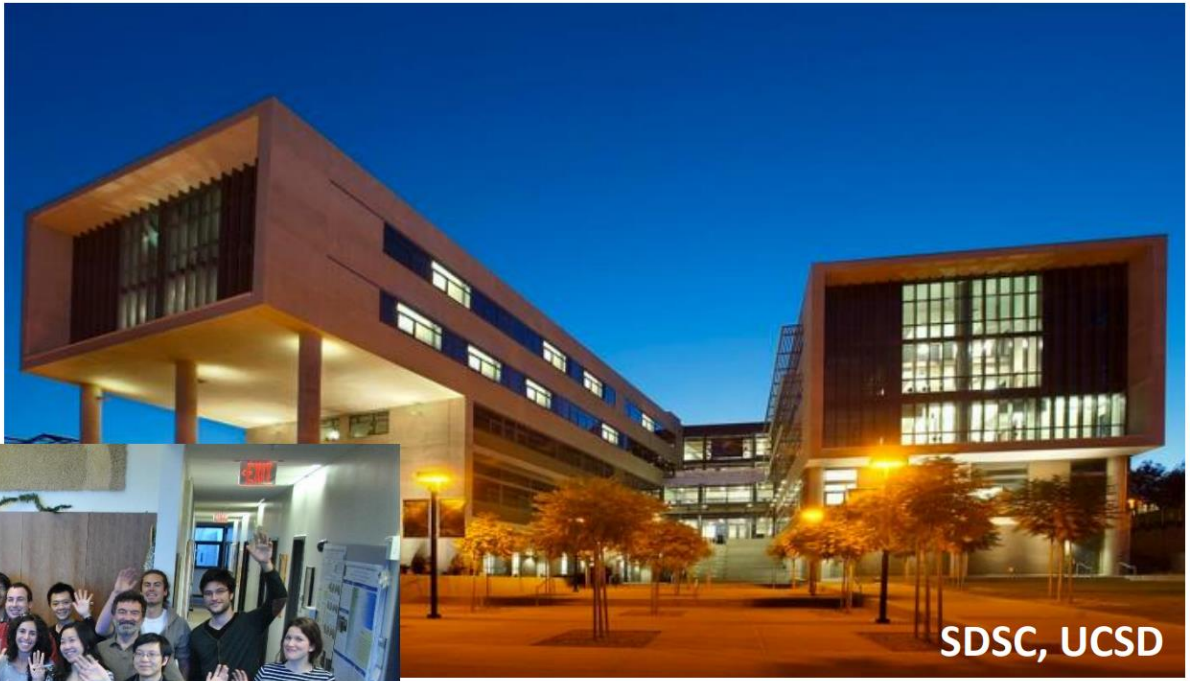
The Lab Streaming Layer – Introduction and Overview

