PyCollect Documentation

Yeison Cardona

Jun 10, 2019

Contents

A Python package for collecting data from the GE patient monitors, for the purpose of conducting research and preparing training materials.

Tested on:

- CARESCAPE Monitor B650
- CARESCAPE Monitor B450

CHAPTER 1

Navigation

1.1 Getting started

1.1.1 Install

From source

```
git clone git clone git@bitbucket.org:gcpds/pycollect.git
cd pycollect
python setup.py install
```

From PyPi (not available yet)

```
pip install pycollect
```

1.1.2 Collect data

There are two main modules, `GEDevice` that handle the connection and data recollecting, and `GEDecode` that parse and sort the received data strings.

```
from pycollect import GEDevice
```

Connection

```
device = GEDevice()
device.connect('/dev/ttyUSB0')
```

Send request

A request is an instruction that enable the data transmission from monitor.

Subrecords

There are three subrecord types for the actual measurement data:

- `device.DISPL` for the displayed values.
- `device.TREND_10S` for the 10 s trended values.
- `device.TREND_60S` for the 60 s trended values.

When `device.DISPL` is used, there is an extra argument, `interval`, that indicate the transmission interval in seconds, this must be possitive and greater or equal to 5.

```
device.request(subtype=device.DISPL)
```

`device.TREND_10S` and `device.TREND_60S` are used for request trends.

```
device.request(subtype=device.TREND_10S)
```

```
device.request(subtype=device.TREND_60S)
```

No further transmission requests are needed after this.

Waveforms

Depending on the total number of samples per second the monitor sends a waveform packet every *1000 ms*, *500 ms* or *250 ms*. Request up to 8 waveforms but total sample rate should be less than 600 samples/sec, sample rate for **ECG** is 300, **INVP** 100, **EEG** 100, **PLETH** 100, respiratory (**CO2**, **O2**, **NO2** . . .) 25 each.

The waveform options can be consulted with:

```
from pycollect.modules import WAVEFORMS_DICT

for wave in WAVEFORMS_DICT:
    print("{label}: {samps}".format(**WAVEFORMS_DICT[wave]))
```

```
ECG1: 300
ECG2: 300
ECG3: 300
INVP1: 100
INVP2: 100
INVP3: 100
INVP4: 1
INVP5: 100
INVP6: 100
PLETH: 100
CO2: 25
NO2: 25
AA: 25
AWP: 25
FLOW: 1
```

(continues on next page)

(continued from previous page)

```
VOL: 25
RESP: 25
EEG1: 100
EEG2: 100
EEG3: 1
EEG4: 100
TONO_PRESS: 1
SPI_LOOP_STATUS: 1
ENT_100: 1
EEG_BIS: 1
```

```
device.request(waveform_set=['PLETH', 'ECG1'])  #400 samp/s
```

The limit of sample rate can not be exceeded:

```
try:
    device.request(waveform_set=['ECG1', 'ECG2', 'ECG3'])  #900 samp/s
except Exception as error:
    print(error)
```

```
Sample rate must be less or equal to 600.
```

No further transmission requests are needed after this.

Mixings

A combination of **Waveforms** and **Subrecords** can be requested at the same time.

```
device.request(subtype=device.DISPL, waveform_set=['PLETH', 'ECG1'])
```

Or in a sequence of requests.

```
device.request(subtype=device.DISPL, interval=10)
device.request(waveform_set=['PLETH', 'ECG1'])
```

```
device.request(subtype=device.DISPL, interval=10)
device.request(subtype=device.TREND_60S)
device.request(waveform_set=['PLETH', 'ECG1'])
```

Read data

The transmitted data is recollected asynchronously with a background thread and appended to a `BUFFER` list.

To activate the data collecting.

```
device.collect(True)
```

To clear the buffer input:

```
device.clear_buffer()
```

To stop the data collecting:

```
device.collect(False)
```

1.1.3 Decode data

The `GEDecode` module is used for parse and sort the received data strings.

```
from pycollect import GEDecode
```

```
decoder = GEDecode(device.BUFFER)
decoder.process(True)
```

1.1.4 Clear buffers

There are a set of methods for clear correctly the stored buffer and recollected data.

```
device.clear_buffer()  # clear input buffer.
decoder.clear_buffer()  # clear decoded data, breaks the synchrony.
decoder.clear_data()  # clear recollected data.
```

The above instructions breaks the synchrony between the collector and decoder, in order to decode input data again `GEDecode` must be reinstantiated:

```
decoder = GEDecode(device.BUFFER)
decoder.process(True)
```

1.1.5 Save data

The data can be saved in two differents formats *csv* and *edf*, each type of data will be saved with their own sufix: **.wave** for waveforms, **.trend10s** and **.trend60s** for trends.

Save data as CSV

```
decoder.save_as_csv('data_out')
```

Save data as EDF+

The *edf* format need extra patient information.

```
decoder.set_edf_header(
    admincode = '',
    birthdate = date(1900, 1, 1), #datetime object
    equipment = '',
    gender = 0, #0 for male, 1 for female
    patientcode = '',
    patientname = '',
    patient_additional = '',
    recording_additional = '',
    technician = '',
```

(continues on next page)

(continued from previous page)

```
)

decoder.save_as_edf('data_out')
```

1.2 Measures

All measurement data is represented as signed 16-bit values. Some control information is embedded into the measurement data by assigning special meaning to certain values.

As values with special meaning start from -32001 downwards, the smallest valid value is always -32000.

1.2.1 Physiological measures

ECG

Label	Description	Unit	Range
ECG HR	Heart rate	1/min	0 - 32768
ECG ST1	St-level	mm	
ECG ST2	St-level	mm	
ECG ST3	St-level	mm	
ECG IMP-RR	Respiration rate	1/min	0 - 32768
ECG: MOD	Measurement module existence	bool	
ECG: ACT	Measurement activity	bool	
ECG: ASY	Asystole	bool	
ECG HR-SRC	Heart rate source • 0: Not selected • 1: ECG • 2: Invasive pressure channel 1 • 3: Invasive pressure channel 2 • 4: Invasive pressure channel 3 • 5: Invasive pressure channel 4 • 6: SpO2 • 7: Invasive pressure channel 5 • 8: Invasive pressure channel 6	str	
ECG: NS	Noise	bool	
ECG: AR	Artifact	bool	
ECG: LRN	Learning	bool	
ECG: PCR	Pacer On	bool	
ECG: CH1	Channel 1 off	bool	
ECG: CH2	Channel 2 off	bool	
ECG: CH3	Channel 3 off	bool	
ECG LEAD-CH1	Lead configuration for channel 1 • 0: Not selected • 1: ECG I • 2: ECG II • 3: ECG III • 4: ECG AVR • 5: ECG AVL • 6: ECG AVF • 7: ECG V	str	
ECG LEAD-CH2	Lead configuration for channel 2 • 0: Not selected • 1: ECG I • 2: ECG II • 3: ECG III • 4: ECG AVR • 5: ECG AVL • 6: ECG AVF • 7: ECG V	str	
ECG LEAD-CH3	Lead configuration for	str	

INV-BP

Label	Description	Unit	Range
INV-BP SYS	Invasive pressure	mmHg	
INV-BP DIA	Invasive pressure	mmHg	
INV-BP MEAN	Invasive pressure	mmHg	
INV-BP HR	Pulse rate	1/min	
INV-BP: MOD	Measurement module existence	bool	
INV-BP: ACT	Measurement activity	bool	
INV-BP: ZR	Zeroing	bool	
INV-BP LBL	Invasive pressure label • 0: Not defined • 1: ART • 2: CVP • 3: PA • 4: RAP • 5: RVP • 6: LAP • 7: ICP • 8: ABP • 9: P1 • 10: P2 • 11: P3 • 12: P4 • 13: P5 • 14: P6	str	

NIBP

Label	Description	Unit	Range
NIBP SYS	Invasive pressure	mmHg	0 - 327
NIBP DIA	Invasive pressure	mmHg	0 - 327
NIBP MEAN	Invasive pressure	mmHg	0 - 327
NIBP HR	Pulse rate	1/min	
NIBP: MOD	Measurement module existence	bool	
NIBP: ACT	Measurement activity	bool	
NIBP CUFF	Invasive pressure cuff type • 0: Not defined • 1: Infant • 2: Reserved • 3: Adult	str	
NIBP: AUTO	Invasive pressure: AUTO mode selected	bool	
NIBP: STAT	Invasive pressure: STAT mode selected	bool	
NIBP: MSR	Invasive pressure: measuring	bool	
NIBP: STASIS	Invasive pressure: STASIS ON	bool	
NIBP: CLBR	Invasive pressure: calibrating	bool	
NIBP: OLD	Invasive pressure: data is older than 60s	bool	

TEMP

Label	Description	Unit	Range
`TEMP`	Temperature	°C	
`TEMP: MOD`	Measurement module existence	bool	
`TEMP: ACT`	Measurement activity	bool	
`TEMP LBL`	Temperature label • 0: Not used • 1: ESO • 2: NASO • 3: TYMP • 4: RECT • 5: BLAD • 6: AXIL • 7: SKIN • 8: AIRW • 9: ROOM • 10: MYO • 11: T1 • 12: T2 • 13: T3 • 14: T4 • 15: CORE • 16: SURF	str	

SpO2

Label	Description	Unit	Range
`SpO2`	Oxygenation percentage	%	
`SpO2 PR`	Pulse rate	1/min	
`SpO2 IR-AMP`	Modulation	%	
`SpO2 [SO2|SaO2|SvO2]`	Modulation, value is specified by the label.	%	
`SpO2: MOD`	Measurement module existence	bool	
`SpO2: ACT`	Measurement activity	bool	
`SpO2 LBL`	Measurement module existence • 0: SO2 • 1: SaO2 • 2: SvO2 • 3: Not used	str	

CO2

Label	Description	Unit	Range
CO2 ET	Expiratory concentration	%	0 - 100
CO2 FI	Inspiratory concentration	%	0 - 100
CO2 RR	Respiration rate	1/min	
CO2 PAMB	Ambient pressure	mmHg	
CO2: MOD	Measurement module existence	bool	
CO2: ACT	Measurement activity	bool	
CO2: AP	Apnea	bool	
CO2: CS	Calibrating sensor	bool	
CO2: ZS	Zeroing sensor	bool	
CO2: OC	Occlusion	bool	
CO2: ALK	Air leak	bool	
CO2 LBL	These bits indicate the respiration rate source • 0: Not selected • 1: CO2 • 2: ECG, Impedance respiratory	str	
CO2: CLBR	Caliabrating	bool	
CO2: MNS	Measurement off	bool	

O2

Label	Description	Unit	Range
O2 ET	Expiratory concentration	%	
O2 FI	Inspiratory concentration	%	
O2: MOD	Measurement module existence	bool	
O2: ACT	Measurement activity	bool	

N2O

Label	Description	Unit	Range
N2O ET	Expiratory concentration	%	
N2O FI	Inspiratory concentration	%	
N2O: MOD	Measurement module existence	bool	
N2O: ACT	Measurement activity	bool	
N2O: CLBR	Caliabrating	bool	
N2O: MNS	Measurement off	bool	

AA

Label	Description	Unit	Range
AA ET	Anesthesia Agents ET	%	
AA FI	Anesthesia Agent FI	%	
AA MAC-SUM	Anesthesia Agents MAC SUM	%	
AA: MOD	Measurement module existence	bool	
AA: ACT	Measurement activity	bool	
AA: CLBR	Caliabrating	bool	
AA: MNS	Measurement off	bool	
AA	Anesthesia Agent • 0: Unknow • 1: None • 2: HAL • 3: ENF • 4: ISO • 5: DES • 6: SEV	str	

