

# SIEMPRE

Social Interaction and Entrainment using Music PeRformance

# SIEMPRE

## D3.1 – Techniques for data acquisition and multimodal analysis of eMAP signals.

<i>Version</i>	<i>Edited by</i>	<i>Changes</i>
0.1	Miguel Ortiz, QUB	
0.2	Paolo Coletta, Barbara Mazzarino, UNIGE	
0.3	Nicholas Gillian, QUB	

## TABLE OF CONTENTS

<b>INTRODUCTION .....</b>	<b>3</b>
<b>HARDWARE.....</b>	<b>3</b>
<b>LAB-SPECIFIC SETUPS. ....</b>	<b>5</b>
<b>SOFTWARE .....</b>	<b>6</b>
<b>XML EXPERIMENT CREATOR.....</b>	<b>7</b>
<b>SYNCHRONISATION TOOL .....</b>	<b>10</b>
SYNCHRONISATION TESTS.....	11
SYNCHRONISATION TEST RECORDING .....	12
SYNCHRONISATION TEST RESULTS .....	12
<b>THE EYESWEB SIEMPRE LIBRARY .....</b>	<b>16</b>
<b>RECORDING PROTOCOL .....</b>	<b>17</b>
OUTLINE OF THE SIEMPRE SYNCHRONISATION PROTOCOL.....	18
RECORDING SETUP.....	18
RECORDING PROTOCOL .....	19
<b>VISUALISATION .....</b>	<b>19</b>
DATAPACKS REPOSITORY & VISUALIZER .....	19
<b>APPENDIX I - TUTORIAL.....</b>	<b>23</b>



## Introduction

This document describes in detail the technological apparatus needed for the acquisition of synchronised low level eMAP signals during the experimental sessions of the SIEMPRE project.

A particular focus is the synchronising of several different sources (having very different properties in terms of e.g., sampling rate, precision, etc.) in order to record a consistent data set. This document explains in detail the required setup and steps to take for the recording of eMAP signals within the SIEMPRE project's proposed experimental sessions.

the releases of the software apps and modules and related documentation are part of this deliverable

## Hardware

In order to carry out the experiment sessions proposed by the SIEMPRE consortium, it is necessary to capture and synchronise large number of multimodal signals. This include hardware and software for professional audio capture, high-resolution and high-speed video-cameras (for full-body, expressive gesture and behaviour analysis), body-worn and ambient sensors (both kinematic and physiological), instrument-attached low intrusion motion sensors (accurate capture of instrumental gesture parameter cues). For physiological measures (electroencephalography, electromyography, skin conductance, cardiac rhythm), SIEMPRE will be employing a broad range of systems from low cost sensors developed by QUB and already being used in live performance settings throughout the world to "gold standard" physiological equipment by Biopac. These will provide an extensive set of unobtrusively measured, yet rigorously validated, expressive Movement, Audio, and Physiological (eMAP) data. As for motion capture systems, data acquisition will be performed with a Qualysis system (Qualysis, Sweden) with five or more cameras recording the position of passive markers placed on players' bows and conductors' baton and left hand.

High spatial and temporal resolution video-cameras from different points of views (e.g., top, lateral, and frontal views) will be used in order to enable the application of computer vision techniques and visual-based expressive gesture processing techniques. Audio will be acquired independently from each of the performers by means of microphones attached to the instruments' bodies. Bowing gesture parameters (bow velocity, bow-bridge distance, and bow pressing force) will be acquired by means of 6DOF sensors attached to the instruments' bodies and to the bows.



20 May 2011



The hardware components of the SIEMPRE system are as follows:

**Table 1 Acquisition interfaces**

Signal	Device	Sampling Frequency	Connection to PC
Audio	MOTU 828 MkII	48 Kz	Firewire 400
Motion Capture	Qualisys Qqus300 system	100 Fps	Ethernet
High Speed Video	Point Grey Grasshopper (only QUB)	100 Fps	Firewire 800
High definition video	JVC GY-HD-251 (only UNIGE)	60Fps	SDI
On-body/instrument sensors	Measurement Computing 64ch analog board	500 Hz	USB
Thermo-graphic imaging			
Active position markers	Polhemus	240 Hz	USB

**Table 2 Sensor systems**

Sensors	Connecting Device	Sampling Frequency	Connection to PC
Microphones	MOTU 828 MkII	48 Kz	Firewire 400
On-body/instrument markers	Qualisys Qqus300 system	100 Fps	Ethernet
Physiological sensors (I-cubeX)	Measurement Computing 64ch analog board	500 Hz	USB
Kinematic sensors	Measurement Computing 64ch analog board	500 Hz	USB
On-body/instrument sensors	Measurement Computing 64ch analog board	500 Hz	USB

#### Additional Hardware

- Audio mixing desk
- Speakers
- Arduino synchronisation test box

These elements constitute the SIEMPRE experimental setup that will be used for the duration of the project.

In order to validate the experiments, we have defined a base hardware setup that is duplicated in QUB and UNIGE. This allows for repetition of experiments at both research centers that warrantee consistency in terms of acquisition equipment. It has also been agreed that all experiments that require the full set of sensors will be carried out at Casa Paganini or SARC exclusively.

### Lab-specific setups.

QUB has a dedicated room to conduct experiments and a separate control room for monitoring and analysis. The system in place at the Sonic Arts Research Centre is demonstrated in Figure 1.

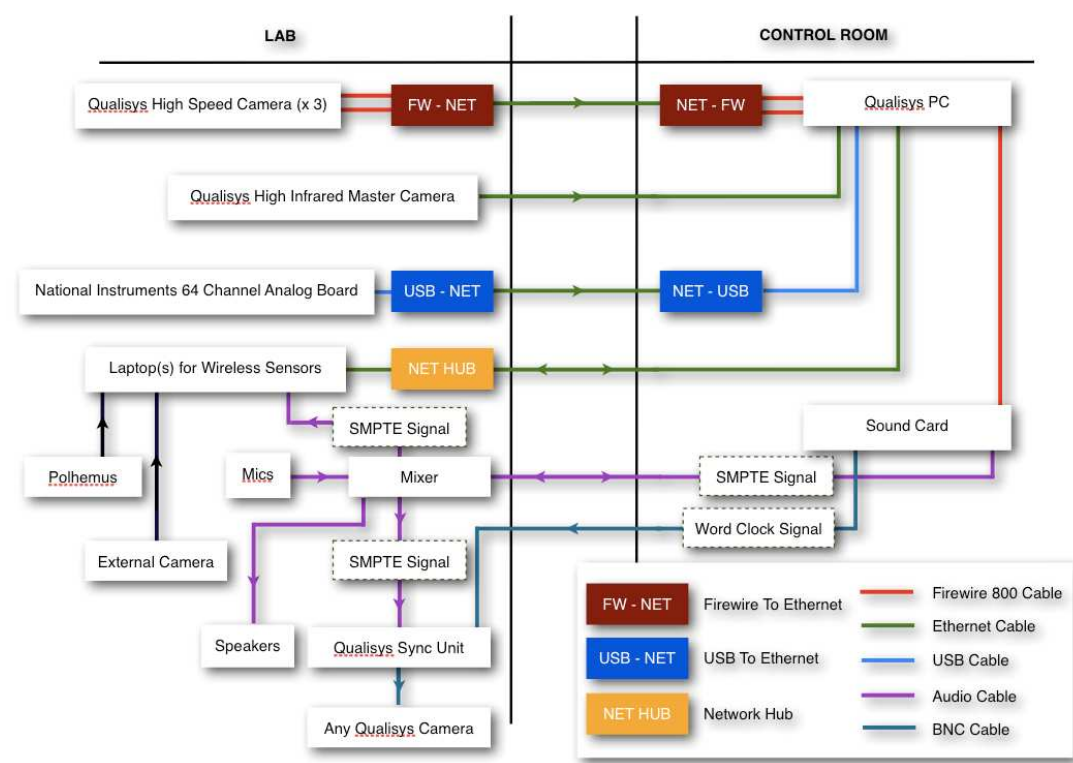


Figure 1 QUB hardware setup.

