# *Deliverable D3.1: Report on modular architecture of AIDE multimodal interface*

*01/06/2016*

**AIDE**

Adaptive Multimodal Interfaces to Assist Disabled People in Daily Activities

**Project number**: 645322

**Start of the project (duration)**: February 1st, 2015 (36 months)

Research and Innovation Action

HORIZON 2020 Programme

LEIT Pilar KET ICT

**Revision:** V.1

| **Project co-funded by the European Commission within the Horizon 2020 Programme (2014-2020)** | |
|---|---|
| **Dissemination Level** | |
| **PU** Public | **X** |
| **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) | |



**List of reviewers**

| Issue | Date | Implemented by | Control of Changes |
|---|---|---|---|
| v.0.1 | 27/05/2016 | UCBM | Draft version |
| v0.2 | 27/05/2016 | UMH | Proof reading and minor changes |
| v1.0 | 01/06/2016 | UCBM | Final version |
| v2.0 | 01/06/2016 | UMH | Approved version with minor changes |

## TABLE OF CONTENTS

# 1. EXECUTIVE SUMMARY

Deliverable D3.1 “Report on modular architecture of AIDE multimodal interface” (delivery date scheduled for June, 2) has been written in the framework of WP3 and is mainly related to Task 3.1 (“Design and development of a modular architecture for the multimodal interface”, closed at month 16) and Task 3.7 (“Experimental tests of the modular architecture”, that will finish at month 18). The goal of these tasks is to develop and test a modular standard architecture in order to guarantee the communication among the several subsystems constituting the AIDE platform.
To this purpose, a literature analysis on the approaches based on Component-Based Software Engineer (CBSE) has been performed in order to find the most appropriate solution for meeting the AIDE requirements. They have been determined on the basis of the results of WP2 and of an internal survey within the consortium. The gathered information led us to define the Yet Another Robot Platform (YARP) as the most suitable component based software for managing communication among the different subsystems of the AIDE platform.

This document describes the research activities and the achieved results on the design and testing of the modular software architecture developed during months 1-16 of the AIDE project. In particular, in the following details about the software architecture, the implementation of the communication nodes, the preliminary tests on the bidirectional communication between YARP and other development environments, and the preliminary experimental tests with part of the AIDE subsystems will be provided.

# 2. INTRODUCTION

The AIDE system aims to provide a multimodal interface to assist compromised individuals in their daily life by analysing and extracting relevant information from the environment and context factors and from the identification of residual abilities, behaviours, emotional state and intentions of the user. This will be achieved by integrating data from different acquisition devices and processing software (Figure 1). In particular, the user's state and intentions will be inferred by using the following sensory system: 1) Brain-machine interface (BMI) based on EEG brain activity; 2) myoelectrical EMG surface interface based on surface EMG; 3) Wearable physiological sensors to monitor physiological signals, such as heart rate, skin conductance level, temperature or respiration rate; 4) Wearable electro-oculography (EOG) system; and 5) Kinetic and dynamic information provided by the upper limb exoskeleton. The context factors, the current position of the users in their home setting, the gaze patterns of the users within the environment, the 3D shape and pose of the objects at which the users stare at and the user's head and mouth pose will be estimated by means of cameras and gaze tracking systems.

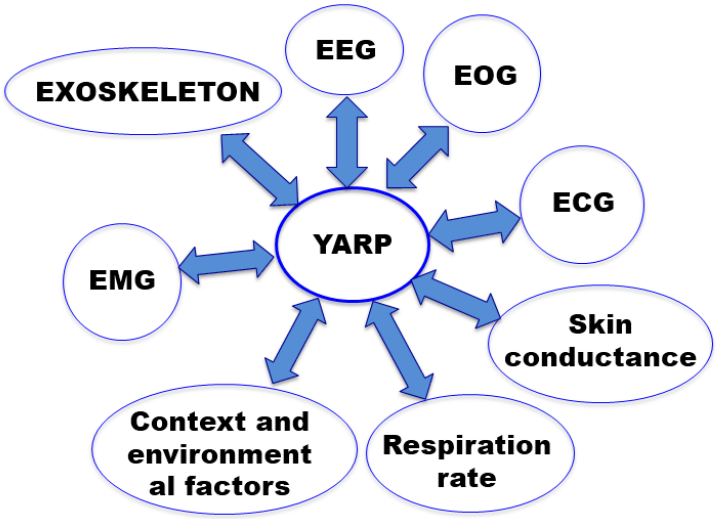


**Figure 1: Schematic representation of the systems that should communicate through YARP.**

One of the required features of this architecture is to guarantee the communication among the different blocks of the AIDE system independently of the chosen platform and software language. Therefore, it has been necessary to develop a modular standard architecture for the multimodal interface supporting its development both at integration and development levels, facilitating a standardized communication between different blocks.