FLOW-VOL

Label	Description	Unit	Range
FLOW-VOL RR	Respiration rate	1/min	
FLOW-VOL PPEAK	Peak pressure	cmH2O	
FLOW-VOL PEEP	Positive end exp. pressure	cmH2O	
FLOW-VOL PPLAT	Plateau pressure	cmH2O	
FLOW-VOL TV-INSP	Inspiratory tidal volume	ml	
FLOW-VOL TV-EXP	Expiratory tidal volume	ml	
FLOW-VOL COMP	Compliance	ml/cmH2O	
FLOW-VOL MV-EXP	Expiratory minute volume	l/min	
FLOW-VOL: MOD	Measurement module existence	bool	
FLOW-VOL: ACT	Measurement activity	bool	
FLOW-VOL: DIS	Disconnection	bool	
FLOW-VOL: CLBR	Calibrating	bool	
FLOW-VOL: ZR	Zeroing	bool	
FLOW-VOL: OBS	Obstruction	bool	
FLOW-VOL: LK	Leak	bool	
FLOW-VOL: MSR	Measurement off	bool	

CO-WEDGE

Label	Description	Unit	Range
CO-WEDGE CO	Cardiac output	ml/min	
CO-WEDGE TEMP	Blood temperature	°C	
CO-WEDGE REF	Right heart ejection fraction	%	
CO-WEDGE PCWP	Wedge pressure	mmHg	
CO-WEDGE: MOD	Measurement module existence	bool	
CO-WEDGE: ACT	Measurement activity	bool	
CO-WEDGE CO-AGE	Age of CO reading is > 60 s	None	
CO-WEDGE PCWP-AGE	Age of PCWP reading is > 60 s	None	

NMT

Label	Description	Unit	Range
NMT T1	Wedge pressure	%	
NMT TRATIO	t4/t1 in TOF mode, t2/t1 in DB mode	%	
NMT PTC-COUNT	Post tetanic count, max. value 21. Has value 31 if count not available	None	
NMT PTC-TOF-COUNT	TOF count in TOF mode	None	
NMT PTC-DB-COUNT	DB count in DB mode	None	
NMT PTC-ST-COUNT	ST count in ST mode	None	
NMT PTC-STIM	Stimulus current	mA	
NMT: MOD	Measurement module existence	bool	
NMT: ACT	Measurement activity	bool	
NMT STM	Stimulus mode • 0: Train Of Four (TOF mode) • 1: Double Burst (DB mode) • 2: Single Twitch (ST mode) • 3: Post-tetanic count • 4: Tetanic • 5: Regional block	str	
NMT TIME	Time • 0: Not used • 1: 100 us • 2: 200 us • 3: 300 us	str	
NMT: SUP	Supramax current found	bool	
NMT: CLBR	Calibrated	bool	

ECG-EXTRA

Label	Description	Unit	Range
ECG-EXTRA: HR	Heart rate as derived from the ecg signal	bool	
ECG-EXTRA: HR-MAX	Maximum heart rate	bool	
ECG-EXTRA: HR-MIN	Minimum heart rate	bool	
ECG-EXTRA: MOD	Measurement module existence	bool	
ECG-EXTRA: ACT	Measurement activity	bool	

SvO2

Label	Description	Unit	Range
SvO2	SvO2	%	
SvO2: MOD	Measurement module existence	bool	
SvO2: ACT	Measurement activity	bool	

ECG-ARRH

Label	Description	Unit	Range
ECG-ARRH HR	Heart rate	1/min	
ECG-ARRH RR	The RR interval	1/min	
ECG-ARRH PVC	Premature Ventricular Contractions	?	
ECG-ARRH: MOD	Measurement module existence	bool	
ECG-ARRH: ACT	Measurement activity	bool	

ECG-12

Label	Description	Unit	Range
ECG-12 STI	St-level	mm	
ECG-12 STII	St-level	mm	
ECG-12 STIII	St-level	mm	
ECG-12 STAVL	St-level	mm	
ECG-12 STAVR	St-level	mm	
ECG-12 STAVF	St-level	mm	
ECG-12 STV1	St-level	mm	
ECG-12 STV2	St-level	mm	
ECG-12 STV3	St-level	mm	
ECG-12 STV4	St-level	mm	
ECG-12 STV5	St-level	mm	
ECG-12 STV6	St-level	mm	
ECG-12: MOD	Measurement module existence	bool	
ECG-12: ACT	Measurement activity	bool	
ECG-12 LEAD-CH1	Lead configuration for channel 1 • 0: Not selected • 1: ECG I • 2: ECG II • 3: ECG III • 4: ECG AVR • 5: ECG AVL • 6: ECG AVF • 7: ECG V	str	
ECG-12 LEAD-CH2	Lead configuration for channel 2 • 0: Not selected • 1: ECG I • 2: ECG II • 3: ECG III • 4: ECG AVR • 5: ECG AVL • 6: ECG AVF • 7: ECG V	str	
ECG-12 LEAD-CH3	Lead configuration for channel 3 • 0: Not selected • 1: ECG I • 2: ECG II • 3: ECG III • 4: ECG AVR • 5: ECG AVL • 6: ECG AVF • 7: ECG V	str	

NMT2

Label	Description	Unit	Range
NMT2 T1	T1 absolute value	None	
NMT2 T2	T2 absolute value	None	
NMT2 T3	T3 absolute value	None	
NMT2 T4	T4 absolute value	None	
NMT2: MOD	Measurement module existence	bool	
NMT2: ACT	Measurement activity	bool	

EEG

Label	Description	Unit	Range
EEG FEMG	Frontal electro-myography	uv	
EEG EEG1-AMPL	RMS amplitude	uv	
EEG EEG1-SFR	Spectral edge frequency	Hz	
EEG EEG1-MNF	Median frequency	Hz	
EEG EEG1-DELTA	Relative power spectral content in delta band	%	
EEG EEG1-THETA	Relative power spectral content in theta band	%	
EEG EEG1-ALPHA	Relative power spectral content in alpha band	%	
EEG EEG1-BETA	Relative power spectral content in beta band	%	
EEG EEG1-BSR	Burst suppression ratio	%	
EEG EEG2-AMPL	RMS amplitude	uv	
EEG EEG2-SFR	Spectral edge frequency	Hz	
EEG EEG2-MF	Median frequency	Hz	
EEG EEG2-DELTA	Relative power spectral content in delta band	%	
EEG EEG2-THETA	Relative power spectral content in theta band	%	
EEG EEG2-ALPHA	Relative power spectral content in alpha band	%	
EEG EEG2-BETA	Relative power spectral content in beta band	%	
EEG EEG2-BSR	Burst suppression ratio	%	
EEG EEG3-AMPL	RMS amplitude	uv	
EEG EEG3-SEF	Spectral edge frequency	Hz	
EEG EEG3-MF	Median frequency	Hz	
EEG EEG3-DELTA	Relative power spectral content in delta band	%	
EEG EEG3-THETA	Relative power spectral content in theta band	%	
EEG EEG3-ALPHA	Relative power spectral content in alpha band	%	

Continued on next page

Table 1 – continued from previous page

Label	Description	Unit	Range
EEG EEG3-BETA	Relative power spectral content in beta band	%	
EEG EEG3 BSR	Burst suppression ratio	%	
EEG EEG4-AMPL	RMS amplitude	uv	
EEG EEG4-SEF	Spectral edge frequency	Hz	
EEG EEG4-MF	Median frequency	Hz	
EEG EEG4-DELTA	Relative power spectral content in delta band	%	
EEG EEG4-THETA	Relative power spectral content in theta band	%	
EEG EEG4-ALPHA	Relative power spectral content in alpha band	%	
EEG EEG4-BETA	Relative power spectral content in beta band	%	
EEG EEG4-BSR	Burst suppression ratio	%	
EEG: MOD	Measurement module existence	bool	
EEG: ACT	Measurement activity	bool	
EEG: MSN	Measurement type • 0: referential • 1: bipolar	bool	
EEG: MONTAGE	Montage (in use: 0...7)	bool	
EEG: HEAD	Headbox off	bool	
EEG: SSEP	SSEP cable off	bool	
EEG: CH1-LEADS	Channel 1 leads off	bool	
EEG: CH2-LEADS	Channel 2 leads off	bool	
EEG: CH3-LEADS	Channel 3 leads off	bool	
EEG: CH4-LEADS	Channel 4 leads off	bool	
EEG: CH1-ARTF	Channel 1 artefact	bool	
EEG: CH2-ARTF	Channel 2 artefact	bool	
EEG: CH3-ARTF	Channel 3 artefact	bool	
EEG: CH4-ARTF	Channel 4 artefact	bool	
EEG: CH1-NS	Channel 1 noise	bool	
EEG: CH2-NS	Channel 2 noise	bool	
EEG: CH3-NS	Channel 3 noise	bool	
EEG: CH4-NS	Channel 4 noise	bool	
EEG: EP	EP selection • 0: AEP • 1: SSEP	bool	
EEG: MSN	Measurement type • 0: referential • 1: bipolar	bool	

EEG-BIS

Label	Description	Unit	Range
EEG-BIS		?	
EEG-BIS SQI		?	
EEG-BIS EMG		?	
EEG-BIS SR		?	
EEG-BIS: MOD	Measurement module existence	bool	
EEG-BIS: ACT	Measurement activity	bool	

ENTROPY

Label	Description	Unit	Range
ENTROPY SE	State entropy	%	0 - 100
ENTROPY RE	Response entropy	%	0 - 100
ENTROPY BSR	Burst suppression rate	%	0 - 100
ENTROPY: MOD	Measurement module existence	bool	
ENTROPY: ACT	Measurement activity	bool	

EEG2

Label	Description	Unit	Range
EEG2 COMMON		?	
EEG2 CH1M		?	
EEG2 CH1P		?	
EEG2 CH2M		?	
EEG2 CH2P		?	
EEG2 CH3M		?	
EEG2 CH3P		?	
EEG2 CH4M		?	
EEG2 CH4P		?	
EEG2: MOD	Measurement module existence	bool	
EEG2: ACT	Measurement activity	bool	

GASEX

Label	Description	Unit	Range
GASEX VO2	Oxygen consumption	ml/min	
GASEX VCO2	Carbon dioxide consumption	ml/min	
GASEX EE	Energy expenditure	kcal/24h	
GASEX RQ	Respiratory quotient	None	
GASEX: MOD	Measurement module existence	bool	
GASEX: ACT	Measurement activity	bool	

FLOW-VOL2

Label	Description	Unit	Range
FLOW-VOL2 IPEEP	Intrinsic PEEP	cmH2O	
FLOW-VOL2 Pmean	Mean pressure	cmH2O	
FLOW-VOL2 RAW	Airway resistance	cmH2O	
FLOW-VOL2 MVINSP	Inspired minute volume	L/min	
FLOW-VOL2 EPEEP	Extrinsic PEEP	cmH2O	
FLOW-VOL2 MVESEX	Spontaneous expired minute volume	L/min	
FLOW-VOL2 IERATIO		None	
FLOW-VOL2 ISPTIME		None	
FLOW-VOL2 EXPTIME		None	
FLOW-VOL2 STCCOMP		None	
FLOW-VOL2 STCPPLAT		None	
FLOW-VOL2 STCPEEPE		None	
FLOW-VOL2 STCPEEPI		None	
FLOW-VOL2: MOD	Measurement module existence	bool	
FLOW-VOL2: ACT	Measurement activity	bool	

