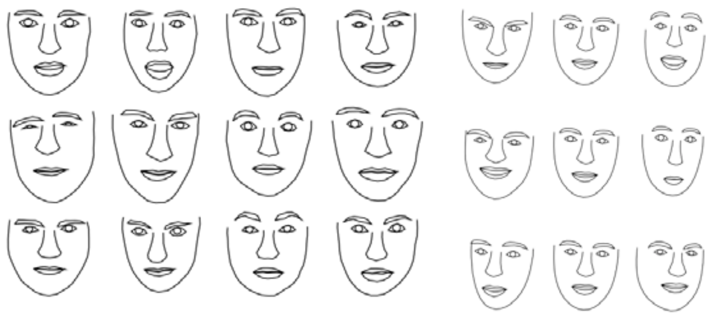

Figure 17 - Left: shapes instances from the training set. Right: shapes resulting when varying the first 3 modes in the model (from Cootes et al 1995)

The **profile** model is a statistical model of the grey-level appearance at each landmark neighbourhood. In the current implementation the values of the normalised gradient along a 1D profile (whisker) around the landmarks are modelled as sampled from a normal distribution. The fitted mean μ and variance Σ characterising the distribution are then used to evaluate the likelihood of a new sample g via the Mahlanobis distance.

The search process is equivalent to find that set of parameters that deform the initial starting shape to match the current instance. The goodness of a set of parameters is evaluated by computing the distance between the local grey-level appearance at each landmark's position and the profile model, which is also learned during the training.

During search such measure is used to update each landmark position to the points that score the highest likelihood (or minimal distance).

Figure 18 – Two examples of the ASM search results.

The current implementation is based on the Stasm[12] library. The library is a straightforward implementation of the original formulation by Cootes et al except for a couple of appealing features that improve robustness in the search results as presented in [13]. The first notable feature is the addition of a 2D shape profile modelling during training. A two dimensional patch captures more information than a cross-border line thus improving the representational power of the ASM.

[12] http://www.milbo.users.sonic.net/stasm/

4.4.3.3 Facial Landmark tracking using FaceTracker

A recent published work [14] presents an alternative approach to the fitting of a deformable model. It borrows the same idea behind active shape models but it is based on a sophisticated version of the constrained local model search (CLM). The response likelihood map that encodes the landmark location likelihood in its neighbourhood is represented by an optimized non-parametric distribution and regularized by the landmark joint motion.

This approach enables the detection of a wider range of head poses and facial expressions compared to Stasm (4.4.3.2).

FaceTraker[13] is a C++ library was made available by the author J. Saragih for inclusion into the AstrRICS

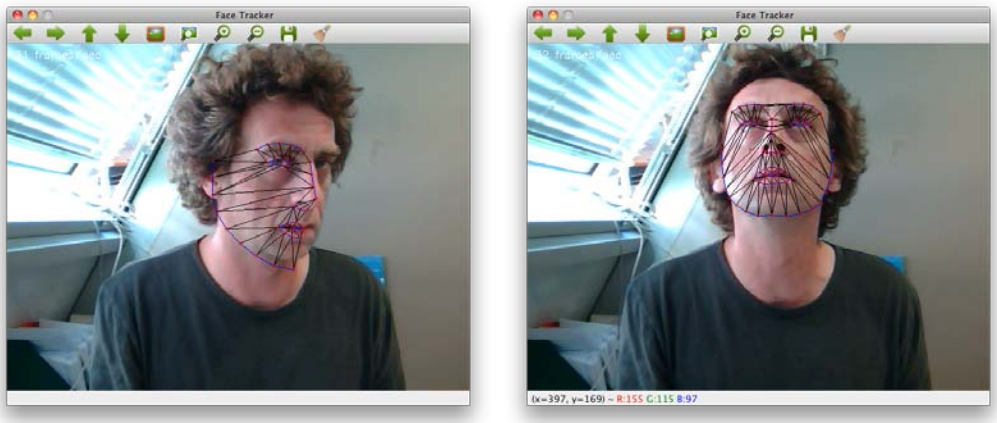

Figure 19 - Two extreme head poses perfectly matched by FaceTracker.

4.4.4 Discussion

A remarkable advantage in using deformable shape approaches in implementing an hands-free interaction resides on the possibility to encode user facial gestures relying on meaningful labelled data (all points in the deformable model are tagged with a label which makes reference to a specific face location) to capture facial gestures while other approaches based on Viola-Jones can deliver only a bounding-box depicting the region where the searched template is likely to be found.

Although the current computational power of the AsTeRICS platform does not provide sufficient performance to take full advantage of such rich information, these approaches will be implemented as ARE plugin by borrowing from the basic head tracking plugin the lightweight approach as explained in Sec. 4.4.3.1 thus performing a detection phase only when a reset of the tracked landmarks is required after a tracking failure.

An immediate advantage of the deformable approach is represented by the chance to track a sparse set of labelled facial locations.