An approach based on CBSE [1] has been adopted to ensure that data can be exchanged efficiently and effectively, through a standardized communication and in such a way that allows resizing and/or modifying the multimodal interface. It will let the user to employ only the modules that are needed in a specific scenario application. CBSE offers a number of advantages, e.g. (i) cost and time reduction for

building large and complicated systems, (ii) improvement of the software quality by ameliorating the performance of the components, (iii) detection of defects within the systems thanks to component approach.
Several software frameworks that are grounded on a CBSE approach have been proposed in the literature. In particular, the Robot Operating System (ROS) [2] is a set of software libraries and tools that help users build robot applications. A ROS system comprises a number of independent nodes, each of which communicates with the other nodes using a publish/subscribe messaging model. This makes ROS flexible and adaptable to the needs of the user. Nodes in ROS do not have to necessarily be on the same system (allowing use of multiple computers) or even on the same architecture. The main limitation of this system is that it has been developed for working under Linux.
The Open Robot Control Software (OROCOS) [3] is a free software and a modular framework for robot and machine control addition; with the Orocos Real-Time Toolkit (RTT) it is possible to perform real-time, on-line interactive and component based applications.
MARIE (Mobile and Autonomous Robotics Integration Environment) [4] is a free software tool using a component-based approach to build robotics software systems by integrating previously-existing and new software components. Its main characteristics lean on distributed computing on heterogeneous platforms and on concurrent use of different communication protocols, mechanisms and standards.

Based on the requirements defined in the framework of the AIDE project, the component based software YARP [2] has been selected. It is a middleware for robotics and supports building a robot control system as a collection of programs communicating in a peer-to-peer way, with an extensible family of connection types (tcp, udp, multicast, local, ...) that can be swapped in and out to match user needs. Namely, it is a framework (i.e., a set of libraries and executable programs) that can be installed on top of Windows OS, permitting it to handle the low level communication with the devices.
Hence, a modular architecture based on the messaging system YARP has been adopted in the AIDE project for connecting the different subsystems of the platform among each other by creating ad hoc nodes. YARP messaging system allows associating a component of the multimodal interface to a node and connecting and disconnecting nodes quickly and easily. Each node has a name and a specific port number. The communication protocol, however, can be multiple and is defined at the time of the link between nodes. The YARP server plays the main role; it can be queried at any time to request information and services to the nodes.
Therefore, the choice of relying on the messaging system YARP guarantees the modularity of the system, making possible to resize the whole system by removing/adding some modules without compromising the system performances. It has been simply realized by turning off the nodes whose counterpart is missing.

The reliability of the proposed approach has been successfully tested during preliminary experimental tests.
The document is structured as follows: the system requirements and the proposed modular architecture are described in Section 3; in Section 4, the results obtained during two sessions of tests are reported. In particular, during the first trials of tests, performance of the communication between the different types of nodes have been assessed. Subsequently, the feasibility and reliability of using YARP for communicating with the AIDE platform subsystems have been demonstrated during an experimental validation with part of the AIDE subsystems. Conclusions and deviations from the planned objectives and the consequent corrective actions are discussed in Sections 5 and 6, respectively.

# 3. DESCRIPTION OF WORK

## SYSTEM REQUIREMENTS

In order to define the system requirements for proceeding with the design of the architecture, it has been necessary to acquire information about the subsystems composing the AIDE platform. To this purpose, a survey within the consortium has been performed; the resulting data are reported in the following:

- Data sending frequency: in the range 20 Hz-3KHz. The optimal value is 2 KHz.
- Length of data in bytes: min: 1, max: 255 bytes.
- Range/scale/options: if this information is provided, YARP can control and pre-elaborate the received/sent data;
- Development language and environment: C++ language, and Matlab and Labview as development environment.

Moreover, in order to reduce the size and the weight of the system to be embedded on the wheelchair it has been decided to use only one high-performance PC running under Windows OS for managing the communication among the AIDE subsystems and all the software packages. The rate of data exchange among the systems is determined by the lowest frequency in the platform.