Christian A Kothe, CTO

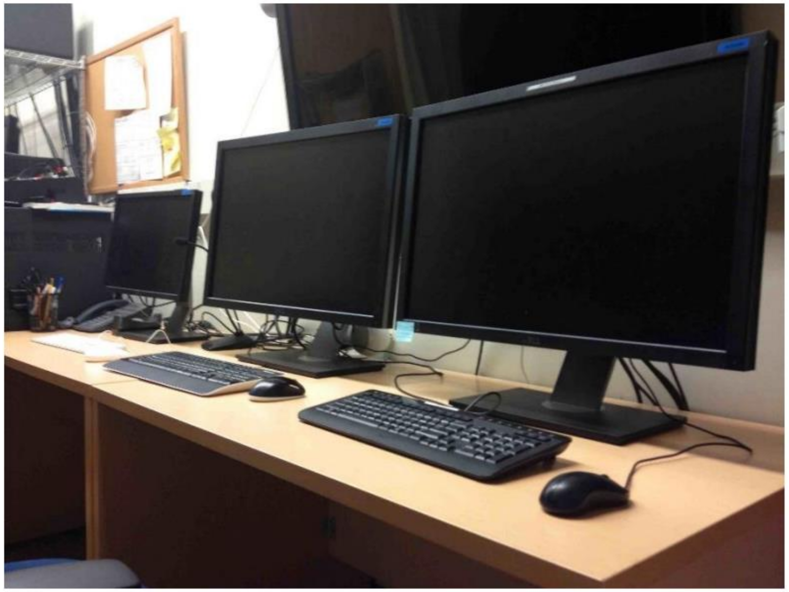
Tim Mullen, CEO

intheon

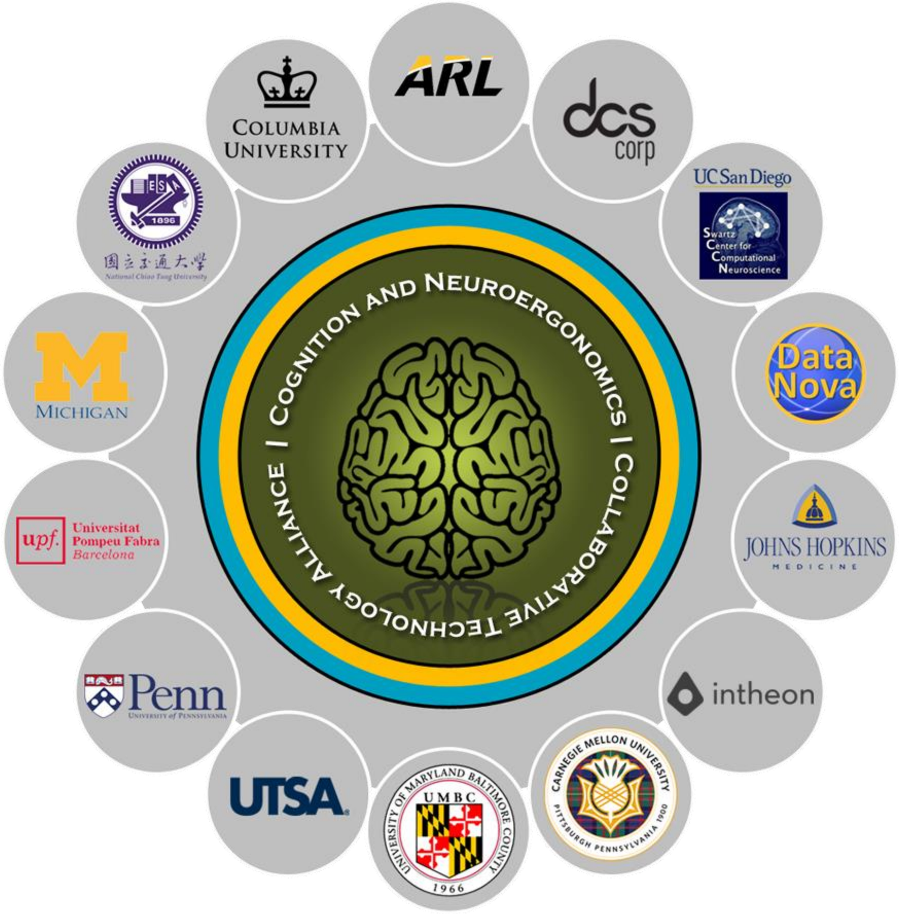
The Birthplace of LSL (SCCN)



The Birthplace of LSL (SCCN)



Seed Funding from CaN CTA

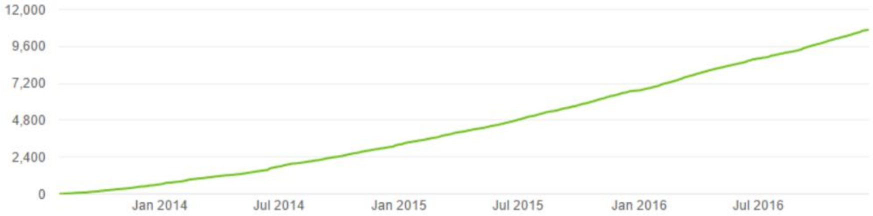

Cognition and Neuroergonomics | Collaborative Technology Alliance |
ARL
dcs corp
UC San Diego
Swartz Center for Computational Neuroscience
Data Nova
Johns Hopkins Medicine
intheon
Carnegie Mellon University Pittsburgh Pennsylvania 1900
University of Maryland Baltimore County
UMBC
1966
UTSA
Penn
University of Pennsylvania
upf. Universitat Pompeu Fabra Barcelona
Michigan
1896
國立交通大學
National Chiao Tung University
Columbia University

Additional government funding sources

ARL ONR Office of Naval Research NIH National Institute of Neurological Disorders and Stroke DARPA NASA

LSL Has Come a Long Way!