Figure 2. shows the hardware setup at Casa Paganini.

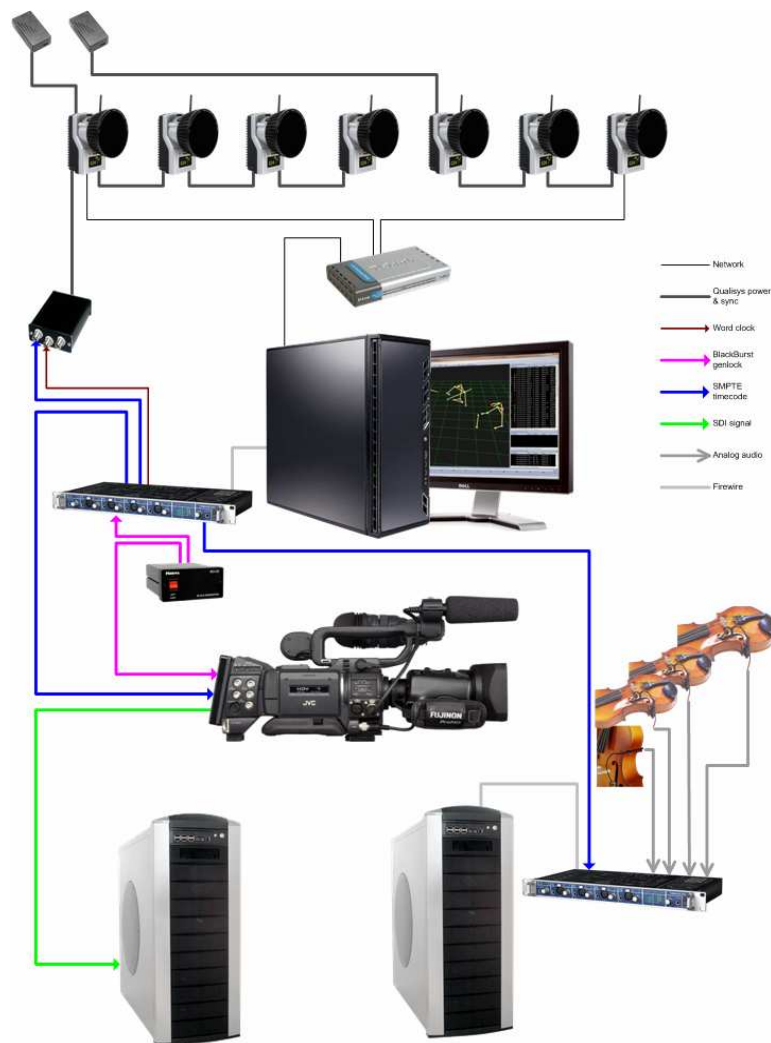


Figure 2 Casa Paganini hardware setup.

## Software

In order to record, synchronise, visualise and analyse data recorded by the various systems a set of software tools have been developed. These include:

- An XML experiment creator.
- Matlab synchronization tool.
- Eyesweb SIEMRE library.
- XML signals visualiser.



20 May 2011



## XML Experiment Creator

The XML experiment creator (see figure 3) is a tool that allows any inexperienced user to create a layout for a multi-modal recording experiment following the SIEMPRE rules defined during the project. It is based on an intuitive GUI where the user designs the structure of the experiment following a set of node templates (Experiment, Take, Subject, Sensors, etc) and visualizes the structure of the experiment in a tree model view, representing this node templates in each of the branches and containing as leaves of each branch the data files that will be created after the multi-modal recording has been done. This node leaves can include *data files* in the SIEMPRE File Format (BWF with an iXML chunk for metadata) for audio or sensor data, *user annotations*, any *description* text or *files* with a video, the score of the piece, a technical paper that explains the experiment, etc.

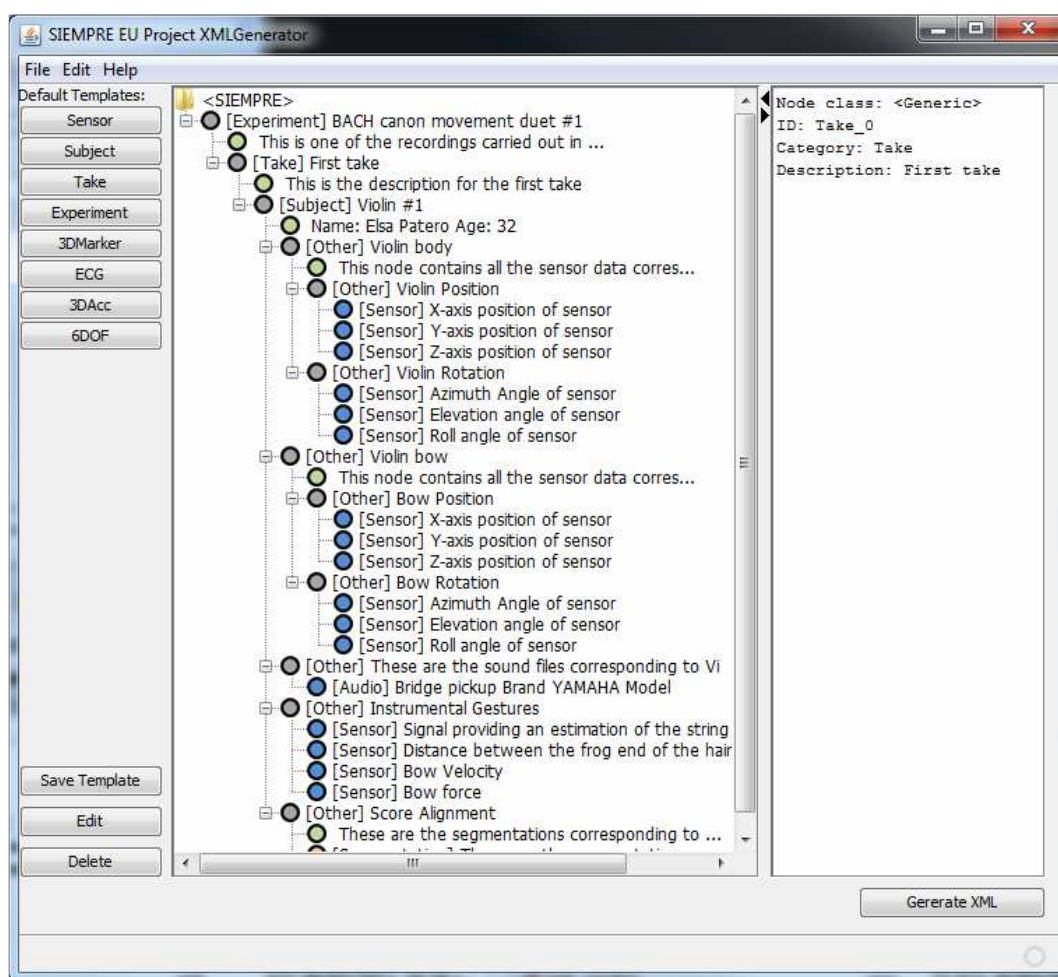


Figure 3 GUI of the XML Experiment creator tool with an opened experiment.

The previous screenshot (figure 3) shows the GUI of the XML Experiment creator tool with an opened recording experiment. As you can see, in the left part, there are some buttons for inserting predefined node templates, and the process to insert basic nodes is based on right clicking with the mouse over a node and a context menu (figure 4) shows up with the allowed options. The allowed children for each node are based on a pre-defined DTD XML schema (figure 6). That won't allow the user to create an experiment that does not follow the DTD XML schema rules. Within the context menu option for nodes there are also Copy/Cut and Paste options in order to make the editing of the experiment easier.