## DESCRIPTION OF THE ARCHITECTURE

As shown in Figure 2, the proposed modular architecture has to manage the communication among the acquisition devices (i.e. the EEG, supervised by EKUT in the framework of WP4, the EOG by UMH in WP3, the EMG by SSSA in WP4, the gaze estimator of competence of UPV in WP5, the physiological signals, monitored and processed by UMH and EKUT in WP, the indoor localization system and the voice recognizer, by ZED in WP5, and the vision systems by UPV in WP5), the software for user intention and affective state estimation (responsible EKUT and SSSA in the

framework of WP4), the module for user activity recognition in real ADL tasks (responsible UMH in WP5), the exoskeletons (developed and controlled in the framework of WP3 and WP6 by UHM, SSSA, FhG-IPA and UCBM) and the communication, control & entertainment devices (provided by BJ in WP6). The communication among all the AIDE subsystems has been guaranteed independently of the chosen platform and software language.

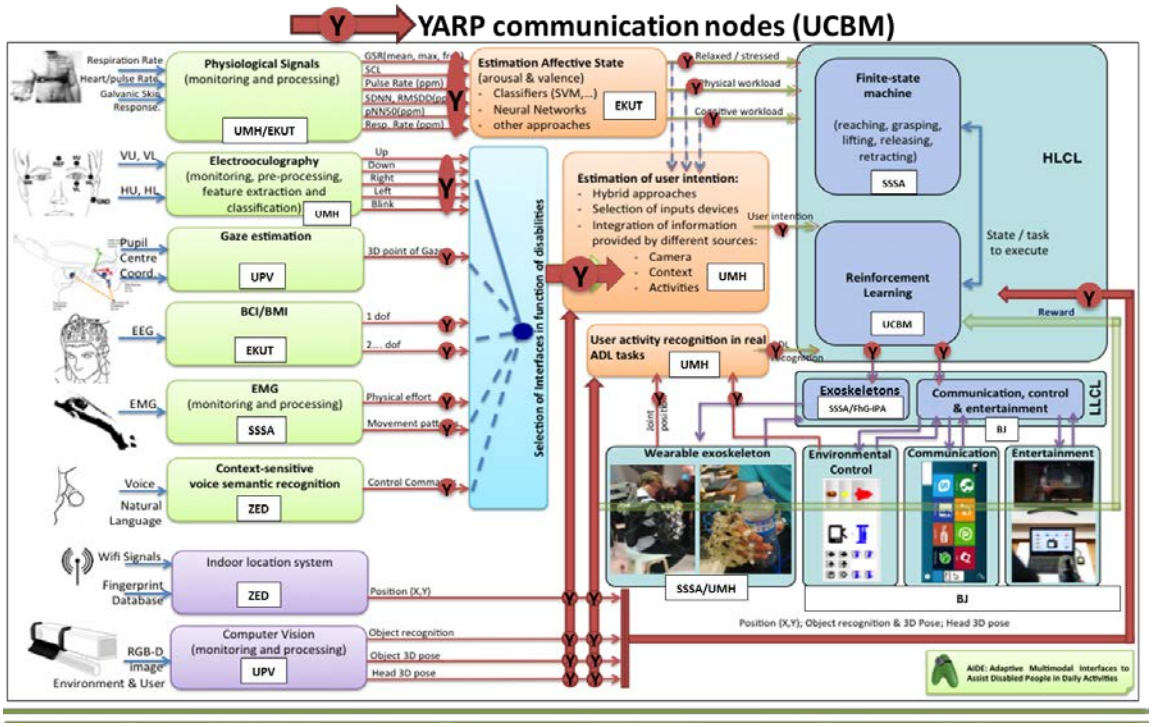


**Figure 2: The AIDE architecture showing the different components of the AIDE system.**

The proposed solution relies on YARP, since it allows addressing all the requirements listed in the previous section, included the use of a single high-performance computer for managing the whole multimodal interface. The overall architecture is shown in Fig. 2, where all the YARP nodes are explicitly drawn.

The chosen communication protocol within the YARP system is the TCP/IP since it is the default communication protocol within YARP and is more stable than the UDP protocol. Each component of the multimodal interface is associated to one node; therefore, the connection or disconnection of a component can be simply performed by connecting and disconnecting nodes. Each node is identified by a label and has a unique communication port on the YARP server that can be queried at any time to request information and services to the nodes (Figure 3). Hence, modularity is easily achieved by removing modules or adding new ones as nodes, without compromising system performance. All messages pass through the central nodes to check their integrity and to be redirected to the correct destination.

