gVARVI: graphic heart rate Variability Analysis in response to audioVisual stImuli

User manual

Nicolás Vila

with colaboration of

Leandro Rodrígez Liñares
Pedro Cuesta Morales
Xosé Antón Vila Sobrino
María José Lado Touriño
Arturo José Méndez Penín

October 8, 2015

Contents

1 Introduction

gVARVI is an open source software for heart rate data acquisition in response to audiovisual stimuli. It has a friendly graphic interface that is very simple to use. It also allows save acquisition configurations and automatically search for nearby heart rate sensors.

2 Basic features

- Heart rate data acquisition in response to audiovisual stimuli: That's the main feature. While application shows any stimuli to the user, it retrieves heart rate data from conventional sensors. The supported sensors by now are *Polar WearLink+* band [1] and all of heart rate sensors that use ANT+ protocol. Specifically gVARVI was tested with *Garmin Premium Soft Strap Heart Rate Monitor* band [2].

- Acquisition configuration: over the "activities" concept (you can see that more in detail at section 3) the user can create an acquisition by setting the stimuli type and configuring associated parameters to that concrete stimuli. For example, if the stimuli is a set of pictures, picture duration can be setted. These configurations can be saved (for future executions of gVARVI), edited or deleted.

- Results plot: acquisition results are saved in a text file, but the user can also watch a graphic representation of heart rate data. Results can be opened with gHRV (a software developed by *Mile Group* to perform heart rate variablity analysis).

- Automatic device search: the user can perform an automatic scan with no specific data knowledge (device name, physic address, etc.).

- Device testing: in order to verify that the device is working properly, the device can be tested. In this way, application show heart rate data (beats per minute and RR-distance) in real time to ensure that there is any issue.

- Complete debugging system: system errors can be managed in four different ways. gVARVI send log messages to graphic interface, standard output (terminal), log file and even over the internet. That allows to perform a remote debug.

- Import/export activities: to save activity configuration and associated media (pictures, audio and/or video) to a single file, share it, and open this activity file in other computer. Very useful to work groups, to avoid configuring the same activity for each computer.

[1] `http://www.polar.com/en/products/accessories/Polar_WearLink_transmitter_with_Bluetooth`

[2] `https://buy.garmin.com/en-US/US/shop-by-accessories/fitness-sensors/soft-strap-premium-heart-rate-monitor/prod15490.html`

3 Activities

Activities are the gVARVI way to configure acquisitions. Each activity is composed of several tags (also know as episodes), and each tag represent an indivual stimuli or a group of them. For example, in a video presentation, each tag represents an invidual video because it's large enough so hear rate varies significantly between to videos. On the other hand, in a picture presentation, each tag represents a group of pictures (specifically a folder that contains a group of pictures).

All activities (and all activity tags) have an identifying name. During activity playing, it can be aborted pressing ESC key.

There are five activity types, based on stimuli to be played:

3.1 Photo presentations

In this activity type, pictures are the main stimuli. Each tag represents a fodler that contains a group of pictures. Formats supported are JPEG, PNG, BMP, TIF, XPM and PCX. Also it is possible to associate sounds to each tag. If every associated sound finished before tag pictures ends, sounds will be played again. Tag playing can be randomly or in a specific order configured previously.

3.2 Video presentations

In this activity typ, videos are the main stimuli. Each tag represents an individual video. Format supported is MPG. Tags acn be played randomly or in a specific order.

3.3 Sound presentations

In this activity type, sons are the main stimuli. Each tag represents an individual sound. Formats supported are MP3 and WAV. Tags can be played randomly or in a specific order. Also it is possible to associate images to each sound tag. Associated images will be played simultaneously to sound, in a random order or in a user-defined order. Duration of each image is calcuted by gVARVI automatically. If there's no associated images, screen turns black during sound playing to avoid any user distraction.

3.4 Activity composed of associated key tags

In this activity type there's no audiovisual stimuli. Simply application shows a custom text to the user. Each tag has to be configured with an associated key so tag playing starts with first tag and the user can change from tag A to tag B by pressing associated key of tag B. In this activity type user can change to a previously played tag. For example, this sequence is allowed: $TAG1 \rightarrow TAG2 \rightarrow TAG3 \rightarrow TAG1$. Acquisition will finish when user press ENTER key.

The main purpose of this activity type is mark off external events with variable duration. For example, a physical exercise can has one tag called "30 abs" and other tag called "10 pushups". In this way, when user do 30 abs will press asociated key of next tag (10 pushups), and when user do 10 pushups it can press associated key to "30 abs" tag again. The proccess can be repeated as often as user wants.

3.5 Manual activity

In this activity type there's no audivisual stimuli too. Again, application shows a custom text to the user. However each tag can finish with key pressing (specifically space bar key) or at a specific time.

Main purpose of this activity is mark off a mixture of events with fixed duration and events with variable duration, so activity can contains tags that will finish at a fixed time and tags that will finish when user press space bar key. Returning to the example of physical exercise, it can have following tags: "30 abs", "20 second relaxation", "10 pushups", "20 second relaxation". In that way, the user can manually finish abs and pushups tags, while relaxation tags finishes at a fixed time (in this case, 20 seconds).

4 Installation

4.1 Installation manual

gVARVI app requires third-party software instalation:

- Python interpreter: on GNU/Linux systems, Python interpreter is installed by default, so its installation it's not necessary. To install the interpreter on Windows systems, there are binaries available at Python webpage [3] for both 64 bits systems and 32 bits systems. Installation is fully guided. It must be installed Python2 instead of Python3 because wxPython library (mentioned below) only works with Python3.

