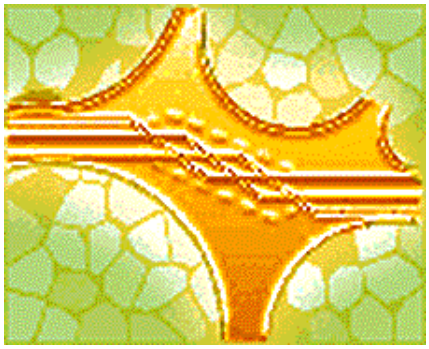
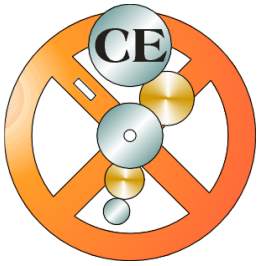
Smart Implantable Medical Systems
The SiMS Project



ImpBench Suite v1.1 – Release Notes

Christos Strydis, Computer Engineering Laboratory, Delft University of Technology



Computer Engineering Laboratory
Electrical Engineering, Mathematics & Computer Science Faculty
Delft University of Technology

DISCLAIMER

The ImpBench Suite has been developed with the explicit aim of providing the widest possible audience with access to the scientific and technological resources it entails.

We have made every reasonable effort to preserve and abide by the intellectual-property rights of the respective holders of all included parts. In some cases this may not be possible because one or more entitled parties or heirs cannot be traced.

Persons or institutions who wish to exercise potential rights to parts of the published ImpBench Suite should contact us (www.erasmusbrainproject.com), specifying the related material. In such cases, the representatives of the SiMS project may have to amend or even remove the material in question.

The ImpBench Suite is designed for educational and other non-profit purposes. No payment is made or received for material submitted or referenced in the ImpBench Suite. We accept no responsibility for improper use of the ImpBench Suite, or for rights or claims deriving from its use.

REVISION HISTORY

Version:	Status:	Details:	Author:	Date:
v1.0	Online	Initial release of ImpBench suite.	C. Strydis	21-02-2008
v1.1	Online	Updated release of ImpBench suite, with new contact info and minor fixes	C. Strydis	01-08-2014

OVERVIEW

ImpBench is a novel benchmark suite meant for designing and evaluating new digital processors for microelectronic implants. In an application field as wide as the various pathoses of the human body, we have conceptualized this suite based on common-sense and market-driven indicators, and we have established its usefulness and uniqueness based on extensive experimental measurement. In its first stable version 1.0, the suite consists of eight carefully selected programs, chosen on the basis of popularity among contemporary and emerging implant applications.

A plethora of benchmark suites has been proposed so far for characterizing a variety of application domains. Of late, workload-characterization programs more suited to the embedded domain have begun spurting in an attempt to better capture the particular characteristics of embedded processors. A special category of embedded systems with particular design constraints is implantable, microelectronic devices. Since their birth, such devices have been traditionally designed in custom style, always attempting to squeeze the desired functionality in an extremely limited size and with the maximum possible safety.

With the advent of mature microelectronics and micromachining technologies as well as more sophisticated computer-architecture and compiler design, this trend has began to change. Continuously more implant designers are willing (and free) to move to software running implant architectures to achieve their goals. In view of more structured and educated implant processors in the years to come, we have carefully put together ImpBench, a collection of benchmark programs and assorted input datasets, able to capture the intrinsic details of new architectures under evaluation. ImpBench has been shown [9] to display considerably different characteristics than other related benchmark collections, justifying introducing an utterly new benchmark suite.

ImpBench is a dynamic construct and, in the future, more benchmarks will be added, subject to our ongoing research. Among others, we anticipate simple DSP applications as potential candidates as well as more "real applications" like the ones we already included.