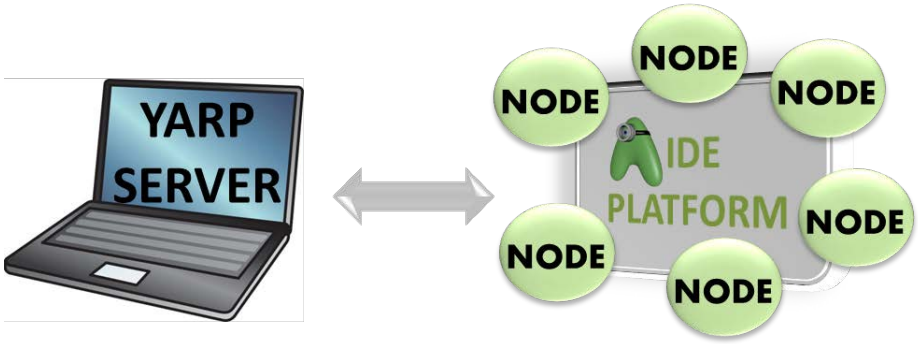


**Figure 3: The proposed solution relying on YARP with the single high-performance computer (on which the YARP server runs) which manage the components of the multimodal interface. Each component is associated to one node.**

Within the YARP communication system, it is possible to use three types of nodes:

- The native node, which is programmed with libraries provided by the YARP platform. This node can send and receive any type of message with any protocol (i.e. full-duplex channel);
- The text node is an external node that can be developed in any programming language supporting TCP/UDP socket. These nodes can send or receive (i.e. half-duplex channel) only ASCII format (text) messages;
- The YARP node is an external node simulating a native node (in the following it is named YARP external node). To simulate a native node means to force a client/server TCP to behave in a way similar to the native nodes. In this way each simulated native node can send or receive all the messages or headers as a native node can do to interact with the YARP server and with all the linked nodes. The strength of this type of nodes is the ability to send one or more data without any formatting (e.g. no ASCII data), even though performance can be lower than the other two types of nodes. The advantage is to to avoid an additional computational burden due to type casting or conversion of incoming data.

In Figure 4 an example of the YARP working principle is shown.

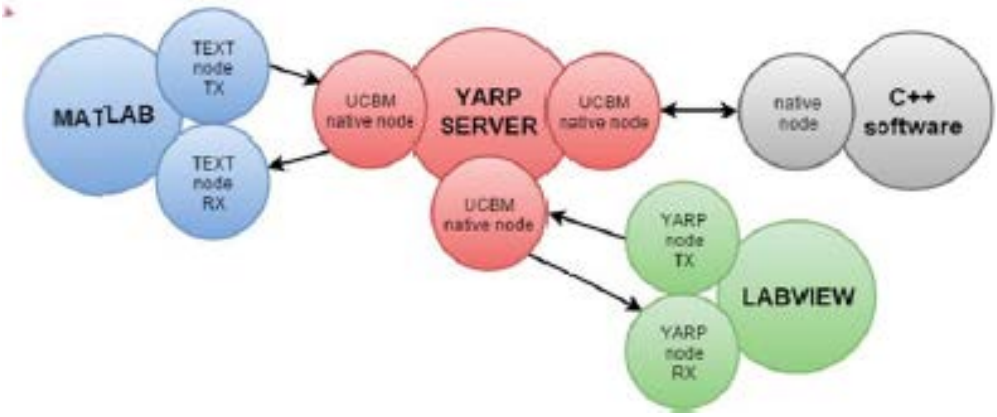


**Figure 4: Example of YARP working principle**

In particular, the YARP server developed by UCBM can transmit to and receive messages from the subsystems of the AIDE platform by using a full-duplex channel and employing native nodes written in C++. On the other hand, the AIDE subsystems can transmit or receive by using one of the aforementioned three types of nodes (native, text or YARP). As shown in Fig. 4, the subsystems developed in languages different from C++ (e.g. MATLAB, LabView, etc…) are connected to the core architecture through text nodes and YARP nodes. Since, they cannot be used for bidirectional communication, two different nodes (in Fig. 4 named TX, i.e. transmitter, and RX, i.e. receiver) are required for the subsystems that need to both send and receive messages from different nodes.

# 4. PRELIMINARY EXPERIMENTAL RESULTS

Experimental tests have been performed within Task 3.7 to verify and assess performance of the proposed software architecture. They have been grouped into two sessions: preliminary tests for assessing performance of communication between the different types of nodes (and consequently extract indications about the most appropriate nodes for a fast enough and reliable communication); subsequently, an experimental validation with part of the AIDE subsystems has been carried out in order to demonstrate feasibility and reliability of using YARP for communicating with the AIDE platform subsystems.

## EXPERIMENTAL SESSION 1

Preliminary tests have been conducted in order to build a platform enabling correct handling of all the signals composing the multimodal interface. In particular, a test to demonstrate the feasibility and the communication performance through YARP between native nodes and non-native nodes has been performed. The YARP server has been installed on a PC running under Windows 10. During the initial tests, data provided by SSSA about joint positions of the arm exoskeleton have been used. 10000 elements from two arrays, called S_AA_DesiredPosition and S_FE_DesiredPosition, have been transmitted from a Matlab script to the YARP server (Figure 3).