- Large & Growing Userbase
  - 19000 lecture views on YouTube as of today



  - 130 GitHub forks, 240 stars, 5000 Google results
  - Workshops, Hackathons, Online Community, ...
- Broad Hardware support
  - Many dozen devices across many vendors
- Used in all sorts of places (e.g., NASA)

Outline

1. Why LSL?
2. What is LSL?
3. Using LSL
4. The LSL Ecosystem
5. Q&A

1 Why LSL?

Issues Addressed by LSL Pt 1

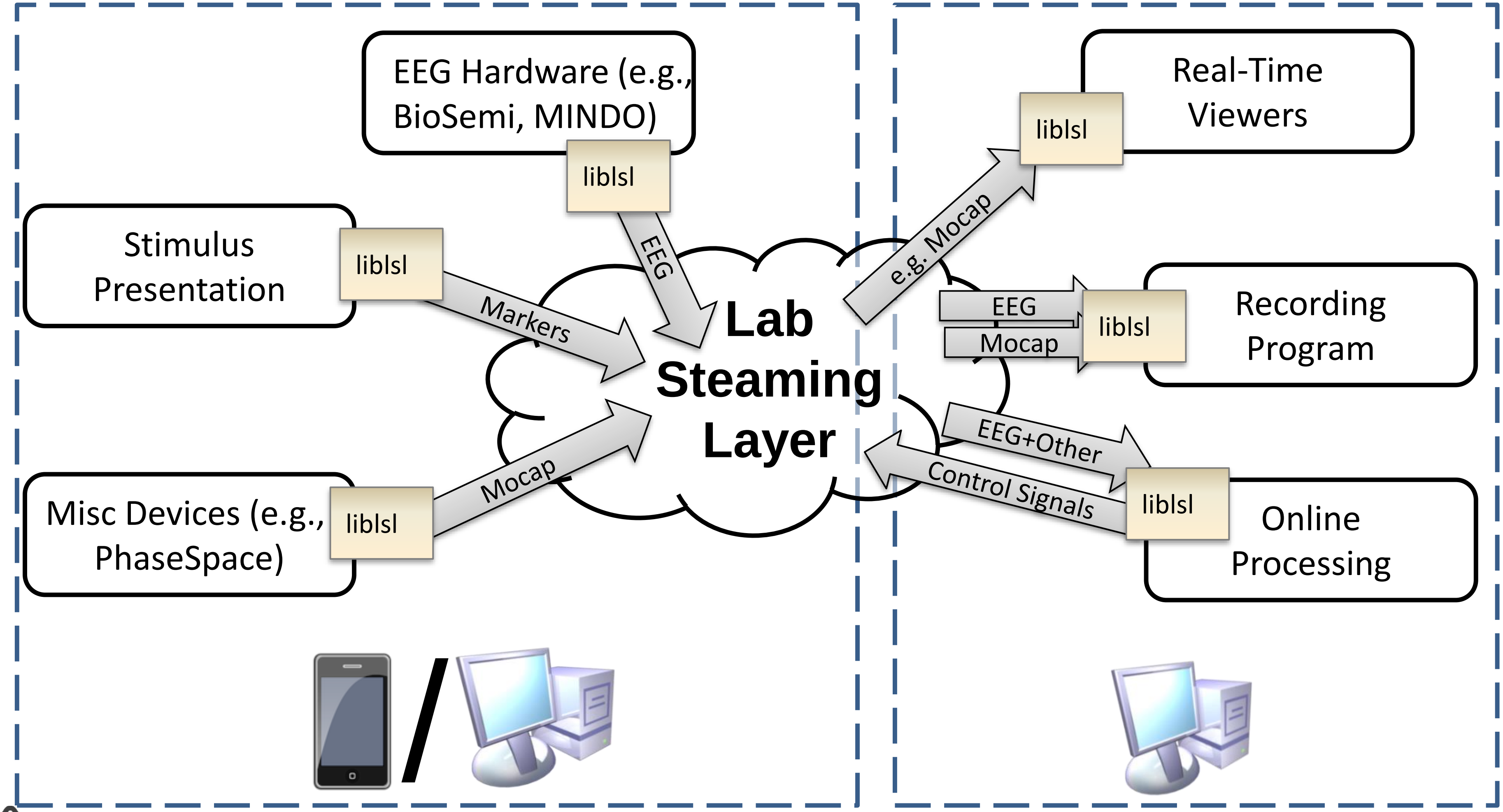
- 😟 **Format mess**
  - Lots of file formats and importer & conversion functions
  - Custom scripts needed to read/write extra files
  - Missing or unreadable meta-data (e.g., channel labels)
- 😟 **Complex hardware time synchronization…**
  - A lot of custom hardware and cabling needed for synchronization (e.g., sync boxes, adapters, creative wiring)
  - Easy to make mistakes at data collection time
  - Lengthy setup and pilot testing stages
- 😟 **… or brittle and wacky post-hoc synchronization**
  - E.g., compare file times for different simultaneous recordings
  - Or find and match peaks in different signals