IMPBENCH STRUCTURE

Although ImpBench is expected to be a continuously evolving and updated tool, still we are confident that we have correctly identified a common subset of programs essential for all current and future implantable systems. To draw a clear structure of our proposed benchmark programs, we have grouped them in four distinct categories of two programs each:

- **Lossless data compression:** miniLZO, Finnish.
- **Symmetric-key encryption:** MISTY1, RC6.
- **Data-integrity:** CHECKSUM, CRC32.
- **Real applications (a.k.a. synthetic programs):** motion, DMU.

Selection of the specific two pairs of compression and encryption algorithms, above, has been based on related works [11], [12] investigating suitable algorithms for highly resource-constrained embedded systems. As explained in those works, lossless as opposed to lossy compression algorithms have been included since information deterioration is not an option for implant applications, in the general case.

Also, symmetrical- as opposed to asymmetric-encryption algorithms have been included since they characterize better the operational profile of implants, that is, a mission-critical, highly-personalized functionality.

The checksum code has been selected for its minimal overhead and effectiveness (it has been used in implantable systems time and again) while CRC32 has already been implemented in various lightweight network protocols including the energy-scavenging ZigBee.

Lastly, we have implemented both real applications (motion and DMU) after extensively investigating the diverse field of implant applications and consider them capable of capturing commonly met operations in contemporary and future implants. Suitable datasets representing biological content have been used to feed all benchmarks. Particularly for the DMU benchmark, actual, field-collected datasets have been used in order to capture the exact behavior of the simulated implantable system. By including pairs of different algorithms performing similar functionality in ImpBench, we attempt to offer some benchmarking diversity able to capture different aspects of new implantable systems, when evaluated through the ImpBench suite.

BENCHMARK DETAILS

The following *legend* has been used for filling in the table:

- Abbreviation: The abbreviated benchmark name, as used in the ImpBench sources.
- Description: A short explanation of the benchmark functionality, origins and suitability for the ImpBench suite.
- Original source: The author(s) of the original benchmark.
- Online resource: Actual source code of the original benchmark.
- ImpBench source: The author(s) of the version used in the ImpBench suite. It applies for either major or minor modifications to the existent benchmark source as well as for the authoring of a new benchmark from scratch.
- Modifications: Concise list of modifications to the original benchmark sources, if applicable. See benchmark sources for a more complete list of modifications.
- Original license: The type of license protecting the original benchmark sources, if any.
- ImpBench license: The type of license protecting the derived ImpBench sources, if any.

ImpBench DETAILS		
		-- COMPRESSION --
miniLZO	Abbreviation	mlzo
	Description	MiniLZO is a light-weight subset of the LZO library (LZ77-variant). LZO is a data compression library suitable for data de-/compression in real-time, i.e. it favors speed over compression ratio. LZO is written in ANSI C and is designed to be portable across platforms.
	Original source	Markus F.X.J. Oberhumer
	Online resource	http://www.oberhumer.com/opensource/lzo/download/minilzo-2.03.tar.gz
	ImpBench source	Christos Strydis
	Modifications	See comments in top-level source file.
	Original license	GNU General Public License – version 2
	ImpBench license	The rights of the initial source(s) are as specified above, under "Original license". This implementation is the property of: Christos Strydis, CE Lab, TU Delft, 2007.
Finnish	Abbreviation	fin
	Description	This is a C version of the Finnish submission to the Dr. Dobbs compression contest. It is considered to be one of the fastest DOS compressors and is, in fact, a LZ77-variant, its functionality based on a 2-character memory window. It is part of the "Lossless Data-Compression Kit, LDS v1.3," by N. de Vries.
	Original source	Jussi Puttonen, Timo Raita and Jukka Teuhola (original assembly version), Nico de Vries (C version)
	Online resource	http://www.nicodevries.com/nico/lds13.zip
	ImpBench source	Christos Strydis
	Modifications	See comments in top-level source file.
	Original license	• Original source submitted to the Dr. Dobb's Journal: Data Compression Contest. • C version implemented by N. de Vries.
	ImpBench license	The rights of the initial source(s) are as specified above, under "Original license". This implementation is the property of: Christos Strydis, CE Lab, TU Delft, 2007.