BAL-GAS

Label	Description	Unit	Range
BAL-GAS ET	Expiratory concentration	%	
BAL-GAS FI	Inspiratory concentration	%	
BAL-GAS: MOD	Measurement module existence	bool	
BAL-GAS: ACT	Measurement activity	bool	

TONO

Label	Description	Unit	Range
TONO PrCO2	PrCO2 concentration	kPa	
TONO P(r-Et)CO2	P(r-Et)CO2 gap	kPa	
TONO P(r-a)CO2	P(r-a)CO2 gap	kPa	
TONO PADELAY	PaCO2 delay	min	
TONO PHI	pHi value	None	
TONO PHIDELAY	pHi delay	min	
TONO PAMB	Ambient pressure	mmHg	
TONO CMPA	Research data	None	
TONO: MOD	Measurement module existence	bool	
TONO: ACT	Measurement activity	bool	
TONO: LEAK	Leak	bool	
TONO: VOLDR	volume dropped in catheter	bool	
TONO: TECHFAIL	Technical failure	bool	
TONO: UNFILL	Unable to fill catheter	bool	
TONO: OVER	PrCO2 over limit	bool	

AA2

Label	Description	Unit	Range
AA2 MAC-AGE-SUM		?	
AA2: MOD	Measurement module existence	bool	
AA2: ACT	Measurement activity	bool	

1.2.2 Waveform measures

Label	Description	Unit	Samps
ECG1		mV	300
ECG2		mV	300
ECG3		mV	300
INVP1	Invasive blood pressure	mmHg	100
INVP2	Invasive blood pressure	mmHg	100
INVP3	Invasive blood pressure	mmHg	100
INVP4	Invasive blood pressure	mmHg	100
INVP5	Invasive blood pressure	mmHg	100
INVP6	Invasive blood pressure	mmHg	100
PLETH	Plethysmograph: modulation	%	100
CO2	CO2 concentration	%	25
N2O	N2O concentration	%	25
AA_WAVE	Anaesthesia agent	%	25
AWP	Airway pressure	cmH2O	25
FLOW	Airway flow	l/min	25
VOL	Airway volume	?	25
RESP	ECG impedance	Ω	25
EEG1		uV	100
EEG2		uV	100
EEG3		uV	100
EEG4		uV	100
TONO_PRESS		?	?
SPI_LOOP_STATUS		?	?
ENT_100	EEG channel from ENT100 module.	uV	300
EEG_BIS		?	?

1.3 Headers

1.3.1 Datex Header

All requests use this header:

Byte No	Description
1	Start flag: FRAMECHAR
2	(start of header)
3	Total length = 0031h = 49d bytes (word r_len)

Continued on next page

Table 2 – continued from previous page

4	Reserved, set to zero (byte res1)
5	Ignored by monitor, set to zero (byte r_dri_level)
6	Reserved = 0000H (byte res2[2])
7	
8	Transmission time = 0x00000000, ignored by monitor when sending transmission request (dword r_time). However, time can be meaningful in outputted messages, which use the header of the same structure (dword r_time).
9	
10	
11	
12	Reserved = 00000000H (dword res3)
13	
14	
15	
16	Main type of record = DRI_MT_PHDB = 0 (r_maintype)
17	
18	Offset to the first subrecord = 0000H (sr_desc[0].offset)
19	
20	Type of first subrecord, DRI_PH_XMIT_REQ = 0 (sr_desc[0].sr_type)
21	Offset to the second subrecord = 0000H, calculated from the beginning of the data area after the header part. Value is not meaningful, since there is only one subrecord in the request (sr_desc[1].offset).
22	
23	"No more subrecords" (sr_desc[1].sr_type)
24	sr_desc[2].offset = 000, no meaning since only one subrecord transmitted. sr_desc[2].sr_type, no meaning
25	
26	
27	sr_desc[3].offset = 000, no meaning sr_desc[3].sr_type, no meaning
28	
29	
30	sr_desc[4].offset = 000, no meaning sr_desc[4].sr_type, no meaning
31	
32	
33	sr_desc[5].offset = 000, no meaning sr_desc[5].sr_type, no meaning
34	
35	
36	sr_desc[6].offset = 000, no meaning sr_desc[6].sr_type, no meaning
37	
38	
39	sr_desc[7].offset = 000, no meaning sr_desc[7].sr_type = 0, no meaning
40	
41	

class pycollect.headers.**DatexHeaderRequest**(**args*, ***kwargs*)

Bases: *pycollect.headers.HeaderHandler*

Header for request transfer.

This header is the combination of *Datex Header* and the follow header:

Byte No	Description
42	Request current values of physiological database = DRI_PH_DISPL (field phdb_rcrd_type of struct phdb_req)
43	Transmission interval in seconds = 00A, i.e., send current values of physiological database every 10 seconds (field tx_interval of struct phdb_req)
44	
45	reserved[0] of struct phdb_req, must be zeroed
46	
47	reserved[1] of struct phdb_req, must be zeroed
48	
49	reserved[2] of struct phdb_req, must be zeroed
50	
51	Checksum
52	End flag: FRAMECHAR

class pycollect.headers.**DatexHeaderResponse**(**args*, ***kwargs*)
Bases: *pycollect.headers.HeaderHandler*

The data transmitted from monitor is writed in this header.

This header is the combination of *Datex Header* and the follow header, is used for store the waveforms inputs.

Byte No	Description
42-1491	Data
1492	Checksum
1493	End flag: FRAMECHAR

class pycollect.headers.**DatexHeaderWaveRequest**(**args*, ***kwargs*)
Bases: *pycollect.headers.HeaderHandler*

Header for request wave transfer.

This header is the combination of *Datex Header* and the follow header, is user for create the recuest of waveforms.

Byte No	Description
42	Request type: one of WF_REQ_CONT_START, WF_REQ_CONT_STOP or WF_REQ_TIMED_START.
43	
44	Duration of snapshot
45	
46-53	An array of the requested waveform subrecords. There is room for up to 8 waveforms, but the monitor sends only the waveforms that fit within the 600 samples/s limitation and ignores the rest. The type array must be terminated using the DRI_EOL_SUBR_LIST constant (0xFF), unless there are 8 waveforms is the request.
54-73	Reserved
74	Checksum
75	End flag: FRAMECHAR

class pycollect.headers.**HeaderHandler**(*data=None*, *init=None*, *size=None*)
Bases: object

Header Handler.

Establish a way to read and write GE protocol headers.

__getitem__(*element*)
Return value from header.

Parameters **element** (*str*) – Define de index and the format of requested value.

Returns Target element, could be of many types due to the nested suported format.

Return type int, list, *HeaderHandler*

Note:

recursive format: header['basic:ecg:hr']

reverse the byte array before to read: header['-basic:ecg:hr']

return the byte array without convert to integer: header['basic:ecg:hr,']

return the boolean for the bit *4*: header['basic:ecg:hr:4]

return the integer generated with the bits *0* to *5*: header['basic:ecg:hr:0-5]

__init__(*data=None*, *init=None*, *size=None*)

Parameters

- **data** (*array*) – Load an array to build the header.
- **init** (*OrderedDict*) – Initialize the the data header with this values.
- **size** (*str*) – Define the size in bytes of the current header.

__length__(*header*)
Calculate the bytes length of the current header.

Parameters **header** (*dict*) – Target header in dict format for calculate the size.

Returns Number of bytes used by the header.

Return type int

array()
Return the sorted byte array with the correct size.

If one element is defined as 2 bytes length, this must be splitted and completed (if necessary) with an empty byte.

Returns Single array with data header values, linke in C, C++, C#.

Return type list

load(*data*)
Load the *data* into the current header values.

Parameters **data** (*array*) – Load an array to build the header.

request()
Generate the final header.

Returns The main data with *start-flag*, *checksum* and *end-flag*.

Return type list

set(*element*, *value*)
Modify the value of one header element.

Parameters

- **element** (*str*) – Header element.
- **value** (*str*) – New value for element.

to32bits(*value*)
Convert a value into a 32 bits array.

Parameters **value** (*array, int*) – A length 4 array or integer value, if integer then it will be converted to [0, 0, 0, value].

Returns List of 32 bits that represents the input value.

Return type list

class pycollect.headers.**PhysiologicalData**(**args*, ***kwargs*)
Bases: *pycollect.headers.HeaderHandler*

The data transmitted from monitor is writed in this header.

This header is the combination of *Datex Header* and the follow header, is used for store the requested subrecords.

Byte No	Description
0-4	Time
5-274	*basic* subclass
275-544	*ext1* subclass
545-814	*ext2* subclass
815-1084	*ext3* subclass
1085	Marker, contains the number of latest entered mark.
1086	contains control information for patient data management functions, used internally by the monitor.
1087-1088	**The last word of the subrecord, contains** • The physiological data record class. • The current D-O Record Interface level. • The subrecord type.

1.4 GE Decode

Synchronize in decoder with a buffer reference, decoder *GEDecode* will attempt to process new data added to the buffer.

```
decoder = GEDecode(device.BUFFER)
decoder.process(True)
```

There are a set of methods for clear correctly the stored buffer and recollected data.

```
device.clear_buffer()  # clear input buffer.
decoder.clear_buffer()  # clear decoded data, breaks the synchrony.
decoder.clear_data()  # clear recollected data.
```

class pycollect.decode.**FormatSubrecord**(*date_*, *header*)
Bases: object

Parse raw data into Pandas DataFrames.

__init__(*date_*, *header*)

Parameters

- **date** (*Datetime object*) – Datetime of the current set.
- **header** (*PhysiologicalData object*) – Header with raw data.

check_module(*label*, *replace=None*)

Check the status module from determinate group.

Parameters

- **label** (*str*) – Group for check their respective module.
- **replace** (*dict*) – Some groups are indexed for multiples modules, in that case, the indexes must be removed.

Returns

- *bool* – Module exist.
- *bool* – Module is active.

format(*active*, *filters=None*)

Process a subrecord (their raw header), aply shifts and get references.

Parameters

- **active** (*array*) – List of groups to parse.
- **filters** (*array, optional*) – A sub list of desired subrecords.

Returns DataFrame with single row that contains all measures with label as headers.

Return type DataFrame

get_short(*key*)

From index in header, convert data to integer, validates and return it.

Parameters **key** (*Header index*) – Location if header of target value.

Returns Int if valus is a signed 16 bits, None if the value is over range and bytearray in other cases.

Return type int, None, bytearray

module_status()

Return the list of present and active modules.

Based in the status bits is possible determine the state of the respective module.

Returns

- *list* – List of groups with modules availables.
- *list* – List of groups with modules availables and actives.
- *list* – List of measures availables and actives.

class pycollect.decode.**GEDecode**(*buffer*, *filter_subrecords=None*, *filter_waveforms=None*)

Bases: object

Decode raw data into Pandas DataFrames.

There are two main objects:

- *DATA_SUBRECORD*: Pandas DataFrame with the subrecords data.

- ***DATA_WAVE*: Dictionary with waveform name as key with the** Pandas DataFrame with the waveform data.

__continuous_processing__(*delay=0.25*)
Continuous processing.

Parameters **delay** (*integer in seconds*) – Delay in seconds between each attempted processing.

__create_framelist__(*byte*)
Search for a complete and validated header.

Parameters **byte** (*byte*) – Byte readed from serial raw data.

__create_recordlist__()
Complete the raw header with the correct/full size.

Returns Full sized DatexHeaderResponse.

