VRPN Physiological Software
University of North Carolina at Chapel Hill
Michael J. Meehan
Sharif Razzaque

Lasted updated August 15th, 2003

Introduction

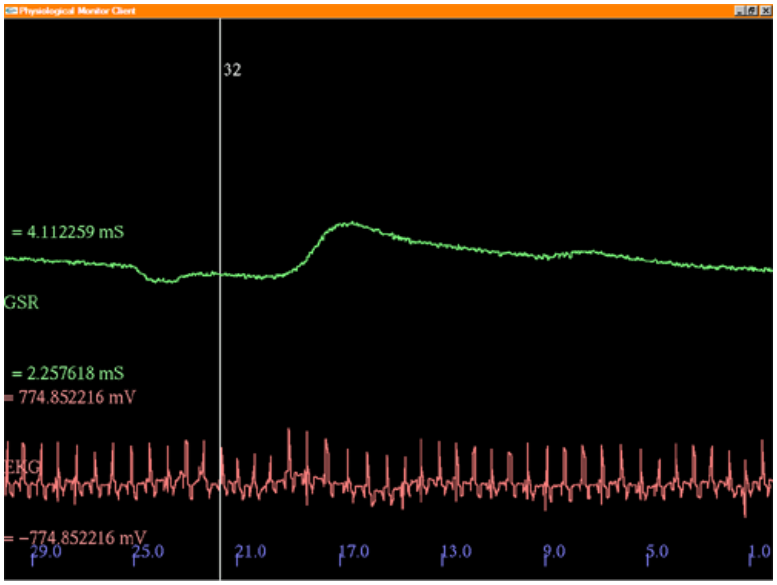

Figure 1 – A screen shot of the client program's graph window showing 30 seconds of data. GSR is shown in green and EKG in red. The white vertical line is an annotation by the experimenter (a keyboard event).

This software records, graphs and analyses physiological data from a human subject. It can record EKG, skin conductance and skin temperature. The data graphed in real time and can be replayed from the recording. It allows allow the experimenter to annotate the data with events (i.e. the experiment begins, the subject sneezes, etc.) during the recording. These events are stored and graphed with the physiological data. This software requires the Thought Technologies LTD ProComp Plus physiological monitor and the associated sensors.

For the purpose of this documentation, there are two types of users. The subject, who is wearing the sensors and who's physiological signals are being recorded; and the operator (or experimenter), who is monitoring the physiological signals and annotating them.

The software has two executable programs, a client and server. Both must be running simultaneously to record or graph the physiological data. The server communicates with the ProComp Plus box and timestamps the data coming from it. It also records the operator's key presses to mark events in the physiological data. The server forwards the data to the client, which records and graphs the data. The client and server can run on the same or different computers – they communicate over the network.

Requirements

An Intel PC running Windows 98, NT, 2000, or XP. The server requires a 400MHz Pentium II or better. The client requires a recent graphics card (low-end AGP Geforce or better). To run the client and server on the same machine requires a dual processor (each being 400MHz or better) and a good graphics card.

Installation & Configuration

This physiological software is packaged as a .zip file. You will need Winzip to uncompress the contents. First, install Microsoft Visual C++ 6. If you don't install it, you will need the file MSVC.dll. It is included in the .zip file. This file may not work with all versions of windows, so it's better to install Visual C++ if possible.

The data and configuration are stored in files. The data files end with a ",log" extension, and the configuration files end with a '.cfg' extension.
Sometimes, Windows hides the file extension. To see the extension, in Windows NT, from the folder toolbar, select the "view" pull-down menu and "options" from that. Then select the "view" tab, and make sure "hide extension for known file types" is NOT checked.

Create two directories, one for the physiological software, and another for the physiological data.

Procomp.cfg syntax

> Any line that starts with # is a comment and is ignored
> Each line which is not a comment must contain a key and value.
> They key and value must be separated by a tab or space a combination of spaces and tabs. There can only be one key and value per line. After the key and value, the rest of the line is ignored. See the comments within the procomp.cfg file for more info.

The server only supports the EKG, skin conductance
and skin temp sensors.