		-- ENCRYPTION --
MISTY1	Abbreviation	misty1
	Description	MISTY1 is one of the CRYPTREC-recommended 64-bit ciphers and is the predecessor of KASUMI, the 3GPP-endorsed encryption algorithm. MISTY1 is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups. MISTY1 is considered secure with full 8 rounds.
	Original source	Hironobu Suzuki
	Online resource	http://www.mirrorservice.org/sites/ftp.wiretapped.net/pub/security/cryptography/
	ImpBench source	Christos Strydis, Di Zhu
	Modifications	See comments in top-level source file.
	Original license	• MISTY1 architecture: Royalty-free open standard, documented in RFC2994 [1]. • Original MISTY1 implementation: Mitsubishi Electric Corporation, 2002 (c) [1].
	ImpBench license	The rights of the initial source(s) are as specified above, under "Original license". This implementation is the property of: Christos Strydis and Di Zhu, CE Lab, TU Delft, 2008.
RC6	Abbreviation	rc6
	Description	RC6 is a parameterized cipher and has a small code size. RC6 is one of the five finalists that competed in the AES challenge and has reasonable performance. Further, Slijepcevic et al. selected RC6 as the algorithm of choice for WSNs. RC6-32/20/16 with 20 rounds is considered secure.
	Original source	Y.W. Law – EYES project
	Online resource	http://wwwes.cs.utwente.nl/eyes/crypto_test.zip
	ImpBench source	Christos Strydis, Di Zhu
	Modifications	See comments in top-level source file.
	Original license	• RC6 architecture: Patented • Original RC6 implementation: Y.W. Law – EYES project
	ImpBench license	The rights of the initial source(s) are as specified above, under "Original license". This implementation is the property of: Christos Strydis and Di Zhu, CE Lab, TU Delft, 2008.

		-- DATA INTEGRITY --
checksum	Abbreviation	csum
	Description	The checksum is an error-detecting code that is mainly used in network protocols (e.g. IP and TCP header checksum). The checksum is calculated by adding the bytes of the data, adding the carry bits to the least significant bytes and then getting the two's complement of the results. The main advantage of the checksum code is that it can be easily implemented using an adder. The main disadvantage is that it cannot detect some types of errors (e.g. reordering the data bytes). In the proposed benchmark, a 16-bit checksum code has been selected which is the most common type used for telecommunications protocols.
	Original source	R. Braden, D. Borman, and C. Partridge
	Online resource	N/A
	ImpBench source	Christos Strydis, Christoforos Kachris
	Modifications	See comments in top-level source file.
	Original license	R. Braden, D. Borman, and C. Partridge [4]
	ImpBench license	The initial source is the property of the original authors. This implementation is the property of: Christos Strydis and Christoforos Kachris, CE Lab, TU Delft, 2008.
crc32	Abbreviation	crc32
	Description	The Cyclic-Redundancy Check (CRC) is an error-detecting code that is based on polynomial division. The main advantage of the CRC code is its simple implementation in hardware, since the polynomial division can be implemented using a shift register and XOR gates.
	Original source	G. Memik, W. H. Mangione-Smith, and W. Hu – NetBench (implementation adapted from [5])
	Online resource	NetBench Website [6]
	ImpBench source	Christos Strydis, Christoforos Kachris
	Modifications	See comments in top-level source file.
	Original license	• Base, optimized, checksum implementation: R. Braden, D. Borman, and C. Partridge [4]. • Subsequent implementation as part of the NetBench suite [5].
	ImpBench license	The initial source is the property of the original authors. This implementation is the property of: Christos Strydis and Christoforos Kachris, CE Lab, TU Delft, 2008.