Return type list

__init__(*buffer*, *filter_subrecords=None*, *filter_waveforms=None*)

Parameters

- **buffer** (*array reference*) – Reference with raw data, the processer will attempt to decoded new data added to this object.
- **filter_subrecords** (*array, optional*) – Sublist with desired subrecords, if empty then will process all available measures.
- **filter_waveforms** (*array, optional*) – List of desired waveforms, if emty then will process all requested waveforms.

__processing__()
Process the input buffer one byte at a time.

Process the input buffer until a header is completed.

clear_buffer()
Clear the processed buffer.

This action brake the synchrony with GEDevice.

clear_data()
Clear the processed data.

process(*flag=True*, *delay=1*)
Enable or disable the continuous data processing.

Parameters

- **flag** (*bool, True*) – Enable or disable the continuous data processing.
- **delay** (*integer in seconds*) – Delay in seconds between each process attempt.

read_shorts(*buffer*)
Convert an array if bytes from signed 16 bits to integer.

Parameters **buffer** (*array*) – Array with raw bytes, It will be processed in pairs (for complete 16 bits).

read_subrecords(*record_list*)
Update the DataFrame of subrecords with new decoded data.

Parameters **record_list** (*array*) – Raw DatexHeaderResponse.

read_waveforms (*record_list*, *ignore_missging=False*)
Update the dictionary of waveforms with new decoded data.

Parameters **record_list** (*array*) – List of raw headers.

save_as_csv (*filename*)
Save the decoded data into a set of CSV files.

Parameters **filename** (*str*) – Absolute or realtive path for CSV file.

Returns A list with filenames generated.

Return type list

save_as_edf (*filename*, *edf_header=None*, *annotations=None*)
Save the decoded data into a set of EDF+ files.

Parameters

- **filename** (*str*) – Absolute or realtive path for EDF+ file.
- **edf_header** (*dict*) – Declare the EDF+ patient header.

Returns A list with filenames genrated.

Return type list

save_as_raw (*filename*)
Save the decoded data into a RAW file.

Parameters **filename** (*str*) – Absolute or realtive path for RAW file.

Returns A list with filenames generated.

Return type list

set_edf_header (**header*)
Set the EDF+ patient header.

Parameters

- **admincode** (*str*) – Sets the admincode.
- **birthdate** (*date object from datetime*) – Sets the birthdate.
- **equipment** (*str*) – Describes the measurement equpipment.
- **gender** (*int*) – Sets the gender, 1 is male, 0 is female.
- **patientcode** (*str*) – Sets the patient code.
- **patientname** (*str*) – Sets the patient name.
- **patient_additional** (*str*) – Sets the additional patient info.
- **recording_additional** (*str*) – Sets the additional recordinginfo.
- **startdate** (*datetime object*) – Sets the recording start Time.
- **technician** (*str*) – Sets the technicians name.

1.5 GE Device

Connect and request Subrecord and Waveform:

```
device = GEDevice()
device.connect('/dev/ttyUSB0')

device.request(subtype=device.DISPL, waveform_set=['PLETH', 'ECG1'])

device.collect(True)
```

To clear the buffer input:

```
device.clear_buffer()
```

To stop the data collecting:

```
device.collect(False)
```

class pycollect.device.**FakeDevice**(*raw_file*)

Bases: object

Debugger class for simulate the input data.

__init__(*raw_file*)

Establish the connection and handle the data input from monitor.

Parameters **raw_file** (*str, optional*) – Input file with raw data.

__read_raw__()

Return generator with the input raw data.

Returns Generator for single bytes.

Return type Generator

close()

Close the serial port.

connect(*port*, *timeout*)

Establish the connection and debug an unrelated serial device.

Parameters

- **port** (*str*) – Serial port.
- **timeout** (*integer*) – Timeout for serial comunication.

read(*size*)

Return the input raw file one byte at time.

Parameters **size** (*int*) – Buffer input size to read.

Returns Array with raw data.

Return type list

writable()

Check if serial port is writable..

Returns Port writable.

Return type bool

write(*data*)

Write on serial port if available.

Parameters **data** (*bytes*) – Desired object to write in serial port.

class pycollect.device.**GEDevice**(*raw_file=None*)
Bases: object

Establish the connection and handle the data input from monitor.

__init__(*raw_file=None*)
Establish the connection and handle the data input from monitor.

Parameters **raw_file** (*str, optional*) – Read from local file, no serial.

clear_buffer()
Crear the data buffer, not the serial input buffer.

close()
Close serial port

collect(*flag=True*, *size=4096*)
Enable or disable the continuous data reading.

Parameters

- **flag** (*bool, True*) – Enable or disable the continuous data reading.
- **size** (*integer, 2**12*) – Input buffer size.

connect(*port*, *timeout=1*)
Establish the connection with the CARESCAPE Monitor Bx50.

- **Baudrate:** 19200
- **Parity:** even
- **Stop bits:** 1
- **bytesize:** 8 bits
- **rtscts:** True

Parameters

- **port** (*str*) – Serial port address.
- **timeout** (*int, in secods*) – Serial port read and write timeout.

create_waveform_set(*waveform_set*)
Generate Waveform set.

Parameters **waveform_set** (*array*) – List of waveform types.

Returns A list of integer that represent the waveforms desired.

Return type list

on_connection_loss()
Overwritable method.

read(*size=4096*)
Write into the data buffer the bytes readed from serial port.

Parameters **size** (*integer, 2**12*) – Input buffer size.

request(*subtype=None*, *waveform_set=None*, *interval=1*)
Create and send a data request of Subrecord or/and Waveform type.

Parameters

- **subtype** (*subrecord type object*) – *DISPL*, *TREND_10S* or *TREND_60S*

- **waveform_set** (*array*) – List of waveform types.
- **interval** (*int, in seconds*) – If subtype is *DISPL* interval is accepted and must be an integer equal or greater to 1.

request_multiple_wave_transfer(*wave_set*, *transmission_type*)
Create and send header of Waveform type.

Parameters

- **wave_set** (*array*) – List of constants that represents each waveform type requested, if list lenghth is less than 8, the last integer must be 255.
- **transmission_type** (*transmission type object*) – *WF_REQ_CONT_START* for start start transmission, *WF_REQ_CONT_STOP* for stop it and *WF_REQ_TIMED_START* for request a timed transmission.

request_transfer(*subtype*, *interval*)
Create and send header of Subrecord type.

Parameters

- **subtype** (*subrecord type object*) – *DISPL*, *TREND_10S* or *TREND_60S*
- **interval** (*int, in seconds*) – If subtype is *DISPL* interval is accepted and must be an integer equal or greater to 5

stop()
Send the request for stop the Subrecords and Waveform transmission.

stop_transfer()
Send the request for stop the Subrecords transmission.

stop_wave_transfer()
Send the request for stop the Waveform transmission.

write_buffer(*data*)
Check if device is writeble and send the data.

Parameters **data** (*bytes array*) – Bytes with the header that bust sent to the device.

class pycollect.edfwriter.**EDF**(*filename*)
Bases: object

Create an EDF+ file format.

__init__(*filename*)
Create an EDF+ file format.

Parameters **filename** (*str*) – Filename for new EDF+ file.

add_channel(*channel*)
Add new chanel to current EDF+.

Parameters **chanel** (*EDFChannel object*) –

save()
Save as EDF+ file.

set_header(***kwargs*)
Sets the file header.

Parameters

- **admincode** (*str*) – Sets the admincode.

- **birthdate** (*date object from datetime*) – Sets the birthdate.
- **equipment** (*str*) – Describes the measurement equpipment.
- **gender** (*int*) – Sets the gender, 1 is male, 0 is female.
- **patientcode** (*str*) – Sets the patient code.
- **patientname** (*str*) – Sets the patient name.
- **patient_additional** (*str*) – Sets the additional patient info.
- **recording_additional** (*str*) – Sets the additional recordinginfo.
- **startdate** (*datetime object*) – Sets the recording start Time.
- **technician** (*str*) – Sets the technicians name.

write_annotation (*onset*, *description*, *duration=-1*)

Writes an annotation/event to the file.

Parameters

- **onset** (*int*) – Second, where happened the annotation
- **duration** (*int, optional*) – Duration of event anotated
- **description** (*str*) – Description of event

class pycollect.edfwriter.**EDFChannel** (*data*, ***kwargs*)

Bases: object

New EDF+ channel.

__init__ (*data*, ***kwargs*)

New channel.

Parameters

- **data** (*array, required*) –
- **label** (*str, recommended*) – channel label (string, <= 16 characters, must be unique
- **dimension** (*str, recommended*) – physical dimension (e.g., mV) (string, <= 8 characters)
- **sample_rate** (*int, required*) – sample frequency in hertz
- **physical_max** (*float, required*) – maximum physical value
- **physical_min** (*float, required*) – minimum physical value
- **digital_max** (*int, optional*) – maximum digital value (int, -2**15 <= x < 2**15)
- **digital_min** (*int, optional*) – minimum digital value (int, -2**15 <= x < 2**15)
- **transducer** (*str, optional*) – sets the transducer used in this channel
- **prefilter** (*str, optional*) – sets the prefilter used in this chanel (“HP:0.1Hz”, “LP:75Hz N:50Hz”, etc.)

1.6 Examples

```
import numpy
from matplotlib import pyplot
from pycollect import GEDevice, GEDecode, database, measures
```

1.6.1 Read RAW file

The **PyCollect** library include a set of prerecorded database for debug purposes.

```
for raw in database.RAWS_ABSPATH:
    print("RAW file path: {}".format(raw))
```

```
RAW file path: /usr/lib/python3.6/site-packages/pycollect-1.0-py3.6.egg/pycollect/
↪database/db02.raw
RAW file path: /usr/lib/python3.6/site-packages/pycollect-1.0-py3.6.egg/pycollect/
↪database/db00.raw
RAW file path: /usr/lib/python3.6/site-packages/pycollect-1.0-py3.6.egg/pycollect/
↪database/db01.raw
```

```
# Read the file
file = open(database.RAWS_ABSPATH[1], 'rb')
data = file.read()
data = data[:int(4e4)]
file.close()

decoder = GEDecode(data)

# Buffer and modules
print("Buffer size: {} bytes".format(len(decoder.BUFFER)))
print("Modules detected: {}".format(decoder.MODULES))
print("Modules active: {}".format(decoder.MODULES_ACTIVE))

# Available labels
print("Trends: {}".format(decoder.DATA_SUBRECORD.columns.tolist()))
print("Waves: {}".format(list(decoder.DATA_WAVE.keys())))

pyplot.figure(figsize=(12,6), dpi=200)
pyplot.plot(decoder.DATA_SUBRECORD['ENTROPY RE'])
pyplot.legend(['ENTROPY RE'])

pyplot.figure(figsize=(12,6), dpi=200)
pyplot.plot(decoder.DATA_WAVE['ENT_100']['values'][:500])
pyplot.legend(['ENT_100'])

pyplot.figure(figsize=(12,6), dpi=200)
pyplot.plot(decoder.DATA_WAVE['PLETH']['values'][:500])
pyplot.legend(['PLETH'])
```