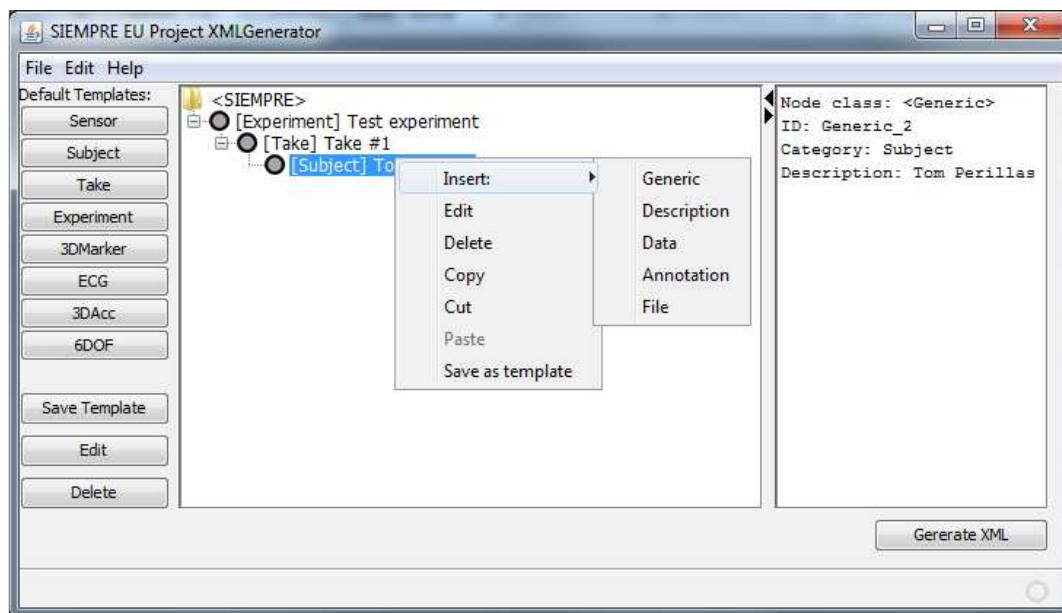


Figure 4 Context menu that appears when right mouse button is pressed over a node.

In the central main window of the tool, the nodes are represented with the following colors:

- Generic nodes (Experiment, Take, Subject, Sensor, etc)
- Annotation nodes (User annotations, segmentations, etc)
- Data nodes (Audio, Data, etc)
- Description nodes (Text descriptions, sensor manufacturer specs, etc)
- File nodes (Video, Score, Research paper, etc)

When the mouse pointer is over a certain node, a tooltip appears showing all attributes and the associated values for this particular node as shown in figure 5. There are some attributes that are not mandatory or does not have to be filled by the user, for instance when specifying the filename for a data node, the tool opens a file browser to allow the user select a BWF file from the hard disk, then the tool opens the BWF file, reads the header information and the iXML chunk and fills the attribute values of the data node (sampling rate, number of channels, etc) with the values defined in the BWF file.



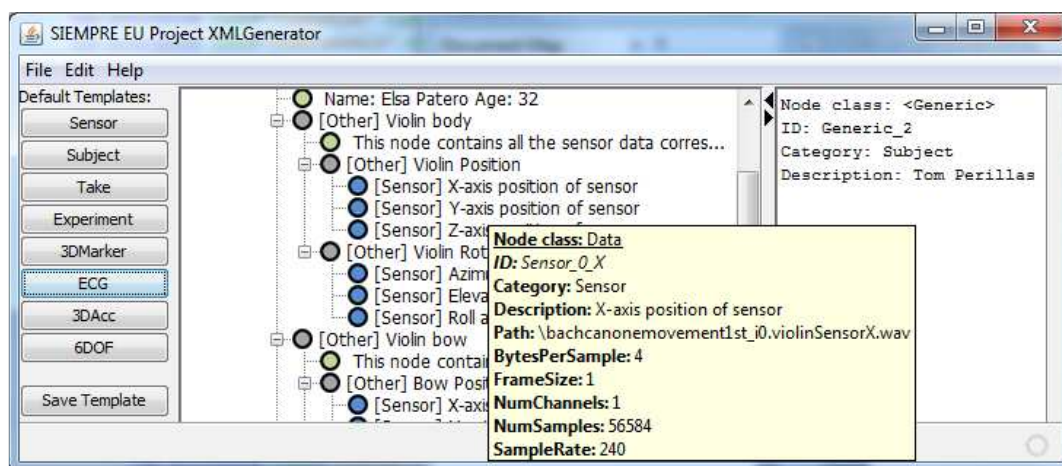


Figure 5 Tooltip showing node attributes and its values.

After creating an experiment, the user just presses “Generate XML” button to save the XML file to disk. This XML file can be opened later for further editing or completion of the experiment by using the menu (File -> open) option. The XML generated is a regular XML so it’s up to the user to use any available XML reader to view the content of the file.

Once the experiment is finished, the user has to create a zip file, *which we will call datapack*, with the XML file and all other files involved (linked to the XML) in the experiment (BWF files, videos, score, segmentation, papers, etc). Once the zip is created it will be uploaded to the datapacks repository for being visualized and shared with the community.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="SIEMPRE">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Generic"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Generic">
    <xs:complexType>
      <xs:attribute name="ID" type="xs:string" editable="0"/>
      <xs:attribute name="Category" type="xs:string" value="Experiment|Take|Subject|Other"/>
      <xs:attribute name="Description" type="xs:string" default="Description text for Generic"/>
      <xs:sequence>
        <xs:element name="Generic"/>
        <xs:element name="Description"/>
        <xs:element name="Data"/>
        <xs:element name="Annotation"/>
        <xs:element name="File"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Data">
    <xs:complexType>
      <xs:attribute name="ID" type="xs:string" editable="0"/>
      <xs:attribute name="Category" type="xs:string" value="Audio|Sensor|Other"/>
      <xs:attribute name="Description" type="xs:string" default="Description text for Data" />
      <xs:attribute name="Path" type="xs:anyURI"/>
      <xs:attribute name="BytesPerSample" type="xs:decimal" editable="0"/>
      <xs:attribute name="FrameSize" type="xs:decimal" editable="0"/>
      <xs:attribute name="NumChannels" type="xs:decimal" editable="0"/>
      <xs:attribute name="NumSamples" type="xs:decimal" editable="0"/>
      <xs:attribute name="SampleRate" type="xs:decimal" editable="0"/>
      <xs:sequence>
        <xs:element name="Description"/>
        <xs:element name="Annotation"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Annotation">
    <xs:complexType>
      <xs:attribute name="ID" type="xs:string" editable="0"/>
      <xs:attribute name="Description" type="xs:string" default="Description text for annotation"/>
      <xs:attribute name="BeginTime" type="xs:decimal"/>
      <xs:attribute name="EndTime" type="xs:decimal"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



```

<xs:sequence>
  <xs:element name="Description"/>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="Description">
  <xs:complexType>
    <xs:attribute name="ID" type="xs:string" editable="0"/>
    <xs:attribute name="Category" type="xs:string" value="Description|Manufacturer|Other"/>
    <xs:attribute name="Text" type="xs:text" default="Description text for description"/>
  </xs:complexType>
</xs:element>

<xs:element name="File">
  <xs:complexType>
    <xs:attribute name="ID" type="xs:string" editable="0"/>
    <xs:attribute name="Category" type="xs:string" value="Segmentation|Score|Video|Other"/>
    <xs:attribute name="Description" type="xs:string" default="Description text for File"/>
    <xs:attribute name="Path" type="xs:anyURI"/>
    <xs:sequence>
      <xs:element name="Description"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>

```

Figure 6 DTD XML schema for multi-modal SIEMPRE recording experiment.

## Synchronisation Tool

After the trials from an experiment have been recorded using the SIEMPRE Synchronisation Protocol (*SPP*, see the specific section on this same document) the data can then be automatically cut using the synchronisation tool so that the data from each sensor all start at TRIAL TIME ZERO and all end at TRIAL TIME END. The synchronisation tool can load data in any of the following file formats:

- **WAVE**
- **VIDEO**
- **DataWithSMPTE** – this is simply a text file containing the sensor data from N sensors saved as a space or tab delimiter separated matrix. The first column in this matrix must contain the decoded SMPTE time code.
- **DataWithClick** – this is simply a text file containing the sensor data from N sensors saved as a space or tab delimiter separated matrix. The first column in this matrix must contain the TTL pulse signal.