The two nodes */matlabY* and */matlabT* have been created in Matlab. The former is a YARP external node exchanging data through TCP/IP connection with TCP protocol, while the latter is a Text external node exchanging data through TCP/IP connection with TEXT protocol.

Afterwards, two additional nodes have been created on C++ Visual Studio, corresponding to the previous ones and called */inBin* and */in.* The first couple of nodes (i.e. */matlabY- /inBin)* through TCP/IP connection and TCP-YARP protocol reached 500 msg/s (500 Hz), the second one (i.e. */matlabT-/in*) reached 2956 msg/s (2956 Hz) on a TCP/IP connection and Text-YARP protocol.

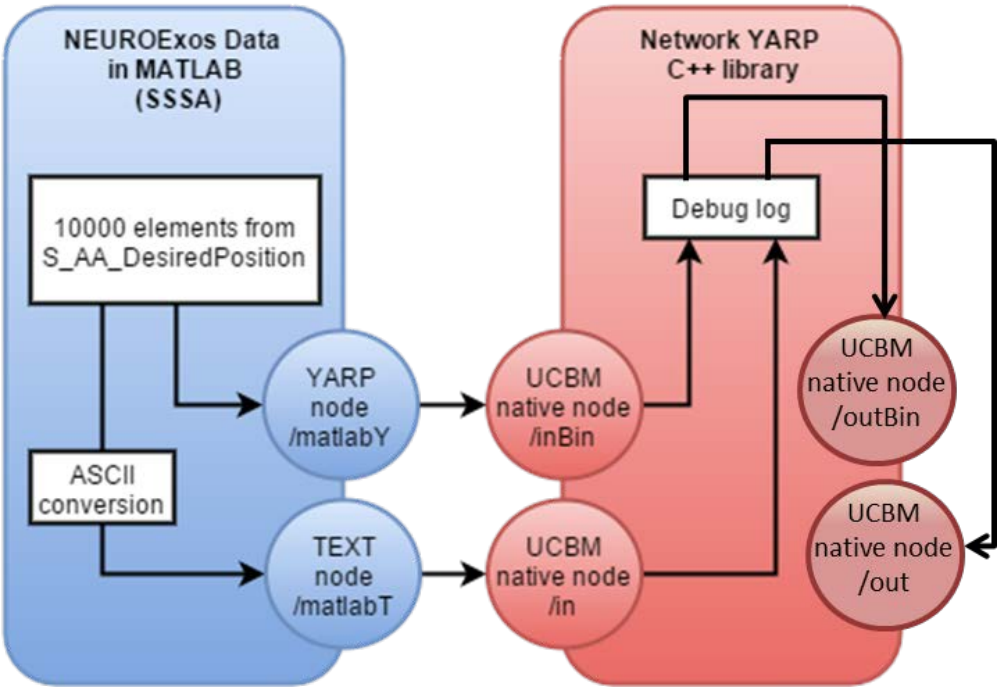


**Figure 5: Example of YARP Communication**

The obtained performance is summarized in Table 1. As shown, when messages are sent through YARP external nodes it is possible to reach a frequency of maximum 500 Hz without losing data. When messages are sent through Text external nodes the maximum frequency without data lost is around 3000 Hz. Therefore, the performed tests have shown that the TEXT-YARP protocol through TCP/IP connection guarantees the best performance in term of data loss. Although it has been found that the best performance can be achieved if messages are sent in Text mode (ASCII) up to 3 KHz, the use of YARP external nodes can reduce the computational burden required by the text nodes for type casting. Therefore, YARP external nodes seems to be the best compromise between performance and computational burden when the working frequency are low, up to a maximum value of 500 Hz.

**Table 1: Obtained performance during preliminary communication tests**

| TX node | RX node | Frequency [Hz] | Data lost [msg] | Delay [ms] |
|---|---|---|---|---|
| /matlabY | /inBin | 500 | 0 | 0 |
| /matlabT | /in | 4235 | 35 | 0 |
| /matlabT | /in | 2956 | 0 | 0.2 |

## EXPERIMENTAL SESSION 2

As a natural prosecution of the experimental tests on YARP performance, the implemented software architecture has been tested during the experimental session carried out at Scuola Superiore Sant'Anna (Pisa, Italy) in May 2016. The involved AIDE partners were UMH, UCBM, SSSA and EKUT.

This first AIDE experimental session aimed at providing a preliminary evidence of the feasibility to allow a user to drink from a glass thanks to the AIDE multimodal robotic system.