[13] http://web.mac.com/jsaragih/FaceTracker/FaceTracker.html

The output interface will be equivalent for the three of them in order to use them transparently depending on the actual lighting conditions.

4.5 Enobio Biosignal Unit - Updated Specification

This section presents the update of the requirements that are related to the integration of the Enobio sensor into the prototype 2 of the AsTeRICS platform. The integration of the Enobio sensor in the prototype 1 was mainly focused in the software part. A plugin that interfaced the Enobio receiver was developed so the electrophysiological signals recorded by Enobio were available to be processed by other plugins running in the ARE. The hardware integration of the sensor (mainly the receiver) was not done at all in the prototype 1 since the receiver was just connected to the Core Computer Module through an USB cable and it was left out of the box.

In prototype 2 this wired connectivity is avoided so the AsTeRICS system will support wireless connectivity with the Enobio system. This is stated in the requirement labeled HSEN8 that is listed in the hardware requirements for the AsTeRICS pluggable components (see Table 2). Thus, the main goal for the integration of the Enobio biosignal unit in the AsTeRICS platform is to fully physically integrate the Enobio receiver in the Core Computer Module as if it was a generic CIM. This requirement is identified with the label HSEN9 in the same table as the previous one.

For making such an integration some considerations have to be taken into account; since the custom receiver will be inside the box and the antenna of the receiver is printed in its PCB board, the box shall allow RF transmissions from inside. This requirement is identified with the label HW13 and is listed in the requirements for the AsTeRICS embedded platform (see Table 1).

In addition, the PCB board of the Enobio receiver shall to be adapted in order to match with the size and the connection that the generic CIM slot will provide. The connector that is currently placed in the Enobio receiver is an USB type-B one which is thought for a wired connection with a host computer so some adaptations might need to be done so it can be plugged to the generic CIM slot connector which is not known yet in this phase of the development.

5 Updated Software Specification and Architecture

The AsTeRICS software framework provides an open and flexible toolkit for enabling the formation of various AT systems. Its architecture is defined in deliverable D2.1. This section provides an update on the current state of the software specification and architecture as well as new requirements emerged since the delivery of D2.1.

There were no significant deviations from the initial software architectural decisions. The originally selected modular architecture of the AsTeRICS system (shown below) has been proven adequate to support the software needs of the system.

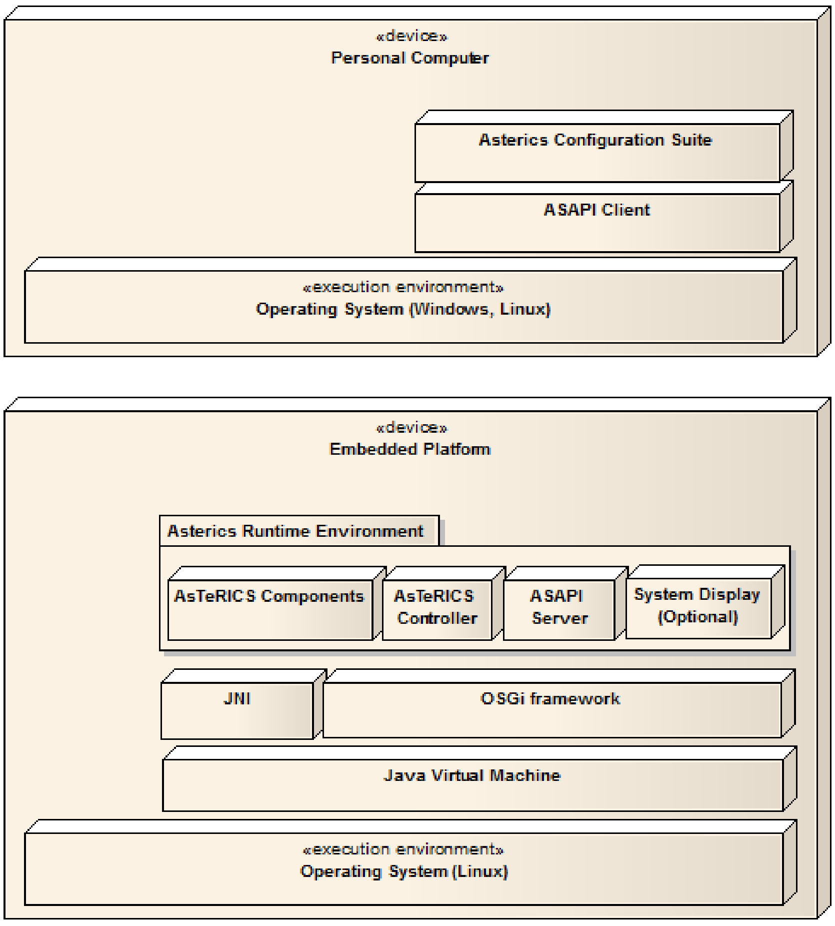

Figure 20: High-level view of the system architecture (deployment model)

The main components of our architecture remain as they were defined in D2.1:

- The AsTeRICS Runtime Environment (ARE)
- The AsTeRICS Configuration Suite (ACS)
- The AsTeRICS Application Programming Interface (ASAPI)

As it was defined in D2.1.: "*The ARE is used to execute a predefined system design in the target environment, which usually is the Embedded Computing Platform. As such, it provides a middleware architecture that controls and manages the components used to form the* *deployed applications. The ACS is mainly used to graphically design the layout of the system as a network of interconnected components. Finally, the ASAPI is used to connect the ACS - or other client applications running on the PC (like AT-software by consortium member SENSORY) - to the AsTeRICS [...] main output of the ACS and the main input of the ARE is the system model. This model describes the components that are used for realizing the intended behavior, along with their input and output ports and the channels connecting them. Since the system model represents the main communication means between the ACS and the ARE it is expected to be a serialisable object, easy to transfer and translate*."

Of course each one of the main AsTeRICS components has been enriched with new features and functionality to support newly emerged requirements. The following sections outline those new features and discuss what is left for the next prototype.

5.1 AsTeRICS Runtime Environment (ARE)

The AsTeRICS runtime environment consists the most complex software part of the system. It provides a runtime platform for executing system models designed in ACS and communicated via ASAPI, based on the currently installed plugins. Below we outline the main functionalities supported by ARE:

5.1.1 ARE Services

One of the most significant additions in the ARE is the ARE Services. This newly added package contains classes that enable the direct interaction between AsTeRICS plugins and other software to directly interact with the runtime environment. The most signrificant ARE Services are:

- **Error Handling:** we needed a uniform way of error reporting in the runtime environment so we have utilized the Java logging libraries and the various severity levels supported. The AsTeRICS error handling mechanism is used extensively from the runtime core classes but also utilized by the AsTeRICS components via the AstericsErrorHandling interface, shown in Apendix A. Each component is allowed to report an error message, a debug information or a simple information to be displayed on the screen. The ARE maintains four separate log files and updates them whenever a new error occurs. In particular there are different loggers for reporting *severe errors*, *warnings*, *fine errors* and one logger that contains them all. ARE also maintains a *status object* for the current status of the runtime environment. Whenever a fatal error occurs (either internally or caused by one of the deployed components) the status changes to *fatal error.* Other possible statuses are *unknown, OK, deployed, running and paused.* The ACS can request the current status of the runtime environment and update its own state accordingly. For example the ACS user can be informed about the current ARE status while the ACS will terminate a connection (or refuse to establish a new one) with a non-working ARE.
- **Remote Connection Service:** the remote connection services allows external software that cannot be integrated into the standard plugin inter communication system used by the ARE, for example because of programming language incompatibilities, to work with the AsTeRICS system. For example, the interconnection of OSKA (the On-Screen Keyboard Application developed by AsTeRICS partner SENSORY) and the ARE uses the Remote Connection Service to send key selection information to the ARE. On the other hand, the ARE can reply with

cell selection commands or other information. The actual communication is done via a protocol that can be understood by the Java ServerSocket implementation. The port number that the external software component connects to identifies the connecting component. Further implementation details on the remote communication service are given in deliverable D4.2.

- **CIM Communication Service:** the ARE CIM Communication service layer is a unified approach to allow software components (plugins) of the AsTeRICS Runtime Environment (ARE) to communicate with their associated Communication Interface Modules (CIMs) attached to the AsTeRICS platform. Further details on the communication protocol can be found in D2.3 [5] and implementation details for the ARE CIM Communication Service in D4.2 [6].
- **Local Storage Service:** the Local Storage Service will allow plugins to store individual working data "per model" and "per plugin-instance". This is necessary when plugins need to store own calibration data, pattern recognition samples or similar data. In course of the architectural refinements for the final prototype, a service class will be provided which generates an according folder and respective file read- and write methods.

5.1.2 New Graphical User Interface for the ARE

The ARE GUI was initially used as a test console for testing the behaviour of the runtime environment without the need of an interconnected ACS. Lately there was a need for extending the ARE graphical user interface to serve as graphical console from where end-users will be able to directly interact with the runtime environment. The new ARE GUI supports a user-friendly quick launch menu for the most needed actions, such as deploy a model from file, start, stop or pause a model, restart ARE, etc. It also provides a "desktop" area where plugin developers can display their graphical elements using a new ARE service.

The ACS will provide a canvas editor that will allow end users to position and resize graphical elements of the plugins participating in a model. Based on this information the ARE displays them on the local device. An example is shown below.