Issues Addressed by LSL Pt 2

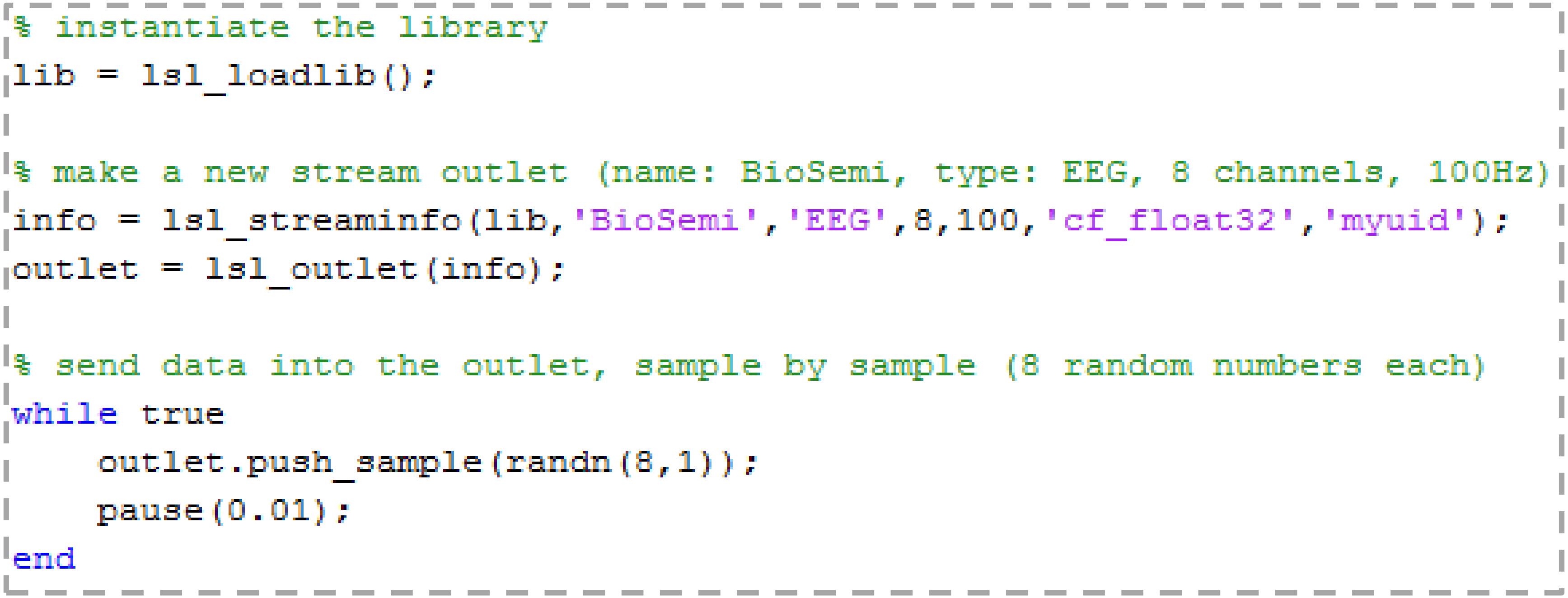
- 😟 **Error prone data collection**
  - chance of failure increases with # of devices, computers, programs and data files involved
  - should be able to reconnect (or hot-swap) a device during experiment
- 😟 **No unified tools for recording, viewing, etc**
  - No centralized viewing / recording across devices
  - Limited to no support for viewing data from custom devices
- 😟 **Every vendor has a different interface (or none)**
  - custom code for each device needed (sometimes driver-level)
  - high development cost for online experiment scripts
  - online time-sync challenging
- 😟 **Not very easy to get data out of most hardware**
  - often need to know a specific programming language (e.g., C++)
  - often little documentation and obscure, rarely-used interfaces