Usage

1) Modify the client's config file (proComp.cfg) to connect to the machine with the server (replace 'nickel-cs')

2) Plug in the following sensors into the following channels:
channel A - EKG
channel D - skin conductance
channel E - skin temp

3) Run the timing to see if you machine is fast enough. You should get at least 32 samples/sec or you will loss data. The server also monitors its sample rate while running, so it will warn you when it runs to slowly.

First start the server program, then the client program.

Server is server_vrpn_procomp.exe
Client is client_vrpn_procomp.exe

Note: the server has a real mode and test mode. The test mode allows you to run the program without being connected to a proCompPlus box. The server generates sinusoid waveforms instead. The test mode allows you to verify that the network and graphing client are working, but does not allows us to verify that the server is running on a fast enough machine.

After you run the client, the graphing window will appear. After a while , the data will appear and it should look something like Figure 1. With the graphing window selected (mouse click inside it), press q,w and e to center the graphs in case the data is above or below the screen. **If the three EKG electrodes are in the proper position, you should (after you run the programs) see the heartbeats in the EKG as upward pointing spikes (see Figure 1). If the spikes point downwards, the positive and negative EKG electrodes are reversed. This can be corrected during the analysis, but it makes things very confusing later!**

Keyboard events must be entered while focus in on server window
Graphing commands must be while focus in on client window

There are three modes in which to run the client program:

1) Have it connect to the vrpn_ProCompPlus server, and simultaneously record and graph the physiological data from the proCompPlus box. The user/volunteer/subject/participant (whatever you call them) must be wearing the sensors and the machine must be turned on and the server_vrpn_procomp.exe program must be running. The data is recorded to a log file. You enter the directory in which this file will be created in the configuration file proComp.cfg
2) Have it play back the graph of a single log file. The log file which you wish to replay should be set in the proComp.cfg config file.
3) Have it do the analysis of many log files, in batch mode. You can run this overnight if needed. The list of log files to analysis should be set in proComp.cfg.

This software Compiles on VC 6.0 with service pack 3, on Windows 98
NT 4 and Windows 2000, and requires a precompiled vrpn.lib for windows (version 6.XX). It also requires the proCompPlus SDK version alpha 1.0, which is included in this directory.

In the client's window there will be three signals graphed against a shared time axis
Top – skin temp (blue) #2
Middle – skin conductance GSR (green) #1
Bottom – EKG (red) #0
These colors above are defaults. They can be changed in the config file, described below.

Key commands:
If the graphing window is active (i.e the mouse was clicked in it), it will respond to the following keyboard commands
Esc - quit. Always quit the problem this way, otherwise, it may not save the log file if you quit by just closing the window
A,a increase/decrease the vertical offset for channel #0 (this is the vertical position on the screen where the line is drawn)
S,s for channel #1
D,d for channel #2

1/!	Increase/decreases vertical scale of channel 1
2/@	increase/decress vertical scale of channel 2
3/#	increase/decrease vertical scale of channel 3
h/H	show detected heart beats (drawn as a vertical line)
f/F	show flatten beats
t/T	increase/decrease horizontal time axis
q/Q	re-center channel 0
w/W	re-center channel 1
e/E	re-center channel 2
a	channel 0: move up
A	channel 0: move down
s	channel 1: move up
S	channel 1: move down
d	channel 2: move up
D	channel 2: move down
l	print the current level of all three channels to the screen
v/V	toggle the EKG display (on or off)
b/B	toggle the GSR display
n/N	toggle the ST display
m/M	toggle the keyboard event display
esc	quit

Procomp.cfg file:

Logfile
directory in which the data will be recorded.
Should be formatted as C:\my_dir\
Directory must exist
Each time you run the program, a new file will be generated. It will be "physio_data.xxxxxxx.log", where xxxxx is some unique number

Serveraddr
The name of the machine on which the proCompserver, vrpn_ProCompPlus.exe, is running

replaylogfile
proCompPlus@file:C:\my_dir\physio_data.45345.log

Don't press keyboard events to close to each other!

Allow at least one heartbeat and a second of GSR data in between. Otherwise, the analysis becomes difficult.