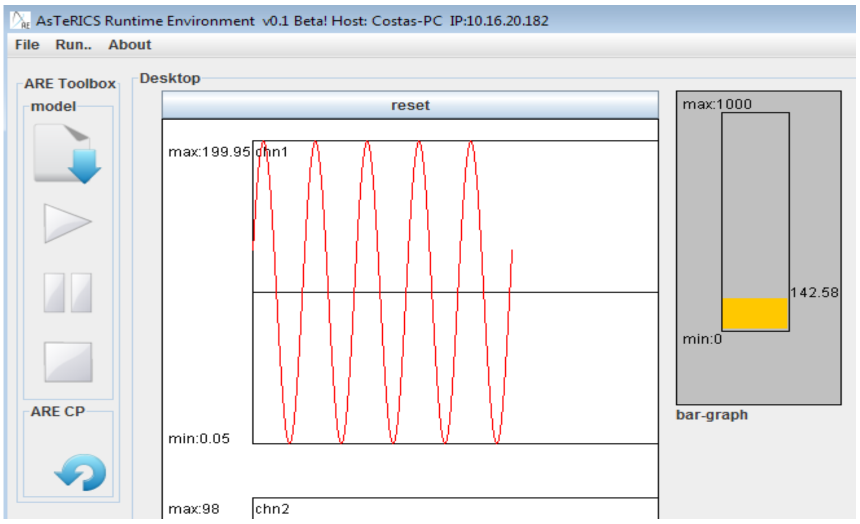

Figure 21: the new ARE GUI

5.2 AsTeRICS Configuration Suite (ACS)

Section 4.3.4 in [2] lists the planned functions of the ACS. During the implementation of PT1, all planned tasks of PT1 have been fulfilled, with the only exception of “Visualisation and graphical elements”. This task was not implemented, because the decision has been made to display all visualisation elements on the ARE. Additionally, several tasks had been fulfilled during PT1 implementation:

- Add and remove event channels between components
- Set and update events within the event channels
- Include the ARE storage concept into the ACS
- Display status changes and logging files of the ARE
- Error messages on incompatible versions of plugins

The full functional description of the PT1 ACS can be found at [5].

The focus of the future work on the ACS will be, fulfilling the requirements for PT2 (compare to Table 6) and the requirements, defined by secondary user’s feedback (see section 2.1.3).

5.3 AsTeRICS Application Programming Interface (ASAPI)

The ASAPI protocol has been updated with new commands in order to satisfy new communication needs between the ACS and ARE. In addition some comments were deemed not necessary and have been removed. The tables below show the current state of the ASAPI protocol.

Method	**Description**
Methods to setup and deploy a model	
`String [] getAvailableComponentTypes();`	Returns an array containing all the available (i.e., installed) component types. These are encoded as strings, representing the absolute class name (in Java) of the corresponding implementation.
`String getModel();`	Returns a string encoding the currently deployed model in XML. If there is no model deployed, then an empty one is returned.
`String getModelFromFile(String filename)`	Returns a string encoding of the model defined in the file name given as argument. If there is no model an empty one is returned.
`void deployModel(String modelInXML) throws AsapiException;`	Deploys the model encoded in the specified string into the ARE. An exception is thrown if the specified string is either not well-defined XML, or not well defined ASAPI model encoding, or if a validation error occurred after reading the model.
`void deployFile(String filename) throws AsapiException;`	Deploys the model associated to the specified filename. An exception is thrown if the specified filename is not found.
`void deployModel(String filename, String modelInXML) throws AsapiException;`	Deploys the model encoded in the specified string into the ARE, which assigns the specified filename to it. An exception is thrown if the specified string is either not well-defined XML, or not well defined ASAPI model encoding, or if a validation error occurred after reading the model. If a model already exists with the specified filename, then it is replaced.
`void newModel();`	Deploys a new empty model into the ARE. In essence, this is

	equivalent to creating an empty model and deploying it using deployModel(String) above. This results in freeing all resources in the ARE (i.e., if a previous model reserved any).
`void runModel() throws AsapiException;`	It starts or resumes the execution of the model. It throws AsapiException if an error occurs while validating and starting the deployed model.
`void pauseModel() throws AsapiException;`	Briefly stops the execution of the model. Its main difference from the stopModel() method is that it does not reset the components (e.g., the buffers are not cleared). It throws an AsapiException if the deployed model is not started already, or if the execution cannot be paused.
`void stopModel() throws AsapiException;`	Stops the execution of the model. Unlike the pauseModel method, this one resets the components, which means that when the model is started again it starts from scratch (i.e., with a new state). It throws AsapiException if the deployed model is not started already, or if the execution cannot be stopped.
`void storeModel(String modelInXML, String filename)`	Used for storing a model deployed from ACS to the local ARE folder.
`boolean deleteModelFile (String filename)`	It is used for deleting a file from the ARE local folder. If the given filename was succesfully deleted true is returned, otherwise false is returned.
`String[] listAllStoredModels()`	Returns a string array with all models' names stored in the ARE local folder.
Methods to read and edit the model	
`String [] getComponents();`	Returns an array that includes all existing component instances in the model (even multiple instances of the same component type).
`String [] getChannels(String componentID);`	Returns an array containing the IDs of all the channels that include the specified component instance either as a source or target.
`void insertComponent(String componentID, String componentType) throws AsapiException;`	Used to create a new instance of the specified component type, with the assigned ID. Throws an exception if the specified component type is not available, or if the specified ID is already defined.
`void removeComponent(String componentID) throws AsapiException;`	Used to delete the instance of the component that is specified by the given ID. Throws an exception if the specified component ID is not defined.
`String [] getAllPorts(String componentID) throws AsapiException;`	Returns an array containing the IDs of all the ports (i.e., includes both input and output ones) of the specified component instance. An exception is thrown if the specified component instance is not defined.
`String [] getInputPorts(String componentID) throws AsapiException;`	Returns an array containing the IDs of all the input ports of the specified component instance. An exception is thrown if the specified component instance is not defined.
`String [] getOutputPorts(String componentID) throws AsapiException;`	Returns an array containing the IDs of all the output ports of the specified component instance. An exception is thrown if the specified component instance is not defined.
`void insertChannel(String channelID, String sourceComponentID,String sourcePortID, String targetComponentID, String targetPortID)throws AsapiException;`	Creates a channel between the specified source and target components and ports. Throws an exception if the specified ID is already defined, or the specified component or port IDs is not found, or if the data types of the ports do not match. Also, an exception is thrown if there is already a channel connected to the specified input port (only one channel is allowed per input port except for event ports that can have multiple event sources).
`void removeChannel (String channelID) throws AsapiException;`	Removes an existing channel between the specified source and target components and ports. Throws an exception if the specified channel is not found.
`Methods to read and edit properties (even while running)`	
`String []`	Reads the IDs of all properties set for the specified