2 What is LSL?

LSL Is A Unified Data Collection Interface



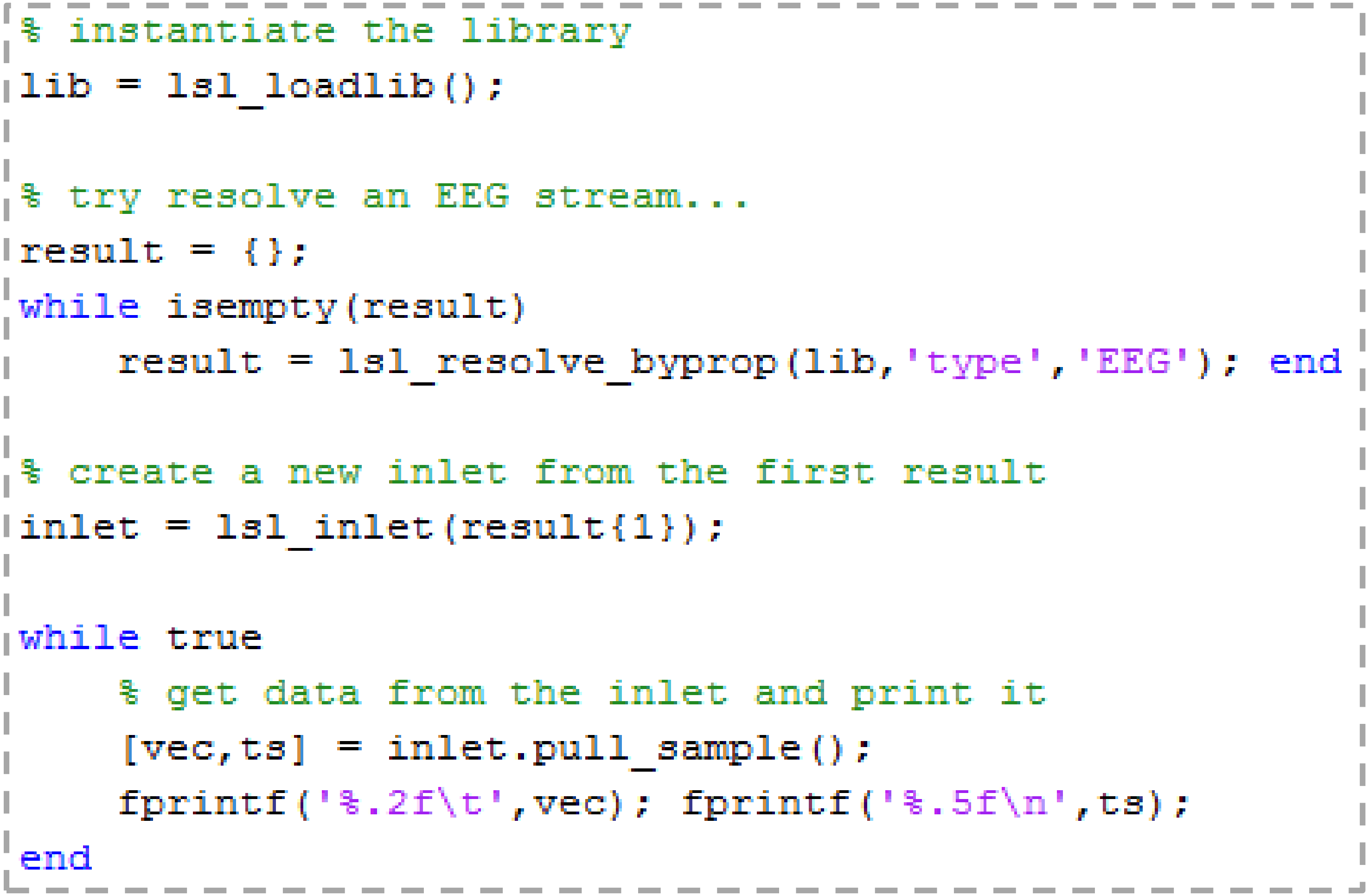
LSL Can Be Easily Integrated Into Programs



```
% instantiate the library
lib = lsl_loadlib();

% make a new stream outlet (name: BioSemi, type: EEG, 8 channels, 100Hz)
info = lsl_streaminfo(lib,'BioSemi','EEG',8,100,'cf_float32','myuid');
outlet = lsl_outlet(info);

% send data into the outlet, sample by sample (8 random numbers each)
while true
    outlet.push_sample(randn(8,1));
    pause(0.01);
end
```

Sample code for sending 8ch EEG (MATLAB)

LSL Can Be Easily Integrated Into Programs



```
% instantiate the library
lib = lsl_loadlib();

% try resolve an EEG stream...
result = {};
while isempty(result)
    result = lsl_resolve_byprop(lib,'type','EEG'); end

% create a new inlet from the first result
inlet = lsl_inlet(result{1});

while true
    % get data from the inlet and print it
    [vec,ts] = inlet.pull_sample();
    fprintf('%.2f\t',vec); fprintf('%.5f\n',ts);
end
```

Sample code for receiving EEG data (MATLAB)