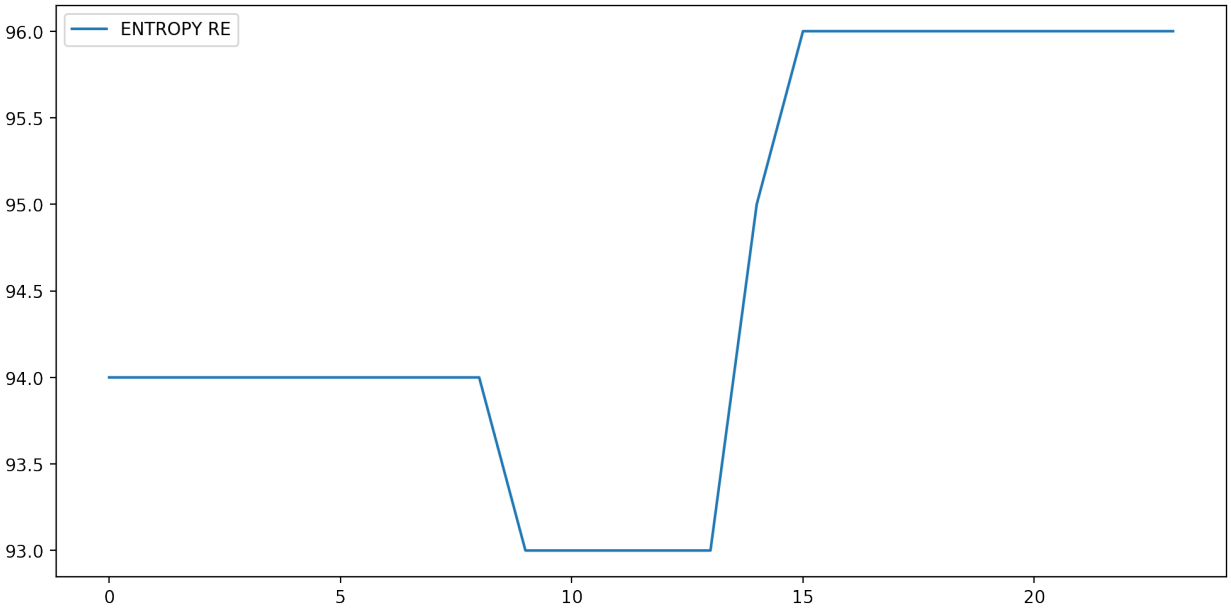
```
Buffer size: 40000 bytes
Modules detected: ['INV-BP (p1)', 'INV-BP (p2)', 'ECG', 'NIBP', 'SpO2', 'NMT', 'ECG-
↪EXTRA', 'ECG-ARRH', 'ECG-12', 'NMT2', 'ENTROPY']
Modules active: ['ECG', 'NIBP', 'SpO2', 'NMT', 'ECG-EXTRA', 'ECG-ARRH', 'ECG-12',
↪'NMT2', 'ENTROPY']
```

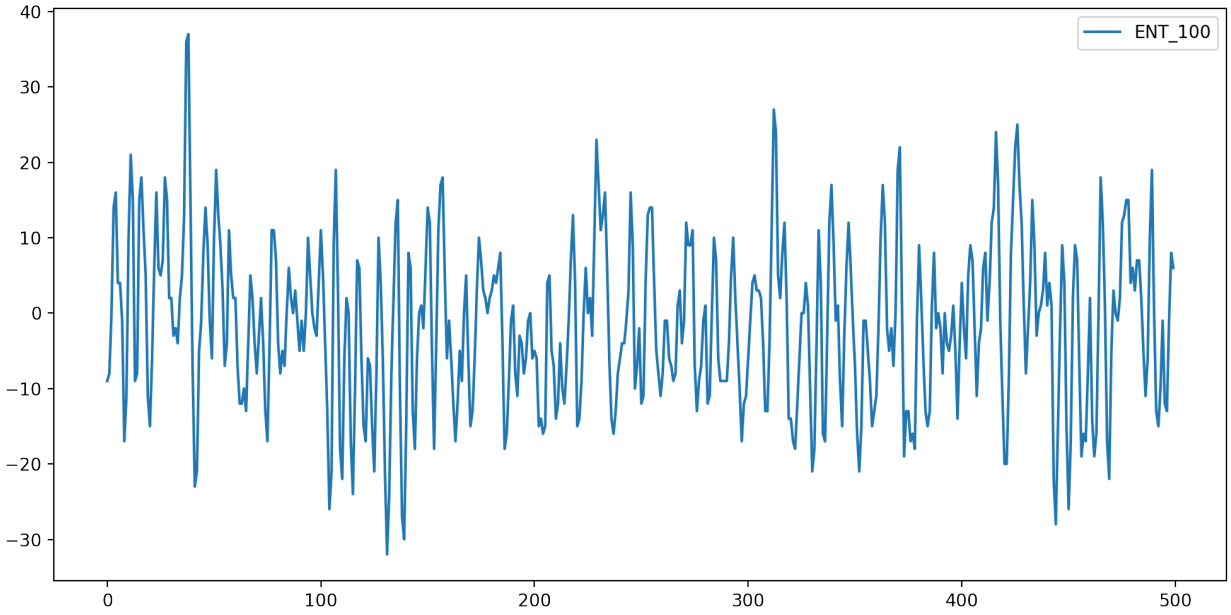
(continues on next page)

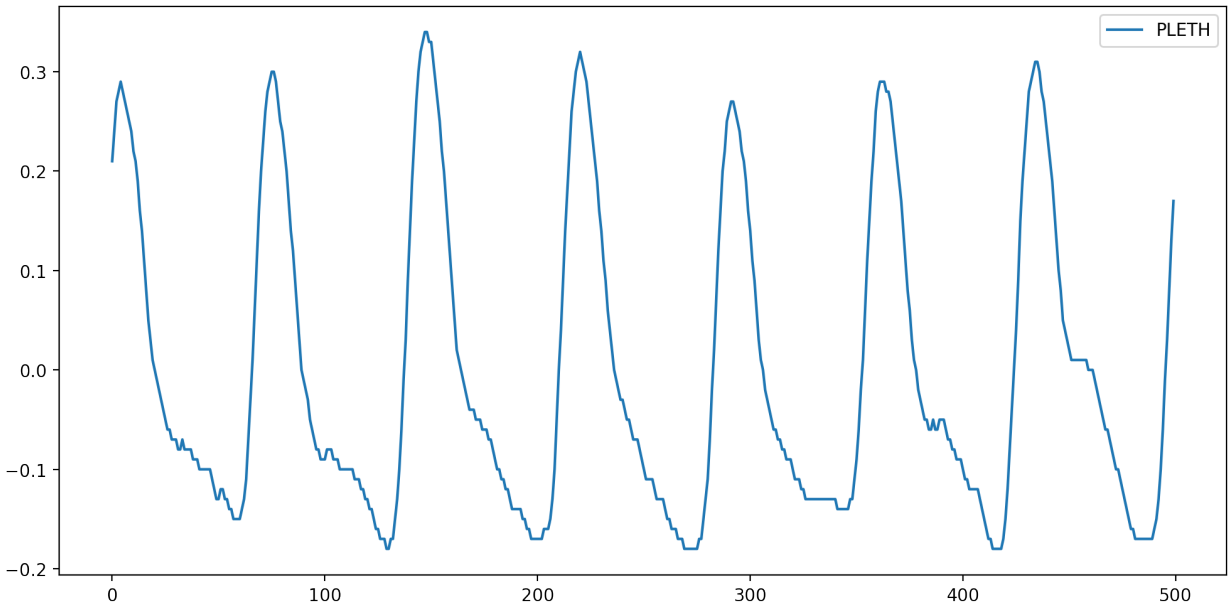
(continued from previous page)

```
Trends: ['AA', 'AA ET', 'AA FI', 'AA MAC-SUM', 'AA2 MAC-AGE-SUM', 'AA: CLBR', 'AA: MNS
↪', 'BAL-GAS ET', 'BAL-GAS FI', 'CO-WEDGE CO', 'CO-WEDGE CO-AGE', 'CO-WEDGE PCWP',
↪'CO-WEDGE PCWP-AGE', 'CO-WEDGE REF', 'CO-WEDGE TEMP', 'CO2 ET', 'CO2 FI', 'CO2 LBL',
↪ 'CO2 PAMB', 'CO2 RR', 'CO2: ALK', 'CO2: AP', 'CO2: CLBR', 'CO2: CS', 'CO2: MNS',
↪'CO2: OC', 'CO2: ZS', 'ECG HR', 'ECG HR-SRC', 'ECG IMP-RR', 'ECG LEAD-CH1', 'ECG␣
↪LEAD-CH2', 'ECG LEAD-CH3', 'ECG ST1', 'ECG ST2', 'ECG ST3', 'ECG-12 LEAD-CH1', 'ECG-
↪12 LEAD-CH2', 'ECG-12 LEAD-CH3', 'ECG-12 STAVF', 'ECG-12 STAVL', 'ECG-12 STAVR',
↪'ECG-12 STI', 'ECG-12 STII', 'ECG-12 STIII', 'ECG-12 STV1', 'ECG-12 STV2', 'ECG-12␣
↪STV3', 'ECG-12 STV4', 'ECG-12 STV5', 'ECG-12 STV6', 'ECG-ARRH HR', 'ECG-ARRH PVC',
↪'ECG-ARRH RR', 'ECG-EXTRA: HR', 'ECG-EXTRA: HR-MAX', 'ECG-EXTRA: HR-MIN', 'ECG: AR',
↪ 'ECG: ASY', 'ECG: CH1', 'ECG: CH2', 'ECG: CH3', 'ECG: LRN', 'ECG: NS', 'ECG: PCR',
↪'EEG EEG1-ALPHA', 'EEG EEG1-AMPL', 'EEG EEG1-BETA', 'EEG EEG1-BSR', 'EEG EEG1-DELTA
↪', 'EEG EEG1-MNF', 'EEG EEG1-SFR', 'EEG EEG1-THETA', 'EEG EEG2-ALPHA', 'EEG EEG2-
↪AMPL', 'EEG EEG2-BETA', 'EEG EEG2-BSR', 'EEG EEG2-DELTA', 'EEG EEG2-MF', 'EEG EEG2-
↪SFR', 'EEG EEG2-THETA', 'EEG EEG3 BSR', 'EEG EEG3-ALPHA', 'EEG EEG3-AMPL', 'EEG␣
↪EEG3-BETA', 'EEG EEG3-DELTA', 'EEG EEG3-MF', 'EEG EEG3-SEF', 'EEG EEG3-THETA', 'EEG␣
↪EEG4-ALPHA', 'EEG EEG4-AMPL', 'EEG EEG4-BETA', 'EEG EEG4-BSR', 'EEG EEG4-DELTA',
↪'EEG EEG4-MF', 'EEG EEG4-SEF', 'EEG EEG4-THETA', 'EEG FEMG', 'EEG-BIS', 'EEG-BIS EMG
↪', 'EEG-BIS SQI', 'EEG-BIS SR', 'EEG2 CH1M', 'EEG2 CH1P', 'EEG2 CH2M', 'EEG2 CH2P',
↪'EEG2 CH3M', 'EEG2 CH3P', 'EEG2 CH4M', 'EEG2 CH4P', 'EEG2 COMMON', 'EEG: CH1-ARTF',
↪'EEG: CH1-LEADS', 'EEG: CH1-NS', 'EEG: CH2-ARTF', 'EEG: CH2-LEADS', 'EEG: CH2-NS',
↪'EEG: CH3-ARTF', 'EEG: CH3-LEADS', 'EEG: CH3-NS', 'EEG: CH4-ARTF', 'EEG: CH4-LEADS',
↪ 'EEG: CH4-NS', 'EEG: EP', 'EEG: HEAD', 'EEG: MONTAGE', 'EEG: MSN', 'EEG: SSEP',
↪'ENTROPY BSR', 'ENTROPY RE', 'ENTROPY SE', 'FLOW-VOL COMP', 'FLOW-VOL MV-EXP',
↪'FLOW-VOL PEEP', 'FLOW-VOL PPEAK', 'FLOW-VOL PPLAT', 'FLOW-VOL RR', 'FLOW-VOL TV-EXP
↪', 'FLOW-VOL TV-INSP', 'FLOW-VOL2 EPEEP', 'FLOW-VOL2 EXPTIME', 'FLOW-VOL2 IERATIO',
↪'FLOW-VOL2 IPEEP', 'FLOW-VOL2 ISPTIME', 'FLOW-VOL2 MVESEX', 'FLOW-VOL2 MVINSP',
↪'FLOW-VOL2 Pmean', 'FLOW-VOL2 RAW', 'FLOW-VOL2 STCCOMP', 'FLOW-VOL2 STCPEEPE',
↪'FLOW-VOL2 STCPEEPI', 'FLOW-VOL2 STCPPLAT', 'FLOW-VOL: CLBR', 'FLOW-VOL: DIS',
↪'FLOW-VOL: LK', 'FLOW-VOL: MSR', 'FLOW-VOL: OBS', 'FLOW-VOL: ZR', 'GASEX EE',
↪'GASEX RQ', 'GASEX VCO2', 'GASEX VO2', 'INV-BP DIA (p1)', 'INV-BP DIA (p2)', 'INV-
↪BP DIA (p3)', 'INV-BP DIA (p4)', 'INV-BP DIA (p5)', 'INV-BP DIA (p6)', 'INV-BP HR␣
↪(p1)', 'INV-BP HR (p2)', 'INV-BP HR (p3)', 'INV-BP HR (p4)', 'INV-BP HR (p5)', 'INV-
↪BP HR (p6)', 'INV-BP LBL (p1)', 'INV-BP LBL (p2)', 'INV-BP LBL (p3)', 'INV-BP LBL␣
↪(p4)', 'INV-BP LBL (p5)', 'INV-BP LBL (p6)', 'INV-BP MEAN (p1)', 'INV-BP MEAN (p2)',
↪ 'INV-BP MEAN (p3)', 'INV-BP MEAN (p4)', 'INV-BP MEAN (p5)', 'INV-BP MEAN (p6)',
↪'INV-BP SYS (p1)', 'INV-BP SYS (p2)', 'INV-BP SYS (p3)', 'INV-BP SYS (p4)', 'INV-BP␣
↪SYS (p5)', 'INV-BP SYS (p6)', 'INV-BP: ZR (p1)', 'INV-BP: ZR (p2)', 'INV-BP: ZR (p3)
↪', 'INV-BP: ZR (p4)', 'INV-BP: ZR (p5)', 'INV-BP: ZR (p6)', 'N2O ET', 'N2O FI',
↪'N2O: CLBR', 'N2O: MNS', 'NIBP CUFF', 'NIBP DIA', 'NIBP HR', 'NIBP MEAN', 'NIBP SYS
↪', 'NIBP: AUTO', 'NIBP: CLBR', 'NIBP: MSR', 'NIBP: OLD', 'NIBP: STASIS', 'NIBP: STAT
↪', 'NMT PTC-COUNT', 'NMT PTC-DB-COUNT', 'NMT PTC-ST-COUNT', 'NMT PTC-STIM', 'NMT␣
↪PTC-TOF-COUNT', 'NMT STM', 'NMT T1', 'NMT TIME', 'NMT TRATIO', 'NMT2 T1', 'NMT2 T2',
↪ 'NMT2 T3', 'NMT2 T4', 'NMT: CLBR', 'NMT: SUP', 'O2 ET', 'O2 FI', 'SpO2', 'SpO2 IR-
↪AMP', 'SpO2 LBL', 'SpO2 PR', 'SpO2 [SO2|SaO2|SvO2]', 'SvO2', 'TEMP (t1)', 'TEMP (t2)
↪', 'TEMP (t3)', 'TEMP (t4)', 'TEMP LBL (t1)', 'TEMP LBL (t2)', 'TEMP LBL (t3)',
↪'TEMP LBL (t4)', 'TONO CMPA', 'TONO P(r-Et)CO2', 'TONO P(r-a)CO2', 'TONO PADELAY',
↪'TONO PAMB', 'TONO PHI', 'TONO PHIDELAY', 'TONO PrCO2', 'TONO: LEAK', 'TONO: OVER',
↪'TONO: TECHFAIL', 'TONO: UNFILL', 'TONO: VOLDR', 'datetime']
Waves: ['PLETH', 'ENT_100']
```

```
<matplotlib.legend.Legend at 0x7f4442e442b0>
```