`getComponentPropertyKeys(String componentID);`	component.
`String getComponentProperty (String componentID, String key);`	Returns the value of the property with the specified key in the component with the specified ID as a string.
`String setComponentProperty (String componentID, String key, String value);`	Sets the property with the specified key in the component with the specified ID with the given string representation of the value.
`String [] getPortPropertyKeys(String portID);`	Reads the IDs of all properties set for the specified port.
`String getPortProperty(String componentID, String portID, String key);`	Returns the value of the property with the specified key in the component with the specified ID as a string.
`String setPortProperty(String componentID, String portID, String key, String value);`	Sets the property with the specified key in the port with the specified ID with the given string representation of the value.
`String [] getChannelPropertyKeys(String channelID);`	Reads the IDs of all properties set for the specified component. Reads the IDs of all properties set for the specified channel.
`String getChannelProperty(String channelID, String key);`	Returns the value of the property with the specified key in the channel with the specified ID as a string.
`String setChannelProperty( String channelID, String key, String value);`	Sets the property with the specified key in the channel with the specified ID with the given string representation of the value.
Methods for status checking	
`StatusObject[] queryStatus(boolean fullList);`	Queries the status of the ARE system. If the single argument is true then the full list of status changes will be sent to ACS, if false the status changes occured from the last time the queryStatus was called will be sent.
`String getLogFile()`	ARE maintains a log file with all logged messages. It can be sent to ACS via this method which serializes the log file and sends it to ACS as a string.

Table 10: The ASAPI Server Interface

The “ASAPI Client”, extends the functionality provided by the “ASAPI Server” by adding commands for discovering and connecting to ARE instances:

Method	**Description**
Methods to discover and connect/disconnect to AREs	
`ASAPI_Server connect(InetAddress ipAddress);`	Connects to the ARE at the specified IP address. The method returns an instance of the ASAPI Server interface (described above), masking the functionality provided by the target ARE through ASAPI.
`void disconnect(ASAPI_Server asapi_server);`	Disconnects from the specified instance of the ASAPI Server, invalidating the reference.

Table 11: The ASAPI Client Interface

5.3.1 Native ASAPI

Native ASAPI is a software development kit for 3rd party developers to help them adapt their application for people with motor disabilities. Native ASAPI will be delivered as a set of DLL libraries and COM objects for the Microsoft Windows Operating system. Native ASAPI works independently of ARE.

The table below lists the devices interfaced by Native ASAPI:

Nr	Device	Priority
1.	3D Mouse	M
2.	Windows Mobile phone	H
3.	Keyboard	L
4.	Android phone	H
5.	GSM modem	M

Table 12: Devices interfaced by Native ASAPI

3D Mouse Interface

The 3D Mouse Library is designed to help in adapting 3Dconnexion 3D Mouse devices for people with motor disabilities. It works with the 3D Mice connected to PC via USB such as: SpacePilot Pro, SpaceExplorer and SpaceNavigator.

The library interface is declared in the 3DMouseLibrary.h file.

Function	Description
int init ()	Initializes the 3D Mouse Library
int close ()	Closes the Library
int get3DMouseState(long *x, long *y, long *z, long *Rx, long *Ry, long *Rz, long* buttons)	Gets the actual state of the 3D mouse. Parameters are axis, axis rotation and button state.

Table 13: 3D mouse library interface

3D Mouse library interface functions are declared with “C” linkage. The 3D Mouse Library function returns a positive value if it succeeds. If the function fails, it returns a value lower than 0 and the returned value is the code of the error.