When loading each file into the synchronisation tool, the user can specify what the file format of that current file is along with any additional information. In addition to loading each of the various data files containing the raw sensor data captured by each hardware device, the user should also input the two SMPTE time code values that represent the TRIAL TIME ZERO and TRIAL TIME END.

When the user has pointed the synchronisation tool to the location and file format of each of the files recorded during the experiment the tool can then load each file and perform the automatic synchronisation process. After the multimodal data has been synchronised, it will be cropped to start from TRIAL TIME ZERO and end at TRIAL TIME END and the cropped data will be exported to the SIEMPRE File Format (**SFF**). The SFF is based on the Broadcast Wave Format<sup>1</sup>, which itself is based on the Microsoft WAVE audio format, and features an iXML chunk which enables metadata to be easily saved with the audio data in the file. The SFF

<sup>1</sup> Broadcast Wave Format: [http://en.wikipedia.org/wiki/Broadcast\\_Wave\\_Format](http://en.wikipedia.org/wiki/Broadcast_Wave_Format)



uses this iXML chunk to write specific information about the contents of the file, such as the data source (i.e. accelerometer X data) or other information such as the sample rate of the capture device. A unique SFF file will be created for each dimension of each modality.

### Synchronisation Tests

A number of tests have been conducted to validate the synchronisation of a large number of hardware devices designed to capture a wide range of heterogeneous signals. The devices used in these tests were:

- Qualysis Motion Capture System with:
  - 5 Oqus IR Tracking Cameras capturing @ 100Hz
  - 3 High speed point grey cameras<sup>2</sup> capturing @ 100Hz
  - 1 Measurement computing USB-2533 analog board with 10 channels @ 500Hz
- MOTU 828 capturing:
  - 2 Mono audio signals @ 48kHz
- 2 I-CubeX microDig wireless acquisition unit<sup>3</sup> capturing:
  - 1 channel analog signal @ 250Hz

The audio was recorded using Adobe Audition, with the I-CubeX data being recorded in EyesWeb on two different computers. The I-CubeX data was time stamped with the decoded SMPTE time code at the software level using EyesWeb. The master SMPTE clock signal was generated using the MOTU 828 at 48kHz with a SMPTE frame rate of 25fps. A third computer was used to record the SMPTE clock signal as an audio signal along with a single TTL pulse that was triggered at TRIAL TIME ZERO and another TTL pulse that was triggered at TRIAL TIME END. These pulses could then be used to synchronise any external hardware device that could not record a SMPTE signal but that could record an external TTL sync signal, such as a Polhemus.

### Setup

A synchronisation test signal was generated using an Arduino micro-controller which consisted of a 200 millisecond pulse that was repeated every 2 seconds for the duration of the 5 minute test period. This signal was sent as an analog signal, an audio signal and also triggered an LED and an IR-LED to light up. The analog signal was connected to the first input in the Measurement Computing USB-2533 analog board and also connected to the first input of each of the I-CubeX wireless acquisition units. The Oqus IR cameras, Point Grey high-speed cameras and Webcam were all positioned so they could see the IR-LED and LED. The audio signal from the Arduino was sent to the MOTU 828 on the first input. The SMPTE audio signal from the MOTU 828 was also feed back into the MOTU on the second input. A TTL pulse signalling the TRIAL TIME ZERO and TRIAL TIME END points was generated using a push switch and second Arduino and sent to another computer that recorded this signal as an audio file along with the SMPTE time stream.

---

<sup>2</sup> Point Grey high-speed Grasshopper 2 cameras:

[http://www.ptgrey.com/products/grasshopper2\\_firewire/grasshopper2\\_firewire\\_camera.asp](http://www.ptgrey.com/products/grasshopper2_firewire/grasshopper2_firewire_camera.asp)

<sup>3</sup> 4 I-CubeX microDig wireless acquisition unit:

[http://infusionsystems.com/catalog/product\\_info.php/products\\_id/204](http://infusionsystems.com/catalog/product_info.php/products_id/204)



### Synchronisation Test Recording

Three five-minute recordings were saved for analysis. After starting the recording processes in the Qualisys Track Manager and Adobe Audition, the test pulse signal generated by the Arduino was started and was stopped a few seconds before the five minute recording session had finished.

### Synchronisation Test results

After the data had been recorded, the Analog, 3D data and SMPTE timestamp data was exported from the Qualisys Track Manager as a Matlab variable and the saved audio data in Adobe was exported as a wav file. This data was then loaded into Matlab for analysis. The analysis process consisted of:

1. Loading each file
2. Decoding the SMPTE audio signal in one channel of the wav file that was exported from Adobe into the SMPTE time stamp values
3. Detecting the pulses in the second channel of the wav file that was exported from Adobe and tagging each pulse with the corresponding decoded SMPTE value
4. Detecting the pulses in the analog data that was exported from Qualisys and tagging each pulse with the corresponding SMPTE value (with the SMPTE value being taken from the SMPTE time stamp stream that was also exported from Qualisys)
5. The SMPTE time stamp stream values were offset by - 40 milliseconds to account for the 1 SMPTE frame offset reported by Qualisys
6. Each pulse in the analog data was then matched with the corresponding pulse in the audio data and the time difference between the two corresponding SMPTE time stamps was computed (as the Euclidean distance between the floating point equivalent SMPTE times)
7. The mean difference and standard deviation of all of the pulses in the five minute recording was then computed

The mean difference and standard deviation for the three tests are shown in Table 3. The results from each take can be found in Figure 7, **Error! Reference source not found.** and **Error! Reference source not found.** for takes 1, 2 and 3 respectively. There are two anomalies in these results:

1. The mean difference across all three tests is much higher than reported by the Qualisys support team
2. In takes 2 and 3 the error is not constant and seems to process over the duration of the five-minute recording. This error is even more evident in Figure 10 which shows how this error grows over the duration of a fifteen minute recording. There also appears to be a pattern in this error, with a jump in the difference at approximately 420 seconds, 620 seconds and 780 seconds. Perhaps the strangest thing about this processing error is that it never appears to occur in take 1 (we have noticed this over many days of different recordings with the various parameters that have been tested over the last two months). This lack of processing error can not be explained and it should be noted that even if a number of takes are recorded and then the counter is reset to take 1 (therefore take 1 is not the first take of the day) that this lack of processing still occurs in take 1 but will reappear in all other takes

Table 3

Analog Vs Audio Pulse Time Difference Sync Results		
Take	Mean Difference	Standard Deviation
1	10.08ms	0.60ms
2	11.75ms	1.30ms
3	13.11ms	1.40ms