LSL Can Scale to Complex Experiments

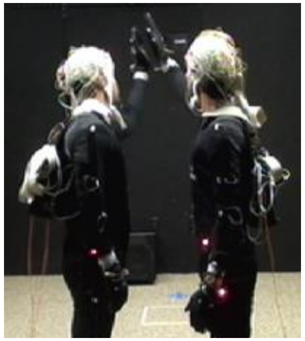
- Acquiring data from multi-modal and multi-vendor brain- and bio-signals

EEG and ExG

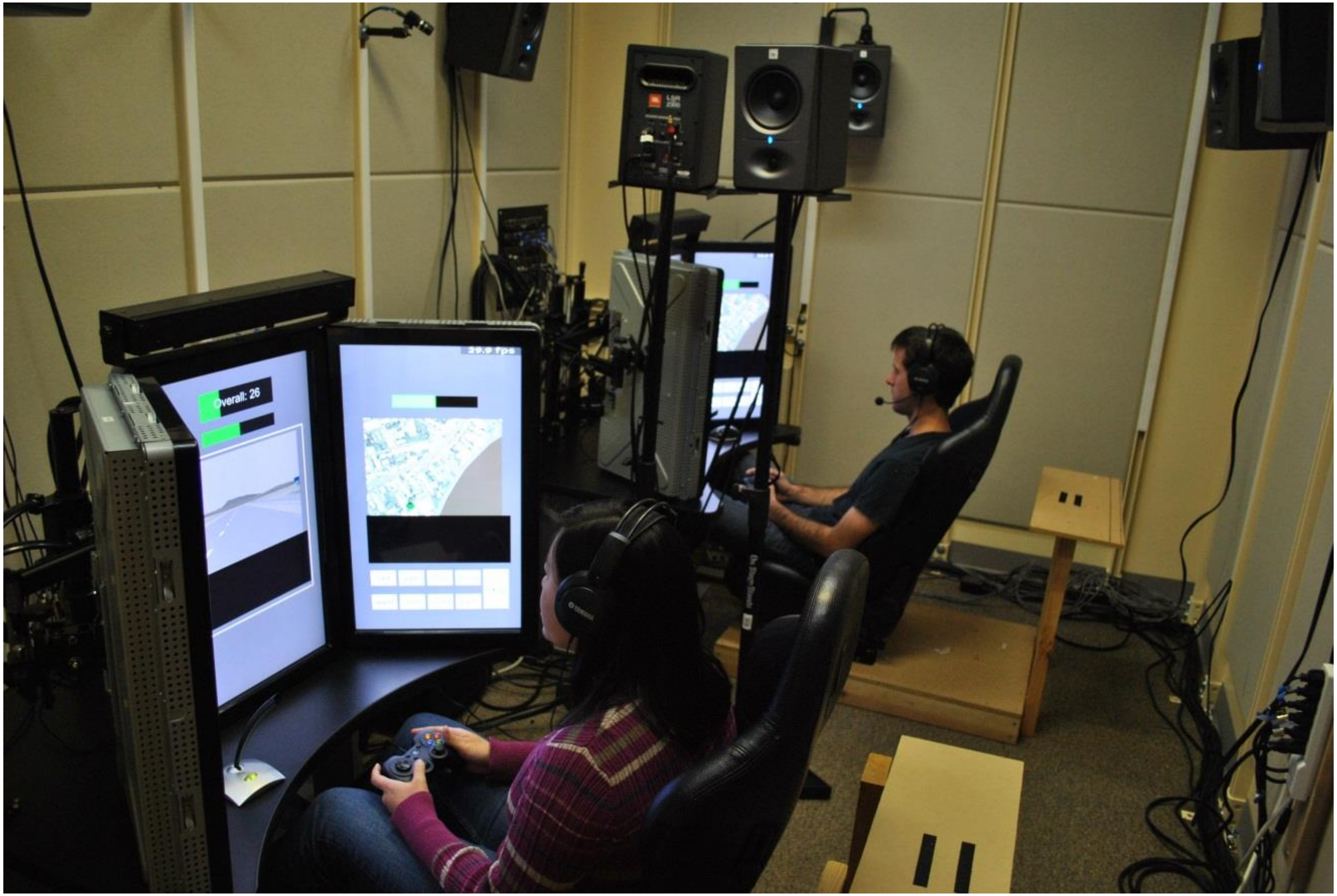
Full-Body Motion Capture

Eye-Tracking

Human Interface Devices, System State, Etc.