		-- REAL APPLICATIONS --
motion	Abbreviation	motion
	Description	This is a synthetic benchmark based on the algorithm described in the work of Wouters et al. [7]. It is a motion-detection algorithm for the movement of animals. In this algorithm, the degree of activity is actually monitored rather than the exact value of the amplitude of the activity signal. That is, the percentage of samples above a set threshold value in a given monitoring window. In effect, this motion-detection algorithm is a smart, efficient, data-reduction algorithm.
	Original source	R. Puers and P. Wouters (concept described in [7])
	Online resource	N/A (Original did not contain a software implementation)
	ImpBench source	Christos Strydis
	Modifications	The whole benchmark has been implemented as a synthetic kernel implementing the functionality described in [7].
	Original license	Implemented as an analog subsystem (HW) of an implant, described in journal publication [7].
	ImpBench license	The motion-detection-algorithm concept and circuit implementation are the property of the paper authors. This implementation is the property of: Christos Strydis, CE Lab, TU Delft, 2008.
DMU	Abbreviation	dmu
	Description	This is a synthetic benchmark based on the system described in the work of Cross et al. [8]. It simulates a drug-delivery & monitoring unit (DMU). This program does (and can) not simulate all real-time time aspects of the actual (interrupt-driven) system, such as sensor/actuator-specific control, low-level functionality, transceiver operation and so on. Nonetheless, the emphasis here is on the operations performed by the implant core in response to external and internal events (i.e. interrupts). A realistic model has been built imitating the real system as closely as possible.
	Original source	P. S. Cross, R. Kunnemeyer, C. R. Bunt, D. A. Carnegie and Michael J. Rathbone (concept described in [8])
	Online resource	Original software sources can be directly requested from the author. See comments in source file.
	ImpBench source	Christos Strydis
	Modifications	The whole benchmark has been implemented as a synthetic kernel implementing the functionality described in [8]. It has been based on the original embedded-C code running in the implant, provided by P. Cross.
	Original license	Implemented as the control SW of an implant, described in journal publication [8].
	ImpBench license	The delivery-monitoring unit (dmu) concept and sources are the property of the paper authors. This implementation is the property of: Christos Strydis, CE Lab, TU Delft, 2008.

A. Comments:

- For further details on the ImpBench characteristics, and related references please refer to [9].
- For more technical details on the structure of the various benchmarks as well as other hands-on aspects, please refer to the extensive comments at the beginning of each top-level, benchmark source file.
- If all else fails, you are welcome to contact the ImpBench primary author, Christos Strydis, at: C.Strydis@erasmusmc.nl.
- All ImpBench benchmarks have been successfully built and run with GNU GCC versions 4.1.1, 3.3.4, 3.3.3, 2.96 and 2.95.3 as well as with the ARM CROSS-GCC version 2.95.2 [10].

B. References

[1] H. Ohta and M. Matsui, A Description of the MISTY1 Encryption Algorithm, United States, 2000.

[2] http://global.mitsubishielectric.com/pdf/advance/vol100/03Vol100_TR2.pdf [unavailable], Copyright (c) 2002 Mitsubishi Electric Corporation.

[3] G. Memik, W. H. Mangione-Smith, and W. Hu, "NetBench: a benchmarking suite for network processors," in IEEE/ACM international conference on Computer-aided design (ICCAD'01), Piscataway, NJ, USA, 2001, pp. 39–42.

[4] R. Braden, D. Borman, and C. Partridge, "Computing the internet checksum", SIGCOMM Comput. Commun. Rev., vol. 19, no. 2, pp. 86–94, 1989.

[5] "Cell Relay Retreat: CRC-32 Calculation, Test Cases and HEC Tutorial", http://cell.onecall.net/cell-relay/publications/software/ [unavailable].

[6] NetBench Web Site: http://istanbul.icsl.ucla.edu/NetBench [unavailable]

[7] R. Puers and P. Wouters, Adaptable interface Circuits for Flexible Monitoring of Temperature and Movement, AICSP, Vol. 14, pp. 193--206, 1997.