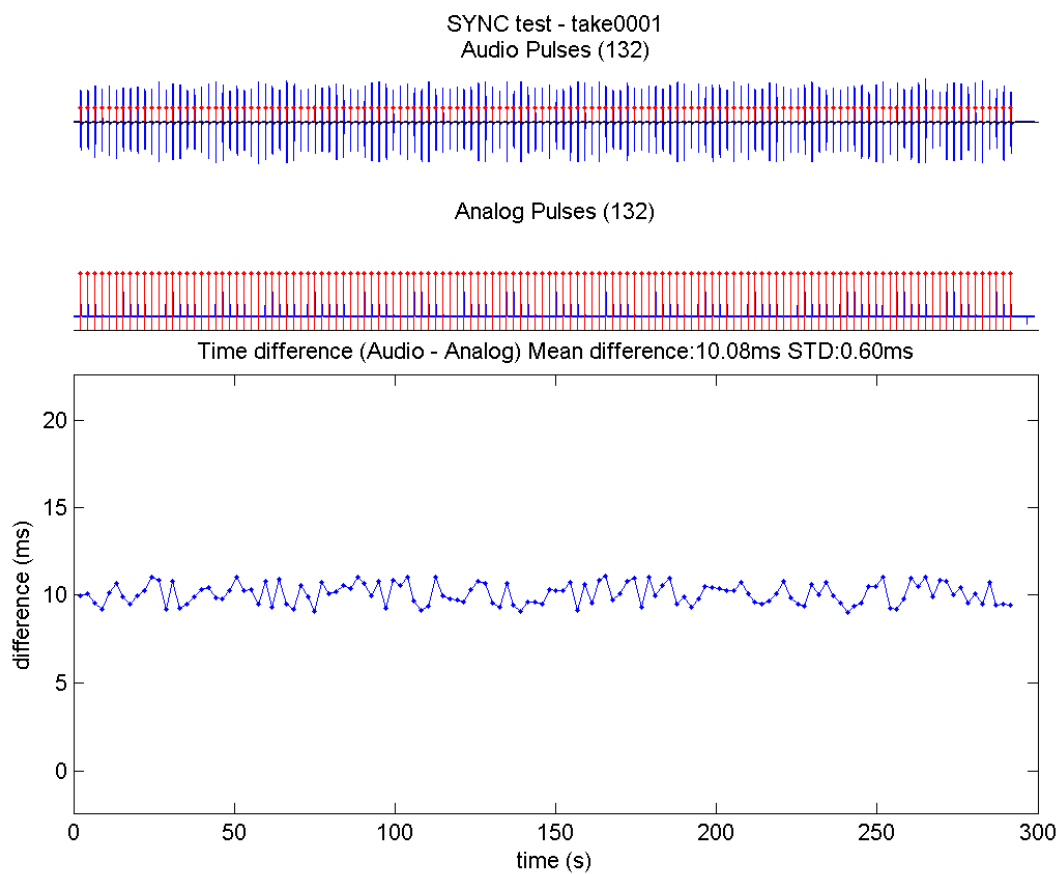


Figure 7



20 May 2011



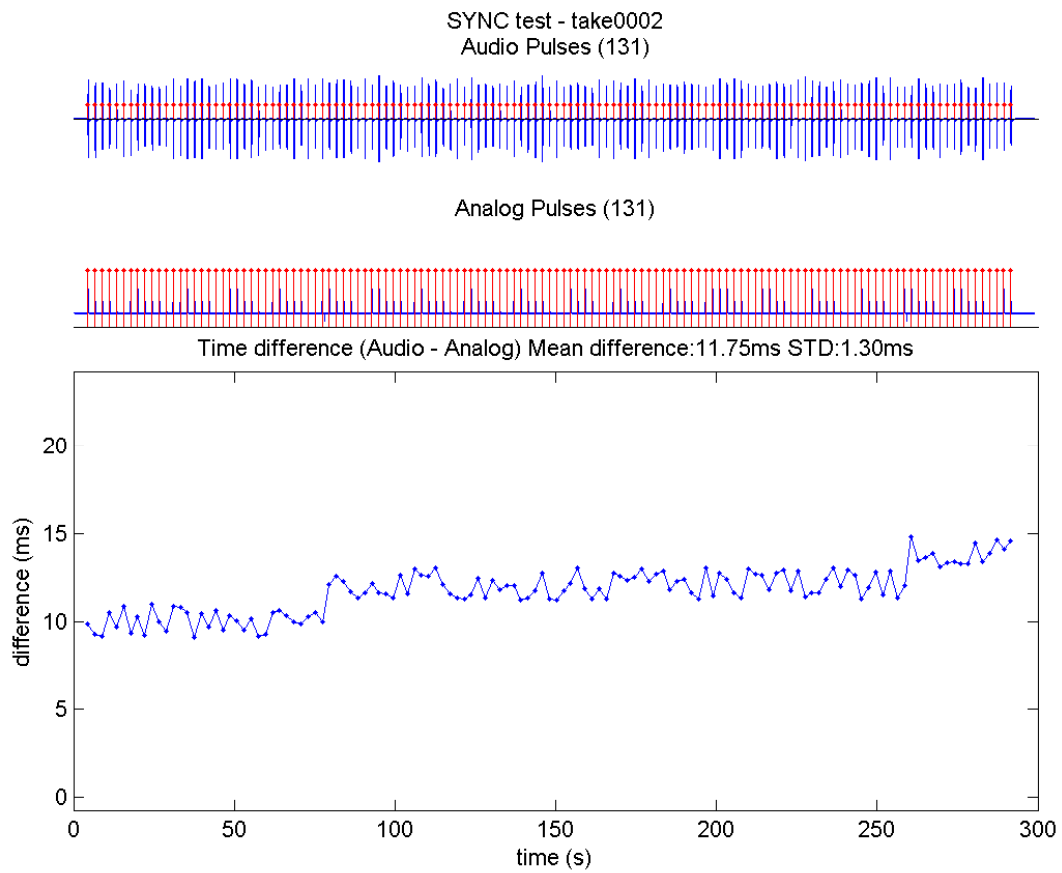


Figure 8



20 May 2011



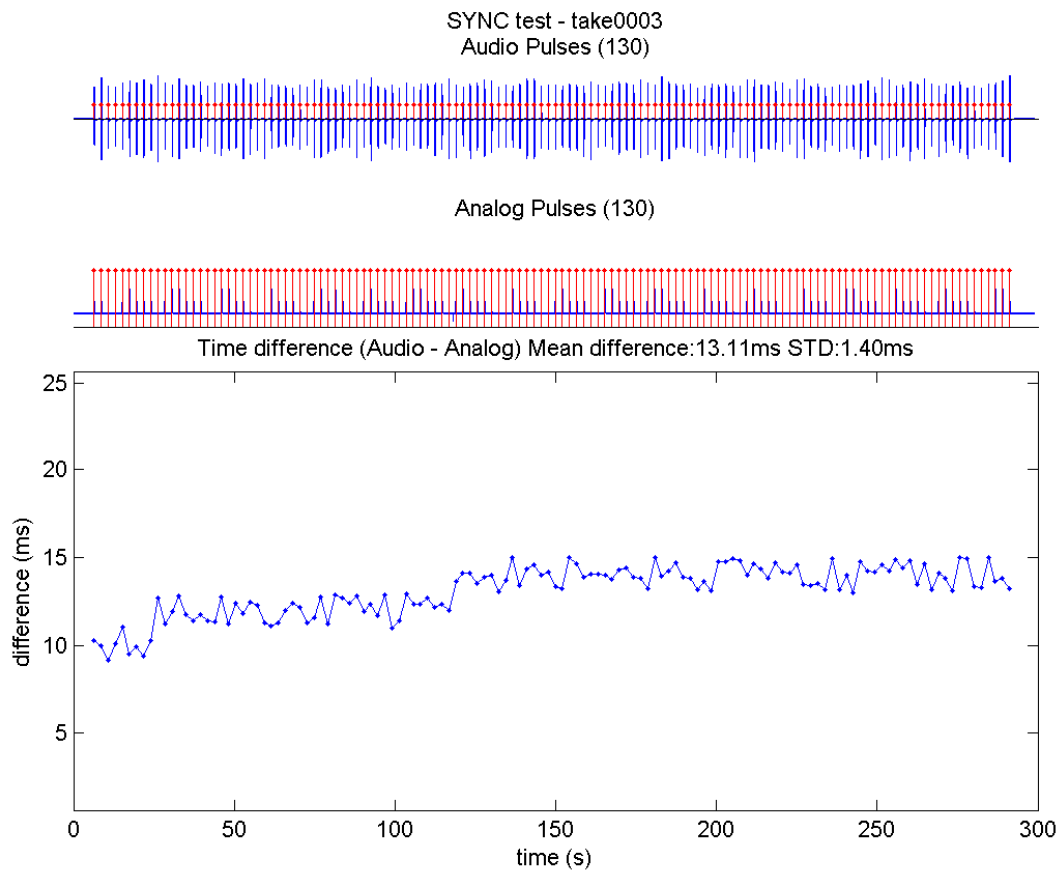


Figure 9



20 May 2011



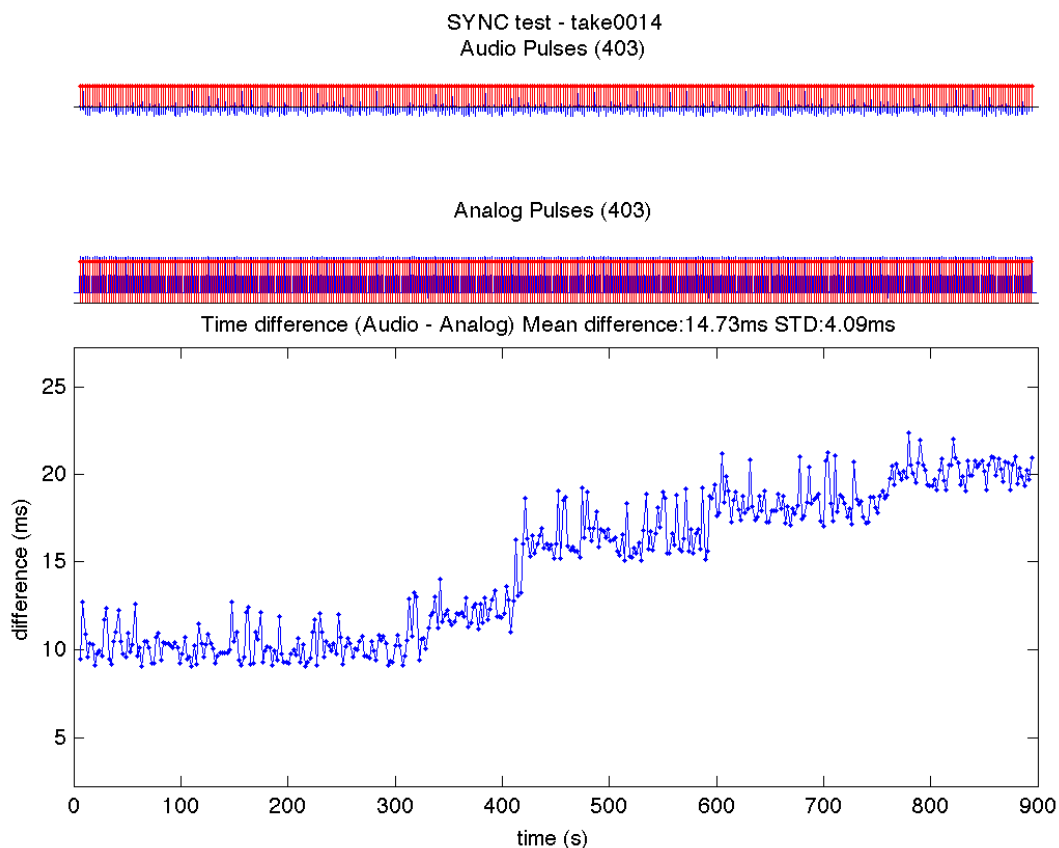


Figure 10

### The EyesWeb SIEMPRE Library

The EyesWeb SIEMPRE Library includes a collection of software modules and patches which have been specifically designed and developed to support the requirements from the EU ICT Siembre project. In particular, the SIEMPRE Library integrates the EyesWeb platform with modules which are required for a distributed multimodal data recording and playback.

- The SMPTE decoder block received an audio signal in inputs and decodes the SMPTE timecode contained in the signal. See SMPTE time code and the references therein for a description of the SMPTE encoding schema.
- The SMPTE encoder block generates an audio signal with an encoded time code.
- The Wave File Writer block writes timeseries in the binary format chosen for the SIEMPRE recordings. The chosen binary format uses a small subset of the Broadcast Wave Format specifications.
- The DeckLink input block add support for the DeckLink family of framegrabbers, which are used in the Casa Paganini setup (see Acquisition setup at Casa Paganini) for synchronized audio/video input from High Definition (HD) videocameras.



20 May 2011





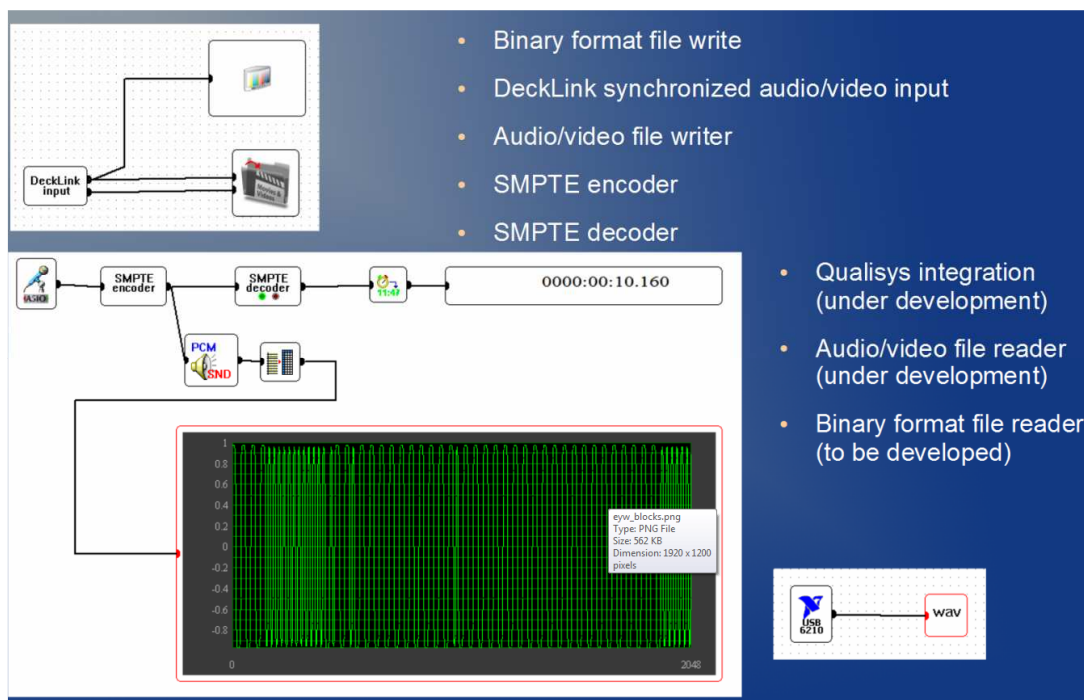


Figure 11 EyesWeb synchronisation tool.

## Recording protocol

SARC has been working to establish and validate a synchronisation protocol that can be applied to a large number of hardware devices designed to capture a wide range of heterogeneous signals. The synchronisation protocol is based on SMPTE time code <sup>4</sup>, a protocol that is used to synchronise film, video and audio material, and can be used to synchronise the data capture from a number of different hardware devices all running at different sample rates. The synchronisation protocol has been developed with a specific set of hardware devices in mind, however, it can be used to synchronise any hardware device that can record SMPTE as an audio stream; that features a synchronisation pin that can accept a TTL pulse; or that can tag a SMPTE time stamp onto each sample of data at the software level.

