Search this site

Muse Developer Resources

Navigation

Home
SDK Overview
Download & Install
Getting Started
LibMuse
MuseIO
MuseLab
MusePlayer
MuseIO Receiver
Muse Hardware
Muse Data Files
Muse Communication Protocol
Multi-Muse Setup
Developer FAQ
Intro to BCI and EEG
Release Notes
Forums
Mailing List
Support

Getting Started

Here's a quick and easy example of how to use the current SDK. This page assumes you have already downloaded and installed the SDK and had a look at the SDK Overview. To get familiar with the Muse hardware, button, and LEDs, check out the consumer quick start guide as well as the Muse Hardware page. The Muse Calm app (it's free!) for Android and iOS also provides a very thorough interactive introduction to Muse.

Contents

At this time, LibMuse is only available for Android and iOS. Versions of LibMuse for desktop platforms are currently being written. To develop for desktop platforms right now we provide MuseIO (as well as the other Muse SDK Tools: MuseLab and MusePlayer) to connect to Muse and stream data over OSC to another program running an OSC server.

In this section we will describe this process and provide some simple examples of how to build such a server. Let's get started!

1. Pair with Muse

Make sure Muse is sufficiently charged before pairing and streaming data. If you are having difficulties initially connecting, charge Muse for 20-30 minutes and then try again.

Switch Muse into Pairing Mode by holding the button down for ~5 seconds. The rest of this process depends on the operating system that you are using, so just follow the standard Bluetooth pairing instructions for your specific operating system. If you are asked to enter a passcode, enter "1234". For more info on using the Muse hardware, the different button presses, LED patterns, etc. see the consumer quick start guide and the Muse Hardware page.

2. Start MuseIO

Open a terminal or command prompt and type:

```
muse-io --osc osc.udp://localhost:5001,osc.udp:
//localhost:5002
```

By default MuseIO will look for paired devices named "Muse" and attempt to connect with them. It will send all data to 127.0.0.1:5000 unless otherwise specified, so here we have instructed it to instead send all OSC data to both port 5001 and 5002 using UDP. We will tell our custom OSC server to listen at port 5001 and at the same time will run MuseLab listening on port 5002 to graph the incoming signals in real time. The "--dsp" flag will enable all the extra data streams that require Muse-IO to perform some Digital Signal Processing. These include FFT coefficients and measures of EEG relevant frequency bands such as alpha, beta, delta, and theta.

For a full explanation of all of MuseIO's options, see here.

What kind of data can I get from MuseIO?

A full list of all the data that MuseIO provides (including background information and explanations) is available here.

3. Run MuseLab

Take a look at the MuseLab tutorial and be sure to point it at port 5002 with UDP selected. Using MuseLab in this way, while you're building your own OSC server and application for Muse, you can double check the data that MuseIO is sending out by observing it as real-time plots.

4. Set up an OSC server

Now that OSC messages containing Muse data are being sent to port 5001 and 5002 from Muse-IO, we just need to set up an OSC server in our language of choice. Once you can receive OSC messages, you can start doing fun things with the data from Muse.

There are OSC libraries available for most major programming languages. Let's start by looking at a simple example in Python.

Python OSC Server

First, make sure you have installed pyliblo which provides Python bindings to liblo.

Note: this example borrows heavily from pyliblo's own documentation, which is very helpful and includes some interesting alternative methods for sending and receiving OSC messages.

WARNING: Windows users, pyliblo is not well supported on your OS. Please see the Processing example in the next section to continue.

The following example sets up and starts an OSC server listening for messages on port 5001. There are callbacks defined for /muse/eeg and /muse/acc (EEG and accelerometer data, respectively), and a generic callback to handle all other messages. All it does is print out the content of whatever message it receives. Grab the code here.

```
from liblo import *
import sys
import time
```

```
class MuseServer(ServerThread):
    #listen for messages on port 5001
    def __init__(self):
        ServerThread.__init__(self, 5001)

    #receive accelrometer data
    @make_method('/muse/acc', 'fff')
    def acc_callback(self, path, args):
        acc_x, acc_y, acc_z = args
        print "%s %f %f %f" % (path, acc_x, acc_y,
acc_z)

    #receive EEG data
    @make_method('/muse/eeg', 'ffff')
    def eeg_callback(self, path, args):
        l_ear, l_forehead, r_forehead, r_ear =
args
        print "%s %f %f %f %f" % (path, l_ear,
l_forehead, r_forehead, r_ear)

    #handle unexpected messages
    @make_method(None, None)
    def fallback(self, path, args, types, src):
        print "Unknown message \
        \n\t Source: '%s' \
        \n\t Address: '%s' \
        \n\t Types: '%s ' \
        \n\t Payload: '%s'" \
        % (src.url, path, types, args)

try:
    server = MuseServer()
except ServerError, err:
```

```
    print str(err)
    sys.exit()


server.start()


if __name__ == "__main__":
    while 1:
        time.sleep(1)
```

Processing OSC Server

This example shows how to create a very simple OSC server in Processing. This is only the code for receiving OSC messages, all the visualization code is left as simple as possible in this example, so the applet simply appears as a blank square. The rest is up to your imagination!

Download and install Processing from processing.org.

Make sure the oscP5 library is installed by going to Sketch -> Import Library -> Add Library and selecting oscP5.

Grab the code here.

```
  //
  //MUSE OSC SERVER EXAMPLE
  //Copyright Interaxon 2015
  //

  import oscP5.*;
  boolean debug = false;

  //OSC PARAMETERS &
PORTS-------------------------------------
  int recvPort = 5001;
  OscP5 oscP5;

  //DISPLAY
PARAMETERS-----------------------------------------
  int WIDTH = 100;
  int HEIGHT = 100;
```

```
//SETUP----------------------------------------------
  void setup() {
    size(WIDTH,HEIGHT);
    frameRate(60);

    /* start oscP5, listening for incoming
messages at recvPort */
    oscP5 = new OscP5(this, recvPort);
    background(0);
  }

  //DRAW LOOP
------------------------------------------------
  void draw() {
    background(0);
  }

  //OSC
HANDLER--------------------------------------------
  void oscEvent(OscMessage msg) {
    /* print the address path and the type string
of the received OscMessage */
    if (debug) {
      print("---OSC Message Received---");
      print(" Address Path: ",
msg.addrPattern());
      println(" Types: ", msg.typetag());
    }
    if (msg.checkAddrPattern("/muse/eeg")==true)
{
        for(int i = 0; i < 4; i++) {
         print("EEG on channel ", i, ": ",
msg.get(i).floatValue(), "\n");
        }
    }
  }
```

osc_server.py (1k) Unknown user, … v.1

processing_osc_… Unknown user, … v.1

Comments

You do not have permission to add comments.