Marking events with the keyboard and remotely

The operator/experimenter can mark events with the keyboard (in the server window), and these are forwarded to the graphics client and logged with the physio data. In addition to the keyboard events, another program can also send events to the procomp server. This program can be running on the same or different machines as the procomp server. The remote events are processed by the procomp server in the same way as the keyboard events.

Consider the example of a virtual environment experiment. If we want to mark the time that the subject opens a virtual door in the physio data, we have two options:

1) The experimenter can press a key (say the spacebar) in the procomp server's window when the subject opens the door. This option is easier to setup, but requires the experimenter to be involved during the experiment.
2) The program which displays the virtual environment knows when the user opens the virtual door. It can automatically send an event to the procomp server. This option is more consistent and reliable and frees the experimenter from this burden. However, the virtual environment program must be modified in order for it to send events to the procomp server. This makes the coding for the experiment more involved. If you want to modify a program to make it send events, see the visual C++ project "physio_remote_events" and the program "physio_remote_event_sender_sample.exe" (included with the distribution of this program) for an example of how to do this.

The procomp client can accept keyboard and remote events simultaneously.

Analysis & Export

Several kinds of analysis can be performed. The settings in the procomp.cfg file control which analyses are computed on which channels. All the files in the batch list are treated the same. For example, if you request "AVE" (average) for the GSR channel, this program computes the average for the GSR on all the log files in the batch list. The batch list also set in the procomp.cfg file.

The most common analysis setting is to perform the IBI and FLT analyses on the EKG channel, and AVE (average signal level) on the GSR channel. FLT (flatten) removes the breathing artefacts from the EKG data, and the IBI (inter-beat-interval) takes the flatten data and computes the time between heart beats. This is the inter-beat-interval, which is the inverse of the heart rate.

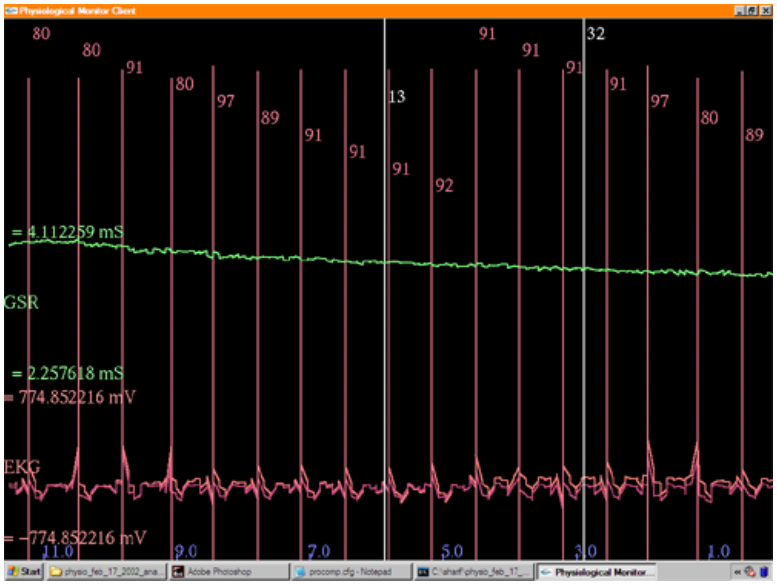

Figure 2 – replay of a .log file with the FLT and IBI analysis turned on. Each vertical red line corresponds to a heartbeat that was detected by the program. The red number to the upper right of each line is the instantaneous heart rate (beats/sec) Pushing 'H' in the graph window toggles the display of the heartbeats.