Using this synchronisation protocol a number of software tools have been developed to:

- Automate the synchronisation process for the multimodal data recorded during an experiment that has used the synchronisation protocol.
- Validate the synchronisation error and possible drift for a given number of modalities over the course of a trial recording.

<sup>4</sup> SMPTE time code: [http://en.wikipedia.org/wiki/SMPTE\\_time\\_code](http://en.wikipedia.org/wiki/SMPTE_time_code)

This report outlines the synchronisation protocol and gives a description of the synchronisation software that has been created to automate the synchronisation of a multimodal recording.

## Outline of the SIEMPRE Synchronisation Protocol

The SIEMPRE Synchronisation Protocol (**SSP**) is based on SMPTE time code and can be used to synchronise the data captured by a large number of multimodal hardware devices as long as the device can:

- (a) Automatically record a SMPTE signal with the data it is recording; such as the Qualysis Motion Capture system<sup>5</sup>
- (b) Record a SMPTE signal as an audio stream along with the data it is recording; such as a video camera
- (c) Record a TTL pulse along with the data it is recording; such as a Polhemus Liberty magnetic tracking system<sup>6</sup>

Alternatively, if a hardware device does not support any of these options then the data can still be synchronised using the SSP as long as the data can be streamed from the device in real-time as each sample can be tagged with a decoded SMPTE time stamp as the data is being saved, using software such as EyesWeb<sup>7</sup>. However, if this later option is used the synchronised data can only be guaranteed to be synchronised at the software level. This is because the time between a sample being sent from the hardware device and being received and stamped in the software program cannot be accounted for and must therefore be naively assumed to be instant and constant, which may not be the case.