ENTROPY RE
96.0
95.5
95.0
94.5
94.0
93.5
93.0
0
5
10
15
20


ENT_100
40
30
20
10
0
−10
−20
−30
0
100
200
300
400
500



1.6.2 Read from monitor

```
device = GEDevice(database.RAWS_ABSPATH[0])
decoder = GEDecode(device.BUFFER)
# device.connect('/dev/ttyUSB0')
```

The monitor will send all `subrecords` data but the waves need an explicit request.

```
default = device.DEFAULT_REQUEST
print("Default data requested: {}".format(default))

# Activate the trends transmission
device.request(subtype=device.DISPL)

# Activate the waves transmission
device.request(waveform_set=['ENT_100', 'PLETH'])
```

```
Default data requested: ['date', 'ENTROPY RE', 'ENTROPY SE', 'ENTROPY BSR', 'NIBP SYS
↪', 'NIBP DIA', 'NIBP MEAN', 'ECG IMP-RR', 'ECG HR', 'TEMP (t1)', 'CO2 FI', 'CO2 ET',
↪ 'ECG1', 'ENT_100', 'PLETH']
```

```
# Start the asynchronous data collecting and process
device.collect(True)
decoder.process(True)

import time
time.sleep(20)
```

```
# Buffer and modules
print("Buffer size: {} bytes".format(len(decoder.BUFFER)))
print("Modules detected: {}".format(decoder.MODULES))
print("Modules active: {}".format(decoder.MODULES_ACTIVE))
```

(continues on next page)

(continued from previous page)

```
# Available labels
print("Trends: {}".format(decoder.DATA_SUBRECORD.columns.tolist()))
print("Waves: {}".format(list(decoder.DATA_WAVE.keys())))
```

```
Buffer size: 17432 bytes
Modules detected: ['INV-BP (p1)', 'INV-BP (p2)', 'ECG', 'NIBP', 'SpO2', 'CO2', 'O2',
↪'N2O', 'AA', 'FLOW-VOL', 'ECG-EXTRA', 'ECG-ARRH', 'ECG-12', 'ENTROPY', 'FLOW-VOL2',
↪'BAL-GAS', 'AA2']
Modules active: ['ECG', 'NIBP', 'SpO2', 'CO2', 'O2', 'N2O', 'AA', 'FLOW-VOL', 'ECG-
↪EXTRA', 'ECG-ARRH', 'ECG-12', 'ENTROPY', 'FLOW-VOL2', 'BAL-GAS', 'AA2']
Trends: ['AA', 'AA ET', 'AA FI', 'AA MAC-SUM', 'AA2 MAC-AGE-SUM', 'AA: CLBR', 'AA: MNS
↪', 'BAL-GAS ET', 'BAL-GAS FI', 'CO-WEDGE CO', 'CO-WEDGE CO-AGE', 'CO-WEDGE PCWP',
↪'CO-WEDGE PCWP-AGE', 'CO-WEDGE REF', 'CO-WEDGE TEMP', 'CO2 ET', 'CO2 FI', 'CO2 LBL',
↪ 'CO2 PAMB', 'CO2 RR', 'CO2: ALK', 'CO2: AP', 'CO2: CLBR', 'CO2: CS', 'CO2: MNS',
↪'CO2: OC', 'CO2: ZS', 'ECG HR', 'ECG HR-SRC', 'ECG IMP-RR', 'ECG LEAD-CH1', 'ECG␣
↪LEAD-CH2', 'ECG LEAD-CH3', 'ECG ST1', 'ECG ST2', 'ECG ST3', 'ECG-12 LEAD-CH1', 'ECG-
↪12 LEAD-CH2', 'ECG-12 LEAD-CH3', 'ECG-12 STAVF', 'ECG-12 STAVL', 'ECG-12 STAVR',
↪'ECG-12 STI', 'ECG-12 STII', 'ECG-12 STIII', 'ECG-12 STV1', 'ECG-12 STV2', 'ECG-12␣
↪STV3', 'ECG-12 STV4', 'ECG-12 STV5', 'ECG-12 STV6', 'ECG-ARRH HR', 'ECG-ARRH PVC',
↪'ECG-ARRH RR', 'ECG-EXTRA: HR', 'ECG-EXTRA: HR-MAX', 'ECG-EXTRA: HR-MIN', 'ECG: AR',
↪ 'ECG: ASY', 'ECG: CH1', 'ECG: CH2', 'ECG: CH3', 'ECG: LRN', 'ECG: NS', 'ECG: PCR',
↪'EEG EEG1-ALPHA', 'EEG EEG1-AMPL', 'EEG EEG1-BETA', 'EEG EEG1-BSR', 'EEG EEG1-DELTA
↪', 'EEG EEG1-MNF', 'EEG EEG1-SFR', 'EEG EEG1-THETA', 'EEG EEG2-ALPHA', 'EEG EEG2-
↪AMPL', 'EEG EEG2-BETA', 'EEG EEG2-BSR', 'EEG EEG2-DELTA', 'EEG EEG2-MF', 'EEG EEG2-
↪SFR', 'EEG EEG2-THETA', 'EEG EEG3 BSR', 'EEG EEG3-ALPHA', 'EEG EEG3-AMPL', 'EEG␣
↪EEG3-BETA', 'EEG EEG3-DELTA', 'EEG EEG3-MF', 'EEG EEG3-SEF', 'EEG EEG3-THETA', 'EEG␣
↪EEG4-ALPHA', 'EEG EEG4-AMPL', 'EEG EEG4-BETA', 'EEG EEG4-BSR', 'EEG EEG4-DELTA',
↪'EEG EEG4-MF', 'EEG EEG4-SEF', 'EEG EEG4-THETA', 'EEG FEMG', 'EEG-BIS', 'EEG-BIS EMG
↪', 'EEG-BIS SQI', 'EEG-BIS SR', 'EEG2 CH1M', 'EEG2 CH1P', 'EEG2 CH2M', 'EEG2 CH2P',
↪'EEG2 CH3M', 'EEG2 CH3P', 'EEG2 CH4M', 'EEG2 CH4P', 'EEG2 COMMON', 'EEG: CH1-ARTF',
↪'EEG: CH1-LEADS', 'EEG: CH1-NS', 'EEG: CH2-ARTF', 'EEG: CH2-LEADS', 'EEG: CH2-NS',
↪'EEG: CH3-ARTF', 'EEG: CH3-LEADS', 'EEG: CH3-NS', 'EEG: CH4-ARTF', 'EEG: CH4-LEADS',
↪ 'EEG: CH4-NS', 'EEG: EP', 'EEG: HEAD', 'EEG: MONTAGE', 'EEG: MSN', 'EEG: SSEP',
↪'ENTROPY BSR', 'ENTROPY RE', 'ENTROPY SE', 'FLOW-VOL COMP', 'FLOW-VOL MV-EXP',
↪'FLOW-VOL PEEP', 'FLOW-VOL PPEAK', 'FLOW-VOL PPLAT', 'FLOW-VOL RR', 'FLOW-VOL TV-EXP
↪', 'FLOW-VOL TV-INSP', 'FLOW-VOL2 EPEEP', 'FLOW-VOL2 EXPTIME', 'FLOW-VOL2 IERATIO',
↪'FLOW-VOL2 IPEEP', 'FLOW-VOL2 ISPTIME', 'FLOW-VOL2 MVESEX', 'FLOW-VOL2 MVINSP',
↪'FLOW-VOL2 Pmean', 'FLOW-VOL2 RAW', 'FLOW-VOL2 STCCOMP', 'FLOW-VOL2 STCPEEPE',
↪'FLOW-VOL2 STCPEEPI', 'FLOW-VOL2 STCPPLAT', 'FLOW-VOL: CLBR', 'FLOW-VOL: DIS',
↪'FLOW-VOL: LK', 'FLOW-VOL: MSR', 'FLOW-VOL: OBS', 'FLOW-VOL: ZR', 'GASEX EE',
↪'GASEX RQ', 'GASEX VCO2', 'GASEX VO2', 'INV-BP DIA (p1)', 'INV-BP DIA (p2)', 'INV-
↪BP DIA (p3)', 'INV-BP DIA (p4)', 'INV-BP DIA (p5)', 'INV-BP DIA (p6)', 'INV-BP HR␣
↪(p1)', 'INV-BP HR (p2)', 'INV-BP HR (p3)', 'INV-BP HR (p4)', 'INV-BP HR (p5)', 'INV-
↪BP HR (p6)', 'INV-BP LBL (p1)', 'INV-BP LBL (p2)', 'INV-BP LBL (p3)', 'INV-BP LBL␣
↪(p4)', 'INV-BP LBL (p5)', 'INV-BP LBL (p6)', 'INV-BP MEAN (p1)', 'INV-BP MEAN (p2)',
↪ 'INV-BP MEAN (p3)', 'INV-BP MEAN (p4)', 'INV-BP MEAN (p5)', 'INV-BP MEAN (p6)',
↪'INV-BP SYS (p1)', 'INV-BP SYS (p2)', 'INV-BP SYS (p3)', 'INV-BP SYS (p4)', 'INV-BP␣
↪SYS (p5)', 'INV-BP SYS (p6)', 'INV-BP: ZR (p1)', 'INV-BP: ZR (p2)', 'INV-BP: ZR (p3)
↪', 'INV-BP: ZR (p4)', 'INV-BP: ZR (p5)', 'INV-BP: ZR (p6)', 'N2O ET', 'N2O FI',
↪'N2O: CLBR', 'N2O: MNS', 'NIBP CUFF', 'NIBP DIA', 'NIBP HR', 'NIBP MEAN', 'NIBP SYS
↪', 'NIBP: AUTO', 'NIBP: CLBR', 'NIBP: MSR', 'NIBP: OLD', 'NIBP: STASIS', 'NIBP: STAT
↪', 'NMT PTC-COUNT', 'NMT PTC-DB-COUNT', 'NMT PTC-ST-COUNT', 'NMT PTC-STIM', 'NMT␣
↪PTC-TOF-COUNT', 'NMT STM', 'NMT T1', 'NMT TIME', 'NMT TRATIO', 'NMT2 T1', 'NMT2 T2',
↪ 'NMT2 T3', 'NMT2 T4', 'NMT: CLBR', 'NMT: SUP', 'O2 ET', 'O2 FI', 'SpO2', 'SpO2 IR-
↪AMP', 'SpO2 LBL', 'SpO2 PR', 'SpO2 [SO2|SaO2|SvO2]', 'SvO2', 'TEMP (t1)', 'TEMP (t2)
↪', 'TEMP (t3)', 'TEMP (t4)', 'TEMP LBL (t1)', 'TEMP LBL (t2)', 'TEMP LBL (t3)',
↪'TEMP LBL (t4)', 'TONO CMPA', 'TONO P(r-Et)CO2', 'TONO P(r-a)CO2', 'TONO PADELAY',
↪'TONO PAMB', 'TONO PHI', 'TONO PHIDELAY', 'TONO PrCO2', 'TONO: LEAK', 'TONO: OVER',
↪'TONO: TECHFAIL', 'TONO: UNFILL', 'TONO: VOLDR', 'datetime']
```