[8] P. Cross, R. Kunnemeyer, C. Bunt, D. Carnegie, and M. Rathbone, Control, communication and monitoring of intravaginal drug delivery in dairy cows, in *International Journal of Pharmaceuticals*, vol. 282, 10 September 2004, pp. 35--44.

[9] C. Strydis, C. Kachris, G. N. Gaydadjiev, ImpBench: A novel benchmark suite for biomedical, microelectronic implants, Int. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation (IC-SAMOS VIII), pp. 86-95, Samos, Greece, July 2008.

[10] SimpleScalar Version 4.0 Test Releases: http://www.simplescalar.com/v4test.html.

[11] C. Strydis and G. Gaydadjiev, Profiling of lossless data-compression algorithms for a novel biomedical-implant architecture, The 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis, pp. 109-114, Atlanta, Georgia, USA, October 2008.

[12] C. Strydis, D. Zhu, and G. Gaydadjiev, Profiling of symmetric encryption algorithms for a novel biomedical-implant architecture, Proceedings of the ACM International Conference on Computing Frontiers (CF 08), pp. 231-240, Ischia, Italy, May 2008.

C. Other resources:

- Compression – http://www.synchrondata.com/pheaven/www/area4.htm
- Compression etc. – http://cd.textfiles.com/blackphilesii/PHILES/COMPUTER/FILES.BBS

INPUT DATASETS

A limited but representative number of input datasets is required for most of the ImpBench programs to consume. Existing literature in the field does not offer any solid pointers or even hints towards a "standard", representative dataset. Of course, a lot of biological-signal databases exist and are even available online. However, since there are no solid hints for favoring one dataset source over another, the datasets selected with the ImpBench suite have been extracted from the "Biopac Student Lab 3.7" (BSL) software for reasons of availability and simplicity. They have the following technical specifications:

dataset type	BSL test name	dataset name	BIN size (B)	ASCII size (B)	samples (#)	duration (sec)	sample rate (sml/sec)	sample rate (KB/sec)
Electromyogram II (EMG)	John-L02	**EMGII_01**	1152	1778	144	0,288	500	3,89
Electromyogram II (EMG)	John-L02	**EMGII_10**	9608	14095	1201	2,402	500	3,91
Electroencephalogram I (EEG)	John-L03	**EEGI_01**	984	1332	123	0,615	200	1,56
Electroencephalogram I (EEG)	John-L03	**EEGI_10**	9616	13005	1202	6,010	200	1,56
Electrocardiogram I (ECG)	John-L05	**ECGI_01**	912	1406	114	0,114	1000	7,81
Electrocardiogram I (ECG)	John-L05	**ECGI_10**	9616	14824	1202	1,202	1000	7,81
Respiratory Cycle I (RC)	John-L08	**RCI_01**	1192	1770	149	1,490	100	0,78
Respiratory Cycle I (RC)	John-L08	**RCI_10**	9520	14581	1191	11,910	100	0,78
Pulmonary Function I (PF)	John-L12	**PFI_01**	1184	1617	148	1,480	100	0,78
Pulmonary Function I (PF)	John-L12	**PFI_10**	9240	12863	1155	11,550	100	0,78
Skin Temperature (AEP)	John-L15	**AEP_01**	1120	1397	140	0,700	200	1,56
Skin Temperature (AEP)	John-L15	**AEP_10**	9736	11739	1217	6,085	200	1,56
Blood Pressure (BP)	John-L16	**BP_01**	1128	1404	141	0,282	500	3,91
Blood Pressure (BP)	John-L16	**BP_10**	9584	12798	1198	2,396	500	3,89

From the previous table, we can see that typical signals of muscle activity (EMG), heart activity (ECG) and brain neural activity (EEG) as well as breath oxygen volume (RC), lung volume (PF), skin surface temperature (AEP) and blood pressure (BP) have been collected.