## Recording Setup

A master SMPTE clock source must be generated for the SSP to be used to synchronise a number of multimodal data signals. The master SMPTE clock source can either be generated using a dedicated hardware unit; generated using the SMPTE output of a suitable audio card (for example the MOTU 828); or alternatively a 24 hour SMPTE stream can be recorded as an audio file and then played back using any audio file player during an experiment.

The master SMPTE clock source signal must be sent to each hardware device that can accept it as an input, either as a native input in the case of the devices that support SMPTE or as an additional input in devices that can accommodate this, such as recording SMPTE as one of the audio channels in a video camera. If a hardware device is being used that cannot accept SMPTE but can accept a TTL signal then a TTL pulse must be sent to this hardware device and sent to a computer that will also receive the master SMPTE clock source. A

---

<sup>5</sup> Qualysis Motion Capture system: <http://www.qualisys.com/>

<sup>6</sup> Polhemus Liberty magnetic tracking system: [http://www.polhemus.com/?page=Motion\\_Liberty](http://www.polhemus.com/?page=Motion_Liberty)

<sup>7</sup> EyesWeb: <http://www.infomus.org/>

software program, such as EyesWeb, must then be used to decode the SMPTE time code and tag each sample of the TTL pulse with a SMPTE time stamp as the data is saved to a file. A single TTL pulse should then be sent at the start of the recording and again at the end of the recording.

## Recording Protocol

The SSP works as follows:

- (1) Start the master SMPTE clock source if it is not already running
- (2) Start recording the data from all hardware devices
- (3) Capture current SMPTE time – this will then become TRIAL TIME ZERO
- (4) At the same time send the TTL pulse if it is needed
- (5) Run trial...
- (6) Capture current SMPTE time – this will then become TRIAL TIME END
- (7) At the same time send the TTL pulse if it is needed
- (8) Stop recording the data from all hardware devices

These 8 steps should be repeated for each trial in the experiment. After the experiment has been completed all the data saved during the trials from each hardware device should be exported to one of the accepted file formats supported by the synchronisation tool (see below for a list of the supported file formats)

## Visualisation

### Datapacks Repository & Visualizer

The *Datapacks Repository* is an online database containing shared multi-modal SIEMPRE recordings and user annotations. The idea of this repository is to contain all the multimodal recordings made during the SIEMPRE project and share the recordings together with all the associated content within the partners and at a later stage with the online community. In the following figure the block diagram of the Repository is shown (figure 8).

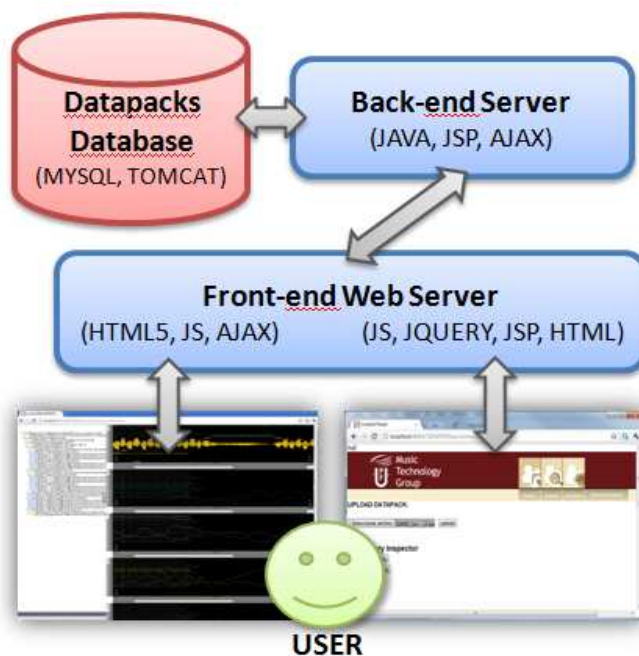


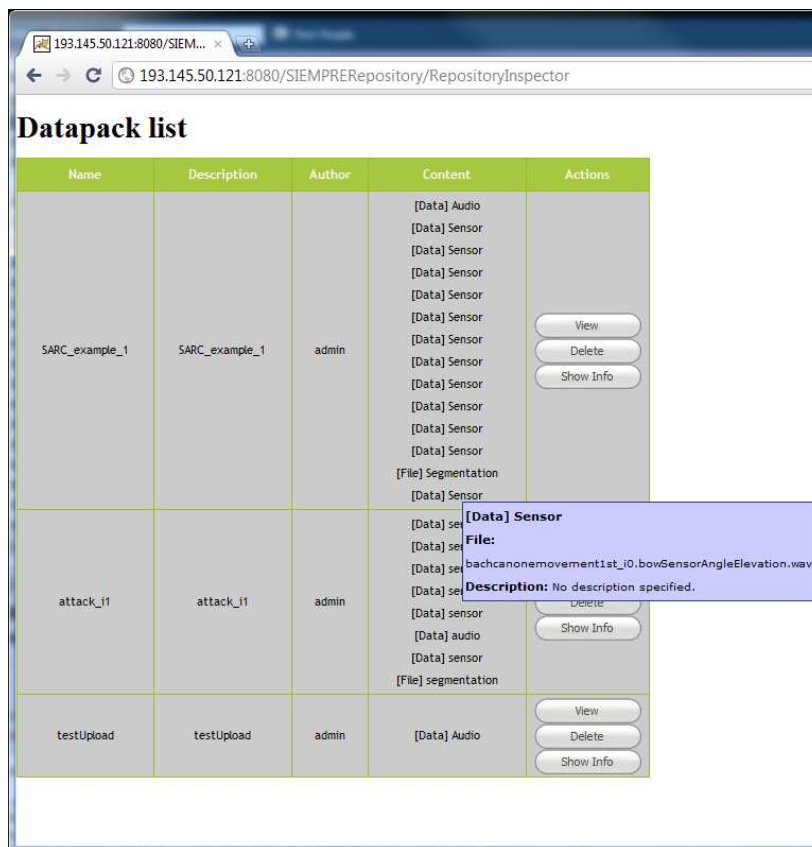
Figure 12 Block diagram of the Datapack Repository

A web-based access (figure 9) allows the user to upload a datapack file, created using the XML Experiment Creator, then the server processes its content and stores it in a structured database. Later when a user enters the web page is able to browse the available datapacks including the ones that have been uploaded by him as well as others uploaded by other users but shared with the community.



Figure 13 Uploading a datapack using the web access of the Datapacks Repository

When browsing the available datapacks, some information of its content can directly visualized in a textual way including the structure of the XML, the files within the datapack or the description associated to each file (figure 10).



The screenshot shows a web browser window with the address bar displaying '193.145.50.121:8080/SIEMPRERepository/RepositoryInspector'. The main content area is titled 'Datapack list' and contains a table with the following columns: Name, Description, Author, Content, and Actions.

Name	Description	Author	Content	Actions
SARC_example_1	SARC_example_1	admin	[Data] Audio [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [Data] Sensor [File] Segmentation [Data] Sensor	View Delete Show Info
attack_i1	attack_i1	admin	[Data] sensor [Data] sensor [Data] sensor [Data] sensor [Data] sensor [Data] sensor [Data] audio [Data] sensor [File] segmentation	View Delete Show Info
testUpload	testUpload	admin	[Data] Audio	View Delete Show Info

A tooltip is displayed over the 'attack\_i1' row, showing the following information:

- [Data] Sensor**
- File:** bachcanonmovementist\_i0.bowSensorAngleElevation.wav
- Description:** No description specified.

Figure 14 Browsing the available datapack and their content.