- **wxPython**: wxPython installation proccess is the same as Python interpreter. Windows binaries are available at wxPython webpage [4]. Installation in GNU/Linux systems is easier. There are wxPython packages available in repositories of main Linux distributions. In particular, for Debian based distributions, there are two different versions of wxPython. To install any of them, you only need to type the following keywords in Linux console:

```
# Version 2.8
$ sudo apt install python-wxgtk2.8

# Version 3.0
$ sudo apt install python-wxgtk3.0
```

- **Pybluez**: this library is available at Python packages repository or PyPI (*Python Package Index*). To install any of theese packages from console it's necessary to have installed *pip* module. From Python 2.7.9 version *pip* module is installed by default instalar directamente unha aplicación deste repositorio é necesario ter instalado o modulo *pip*. So open a console (Windows or Linux) and type the following:

[3] `https://www.python.org/downloads/windows/`
[4] `http://wxpython.org/download.php`

```
# Windows
$ pip install PyBluez
# GNU/Linux
$ sudo pip install PyBluez
```

This library has some requirements, explained in detail at *README* document on Github repository webpage [5].

- **PyGame**: To install PyGame we can download it from its webpage, at "downloads" section [6]. There are binaries to Windows systems, with easy step-to-step installation. For Linux systems, there are packages available on well know distributions. For Debian based distributions, we can install Pygame in this way:

  ```
  $ sudo apt install python-pygame
  ```

  NOTE: there are no official binaries for 64 bits Windows systems so in that case we need unofficial binaries, available on California University web server [7]. Theese binaries (*.whl* extension) can be installed using *pip*.

- **Matplotlib**: also available in PyPI repository. In this way, the command to install it could be:

  ```
  # Windows
  $ pip install matplotlib
  # GNU/Linux
  $ sudo pip install matplotlib
  ```

- **Pyusb**: it's a requeriment of *pyant* library. It's available at PyPI, but gVARVi needs the last version (that isn't available at PyPI), so we have to download it from Sourceforge [8] and install it with *pip*:

  ```
  # Windows
  $ pip install pyusb-1.0.0b2.zip
  # GNU/Linux
  $ sudo pip install pyusb-1.0.0b2.zip
  ```

- **Pyserial** and **msgpack-python**: in the same way that Pyusb, these libraries are *pyant* requeriments that have to be installed manually. There are available at PyPI:

  ```
  # Windows
  $ pip install pyserial msgpack-python
  # GNU/Linux
  $ sudo pip install pyserial msgpack-python
  ```

[5] https://github.com/karulis/pybluez
[6] http://www.pygame.org/download.shtml
[7] http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame
[8] http://sourceforge.net/projects/pyusb/files/latest/download

When all requeriments are installed, you need to download gVARVI source code at GitHub [9], unpack compressed file and place project folder anywhere. To create a launcher that eases the process of start the application, you can do this:

- In Windows systems, you can create a shortcut. The process is explained in ZonasSystem tutorial [10]. You only have to change this command

  ```
  shutdown /s /t00
  ```

  for this command

  ```
  python <Path to gVarvi.py file inside the directory>
  ```

  This shortcut can be placed anywhere, for example in the desktop.

- In GNU/Linux the way is similar. You only have to create a file with *.desktop* extension and place it at *$HOME/.local/share/applications*. File structure can be something like this:

  ```
  [Desktop Entry]
  Encoding=UTF-8
  Version=0.1
  Name=gVARVI
  Exec=python <Path to gVarvi.py file inside the directory>
  Terminal=false
  Type=Application
  ```

  Once you create this file, operative system detects it as an installed application. After that, you can already create a desktop shortcut.

4.2 Available binaries

To ease installation process, there is a binary for Debian-based Linux distributions (*.deb* file). It's available at GitHub repository.

5 Operation guide

User interface has the same structure in Windows and Linux. Simply application will adapt to each operative system look and feel. For this reason, altough this screenshots have been taken in Ubuntu OS (whit Gtk toolkit), they are extrapolated to any system.

5.1 Main window

As you can see at figure 1, the main window of gVARVI is divided into four sections.

- Activities panel (top left) shows activities saved in the system, indicating for each an unique identifier, the name, and activity type. We also have four

[9] `https://github.com/milegroup/gVarvi/`

[10] `http://www.zonasystem.com/2010/12/crear-acceso-directo-para-apagar-equipo.html`

buttons. *Add* button allows us to create a new activity; with *Edit* button, we can change configuration parameters for selected activity; with *Remove* button, we can delete the activity; and *Export* button allows us to save selected activity to a single file.

- Devices panel (top right) shows found devices after a scan operation. We can see the name and physical address (for Bluetooth devices) for each device. There are two button below. By pressing *Scan* button, we can search for nearby devices and show their information on the panel; *Test* button is used to check device performance, showing us real time data retrieved from de device.
- Global options (bottom left) are considered as "mandatory" for every application. *Quit* option to close application; *About* option to show general information about application; and *Preferences* options to manage application configuration.
- Acquisition panel (bottom right) shows us which activity and device are selected and allows us to begin the acquisition.