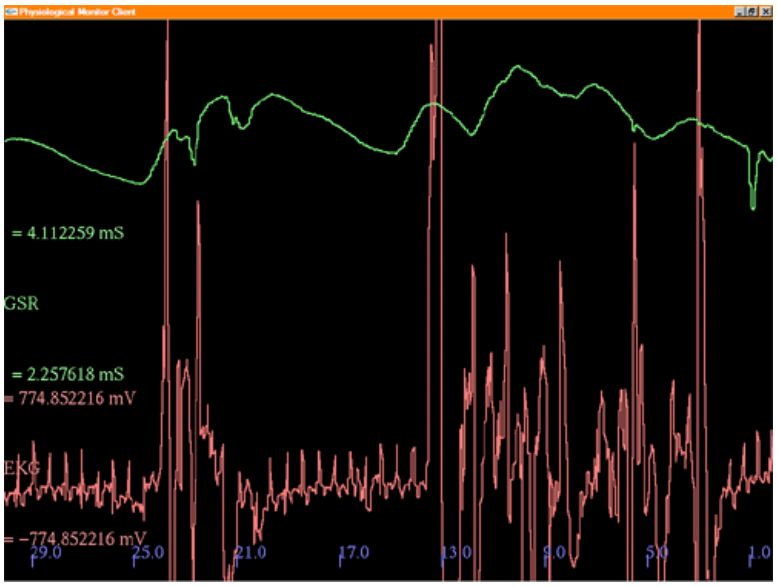

Figure 3 – a glitch in the EKG data. Perhaps the sensors or the proComp+ box were knocked? The analysis for this 20 second segment is not reliable – this software cannot detect heartbeats well because of the noise. Since the IBI is averaged over a long time, it does not have a big affect on the overall results.

To configure the client program to compute the above analysis, the procomp.cfg would have the following:

```
analysis_EKG          ibi
analysis_EKG          flt
analysis_GSR          ave
avespan_GSR           4000
```

If the batch list contained one file:

C4p4.log

The client program would produce the following output files:

1. c4p4_all_data_raw_export_32Hz.txt
2. c4p4_EKG_data_raw_export_256Hz.txt
3. c4p4_ibi_EKG.txt
4. overallibi_EKG.txt
5. overallave_GSR.txt

The raw export files are only produces if the procomp.cfg has the following line:

```
export_raw_data       YES
```

c4p4_all_data_raw_export_32Hz has the following 8 columns:

- ***Absolute time - seconds***
- ***Absolute time - microseconds***.
  These two fields make up the UNIX standard absolute time (number of seconds & microseconds since Jan 1 1970, I think)
- ***time since beginning of recording - seconds***
- ***time since beginning of recording -microseconds***
- ***keyboard event number***.
  If the experimenter pressed a keyboard key during the recording, the number corresponding to that key will show up here. Most of the time, no key was pressed, so for most of the lines this column has a 0 in it
- ***raw EKG data***
  (note this data is down-sampled from 256Hz – see the note below)
- ***raw GSR data***
- ***raw ST (skin temperature) data***

There is one line for each 1/32th of a second of recording time.

The proComp+ box samples the GSR and ST data at 32Hz (32 samples every second). On the other hand, the EKG data is sampled at 256Hz. This data is recorded in the .log files at these rates. For the purpose of graphing and computing the analysis, the 256Hz data EKG is converted (or down-sampled) to 32Hz. It collects 8 samples at 256Hz, and throws out all but the highest sample level. For example, if the 256Hz stream has "20 23 25 80 100 80 20 21 ", the 32Hz stream will have "100" in it. This technique is crude, but works as long as the heartbeat spikes point up (see figure).
Note: as of Summer 2002, the analysis is done on the full 256Hz EKG data stream. This is more reliable and less likely to miss heartbeats. It does require a faster computer in order to do the analysis in real-time, while recording the physio data.

If you want to see the higher-fidelity data, use the EKG_data_raw_export_256Hz.txt file. It has the EKG at the original sample rate.

EKG_data_raw_export_256Hz.txt has the following 4 columns:

- ***Absolute time - seconds***
- ***Absolute time - microseconds***.
  These two fields make up the UNIX absolute time (number of seconds & microseconds since Jan 1 1970, I think). Each set of 8 lines has the same time stamp. The clock we use is not very accurate, but is good enough for any analysis you might want to do.