(continues on next page)

(continued from previous page)

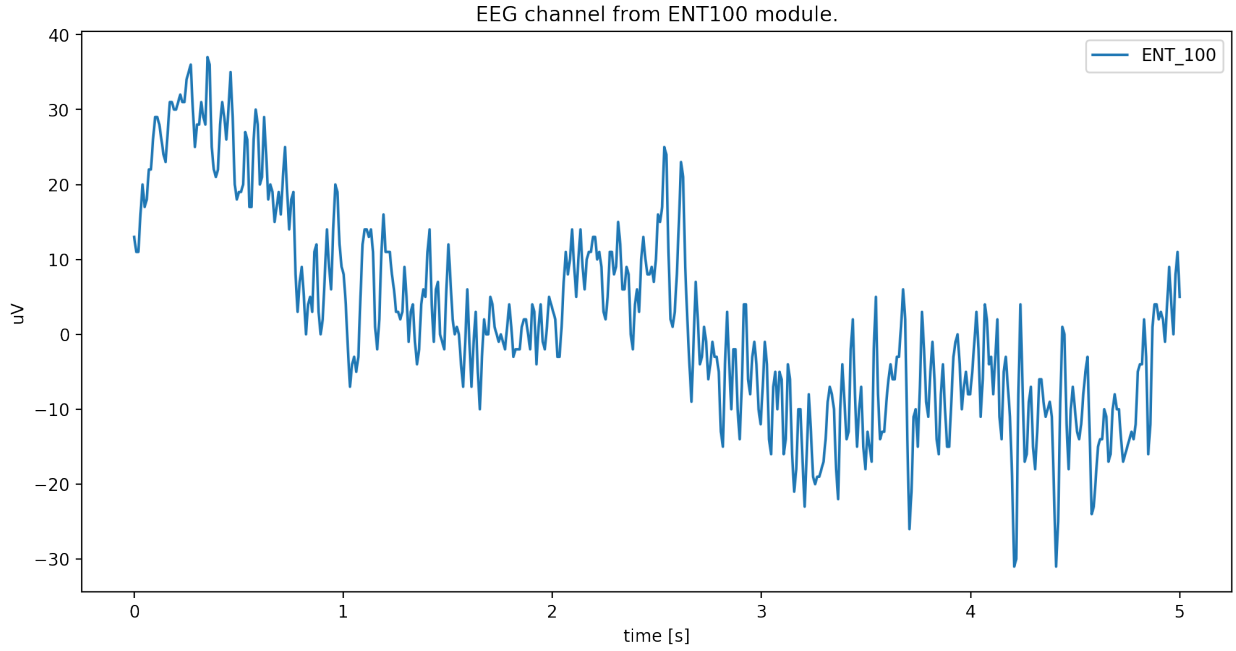
```
Waves: ['PLETH', 'ENT_100']
```

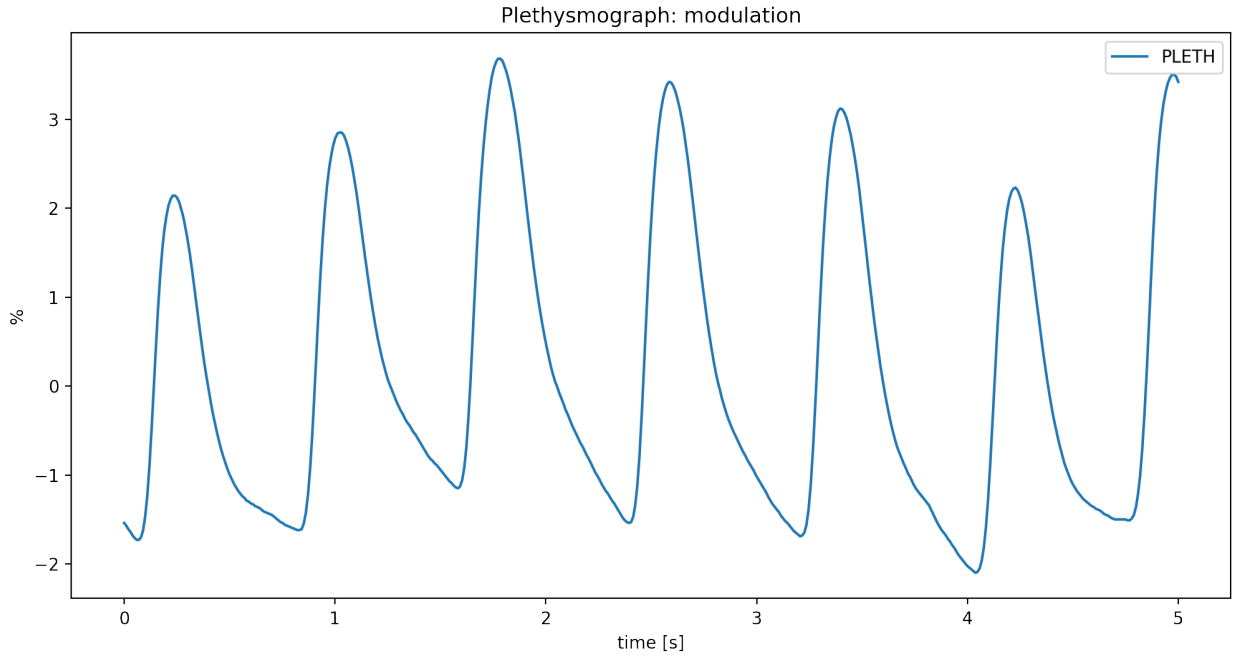
```
# pyplot.figure(figsize=(18,9), dpi=90)
# pyplot.plot(decoder.DATA_SUBRECORD['ENTROPY RE'])
# pyplot.legend(['ENTROPY RE'])

pyplot.figure(figsize=(12,6), dpi=200)
pyplot.plot(numpy.linspace(0, 5, 5*measures.WAVEFORMS_DICT['ENT_100']['samps']),␣
↪decoder.DATA_WAVE['ENT_100']['values'][:500])
pyplot.title(measures.WAVEFORMS_DICT['ENT_100']['desc'])
pyplot.xlabel('time [s]')
pyplot.ylabel(measures.WAVEFORMS_DICT['ENT_100']['unit'])
pyplot.legend(['ENT_100'])

pyplot.figure(figsize=(12,6), dpi=200)
pyplot.plot(numpy.linspace(0, 5, 5*measures.WAVEFORMS_DICT['PLETH']['samps']),␣
↪decoder.DATA_WAVE['PLETH']['values'][:500])
pyplot.title(measures.WAVEFORMS_DICT['PLETH']['desc'])
pyplot.xlabel('time [s]')
pyplot.ylabel(measures.WAVEFORMS_DICT['PLETH']['unit'])
pyplot.legend(['PLETH'])
```

```
<matplotlib.legend.Legend at 0x7f44425c3d30>
```





1.6.3 Save data

The data can be saved in two differents formats *RAW*, *SCV* and *EDF+*.