5.2 Creating activites

To create activitities we only have to press *Add* button at the main window and select an activity type at the shown dialog (figure 2). In this dialog there are only pictures. However, if we leave the mouse above any picture, we can obtain information of each

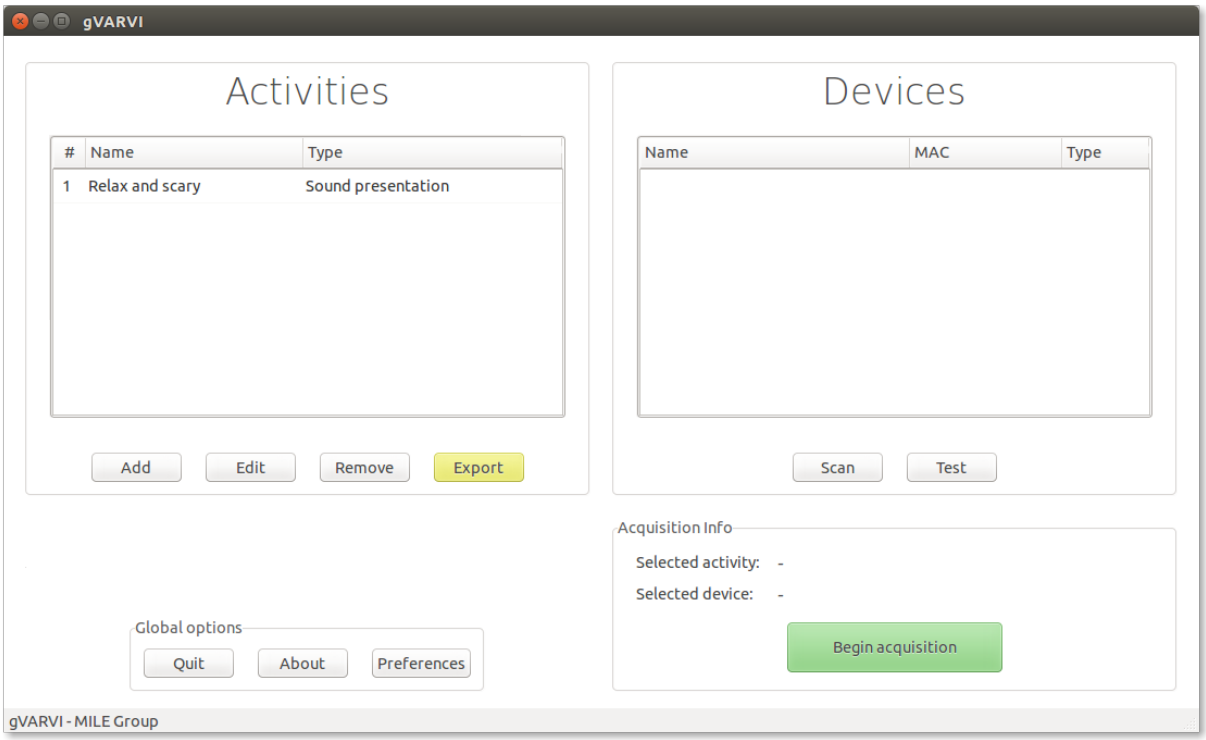

Figure 1: Main window.

activity type. After select activity type, we must fill configuration fields for that activity.

5.2.1 Picture presentation activity

To create an activity we have to fill the following parameters (figure 3):

- *Name*: activity name.
- *Gap (seconds)*: duration of each picture.
- *Random*: this option will be selected if randomly tags playback is desired.
- *Tags*: tags of the activity. These tags can be added, removed and modified. Also tags playback order can be modified (the order will only be taken into account if *Random* option is unchecked).

For each tag the following parameters are mandatory (figura 4):

- *Name*: tag name.
- *Path*: absolute path of the folder where tag images are located.
- *Associated sound*: this option will be selected if we want to accompany pictures with sounds.
- *Sounds*: associated sounds of the tag. They can be added, removed and modified. Playback order changing is also allowed. If *Associated sound* option is unchecked, sounds will not be played.

5.2.2 Sound presentation activity

To create a sound presentation activity we have to fill the following parameters (figure 5):

- *Name*: activity name.

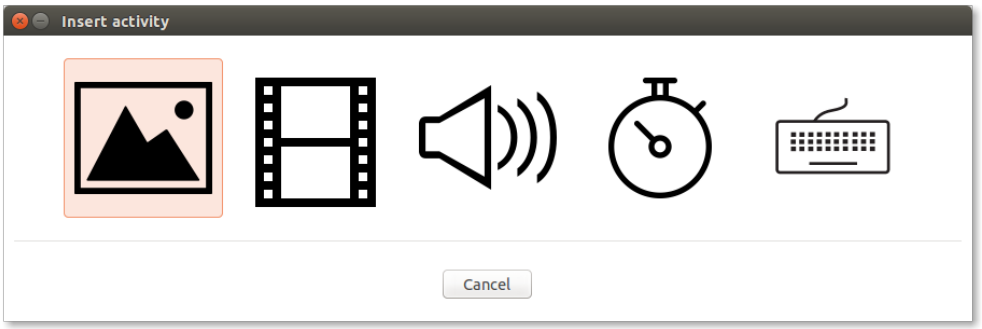

Figure 2: Add activity dialog.