The error codes are declared in the 3DMouseLibraryErrors.h file:

Number	Description
-1	Default error.
-2	Library is not initialized.
-3	Library is initialized.
-4	Library initialization error
-5	The 3D Mouse device not found.
-6	Data acquire error.

Table 14: 3D mouse library error codes

Windows Mobile phone library

The Phone Library is designed to control mobile phones with Windows Mobile operation system. The library uses Bluetooth connection to connect to the Phone Library Server Application run on the mobile phone

The library interface is declared in the PhoneLibrary.h file.

Function	Description
int init(DeviceFound deviceFound, NewSMS newSMS, PhoneStateChanged phoneStateChanged, LPVOID param)	Initializes the Phome Library. The deviceFound, newSMS and phoneStateChanged parameters are pointers to the call-back functions implemented in the Phone Library user application. The param parameter is a parameter defined by the user and passed to the call-back functions.
int close()	Closes the library.
int searchDevices()	Starts searching for devices. For each discovered device the DeviceFound call-back function is called
int connectToDevice(unsigned _int64 deviceAddress, int port)	Connects to the device defined by deviceAddress parameter.
int disconnect()	Disconnects device.
int makePhoneCall (LPWSTR recipientID)	Makes a phone call. The recipientID parameter is the recipient phone ID.
int acceptCall()	Accepts incoming phone calls.
int dropCall()	This function drops incoming phone call or disconnects phone calls.
int getPhoneState(PhoneState &phoneState)	Gets actual phone state of the mobile phone.
int sendSMS(LPWSTR recipientID, LPWSTR subject)	Sends SMSs. The recipientID parameter is the recipient phone ID, the subject parameter is the message content.

Table 15: Windows Mobile phone library interface

Phone library interface functions are declared with “C” linkage. The Phone Library function returns a positive value if it succeeds. If the function fails, it returns a value lower than 0 and the returned value is the code of the error.

Windows Mobile library phone call-back functions definitions:

Function	Description
typedef void (__stdcall *DeviceFound) (unsigned _int64 deviceAddress, LPWSTR deviceName, LPVOID param)	This function is called when a new device is found. The deviceAddress parameter is the address of the discovered device. The deviceName parameter is the name of the device. If the returned deviceAddress parameter is equal to 0, the device search process finishes.
typedef void (__stdcall *NewSMS) (LPWSTR PhoneID, LPWSTR subject, LPVOID param)	This function is called when there is a new SMS available. The PhoneID parameter is the sender phone ID. The subject parameter is the SMS content.
typedef void (__stdcall *PhoneStateChanged) (PhoneState phoneState, LPWSTR phoneID , LPVOID param)	This function is called when the state of the phone is changed. The PhoneStates parameters indicates the actual phone state. The PhoneID parameter is the caller ID.

Table 16: Windows Mobile phone library call-back functions

The error codes are declared in the PhoneLibraryErrors.h file:

Number	Description
-1	Default error.
-2	Library is not initialized.
-3	Library is initialized.
-4	Library initialization error.
-5	No respond from remote device.
-20	Library is searching for the devices now.
-21	Device is not found.
-31	Device is connected.
-32	Error during connecting to the device.
-33	Device is not connected.
-34	Default port error.
-50	Phone ID or SMS content is empty

-1001	Remote device default error.
-1011	Bluetooth initialization error on the remote device.
-1015	Packet error.
-1031	Messanger module initialization error on the remote device.
-1032	Messanger module is not initialized on the remote device.
-1033	Message send error on the remote device.
-1051	Phone module initialization error on the remote device.
-1052	Phone module is not initialized on the remote device.
-1053	Phone accept the call error on the remote device.
-1054	Phone drop the call error on the remote device.
-1055	Phone make the call error on the remote device.
-1072	Messanger module and Phone module is not initialized on the remote device.

Table 17: Windows Mobile phone library error codes

Other interface information:

Interface	Description

enum PhoneState
{
PS_IDLE=1,
PS_RING,
PS_CONNECTED
};
	Indicates current phone state.

#define Default_port 1	Indicates the default port number, which can be use in the connectToDevice method.

Table 18: Windows Mobile phone library other interface information

Keyboard library

The Keyboard Library is designed for developers who need to adapt the computer keyboard for the specialized needs of motor disabled people. For example if the application has to use standard keyboard input for the scanning and send the keys in different way. Developers using this library will be able to get information about all system key events and send key events to other applications.