The most important feature of the repository and what makes it more attractive is the visualizer that allows visualizing graphically the content of a datapack using a web-browser based interface. The next figures (figures 11 & 12) show examples of the visualization of data files in the datapack. The online GUI allows the user to select which data files want to view at each moment, adjust the level of zoom, play the audio and video tracks, define multiple panels and select which data is visualized in each panel, select the data files to be visualized by using the XML of the experiment represented as a tree (like in the XML Experiment Creator), highlight a certain data signal to be visualized over the rest, show the notes segmentation over the audio file, etc...

In a later stage, the Visualizer will allow the user to make annotations online and also to select a segment of a datapack and download it to analyze it using custom tools.



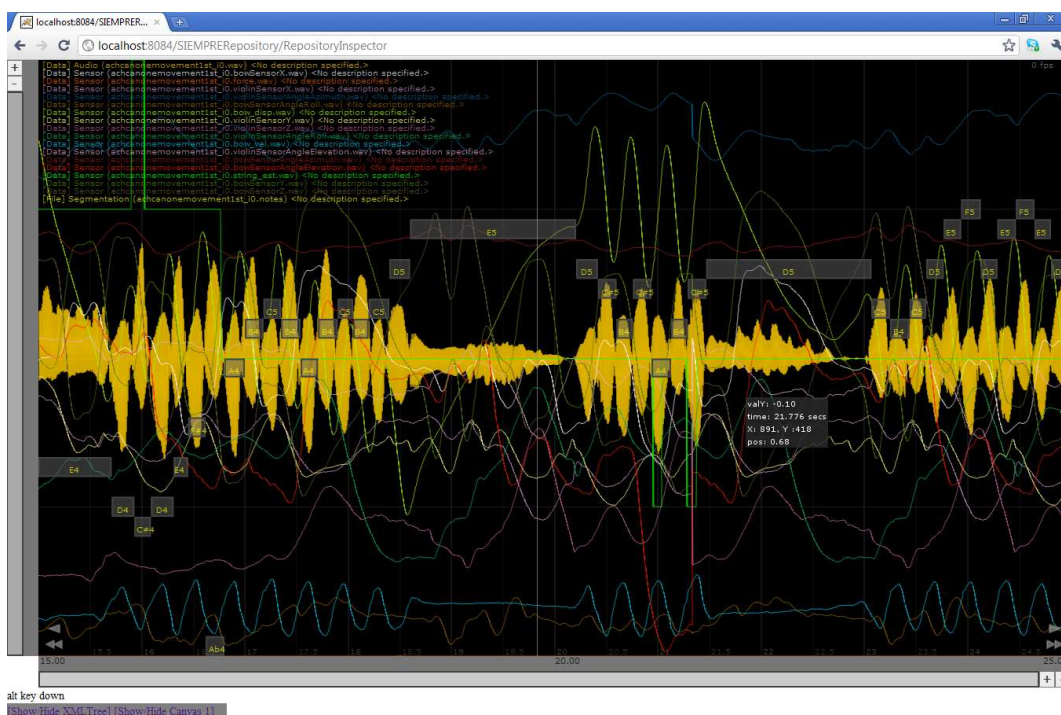


Figure 15 Repository visualisation tool showing a complete experiment in a single panel.

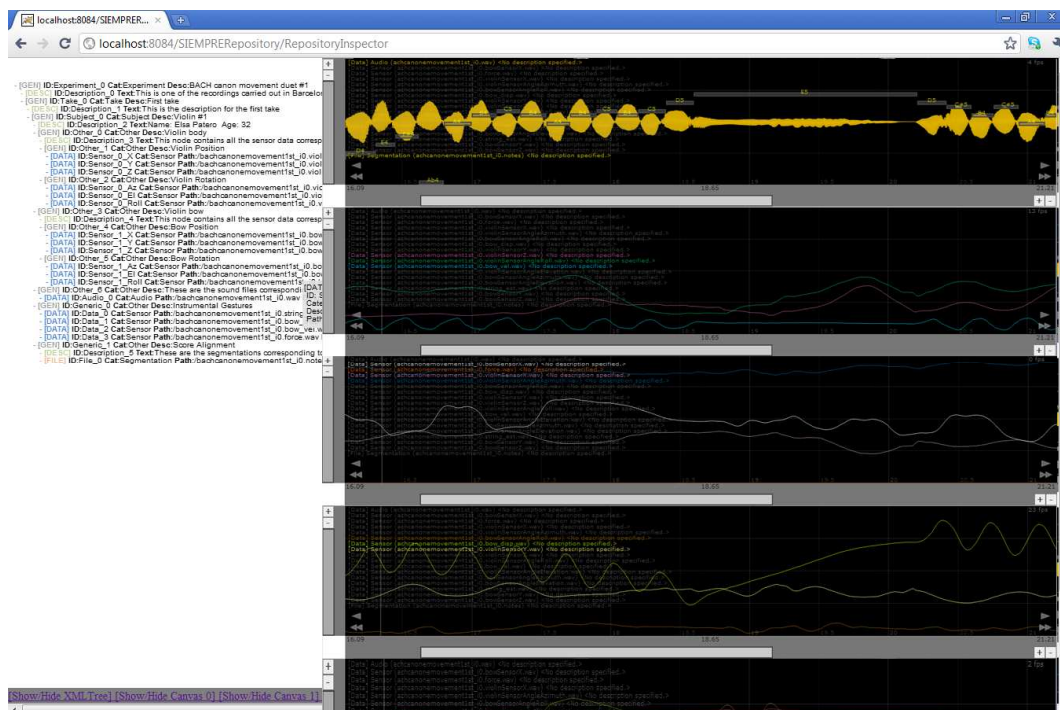


Figure 16 Repository visualisation tool showing a multi-modal recording experiment in a tree view and multiple user defined panels for data signal visualisation.



## APPENDIX I - TUTORIAL

The present document describes in detail the required technical steps to undertake experiments as defined by the SIEMPRE Consortium in Deliverable 3.1. It assumes that the experiments are taking place at either Casa Paganini or the Sonic Arts Research Centre. Meeting this requirement, means that all hardware has been placed correctly as defined by the SIEMPRE Consortium for the project. The nature and specific variables of each experiment can vary as defined in Deliverables 1.1 and 2.1 yet the technical setup should remain fixed as agreed by all partners. This document provides a step-by-step guide and checkbox list to ensure any measurement meets the requirements specified by the team.

This is a 'living' document and will be updated as required by issues not envisaged at the time and will be correctly changed and documented as new needs arise.

For any additional information please refer to Deliverables 1.1, 2.1 and 3.1.

### SIEMPRE Synchronisation Protocol

The SIEMPRE Synchronisation Protocol (SSP) aims at consistently recording and synchronising multi-modal signals acquired from a vast range of Hardware/Software modules as defined by the SIEMPRE Consortium.

Deliverable 3.1 describes the specific hardware and software modules required for all experimental sessions. Different types of signals are recorded using the defined Hardware/ Software tools to ensure that we have raw signals in their native formats. The SSP implements a synchronisation protocol that makes sure all signals recorded by various systems are correctly synchronised and can be used for analysis during the SIEMPRE project.

After recording, all signals are post processed and normalised so that we can compare and run analysis algorithms on the data. This allows for the transparent exchange of data sets between the SIEMPRE partners as well as to disseminate these data sets to the research community at large, thus validating all the findings of the project.

The SSP works as follows:

- (1) Start the master SMPTE clock source if it is not already running
- (2) Start recording the data from all hardware devices
- (3) Capture current SMPTE time – this will then become TRIAL TIME ZERO
- (4) At the same time send the TTL pulse if it is needed
- (5) Run trial...
- (6) Capture current SMPTE time – this will then become TRIAL TIME END
- (7) At the same time send the TTL pulse if it is needed
- (8) Stop recording the data from all hardware devices

These 8 steps should be repeated for each trial in the experiment. After the experiment has been completed all the data saved during the trials from each hardware device should be exported to one of the accepted file formats supported by the synchronisation tool.