During this experimentation, the following AIDE subsystems were used together (Fig. 6) and communication was managed by YARP:

- Shoulder-elbow exoskeleton;
- Hand exoskeleton and forearm pronation/supination device;
- Biosignal processing and recording system: i.e. the wireless EEG recording system, the wearable glasses for EOG acquisition, the device for measuring skin conductivity, a wearable strap placed around the chest for acquiring ECG and respiratory plethysmography and a main processing unit;
- Gaze tracking and context recognition devices.

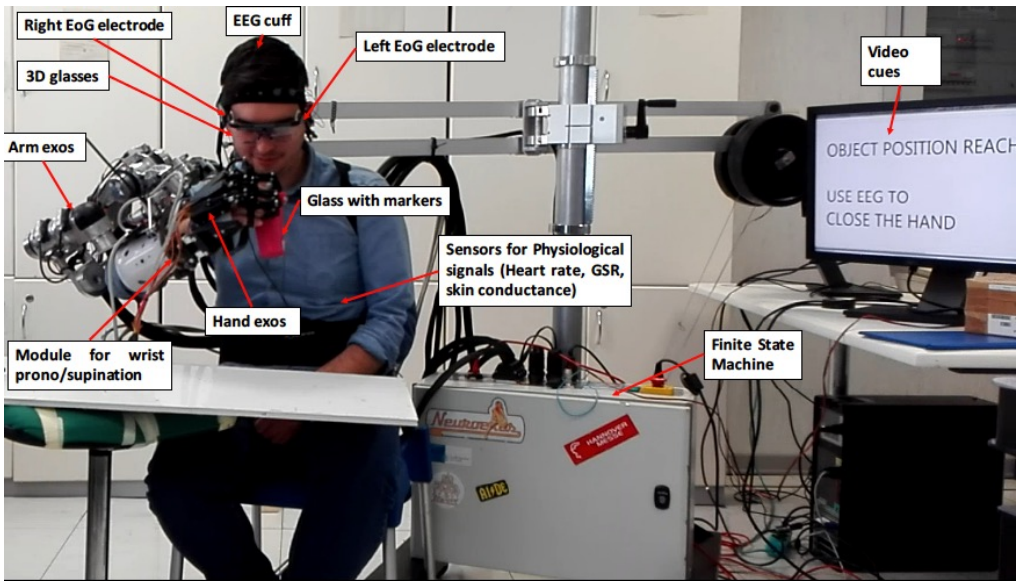


**Figure 6: Experimental scenario**

UCBM has been the responsible for creating all the communication nodes within the aforementioned subsystems of the AIDE platform. The proposed experimental scenario is shown in Figure 6. The arm, wrist and hand exoskeletons have been adapted to the subject arm and hand. In the starting configuration, the shoulder-elbow and wrist exoskeletons have been placed in a configuration comfortable for the user and the hand exoskeleton have been opened.

The gaze tracking glasses and cameras have been used to identify the object and its location in the workspace. EoG Left signal has been used to trigger the exoskeleton

to move to the target (the object or the mouth). The control of the arm exoskeleton ensured the target reaching. After opening the hand exoskeleton has been moved back to the initial position automatically. The EoG Right signal could be used at any times by the user to stop the execution of the movement and go back to the initial position.

The main processing unit has been a computer with Windows 10 OS representing the core of the architecture (i.e. the YARP server). It is a permanent means of communication among the architecture nodes. The software developed in C++ exchange messages with the other subsystems of the platform by using native nodes, which employ full-duplex channels. YARP external nodes simulating the behaviour of native nodes have been adopted for allowing the communication with software modules that are not developed in C++ language, such as MATLAB or LabView, and for communicating with systems running under Linux (e.g. cameras). It has been chosen to use YARP external nodes instead of text nodes since they can optimally work without data loss in the AIDE communication frequency range (as detailed in the following). Furthermore, the use of YARP external nodes notably reducing the computational load required by the text nodes for type casting significantly improving the system performance.
Hence, YARP external nodes have been created under MATLAB, LABVIEW and Linux for sending or receiving data to/from the other nodes of the architecture. A TCP/IP connection with TCP protocol has been adopted for managing the communication among the external nodes and between external nodes and the YARP server.
An overview of the whole communication system adopted in this experimental session is shown in Figure 5. The YARP external nodes under Matlab are in light blue, the YARP external nodes under LabView are in blue, the YARP external nodes under Linux are in yellow and the native nodes are in green (C++ application with official YARP libraries).