All considered datasets are representing one-dimensional physiological-signal measurements of varied sampling rates, depending on the particular application. As can be also seen from the table, two dataset sizes have been used: roughly 1-KB and 10-KB data. These readouts are actually double-precision, floating-point numbers and have been stored in both ASCII- and binary-encoded files:

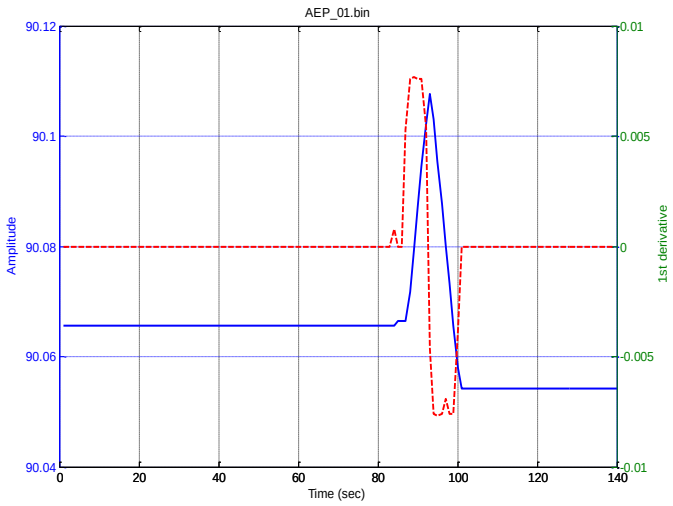
- 8-bit ASCII representation: Each readout is a text string terminated (on each line) with CR (carriage-return) and LF (line-feed) characters. Each string consists of as many Bytes as the number of digits plus two extra Bytes for the decimal point and the sign characters; thus, about 8 to 10 Bytes are used per entry, in the general case.
- Binary representation: Each readout is a packed, double-precision, floating-point number (IEEE-754) of 64 bits; thus precisely 8 Bytes are used per entry.

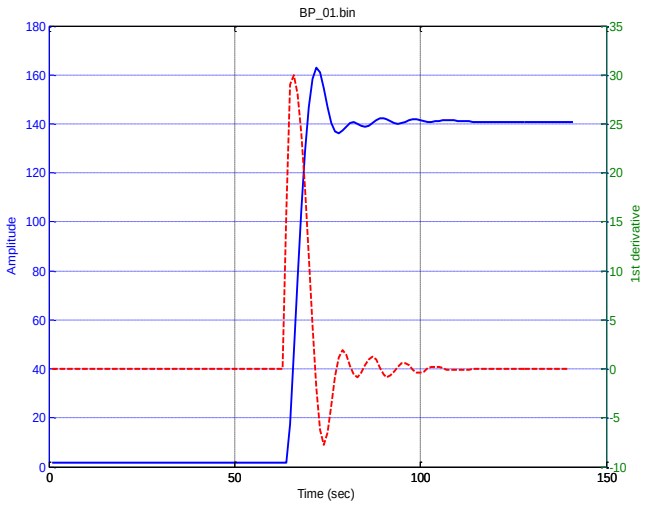
Except for the *dmu* benchmark which uses its own datasets, all other benchmarks can process either ASCII- or binary-formatted data with no problem. The only current exception lies with the *motion* benchmark which (when compiled with the arm cross-gcc-2.95.2) does not generate correct results when reading binary data.