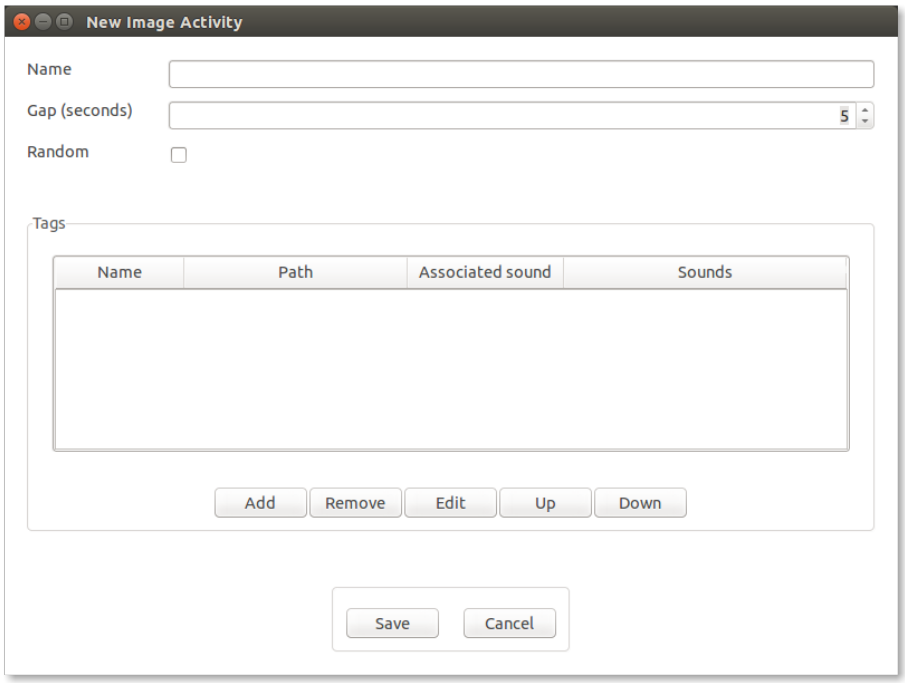

Figure 3: Window for managing a picture presentation activity.

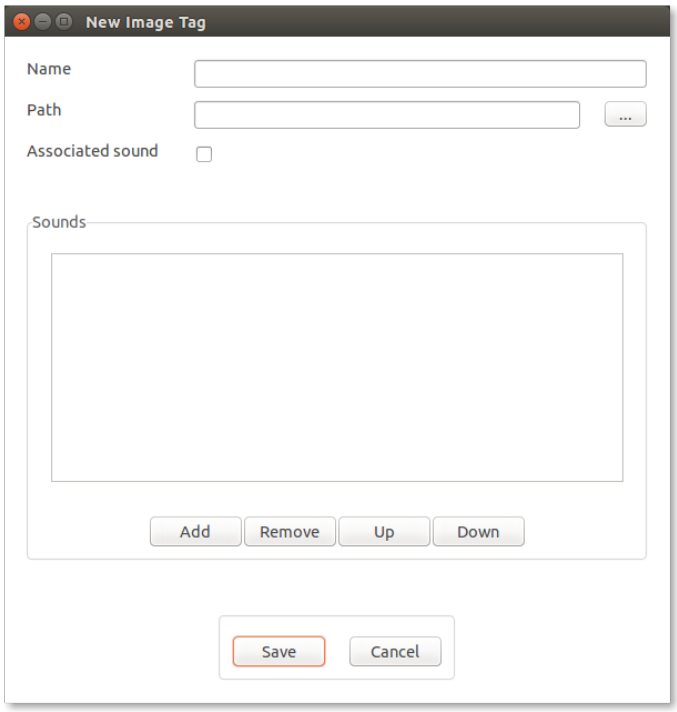

Figure 4: Window for add a new picture presentation tag.

- *Random*: this option will be selected if randomly tags playback is desired.
- *Tags*: activity tags. As in the other activities, tags can be added, removed and modified, and also playback order can be chenged.

For each tag the following parameters are mandatory (figura 6):

- *Name*: tag name.
- *Path*: path to tag sound file.
- *Associated image*: this option will be selected if we want to accompany sound with images.
- *Randomize images*: we will check this if we want to play associated images in a random way.
- *Images*: associated images for the tag. They can be added, removed and modified, and their order can be changed. If option *Associated image* is unchecked, these images will not be taken into account. In the same way, if option *Randomize images* is checked, image order will not be taken into account.

5.2.3 Video presentation activity

To add an activity of this type, mandatory parameters are (figure 7):

- *Name*: activity name.
- *Random*: this option shows if tags will be played in a random way.
- *Tags*: activity tags. Playback order will be taken into account if option *Random* is unchecked.

For each activity tag is mandatory to fill following parameters(figure 8):

- *Name*: tag name.
- *Path*: absolute path to video file.

5.2.4 Associated key activity

To add an activity, required information is (figure 9):

- *Name*: activity name.
- *Tags*: activity tags. In this activity type order is not important. Only will be taken into account the first tag of the list. After that, the user has the control and decides at playing time the order of the tags by pressing associated keys of each tag.

For each activity tag is necessary the following parameters (figure 10):

- *Name*: tag name.
- *Screen text*: text that will be shown on the screen.
- *Key*: associated key.

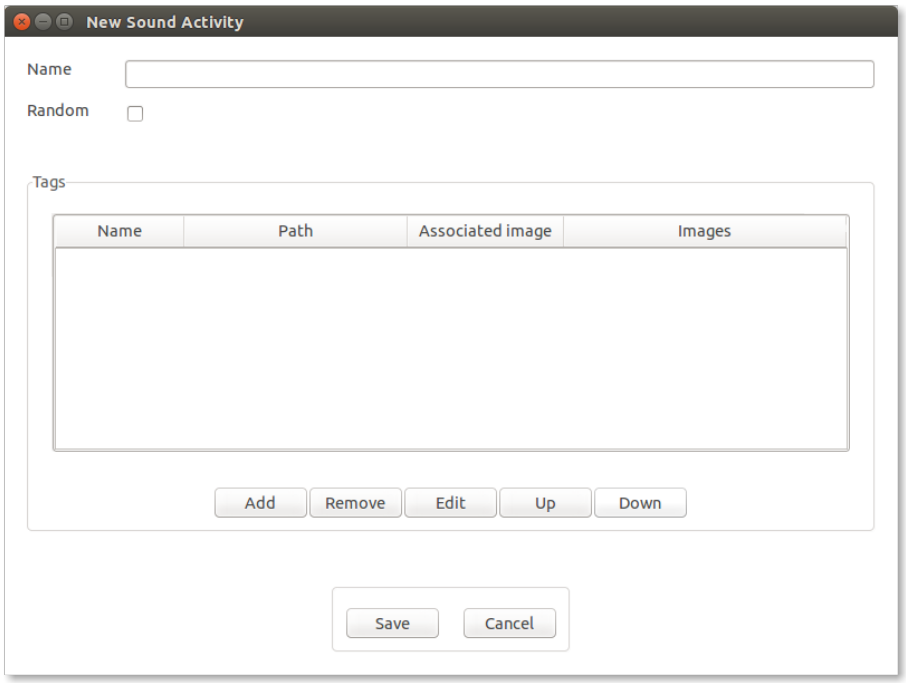

Figure 5: Window to add a sound presentation activity.