In Figure 7, the label of each transmitter and receiver nodes are also reported. Only the transmission of the EEG signals (processed by the BCI2000 software) has been handled through a direct TCP connection (localhost:8000) without YARP, because the software (i.e. BCI2000) did not allow modifications. In this way the utility of YARP (i.e. the control of data loss and of active communication) has been lost, but it did not caused problems. In fact, data from the BCI module consisted of one byte of “0” and “2” (ASCII) characters, corresponding to trigger signals, transmitted in a continuous way with a frequency higher than the one of the Finite State Machine. Furthermore, the Finite State Machine detects the data from the BCI module only if the same value (i.e. “0” or “2”) is received at least two consecutive times. Therefore, the loss of a character has not been critical for the system global performance.

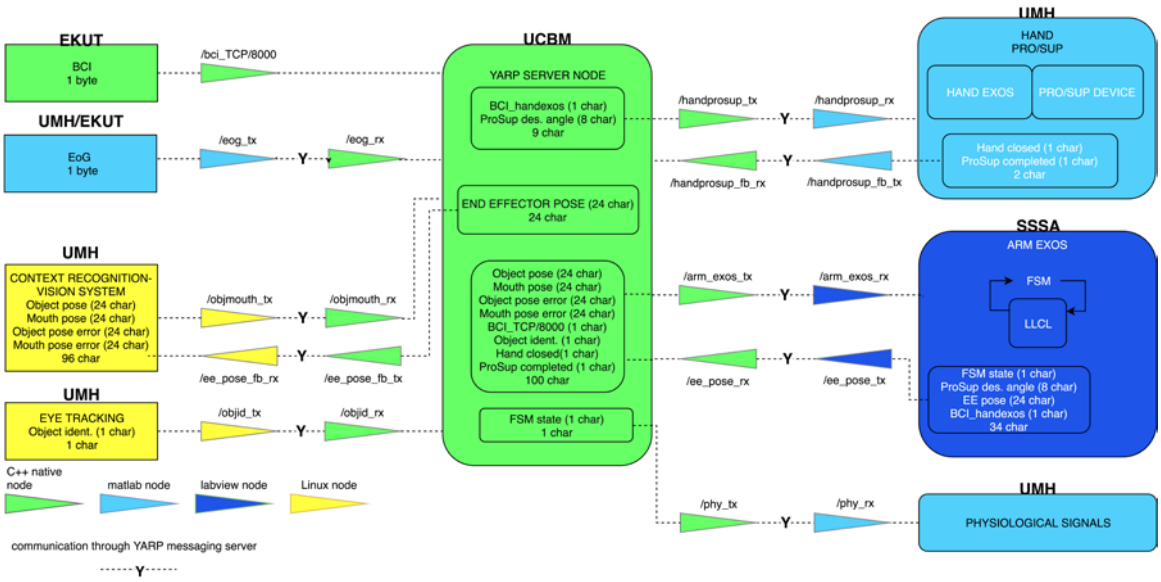


**Figure 7: Overview of the YARP communication system. The native nodes are in green and all the remaining nodes are simulated native nodes written in several programming languages or environment.**

Each node (in YARP application) had a specific dedicated thread making the communication a multithread framework and therefore making the interaction (i.e. access to a node or its disconnection) with the various nodes decoupled from the main linear loop. In this way, each node could be connected or disconnected at any time without causing delays in systems communications. It made the YARP server application, and consequently the communication, more dynamic and reactive.

During the experimental sessions, the communication between YARP and each subsystem of the AIDE platform has been tested.
Furthermore, the communication among all the subsystems described above has been tested during the experimental tests on one participant (a volunteer healthy subject). He has been asked to identify the object to be grasped (i.e. a glass) and to drink from it for 10 times. For measuring performance of the YARP architecture, the timing of sent and received data has been measured and a screenshot of the obtained values during one trial is shown in Figure 8.
The messages among the several parts of the AIDE platform have been exchanged through TCP at a frequency of about 20 Hz (50 ms), which is the frequency of the exoskeleton Finite State Machine (i.e. the lowest frequency in the platform). No data have been lost during the trials due to the adopted low frequency.

The performed tests demonstrated the feasibility and reliability of using YARP for communicating with the AIDE platform subsystems. In particular, the communication with YARP server has been tested when all the nodes were running and exchanging information each other. The tests revealed no data loss for data exchange rate around 45-55 ms, imposed by the exoskeleton Finite State Machine. Tests have been performed up to 4.5 KHz and the results revealed that no data have been lost up to 2 KHz.

