bioPlux java API v. 2

Programming Interface Reference

Document Version 1.1

PLUX - Wireless Biosigals
Headquarters
Parkurbis | 6200-865 Covilhã | Portugal
plux@plux.info

Lisbon Office
Av. 5 de Outubro n º70 – 8º | 1050-059 Lisboa | Portugal
Tel.. 217 959 120 | Fax. 217 976 007

Change Log

Version	Date	Changes
1	30-04-10	● Initial Release
1.1	15-11-10	● Constructor Correction

Introduction

The bioPlux java API brings to java applications all the functionality of bioPlux devices. The API is implemented as two classes called *Device* and *BPException*.
The class *Device* encapsulates the communication channel to a bioPlux device and the state of that device. This device can be physical and accessed through a Bluetooth connection, or it can be software-emulated for testing purposes when a physical device is not available.
The class *BPException* implements the exception which is thrown by the API when an error condition is met while calling any of the API functions or class methods.
The API consists of a jar file (NewBioPlux,jar) containing the API object code and a dll with the JNI wrapper (Java Native Interface) called Lib_java.dll. The dll file shall be installed in the Windows/system32 directory.

Class Device Overview

The class *Device* encapsulates the communication channel to a bioPlux device and the state of that device. This device can be one of the following:

- a bioPlux device connected through Bluetooth using a virtual COM port;
- a bioPlux device connected through Bluetooth using its MAC address;
- a software-emulated device implemented locally by the API.

The system resources are allocated at the creation of the *Device* instance. These resources are then released when this instance is destroyed. During instance destruction, if the bioPlux device is in acquisition mode, it is automatically stopped.

A *Device* instance has exclusive access to a bioPlux device, so it cannot share the device with another instance. If a bioPlux device is in use by an instance, it cannot be opened by a new *Device* instance until the first instance is destroyed.

Class Device Static Functions

static void FindDevices(Vector devs)
parameters:
[out] *devs*
vector of strings where the MAC addresses will be stored

description:
Search for bioPlux Bluetooth devices that are switched on and within range of the local Bluetooth adapter. The MAC address of each device found is added to the *devs* vector. This vector is first emptied.

Class Device Methods and Constructor

Device(String port)
parameters:
[in] *port*
string with either a valid COM port or MAC address, or *"test"*

description:
Constructor of the *Device* instance. Depending on the format of *port*, the new instance is connected to the software-emulated test device (described later), or to a bioPlux Bluetooth device through the provided COM port or through the provided Bluetooth MAC address.

void String GetDescription()

description:
Retrieve the bioPlux device description and store it in *str*. This method can be called anytime during instance lifetime.

returns:
A string in which the bioPlux device description will be stored

void BeginAcq()
description:
Start the acquisition mode in the device. An exception is thrown if the device is already acquiring. In acquisition mode, one frame of data (described later) is received every millisecond (sampling rate of 1000 Hz) with 12-bit samples of the eight channels.

void BeginAcq(int freq, int chmask, int nbits)
parameters:
[in] *freq*
requested sampling frequency in Hz
[in] *chmask*
bit-mask for selection of analog channels to sample
[in] *nbits*
number of bits per sample

description:
Start the acquisition mode in the device. An exception is thrown if the device is already acquiring. The *freq* parameter must be between 36 Hz and 1000 Hz. Parameter *chmask* is one byte in an integer where the LSB refers to channel 1 and the MSB refers to channel 8. If a bit is set, the corresponding channel will be sampled. Parameter *nbits* can only be 8 or 12. In acquisition mode, one frame of data (described later) is received with a sample for each of the selected channels, at a frequency *freq*. This function is available only on devices with firmware version 2, otherwise an exception is thrown.

void GetFrames(int nframes, Frame[] frames)
parameters:
[in] *nframes*
number of frames to acquire from the bioPlux device
[out] *frames*
array with the acquired frames

description:
Fill the *frames* array with *nframes* (at most) frames acquired from the device. The acquisition mode must be started before calling this method, otherwise an exception is thrown. If the input buffer does not contain yet the requested frames, the method blocks until all the requested frames arrive, or until a communications time-out error occurs. The application shall verify the sequence number of each returned frame in the array *frames* (member *seq* of the *Frame* structure), as discussed later in the *Frame* structure section.

void SetDOut(boolean dout)
parameters:
[in] *dout*
value with which to set the bioPlux device digital output signal

description:
Set the digital output signal of the bioPlux device with *dout*. This method can be called anytime during instance lifetime.

void EndAcq()
description:
Stop the acquisition mode in the device. An exception is throw if the acquisition mode is not active.

class Device.Frame

As discussed earlier, the GetFrames() method fills an array of *Frame* classes. This class has the following data members:

- seq (type byte) : the sequence number of the frame. This number is an incrementing 7-bit unsigned integer running from 0 to 127, and then overflowing to 0. The application should verify this number, since a difference of more than one unit between two consecutive received frames (correcting the overflow from 127 to 0) means that there was some missing sent frames between these two received frames. This loss of frames can occur near the limit of the Bluetooth transmission range.
- dig_in (type boolean) : the state of the digital input.
- an_in (type short[8]) : an array with the signal sample value at the selected analog channels, from channel 1 to channel 8, as unsigned integers. For 12-bit samples, each value spans from 0 to 4095. For 8-bit samples, each value spans from 0 to 255. If the acquisition mode was started calling the BeginAcq(void) method (not the BeginAcq(freq, chmask, nbits) method), the values for the two last channels are updated only for frames whose *seq* value is multiple of 8. For all other frames, these two channels retain the value of last frame. So, the first six channels are sampled at 1000 Hz and the last two channels are sampled at 125 Hz.

The Software-Emulated Test Device

As discussed earlier, the Create() function accepts the string “test” (case insensitive) to create a virtual bioPlux device implemented in software by the API. This device is useful to quickly test an application when a real bioPlux device is not available or is not convenient to use.

This device behaves exactly as a real bioPlux device in terms of the application interface, the device state management (the acquisition mode), the exception mechanism (described later) and even the waiting times on GetFrames() for new “frames”. The main difference is of course the values on the returned frames. Each of the 8 channels of this device returns a mathematical function listed in the following table.

Channel	Function Description
1	Sine wave, mean value: *max*/2, amplitude: *max*/2, frequency: f/100
2	Saw tooth, from 0 to *max*, frequency: f/100
3	Saw tooth, from *max* to 0, frequency: f/100
4	Square wave, amplitude: *max*, duty-cycle: 50 %, frequency: f/100
5	Alternating 0 with *max*, each sample
6	Uniform pseudo-random value, between 0 and *max*
7	Sine wave, mean value: *max*/2, amplitude: *max*/2, frequency: f/1000
8	Saw tooth, from 0 to *max*, frequency: f/1000

where *max* is the maximum value (255 for 8-bit samples or 4095 for 12-bit samples) and f is the sampling frequency.

If the acquisition mode was started calling the BeginAcq(void) method (not the BeginAcq(freq, chmask, nbits) method), the values of the last two channels are updated only on frames whose *seq* value is multiple of 8, as with a real bioPlux device.

The state of the digital input (present on each frame) is the state defined in the last call to SetDOut(). Before the first call, the input state is initialized to *false*.

There is no restriction on opening multiple test devices, operating independently from each other.

The test device does not simulate loss of frames, frame transmission delays or any other transmission effect.

Class BPException Overview

The class BP*Exception* implements the exception which is thrown by the API when an error condition is met while calling any of the API functions or class methods. The application shall expect that any call to API functions or methods can throw this exception and shall catch the exception. It derives from the java.lang. Exception and inherits the same methods.

There are two types of conditions causing this exception to be thrown:

- Notification : an external condition that should be presented to the user so that he can solve the problem;
- Error : a condition due to application logic failure or an unexpected error that the application should handle.

Class BPException Methods

String getMessage(void)
description:
Retrieve the error description string associated with the exception.

returns:
The description of the Error

Class BPException Data Members

int code
description:
A integer value identifying the condition associated with the exception. The possible values are enumerated in the first column of the Error Codes Table.

Error Codes Table

The following table shows all the conditions returned by the BP*Exception* exception. The *Code* column is the value assigned to the *code* data member. The *Description* column is the string returned by the GetMessage() method. The *Type* column indicates the type of exception, where 'N' means a notification and 'E' means an error.

Code	**Description**	**Type**
BT_ADDRESS(1)	The specified address is not correct. Please verify that it is composed by six (6) pairs of comma separated alphanumeric characters [xx:xx:xx:xx:xx:xx].	N
BT_ADAPTER_NOT_FOUND(2)	No Bluetooth adapter was found. Verify if an adapter is connected and turned on.	N
BT_DEVICE_NOT_FOUND(3)	The Plux device could not be found. Verify that it is within range, turned on, and that the battery is ok (red led turned off).	N
CONTACTING_DEVICE(4)	The computer lost communication with the Plux device. Verify that the Plux device is within range, and that the battery is ok (red led turned off).	N
PORT_COULD_NOT_BE_OPENED(5)	The communication port does not exist or it is already being used.	N
PORT_INITIALIZATION(6)	The communication port could not be initialized.	E
DEVICE_NOT_IDLE(7)	The device is not idle.	E
DEVICE_NOT_IN_ACQUISITION_MODE(8)	The device is not in acquisition mode.	E
PORT_COULD_NOT_BE_CLOSED(9)	The communication port could not be closed.	E
BT_DEVICE_NOT_PAIRED(10)	The Plux device is not paired.	E
INVALID_PARAMETER(11)	Invalid parameter.	E
FUNCTION_NOT_SUPPORTED(12)	Function not supported by the device firmware.	E