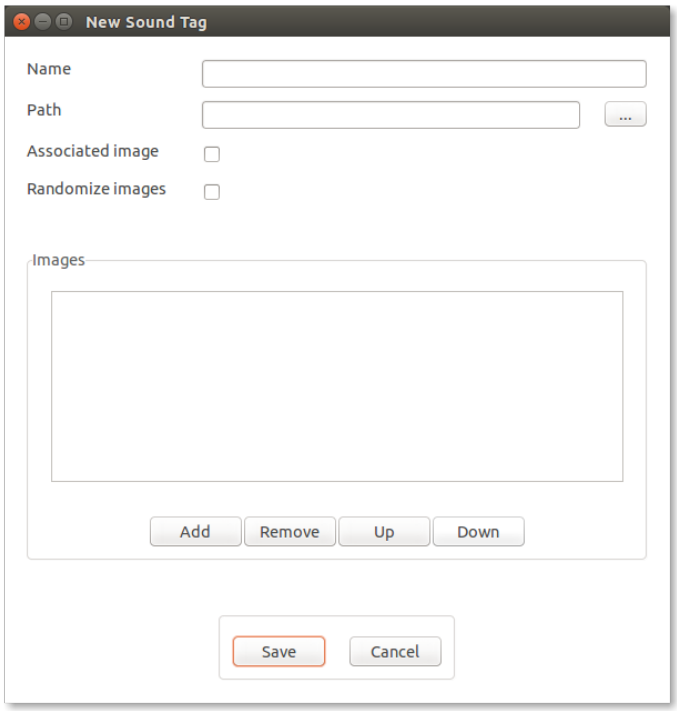

Figure 6: Window to add a sound presentation tag.

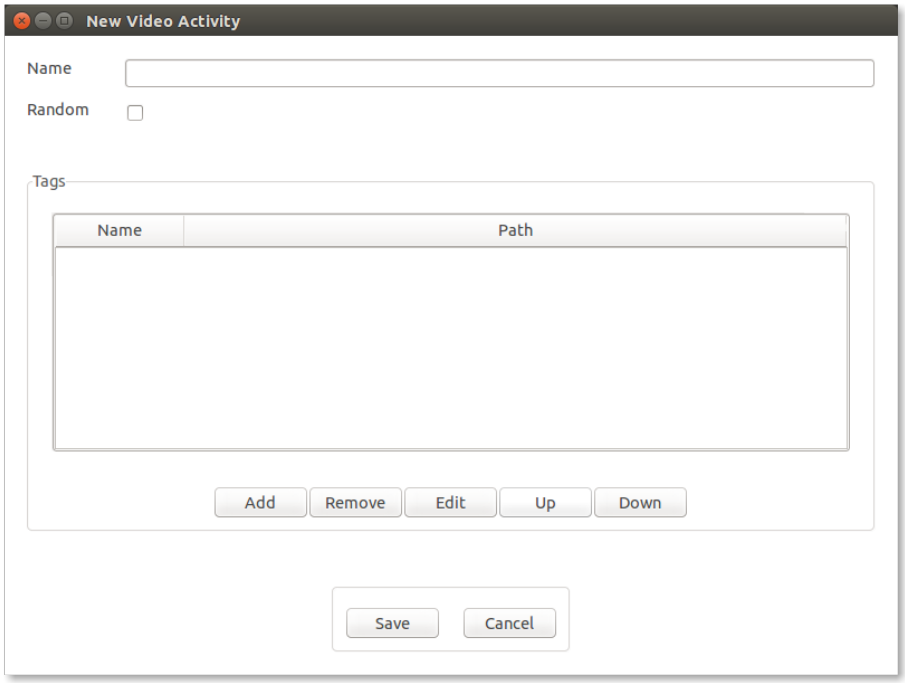

Figure 7: Window to add a video presentation activity.

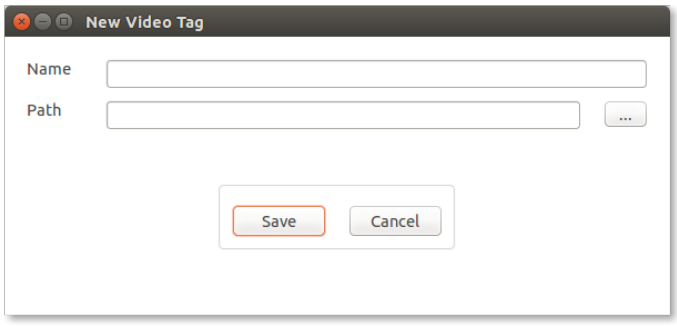

Figure 8: Window to add a video presentation tag.

