BioRadio
Software Development Kit
For MATLAB®

User's Guide

GREAT LAKES NEUROTECHNOLOGIES

Telephone: (216) 361-5410 or toll-free 1-855-GLNeuro (1-855-456-3876)
9:00 a.m. - 5:00 p.m. EST
Monday – Friday
Fax: (216) 361-5420

Customer Support: support@GLNeuroTech.com

Sales: sales@GLNeuroTech.com

Website: http://www.GLNeuroTech.com

Mailing Address: Great Lakes NeuroTechnologies Inc.
10055 Sweet Valley Drive
Cleveland, Ohio 44125

BioRadio SDK for MATLAB User's Guide REF No. G392-0051, Rev A
DCO G174

TABLE OF CONTENTS

Chapter 1. Introduction

This document describes MATLAB example functions and routines that communicate with the BioRadio through the following software DLL (dynamic link library): BioRadioSDK.dll

The BioRadio must initially be configured in the included BioCapture program before streaming data into the MATLAB environment.

During real-time streaming, acquired data is sent from the BioRadio and buffered at the PC's virtual serial port, where it is collected by background processes in the software device object. Regardless of how often data is read from the data buffer, biopotential data is sampled at the sample rate configured in BioCapture, while other signals acquired through the BioRadio Sensor Pods are sampled at 250 Hz. Data can be collected regularly from the BioRadio device object by the developer's application for display, processing, or archival. The most efficient interval between data collections will vary slightly by system and environment. The MATLAB examples attempt to collect data every 80 milliseconds to minimize system overhead while ensuring that the buffers are serviced frequently enough that they do not fill and overflow.

Installation

Download the BioRadio software development kit (SDK) and install it to a suitable location. This SDK contains the application programming interface (API) assembly BioRadioSDK.dll. It also contains detailed information on the constructors, classes, methods, and properties in the html documentation files.

System Requirements

Bluetooth adapter (v2.1 or greater)

The API requires Microsoft .NET 4.5.1 to be installed. It can be downloaded directly from Microsoft's website at http://www.microsoft.com/

Because the API requires Microsoft .NET, interacting with the BioRadio from the MATLAB environment requires a version of MATLAB that includes a .NET interface, which was first added in 2009b.

Chapter 2. Getting Started

Start MATLAB and set the current working directory to the MATLAB folder contained in the Software Development Kit you installed previously. The folder contains several modular .m files, described in the table below. Various API-level commands are used within these .m files.

Filename	Description
`load_API.m`	Makes the BioRadio .NET assembly visible to MATLAB
`BioRadio_Find.m`	Searches for BioRadios in the vicinity and prompts the user to select one for connection
`BioRadio_Connect.m`	Initializes a BioRadio object selected by `BioRadio_find.m`. If the MAC address and name of the BioRadio are known, these can be input directly without searching.
`BioRadio_Stream.m`	Streams data from BioRadio and imports the data into MATLAB
`BioRadio_Disconnect.m`	Terminates the connection with BioRadio device

There is a complete example that illustrates the overall process of interacting with the BioRadio in MATLAB.

Filename	Description
`BioRadio_example.m`	A script that goes through each step of creating BioRadio objects and streaming data. Uses modular functions above.

Note: The MATLAB code relies on the .NET assembly (BioRadioSDK.dll) for operation. Once loaded, the MATLAB command line can be used to make calls to DLL functions for operating the device.

Chapter 3. Interacting with the BioRadio in MATLAB for Real-Time Streaming

The best way to start interacting with the BioRadio for streaming data into MATLAB is to go through `BioRadio_example.m`, which calls modular sub-functions that can be easily integrated into a custom application. In this section of the User Guide, the major steps in interacting with the BioRadio are outlined.

1. Configure the BioRadio in BioCapture

a. Configure the biopotential channels and sensor pods in BioCapture. Once this is completed, you can connect to and stream data from the BioRadio in other software environments.