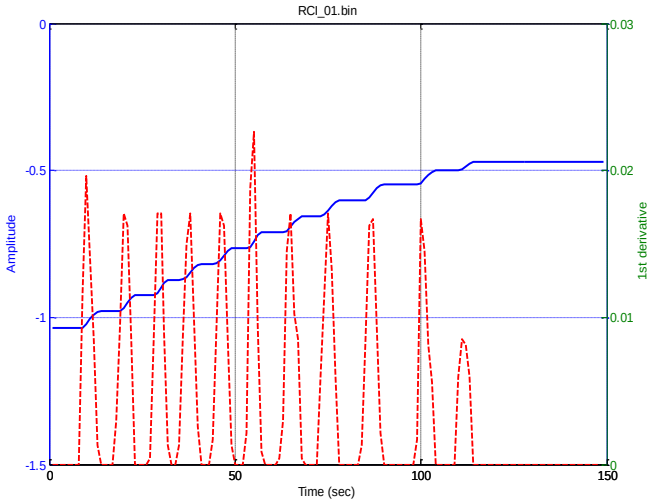
Most of the input datasets come with a sufficiently extensive description of the experimental setup and the acquisition parameters, as shown in the table below (two entries for each dataset type, one for each dataset size):

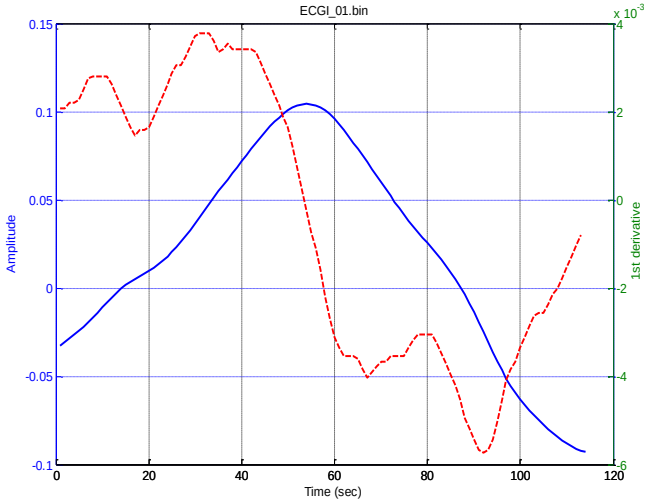
dataset type	description
Electromyogram II (EMG)	First procedure, Forearm 1 (dominant): Increased clenches in increments of 5 Kg until maximum clench force is obtained.
Electromyogram II (EMG)	Second procedure, Forearm 1 (dominant): Continued maximal clench until fatigue causes 50% reduction in measured force.
Electroencephalogram I (EEG)	This recording shows the Subject in a relaxed state, with eyes closed for about 10 seconds. The eyes were then opened for approx. 12 seconds, closed approx. 12 seconds, then opened for the remainder of the recording time.
Electroencephalogram I (EEG)	N/A
Electrocardiogram I (ECG)	First procedure: Relaxed, lying down. Second procedure: Relaxed, sitting up. Third procedure: Relaxed, sitting up state, taking prolonged breaths. Markers at the start of inhales and exhales. Fourth procedure: Relaxed, sitting up state, recovering from exercise.
Electrocardiogram I (ECG)	N/A
Respiratory Cycle I (RC)	First procedure: Seated in a chair and breathing normally. Second procedure: hyperventilation for 30 seconds, then recovery. Third procedure: hypoventilation, then recovery. Fourth procedure: coughing, then reading aloud.
Respiratory Cycle I (RC)	N/A
Pulmonary Function I (PF)	This recording shows normal breathing for 3 breaths, full inhale, return to normal breathing, full exhale, then a return to normal breathing. Note that a Residual Volume of 1.0 Liters was used.
Pulmonary Function I (PF)	N/A
Skin Temperature (AEP)	Aerobic Exercise Physiology
Skin Temperature (AEP)	N/A
Blood Pressure (BP)	First procedure: Subject has cuff on LEFT arm, and is sitting up at rest. Second procedure: Repeat of the first procedure. Third procedure: Subject has pressure cuff on RIGHT arm, and is sitting up at rest. Fourth procedure: Repeat of the third procedure. Fifth procedure: Subject is lying down at rest, with pressure cuff on RIGHT arm. Sixth procedure: Repeat of the fifth procedure. Seventh procedure: After mild exercise, Subject should sit up to recover with the pressure cuff on the RIGHT arm.
Blood Pressure (BP)	N/A