The library interface is declared in the KeyboardLibrary.h file:

Function	Description
KEYBOARDLIBRARY_API int __stdcall init(HookCallBack hookCallBack, LPVOID param)	Initializes the library. The hookCallBack parameter is a pointer to the call-back function. The param parameter is a parameter defined by user.
KEYBOARDLIBRARY_API int __stdcall close()	Closes the library.
KEYBOARDLIBRARY_API int __stdcall startHook()	Starts key events hooking
KEYBOARDLIBRARY_API int __stdcall stopHook()	Stops key events hooking.
KEYBOARDLIBRARY_API int __stdcall sendKeyByScanCode(int scanCode, SendKeyFlags flags)	Simulates a key event using a key scan code.
KEYBOARDLIBRARY_API int __stdcall sendKeyByVirtualCode(int virtualCode, SendKeyFlags flags)	Simulates a key event using a virtual key code.
KEYBOARDLIBRARY_API int __stdcall sendText(LPWSTR text)	Simulates text being typed in, defined by the text parameter.
KEYBOARDLIBRARY_API int __stdcall blockKeys(BlockOptions blockOptions)	Blocks or Passes key events. The blockOptions parameter defines the function's behaviour.

Table 19: Keyboard library interface

Keyboard library interface functions are declared with “C” linkage. The Keyboard Library function returns a positive value if it succeeds. If the function fails, it returns a value lower than 0 and the returned value is the code of the error.

Keyboard library phone call-back function definition:

Function	Description
typedef int (__stdcall *HookCallBack)(int scanCode, int virtualCode,HookMessage message, HookFlags flags, LPVOID param);	This function is called if there is a new key event. The scanCode parameter defines the scan code of the key, the virtualCode parameter defines the virtual key code, the message defines message type, the flags parameter defines additional information about the key event, the param parameter is a parameter passed by the user. If the returned vaule is lest than 0, the library will block the event, if the returned value is greather than 0 the library will pass the event. If the returned value is 0 the library will block or pass the event according to the BlockKeys function.

Table 20: Keyboard library call-back function

The error codes are declared in the KeyboardLibraryErrors.h file:

Number	Description
-1	Default error.
-2	Library is not initialized.
-3	Library is initialized.
-4	Library initialization error.
-5	Hook in not initialized.
-6	Hook is initialized.
-7	Hook initialization error.
-8	Hook stopping error.
-9	Error during key send.

Table 21: Keyboard library call-back function

Other interface information:

Interface	Description

enum HookFlags
{
HF_None=0,
HF_ExtendedKey=1,
HF_InjectedKey=2,
HF_AltKeyPressed=4,
HF_KeyPress=8,
HF_SentFromLibrary =0x10
};
	Flags which defines additional information about the event: HF_ExtendedKey - the extended key is sent, HF_InjectedKey - the key event is sent by application not by the keyboard, HF_AltKeyPressed - the Alt key is pressed, HF_KeyPress – the key is pressed down, HF_SentFromLibrary – the key is sent from the library.

enum HookMessage
{
HM_None=0,
HM_KEYDOWN=1,
HM_KEYUP,
HM_SYSKEYDOWN,
HM_SYSKEYUP
};
	Defines message type: key event down, key event up, system key event up or system key event up

enum SendKeyFlags
{
SKF_KeyDown=1,
SKF_KeyUP=2,
SKF_KeyPress=3,
SKF_KeyExtended=4,
};
	Used in the SendKeyByScanCode and SendKeyByVirtualCode functions. These flags defines the event type: key event up, key event down, extended key sent. The SKF_KeyPress flag is defindes as SKF_KeyPress=SKF_KeyDown|SKF_KeyUP.

enum BlockOptions
{
BO_BlockAll=1,
BO_PassSentFromLibrary=2,
BO_PassAll=3
};
	Used in the BlockKeys function. It defines the function's behaviour. It can take the following values: BO_PASS_ALL, BO_PASS_SENDED_ONLY, BO_BLOCK_ALL. If it takes the BO_PASS_SENDED_ONLY value, the function passes keyboard events generated by SendKeyByScanCode, SendKeyByVirtualCode and SendText functions and blocks all other keyboard events.

Table 22: Keyboard library other interface information

Native ASAPI will be based on the experience gained in developing of ARE, so some of its requirements may change during development. The integration of other devices with Native ASAPI may be considered. The detail specification of the Android phone and GSM modem interface will be described in D6.4

6 Updates on BNCI Evaluation Suite

This section provides an update on the requirements for the different software implementations related to the BNCI Evaluation Suite. These are naturally related to the requirements given in Table 8 labeled as BNCI#, but relate as well to the requirements labeled as SPROC# and SBPROC# in Table 8. This is based on the fact that the final goal of the BNCI Evaluation Suite is to make flow evaluated BNCI functions into more hard-coded plugins.

Therefore we have changed the priority of some requirements SPROC#, SBPROC#, and BNCI# for the completion of prototype 2 after the experience gained in prototype 1 (PT1) completion. Moreover this same experience has driven the addition of new requirements. In general we can state that the experience during PT1 discourage the implementation of methodologies per se, but just those whose usefulness have been tested in the Evaluation Tool. Therefore we will implement in PT2 only those methodologies that demonstrate in previous experiments to be useful and with a priority of Medium (M) or High (H) level.