2. Load API into MATLAB environment

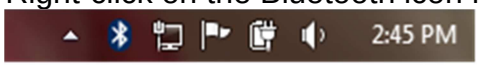
a. The BioRadio .NET assembly must be visible to MATLAB in order to interact with a device.
b. Key MATLAB function call: `NET.addAssembly(filepath)`, where `filepath` is a string pointing to the location (i.e., full file path) of the BioRadio API dll file, BioRadioSDK.dll. See `load_API.m` for an example on usage

3. Create Device Manager

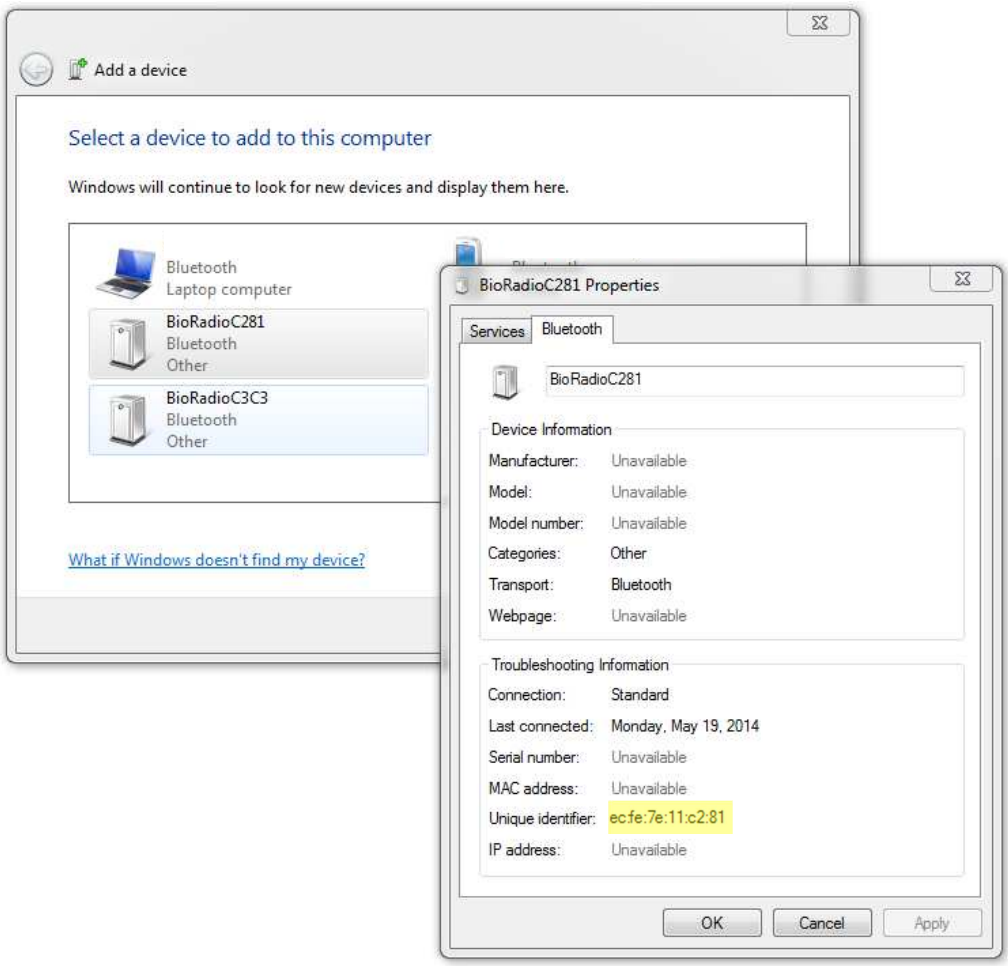
a. Once the API is loaded, calling `GLNeuroTech.Devices.BioRadio.BioRadioDeviceManager` returns a handle to a Device Manager object that is used to help with the searching and creation of BioRadioDevice objects. See `load_API.m` for an example on usage

4. Initialize BioRadio

a. BioRadioDevice objects can be instantiated by entering the MAC address of the BioRadio directly. The MAC address for each BioRadio is included in the shipping documentation and can be looked up in Windows as follows:
    i. Right-click on the Bluetooth icon in your system tray and select “Add a Device”



    ii. Locate the BioRadio you want to connect to in the list, right-click it, and select Properties. The unit ID must match that provided on the back of your device.