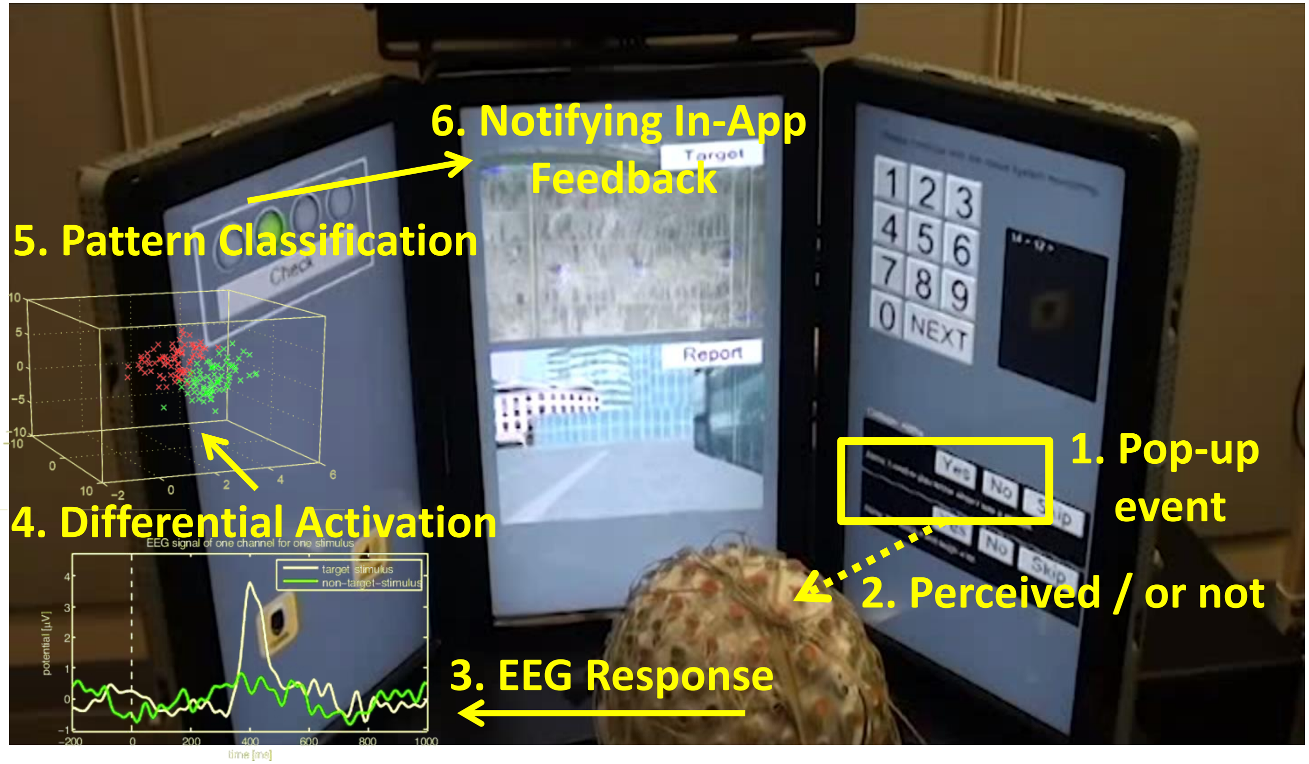
LSL Can Scale to Complex Experiments



STRUM: Small-Team Reconnaissance Urban Missions

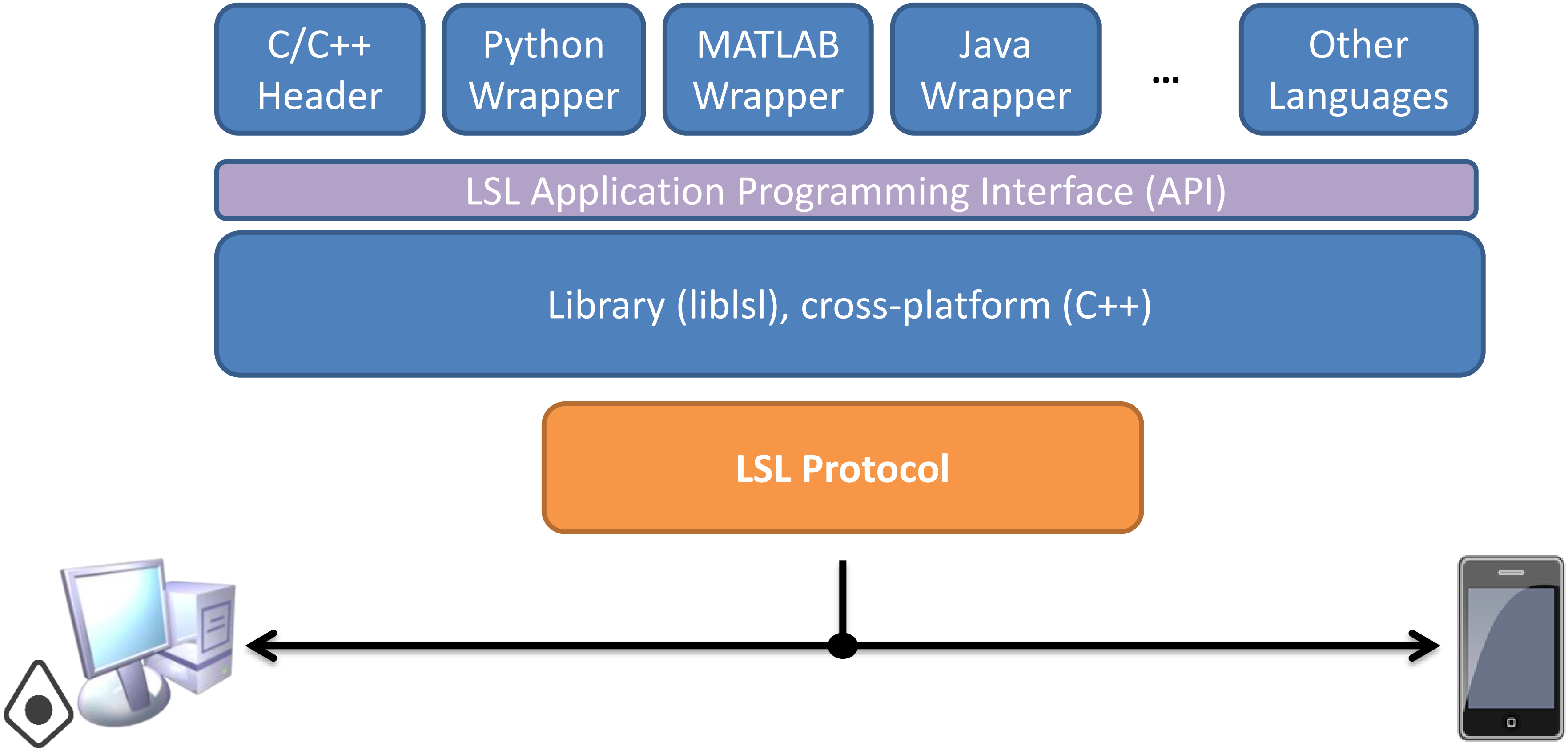
LSL Can Scale to Complex Experiments

- May require online access to multiple device streams from one experiment script



The LSL Software Stack

- The core piece of LSL is a network protocol, a library, and various language interfaces for it



C/C++ Header

Python Wrapper

MATLAB Wrapper

Java Wrapper

…

Other Languages