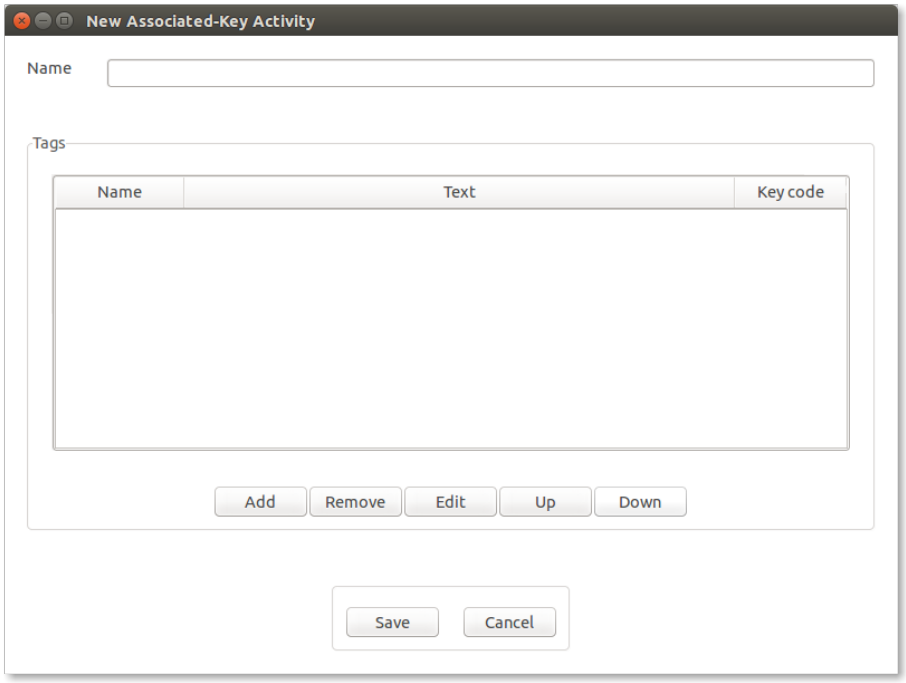

Figure 9: Window to add a set of actions controlled by the keyboard.

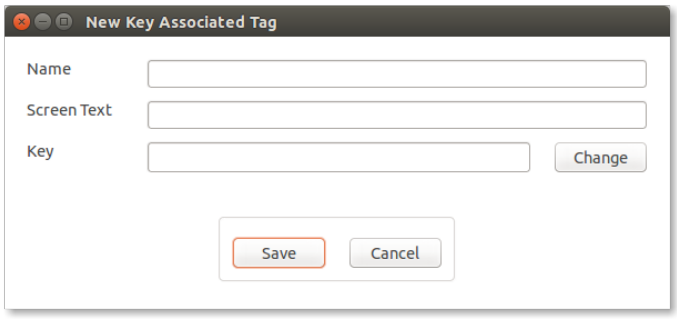

Figure 10: Window to add a tag with an associate key.

5.2.5 Manual activity

To add a manual activity, following attributes are mandatory (figure 11):

- *Name*: activity name.
- *Tags*: activity tags. These tags will be played according to the list order.

For each tag, user must fill these parameters (figure 12):

- *Name*: tag name.
- *Screen text*: text that will be shown in the screen.
- *Finish type*: parameter indicating how tag will end. *Timed* option makes the tag end in a predetermined time. *Key (SPACE BAR)* option makes the tag end when user has pressed space bar key.
- *Time*: duration of the tag in seconds (if *Timed* option is selected in the previous parameter.

5.3 Editing and removing activities

Activity and tag modification is done in the same way that the creation of them. To editing an activity user must select it in the main window and press *Edit* button. To edit a tag, it must be selected in tag panel of the activity. Then, *Edit* button must be pressed. The window shown is the same as in the creation process, but parameter data are filled with actual configuration. (figures 13 and 14).

To remove an activity the process is even easier. User only must select an activity in the main window and press *Remove button*. Confirm dialog will be shown.

5.4 Devices

Device discover process is so easy too. By pressing *Scan* button, application starts the search. When it finishes, discovered devices are shown in devices panel of the main window.

If we want to test some device, we have to select it and press *Test* button. A pop-up dialog will appear and it will show real time data retrieved from the device (figure 15).

5.5 Application preferences

We have to press *Config* to show configuration panel (figure 16).

- *Language*: language of the application. Language changing requires reboot the application.
- *Check for updates on start*: if this is checked, gVARVI will search for new available versions when it starts.

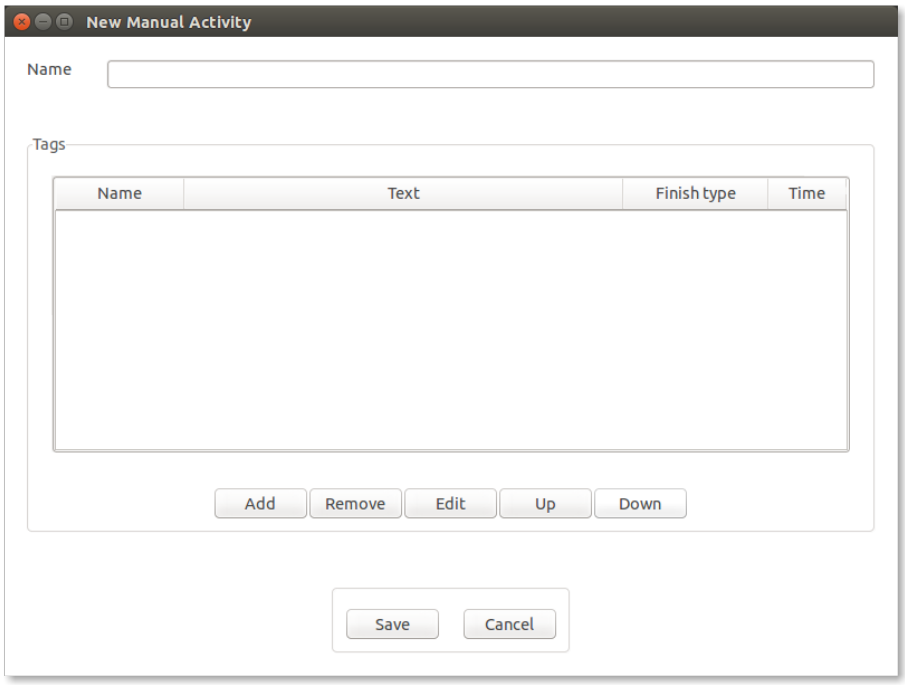

Figure 11: Window to add a manual activity.