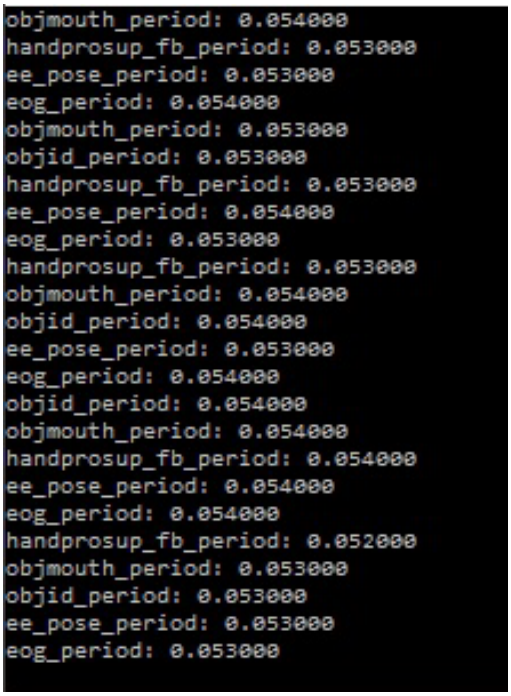


**Figure 8: Screenshot of the measured time for sending and receiving data. The values are reported in seconds.**

## 5. CONCLUSIONS

This deliverable is focused on the design and development of a modular architecture for the multimodal interface able to guarantee a reliable and robust communication among the several subsystems constituting the AIDE platform.

Starting from a detailed literature analysis, the YARP system has been chosen as the most appropriate messaging system for satisfying the set of requirements for the AIDE platform. The demanded architecture for the communication among the several subsystems constituting the AIDE platform has been developed and successfully tested. All related tasks contributing to the deliverable have been accomplished according to defined system requirements.

YARP communication among nodes regarding EEG, EoG, physiological signals (i.e. heart rate, GSR, skin conductance etc.), Finite State Machine, gaze tracking and object identification system has been tested using TCP/IP protocol and a frequency rate of 20 Hz. The YARP server and the receiver and transmitter nodes did not reported errors that could imply system stop. The dynamic connection and disconnection of the nodes allowed us not following a precise order in starting each component.

Therefore, the YARP communication system ensured effective and efficient data exchange and offered high reliability, control on data loss, control on active communication and modularity. Moreover, the system guaranteed the communication among some the AIDE subsystems independently of the chosen platform and software language. The first series of the tests revealed the feasibility

of the YARP messaging system for meeting the AIDE project requirements in terms of performance and control.
The system is ready to use for upcoming experimental tests.

## 6. DEVIATIONS FROM THE PLANNED OBJECTIVES AND CORRECTIVE ACTIONS

During the project, two modifications to the architecture depicted in the proposal were carried out: 1) removal of the data acquisition system, called “Signal Server” and 2) replacement of Robot Operating System (ROS) with Yet Another Robot Platform (YARP) messaging system. The main reasons of these choices can be summarized as follows:
1) The Signal Server was responsible for receiving messages from all the devices and sensors and processing all these signals to produce standardized information for ROS. The complexity of the Signal Server, due to the diversity of information to receive in terms of its data type and frequency, led to remove the Signal Server. In fact, the Signal Server would provide interfaces of many different types and include different processing pipelines to transform the data in parallel. As a consequence of this, the Signal Server was potentially a bottle neck for the whole system and it was agreed by the consortium to find a better approach by giving more intelligence to all the input devices and sensors. In other words, the input information would be restricted to a small amount of data types, and the developer of each input device would be the responsible to define and develop the appropriate processes to adapt its data.
2) Most of the partners were developing software modules running under Windows OS, while ROS is a software running under Linux OS. The adoption of ROS did not allow using only one PC, or else would require the installation of a virtual machine with Linux (thus entailing loss of performance). Therefore, since the target hardware for the entire solution became a Windows computer attached to a wheelchair, it has been decided to replace ROS with YARP. Using this design, the Signal Server disappears and all the communications are carried by YARP.

## REFERENCES


1. C.Schlegel, A. Steck and A. Lotz (2011). Robotic Software Systems: From Code-Driven to Model-Driven Software Development. Book of Robotic Systems - Applications, Control and Programming, Chap 23, 473-502
2. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng. (2009). ROS: an open-source Robot Operating System. ICRA workshop on open source software, Vol.3, no. 3.2
3. (http://www.orocos.org/)
4. http://marie.sourceforge.net/wiki/index.php/Main_Page]
5. Fitzpatrick, P., Ceseracciu, E., Domenichelli, D., Paikan, A., Metta, G., and Natale, L., “A middle way for robotics middleware.”, Journal of Software Engineering for Robotics, 5(2): 42-49, 2014 - http://www.yarp.it/