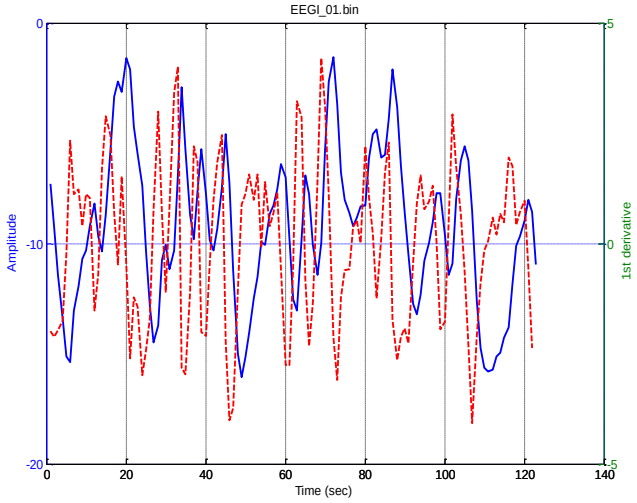
In the following figures, the various datasets for 1-KB data are plotted (amplitude vs. time) along with their first-order derivatives, to give a feeling of the morphology of the various datasets as well as to illustrate the rate of change of their values:

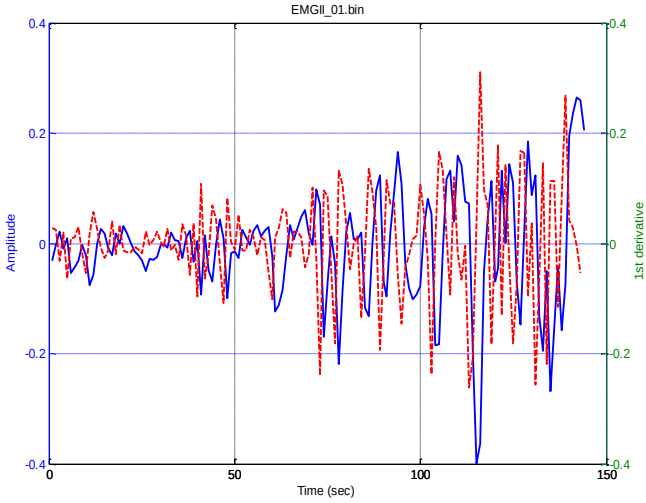

AEP_01.bin
Amplitude
1st derivative
Time (sec)

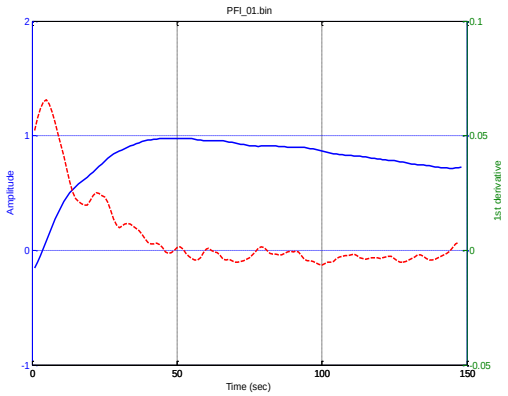

BP_01.bin
Amplitude
1st derivative
Time (sec)


RCI_01.bin
Amplitude
1st derivative
Time (sec)


ECGI_01.bin
x 10^-3
Amplitude
1st derivative
Time (sec)


EEGI_01.bin
Amplitude
1st derivative
Time (sec)


EMGII_01.bin
Amplitude
1st derivative
Time (sec)



From the figures above, it becomes apparent that all datasets are distinctly different in terms of morphology; i.e. amplitude, duration and rate of change, as well as sampling rate. This has been our original intention in defining a diverse yet contained set of input vectors and, subsequently, application scenarios.