Save data as CSV

```
decoder.save_as_csv('data_out');
```

Save data as EDF+

The *edf* format need extra patient information.

```
from datetime import datetime

decoder.set_edf_header(
    admincode = '',
    birthdate = datetime(1900, 1, 1).timestamp(), #datetime object
    equipment = '',
    gender = 0, #0 for male, 1 for female
    patientcode = '',
    patientname = '',
    patient_additional = '',
    recording_additional = '',
    technician = '',
)

decoder.save_as_edf('data_out');
```

Save RAW data

```
decoder.save_as_raw('raw');
```

1.6.4 Channels information

List all labels.

```
subrecords = [g['label'] for g in measures.GROUPS]
print('All subrecords: {}'.format(subrecords))

waves = [g['label'] for g in measures.WAVEFORMS]
print('All waveforms: {}'.format(waves))
```

```
All subrecords: ['ECG HR', 'ECG ST1', 'ECG ST2', 'ECG ST3', 'ECG IMP-RR', 'ECG: MOD',
↪'ECG: ACT', 'ECG: ASY', 'ECG HR-SRC', 'ECG: NS', 'ECG: AR', 'ECG: LRN', 'ECG: PCR',
↪'ECG: CH1', 'ECG: CH2', 'ECG: CH3', 'ECG LEAD-CH1', 'ECG LEAD-CH2', 'ECG LEAD-CH3',
↪'INV-BP SYS', 'INV-BP DIA', 'INV-BP MEAN', 'INV-BP HR', 'INV-BP: MOD', 'INV-BP: ACT
↪', 'INV-BP: ZR', 'INV-BP LBL', 'NIBP SYS', 'NIBP DIA', 'NIBP MEAN', 'NIBP HR',
↪'NIBP: MOD', 'NIBP: ACT', 'NIBP CUFF', 'NIBP: AUTO', 'NIBP: STAT', 'NIBP: MSR',
↪'NIBP: STASIS', 'NIBP: CLBR', 'NIBP: OLD', 'TEMP', 'TEMP: MOD', 'TEMP: ACT', 'TEMP␣
↪LBL', 'SpO2', 'SpO2 PR', 'SpO2 IR-AMP', 'SpO2 [SO2|SaO2|SvO2]', 'SpO2: MOD', 'SpO2:␣
↪ACT', 'SpO2 LBL', 'CO2 ET', 'CO2 FI', 'CO2 RR', 'CO2 PAMB', 'CO2: MOD', 'CO2: ACT',
↪'CO2: AP', 'CO2: CS', 'CO2: ZS', 'CO2: OC', 'CO2: ALK', 'CO2 LBL', 'O2 ET', 'O2 FI',
↪ 'O2: MOD', 'O2: ACT', 'CO2: CLBR', 'CO2: MNS', 'N2O ET', 'N2O FI', 'N2O: MOD',
↪'N2O: ACT', 'N2O: CLBR', 'N2O: MNS', 'AA ET', 'AA FI', 'AA MAC-SUM', 'AA: MOD',
↪'AA: ACT', 'AA: CLBR', 'AA: MNS', 'AA', 'FLOW-VOL RR', 'FLOW-VOL PPEAK', 'FLOW-VOL␣
↪PEEP', 'FLOW-VOL PPLAT', 'FLOW-VOL TV-INSP', 'FLOW-VOL TV-EXP', 'FLOW-VOL COMP',
↪'FLOW-VOL MV-EXP', 'FLOW-VOL: MOD', 'FLOW-VOL: ACT', 'FLOW-VOL: DIS', 'FLOW-VOL:␣
↪CLBR', 'FLOW-VOL: ZR', 'FLOW-VOL: OBS', 'FLOW-VOL: LK', 'FLOW-VOL: MSR', 'CO-WEDGE␣
↪CO', 'CO-WEDGE TEMP', 'CO-WEDGE REF', 'CO-WEDGE PCWP', 'CO-WEDGE: MOD', 'CO-WEDGE:␣
↪ACT', 'CO-WEDGE CO-AGE', 'CO-WEDGE PCWP-AGE', 'NMT T1', 'NMT TRATIO', 'NMT PTC-COUNT
↪', 'NMT PTC-TOF-COUNT', 'NMT PTC-DB-COUNT', 'NMT PTC-ST-COUNT', 'NMT PTC-STIM',
↪'NMT: MOD', 'NMT: ACT', 'NMT STM', 'NMT TIME', 'NMT: SUP', 'NMT: CLBR', 'ECG-EXTRA:␣
↪HR', 'ECG-EXTRA: HR-MAX', 'ECG-EXTRA: HR-MIN', 'ECG-EXTRA: MOD', 'ECG-EXTRA: ACT',
↪'SvO2', 'SvO2: MOD', 'SvO2: ACT', 'ECG-ARRH HR', 'ECG-ARRH RR', 'ECG-ARRH PVC',
↪'ECG-ARRH: MOD', 'ECG-ARRH: ACT', 'ECG-12 STI', 'ECG-12 STII', 'ECG-12 STIII', 'ECG-
↪12 STAVL', 'ECG-12 STAVR', 'ECG-12 STAVF', 'ECG-12 STV1', 'ECG-12 STV2', 'ECG-12␣
↪STV3', 'ECG-12 STV4', 'ECG-12 STV5', 'ECG-12 STV6', 'ECG-12: MOD', 'ECG-12: ACT',
↪'ECG-12 LEAD-CH1', 'ECG-12 LEAD-CH2', 'ECG-12 LEAD-CH3', 'NMT2 T1', 'NMT2 T2',
↪'NMT2 T3', 'NMT2 T4', 'NMT2: MOD', 'NMT2: ACT', 'EEG FEMG', 'EEG EEG1-AMPL', 'EEG␣
↪EEG1-SFR', 'EEG EEG1-MNF', 'EEG EEG1-DELTA', 'EEG EEG1-THETA', 'EEG EEG1-ALPHA',
↪'EEG EEG1-BETA', 'EEG EEG1-BSR', 'EEG EEG2-AMPL', 'EEG EEG2-SFR', 'EEG EEG2-MF',
↪'EEG EEG2-DELTA', 'EEG EEG2-THETA', 'EEG EEG2-ALPHA', 'EEG EEG2-BETA', 'EEG EEG2-BSR
↪', 'EEG EEG3-AMPL', 'EEG EEG3-SEF', 'EEG EEG3-MF', 'EEG EEG3-DELTA', 'EEG EEG3-THETA
↪', 'EEG EEG3-ALPHA', 'EEG EEG3-BETA', 'EEG EEG3 BSR', 'EEG EEG4-AMPL', 'EEG EEG4-SEF
↪', 'EEG EEG4-MF', 'EEG EEG4-DELTA', 'EEG EEG4-THETA', 'EEG EEG4-ALPHA', 'EEG EEG4-
↪BETA', 'EEG EEG4-BSR', 'EEG: MOD', 'EEG: ACT', 'EEG: MSN', 'EEG: MONTAGE', 'EEG:␣
↪HEAD', 'EEG: SSEP', 'EEG: CH1-LEADS', 'EEG: CH2-LEADS', 'EEG: CH3-LEADS', 'EEG: CH4-
↪LEADS', 'EEG: CH1-ARTF', 'EEG: CH2-ARTF', 'EEG: CH3-ARTF', 'EEG: CH4-ARTF', 'EEG:␣
↪CH1-NS', 'EEG: CH2-NS', 'EEG: CH3-NS', 'EEG: CH4-NS', 'EEG: EP', 'EEG: MSN', 'EEG-
↪BIS', 'EEG-BIS SQI', 'EEG-BIS EMG', 'EEG-BIS SR', 'EEG-BIS: MOD', 'EEG-BIS: ACT',
↪'ENTROPY SE', 'ENTROPY RE', 'ENTROPY BSR', 'ENTROPY: MOD', 'ENTROPY: ACT', 'EEG2␣
↪COMMON', 'EEG2 CH1M', 'EEG2 CH1P', 'EEG2 CH2M', 'EEG2 CH2P', 'EEG2 CH3M', 'EEG2 CH3P
↪', 'EEG2 CH4M', 'EEG2 CH4P', 'EEG2: MOD', 'EEG2: ACT', 'GASEX VO2', 'GASEX VCO2',
↪'GASEX EE', 'GASEX RQ', 'GASEX: MOD', 'GASEX: ACT', 'FLOW-VOL2 IPEEP', 'FLOW-VOL2␣
↪Pmean', 'FLOW-VOL2 RAW', 'FLOW-VOL2 MVINSP', 'FLOW-VOL2 EPEEP', 'FLOW-VOL2 MVESEX',
↪'FLOW-VOL2 IERATIO', 'FLOW-VOL2 ISPTIME', 'FLOW-VOL2 EXPTIME', 'FLOW-VOL2 STCCOMP',
↪'FLOW-VOL2 STCPPLAT', 'FLOW-VOL2 STCPEEPE', 'FLOW-VOL2 STCPEEPI', 'FLOW-VOL2: MOD',
↪'FLOW-VOL2: ACT', 'BAL-GAS ET', 'BAL-GAS FI', 'BAL-GAS: MOD', 'BAL-GAS: ACT', 'TONO␣
↪PrCO2', 'TONO P(r-Et)CO2', 'TONO P(r-a)CO2', 'TONO PADELAY', 'TONO PHI', 'TONO␣
↪PHIDELAY', 'TONO PAMB', 'TONO CMPA', 'TONO: MOD', 'TONO: ACT', 'TONO: LEAK', 'TONO:␣
↪VOLDR', 'TONO: TECHFAIL', 'TONO: UNFILL', 'TONO: OVER', 'AA2 MAC-AGE-SUM', 'AA2: MOD
```

(continues on next page)

(continued from previous page)

```
All waveforms: ['ECG1', 'ECG2', 'ECG3', 'INVP1', 'INVP2', 'INVP3', 'INVP4', 'INVP5',
↪'INVP6', 'PLETH', 'CO2', 'N2O', 'AA_WAVE', 'AWP', 'FLOW', 'VOL', 'RESP', 'EEG1',
↪'EEG2', 'EEG3', 'EEG4', 'TONO_PRESS', 'SPI_LOOP_STATUS', 'ENT_100', 'EEG_BIS']
```

Get all measures from desired group.

```
group_aa = [g['label'] for g in measures.GROUPS_DICT['AA']]
print('All measures from group AA: {}'.format(group_aa))
```

```
All measures from group AA: ['AA ET', 'AA FI', 'AA MAC-SUM', 'AA: MOD', 'AA: ACT',
↪'AA: CLBR', 'AA: MNS', 'AA']
```

Inspect a single measure from subrecord or waveform.

```
# Information about ``AA ET`` (Subrecord)
print('AA ET: {}'.format(measures.LABEL_TO_DICT['AA ET']))

# Information about ``ECG1`` (Waveforms)
print('ECG1: {}'.format(measures.WAVEFORMS_DICT['ECG1']))
```

```
AA ET: {'label': 'AA ET', 'name': 'FeAA', 'desc': 'Anesthesia Agents ET', 'key':
↪'aa:et', 'unit': '%', 'shift': 0.01, 'subclass': 'basic'}
ECG1: {'label': 'ECG1', 'desc': '', 'unit': 'mV', 'shift_': 0.001, 'samps': 300,
↪'transducer': '', 'prefilter': '', 'physical_min': -0.001, 'physical_max': 0.001}
```

1.6.5 Other control funtions

```
# Stop data tranfer from monitor
device.stop()

# Close serial port
device.close()

# Stop internal asynchronous process
device.collect(False)
decoder.process(False)

# Restart decode process
device.collect(True)
decoder.BUFFER = device.BUFFER  #Force reference
decoder.process(True)

# Clear buffer, will require 'restart decode process'
device.clear_buffer()
decoder.clear_data()
decoder.clear_buffer()
```

1.7 Indices and tables

- genindex
- modindex

- search

1.8 References

- AS/3, CS/3 Monitoring System Main Software S/5 Monitor System Main Software Computer Interface
- Data acquisition from S/5 GE Datex anesthesia monitor using VSCapture: An open source.NET/Mono tool
- An Open-Source Anaesthesia Workstation (Linux)

Python Module Index

p

Index

Symbols

A

C

D

E

F

G

H

L

M

O

P

R

S

T

W