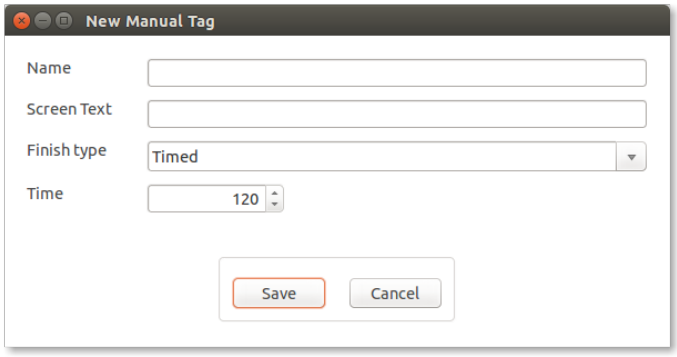

Figure 12: Window to add a manual activity tag.

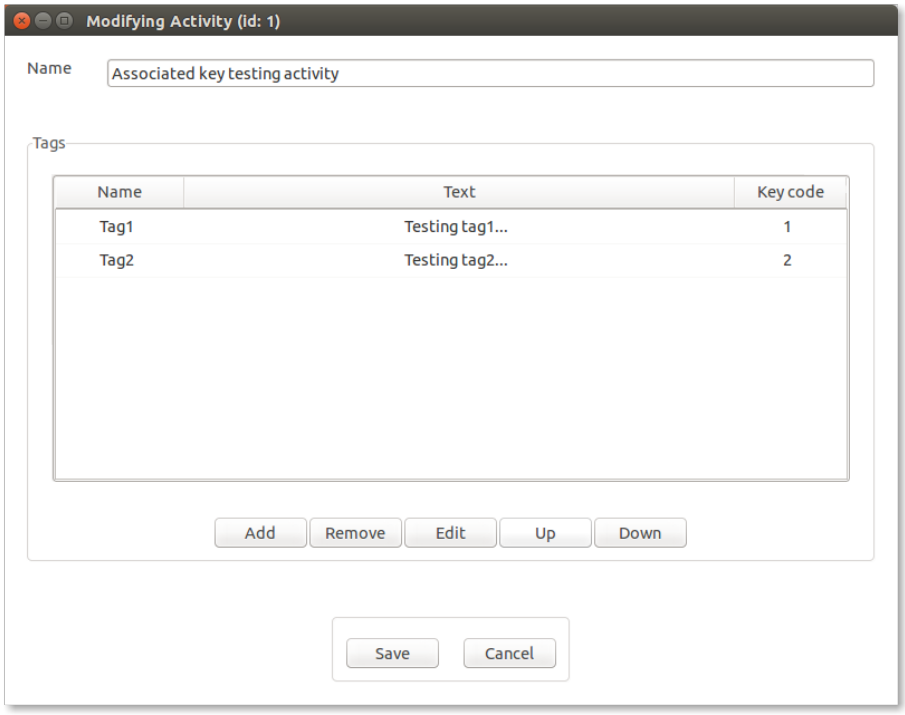

Figure 13: Window to modify an activity.

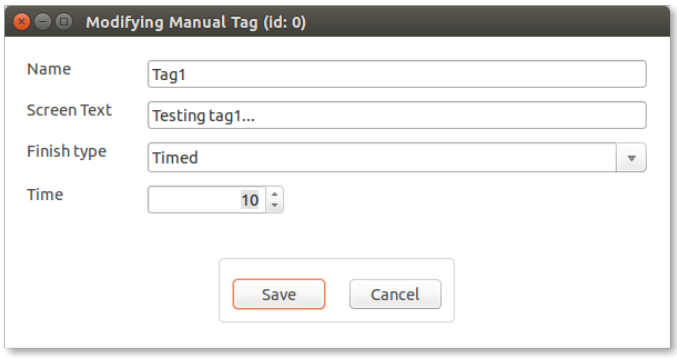

Figure 14: Window to modify a tag.

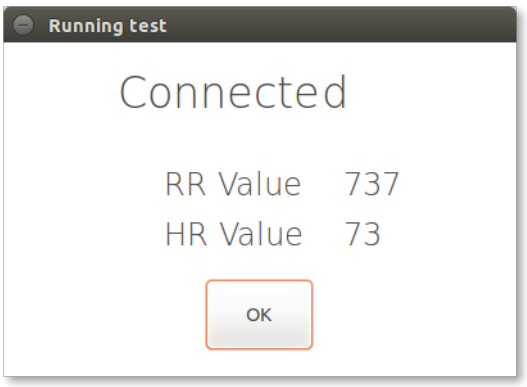

Figure 15: Window to test acquisition devices.