As a consequence we have exchanged the priority of requirements SPROC9 and SPROC14, since we have not found heretofore any concrete example for using PCA. Moreover we have added a requirement for a general EMG detection methodology (SBPROC11), which should substitute the more particular cases described in SBPROC5-7. Furthermore and after demonstrating the usefulness of BCI2000 motor imagery implementation based on Enobio signals, we will focus on the implementation of the corresponding methodology in form of plugins SBPROC9. Lastly we have added requirement (SBPROC12) for an interface based on Visual Evoked Potentials in case the Evaluation results are satisfactory.

For the BNCI requirements we focused in the PT1 phase in the p300 paradigm. However this showed to be too limited and complex in the intermediate evaluation phase. As a consequence we have changed the priority of BNCI29-30 for PT2, which indicates we will further our works in p300 evaluation. As well we decided not to focus only on p300 but to extend the experimentation to motor imagery, which we undertook via integration with BCI2000. We will pursue the implementation of this paradigm in form of AsTeRICS plugins as described in the former paragraph and reflected in requirement BNCI31. Moreover we will diversify far more our target by attaining the evaluation of Visual Evoked Potentials, which are reflected in BNCI34-37. Particularly the fulfillment of the 3 last requirements substitutes this of BNCI14-17, which have changed their priority because we can not envisage any practical use.

7 Summary and Conclusions

In this document, the requirements and specifications of the hardware and software architecture of the AsTeRICS system have been refined from the original version discussed in D2.1. An important source of information for these refinements were the results of the user evaluation phase, where the first AsTeRICS system prototype has been presented to primary and secondary users at different user site in Spain, Poland and Austria (see D1.5 [4]). Furthermore a short state-of-the-art analysis has been performed to update the AsTeRICS design goals and include interesting technology where feasible.

The comprehensive list of requirements is grouped into hardware and software sections for the main AsTeRICS components and will guide the development process for the second system prototype. The subsequent specification sections give detailed insight into changes and updates of the architectural concept for the software layers and describe the features of hardware components which either have been chosen yet or are being designed in course of the project. The most important improvements in the second system prototype include:

- Utilisation of a new hardware platform, based upon the ultra-small form factor NanoETXexpress board with Intel Atom CPU by Kontron
    - Reduction of form factor and power consumption of the personal platform, optional extension container for CIM's
    - Integration of a rechargeable battery and essential CIM functions (digital and analogue inputs and outputs) into the main platform
- Switch to Windows-7 embedded to replace the outdated XP operating system and enable volume licensing and deployment of images to multiple platforms
- Improvement of the usability of the ACS (element grouping, easyGUI editor)
- Internationalisation of ARE plugin- and property labelling, user-friendly names for plugins
- Improvement of the Enobio BCI functionalities (VEP, motor imagery, signal processing)
- Development of a head-mounted SVM module (iris tracker)
- Extension of ARE framework and plugins

References

1 AsTeRICS Description of Work, technical annex 1 of the AsTeRICS Grant agreement

2 AsTeRICS Deliverable D2.1 – “System Specification and Architecture”

3 AsTeRICS Deliverable D2.4 – “Report on the State of the Art”

4 AsTeRICS Deliverable D1.5 – “Evaluation Report Integrated Prototype”

5 AsTeRICS Deliverable D2.3 “Report on API specification for sensors to be integrated to the personal platform”

6 AsTeRICS Deliverable D4.2 “Prototype 1 of the AsTeRICS Runtime System”

7 AsTeRICS Deliverable D4.8 “Porting OSGi to the AsTeRICS Personal Platfrom Prototype-2”

8 http://www.jedec.org/news/pressreleases/jedec-publishes-registered-outline-msata-ssd-assembly

9 http://www.transcendusa.com/Products/ModDetail.asp?ModNo=313&SpNo=1&LangNo=0

10 Christoph Veigl: An Open-Source System for Biosignal- and Camera-Mouse Applications; Submission for the Young Researchers Consortium of the ICCHP 2006, Linz

11 Cootes et al. Active shape models-their training and application. Computer vision and image understanding (1995) vol. 61 (1) pp. 38-59.

12 Kass et al. Snakes: Active contour models. International Journal of Computer Vision (1988) vol. 1 (4) pp. 321-331

13 Milborrow and Nicolls. Locating facial features with an extended active shape model. Computer Vision–ECCV 2008 (2008)

14 J. Saragih, S. Lucey and J. Cohn, ``Deformable Model Fitting by Regularized Landmark Mean-Shifts", *International Journal of Computer Vision* (IJCV), 2010.

15 The BioSig Project [Online], Available: http://biosig.sourceforge.net/ [Accessed: June 9 2010].

16 AsTeRICS Deliverable D1.1 – “Report on Users, Preferences and Needs”

17 AsTeRICS Deliverable D1.3 – “Technical Specifications”

Appendix A

```
public interface IAstericsErrorHandling
{

      public void reportError (IRuntimeComponentInstance component,
          String errorMsg);
      public void reportInfo (IRuntimeComponentInstance component,
          String info);
      public void reportDebugInfo (IRuntimeComponentInstance component,
          String info);

}
```