- ***Keyboard Event Number***
  Described above
- ***Raw EKG Data***

overallibi_EKG.txt has the 7 following columns:

- ***Log file name,***
- ***Keyboard event #,***
- ***Number of heartbeats during this segment,***
- ***Average ibi of during the segment (in milliseconds),***
- ***variance of ibi during this segment,***
- ***time over which this ibi was computed = the length of this segment (in milliseconds)***
- ***Average heart rate for this segment (in beats per minute)***

There is one line for each keyboard event (or segment) in the .log file.
For event keyboard event, this file will have statistics for the segment of the log file BEFORE the key was pressed. For example, if you start recording a new file, then press keyboard events 300 and 400 and then quit the recording, The overallibi_EKG file will have two lines for this file:

```
Blah.log   300  number of heartbeats that happened since
begin of recording until you pushed the key #300 and more
stats
Blah.log   400  number of heartbeats that happened since
pushing the key #300 and the key #400 etc.  and more
stats
```

The data between the very last keyboard event and the end of the file is not analysed (but it is recorded in the .log file)

If there are no heartbeats between two keyboard events, the 3rd column will have a zero in it, and columns 4-6 will be empty (so not every line will have 6 columns). If there is only one heartbeat between keyboard events, the variance for that segment will be infinite (listed as 1.#IND00).

To convert the average ibi (in milliseconds) to heat rate (in beats per minute), use to formula 60/(ibi/1000).

c4p4_ibi_EKG.txt has information similar to overall_ibi_EKG.txt, but in more detail. It contains the ibi for each and every heart beat that it detected, not just the average ibi. Usually, we don't look at this file. It's for debugging. Also, this file is for one log file, while overall_ibi_EKG.txt has the heart rate & ibi info for all the log files in the batch list.

Overallave_GSR.txt has the 5 following columns:

- ***Log file name,***
- ***Keyboard event #,***
- ***Analysis Channel #***
  (0=EKG, 1=GSR 2=Skin Temp. There's no need for this column, because the information is also in the filename)

- ***Average signal level during the segment,***
- ***time over which this average was computed = the length of this segment (in seconds)***
  This is rounded to the nearest second. Hence, if you have many events marked very close to each other (i.e. only ½ second apart), there may be much error in the analysis. If this is the case, you can have the analysis ignore certain events. See the procomp.cfg file for more information.

Programmer's Notes

Useful to anyone that might have to read/debug or otherwise maintain this code.

Language – This is written in C++. However, the only feature of C++ we use are iostreams. Other than that, the code is mostly C. We don't use templates or classes. The projects files are for Microsoft Visual C++ 6.0.

Libraries – We make use of many libraries:
VRPN – networking, logging and timestamping. See
http://www.cs.unc.edu/research/vrpn for documentation
GLUT & OpenGL for graphing and handling keyboard presses
Procomp API – for reading data from the pro comp box

Channels – the word channels is heavily overloaded in this code. There are three kinds of channels: proCompPlus channels (the 6 plugs ABCDEF on the front of the proComp box), VRPN channels, and analysis channels. Hopefully it is clear which channel is which in the code.

VRPN Analog channels:
- 0 not used
- 1 GSR
- 2 Skin Temp
- 3 keyboard events
- 4-11 EKG Data

Analysis channels:
(These are what the user sees when running the client)
- 0 EKG
- 1 GSR
- 2 ST skin temp

How to build code from the UNC EVE CVS repository:
Pre-compiled .zip files contain everything need to install and use the program. If you plan on modifying the code and checking in the changes, you will need to get the code from the UNC EVE CVS repository. Contact Sharif to get access this repository if you don't already have it. Once you check out the physio directory from CVS, you will need to copy the VRPN (and GL directory into the physio directory (…/physio/vrpn & …/physio/GL). You will also need to copy the GLUT and GL dlls and .libs into the physio directory. We use local copes of the file so that users can just copy the physio folder to a new machine and it will still work. Finally, build the VRPN libraries, then build the sample_client and vrpn_proCompPlus projects (from the vrpn_proCompPlus.dsw workspace).

How to build code from the UNC EVE website:
If you downloaded the code, you it should already be built. You can also click on "start_here.dsw" and Visual C++ 6.0 should load everything. We have not tested with Visual Dev.NET…

For more info

VRPN documentation (http://www.cs.unc.edu/research/vrpn)
ProComp API (pdf included)
Mike Meehan's thesis (http://www.cs.unc.edu/publications for electrode placement and physio info.