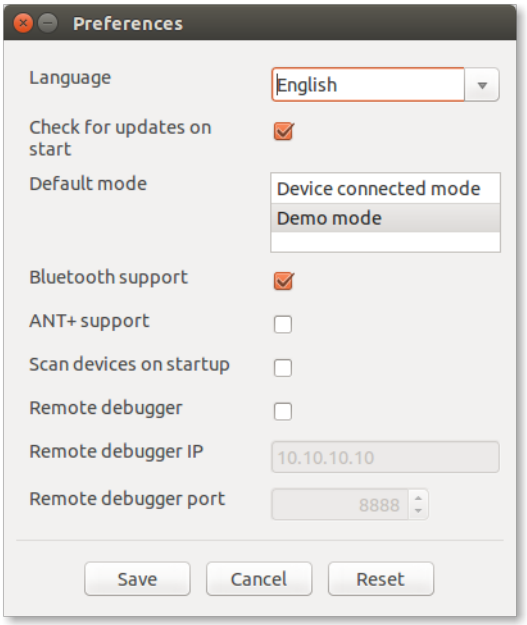

Figure 16: gVARVI preferences panel.

- *Default mode*: acquisition mode. If *Device connected mode* is selected application will connect to selected acquisition device and will acquire real data. On the other hand, if option *Demo mode* is selected, acquisition data will be generated randomly.
- *Bluetooth support*: if this option is selected, gVARVI will search for Bluetooth devices.
- *ANT+ support*: if this option is selected, gVARVI will search for ANT+ devices.
- *Scan devices on startup*: we can select this option to searching devices on application boot.
- *Remote debugger*: if this option is selected, remote debugging (explained on section 5.6) will be activated.
- *Remote debugger IP*: IP of remote host where debugger is located.
- *Remote debugger port*: port where remote debugger is listening.

5.6 Application debugging

gVARVI has several debugging systems:

- We can execute gVARVI from a terminal and we will see debug that on it.
- We have log file (gVarvi.log) at *C:\\Documents and Settings\<User name>\.gvarvi* (in Windows) or *$HOME/.gvarvi* (in Linux). In this file information is more detailed, and it contain time information, as well as which line of which module print each log message.
- In title bar of gVARVI we have *Advanced →Toggle debug window* opton. By pressing it, debugging data will be shown in a new window, so we avoid to execute application from a terminal (figure 17).
- By last, we have the ability to do a remote debugging. This is so useful beacuse activities are played in fullscreen mode so user can not see the debug window. To do this remote debugger script *RemoteDebugger.py* has to be downloaded and executed in a remote host. It is available in GitHub repository. This script allow us to configure listening port and show us our IP. These parameters has to be filled in gVARVI configuration, in fields *Remote debugger IP* and *Remote debugger port*. Once configured these parameters, in the remote host will appear all debug messages.

5.7 Acquisition

To start an acquisition, user has to press *Begin acquisition* button on the main window. A window will appear asking for the name of the acquisition. File extension will be completed by gVARVI, so user does not need to fill it. After accept, activity will be played. At the same time, application will start to acquire heart rate data

from the device. Once activity has finished, results file will be generated (extension .tag.txt and extension .rr.txt) and a dialog popup will be shown. Available options are:

- *Do nothing*: non fai nada.
- *Open in gHRV*: open results in gHRV tool to analyze them.
- *Plot results*: plot heart rate data (figure 18).
- *Open tag file*: open tag file with the default text file reader of the operative system.
- *Open rr file*: open rr file with the default text file reader of the operative system.

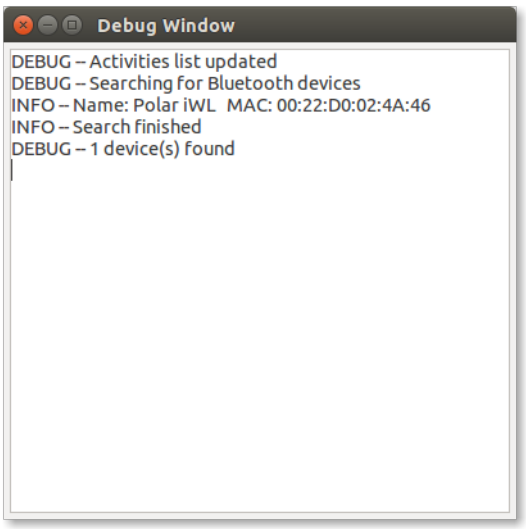

Figure 17: Debug window of gVARVI.

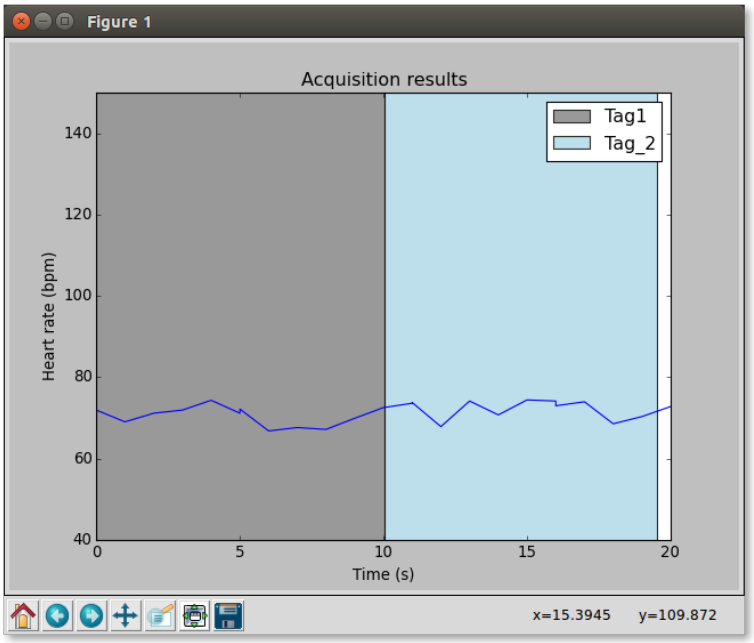

Figure 18: Plot of the results.