iii. On the Bluetooth tab of the popup window, the window will list a Unique identifier (highlighted in yellow) which is the Mac ID

b. The device manager can also search for BioRadio devices in the vicinity by calling `deviceManager.DiscoverBluetoothDevices`. See `BioRadio_Find.m` for an example on usage.
c. The 64-bit MAC ID must be used. In MATLAB, using the example above, `macID = int64(hex2dec('ECFE7E11C281'))` would return the 64-bit MAC ID
d. Key API function call: `myDevice = deviceManager.GetBluetoothDevice(macID)`. This returns a handle to a BioRadioDevice, which is automatically connected. See `BioRadio_Connect.m` for an example on usage.
e. NOTE: `BioRadio_Connect.m` is designed to work with `BioRadio_Find.m` as a flexible solution for interacting with an unknown set of BioRadios. Within the context of a custom application, it may be more efficient to store the BioRadio MAC address and automatically connect to it, rather than searching and manually selecting.

5. Read BioRadio Configuration Settings

a. Reading the configuration of the BioRadio may be a useful first step before acquiring, plotting, or analyzing data.
b. The biopotential and sensor pod signals can be accessed via signal group properties of the BioRadioDevice object (e.g., `myDevice.BioPotentialSignals`, `myDevice.PulseOxSignals`, or `myDevice.AuxiliarySignals`).
c. The number of channels can be accessed via the "Count" property of each signal group (e.g., `myDevice.BioPotentialSignals.Count`).
d. The sample rate can be accessed via the "SamplesPerSecond" property of each signal group (e.g., `myDevice.BioPotentialSignals.SamplesPerSecond`).
e. The sample rate of any signals acquired using a sensor pod is 250 Hz.

6. Start Acquisition

a. Once a BioRadio has been instantiated and assigned to a MATLAB variable (let's say that variable name is `myDevice` for this example), data streaming can be initialized by calling `myDevice.StartAcquisition.` See `BioRadio_Stream.m` for an example on usage.

7. Stop Acquisition

a. Data streaming can be terminated by calling `myDevice.StopAcquisition.` See `BioRadio_Stream.m` for an example on usage.
b. NOTE: Built-in MATLAB timing functions such as `tic` and `toc` may be useful for acquiring data for a specified time interval.

8. Retrieve Data from Buffer

a. Data is stored in a buffer on the host PC during streaming.
b. The individual channels of data associated with each signal group can be accessed via the Item method (e.g., `myDevice.BioPotentialSignals.Item(0)`).
c. NOTE: The index of the Item method is zero-based. For example, the first channel would be accessed with `Item(0)`, while the fourth channel would be accessed with `Item(3)`.
d. The GetScaledValueArray method pulls the data from the buffer and can be used to assign it to a MATLAB variable `myData_0 = myDevice.BioPotentialSignals.Item(0).GetScaledValueArray.double`
e. See `BioRadio_Stream.m` for an example of pulling data from the buffer.

9. Disconnect from BioRadio

a. Calling Disconnect closes the Bluetooth connection with the BioRadio (e.g., `myDevice.Disconnect`). See `BioRadio_Disconnect.m` for an example on usage.

Chapter 4. Troubleshooting

1. Why won't my computer connect to the BioRadio?

If you are unable to successfully connect to a BioRadio, please try the following:

(A) Ensure that the BioRadio is powered on and the Bluetooth adapter in your computer is enabled.
(B) Make sure that you are not attempting to simultaneously connect to more than 7 Bluetooth devices, counting any BioRadios and other accessories (e.g., mouse, keyboard, printer, cell phone)
(C) Ensure that Microsoft .NET 4.5.1, which the API requires, is installed.
(D) Check to see if your Bluetooth adapter is using the latest drivers. Updating to the most recent drivers may resolve some communication issues.

2. My BioRadio stopped streaming in the middle of data collection.

This will happen if the battery dies or the BioRadio is out of range of the Bluetooth receiver on the host PC. Please make sure the battery is fully charged. The range will vary depending on the environment.

3. I can connect to my BioRadio, but it does not stream any data when I start acquisition and then loses its connection.

Consider lowering the sample rate configured in BioCapture. If the device configuration exceeds the bandwidth for transmission and too many packets are dropped, the connection is terminated. Lowering the sample rate will improve data transmission.

4. My BioRadio is powered on but it is not discovered by the search.

Repeat the search. Occasionally environmental factors (e.g., noise on 2.4 GHz band) interfere, but the issue is typically resolved by repeating the search. You might also change where the BioRadio is located relative to the Bluetooth adapter if the problem persists.