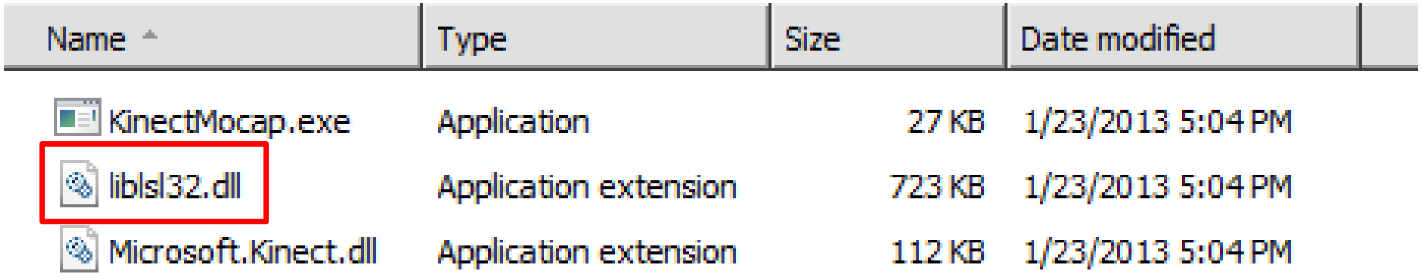
LSL Application Programming Interface (API)

Library (liblsl), cross-platform (C++)

LSL Protocol

The liblsl Library

- Cross-platform C++ library (compiles out-of-the-box for Windows, Mac OS, Linux, Android, 32/64 bit) , IPv4/6
- Stable API (no breaking change since 1y / 1st release)
- Extensive documentation and example code
- High code quality / very few bugs
- Low-overhead implementation (memory, IO, threads, complexity, …), low binary footprint

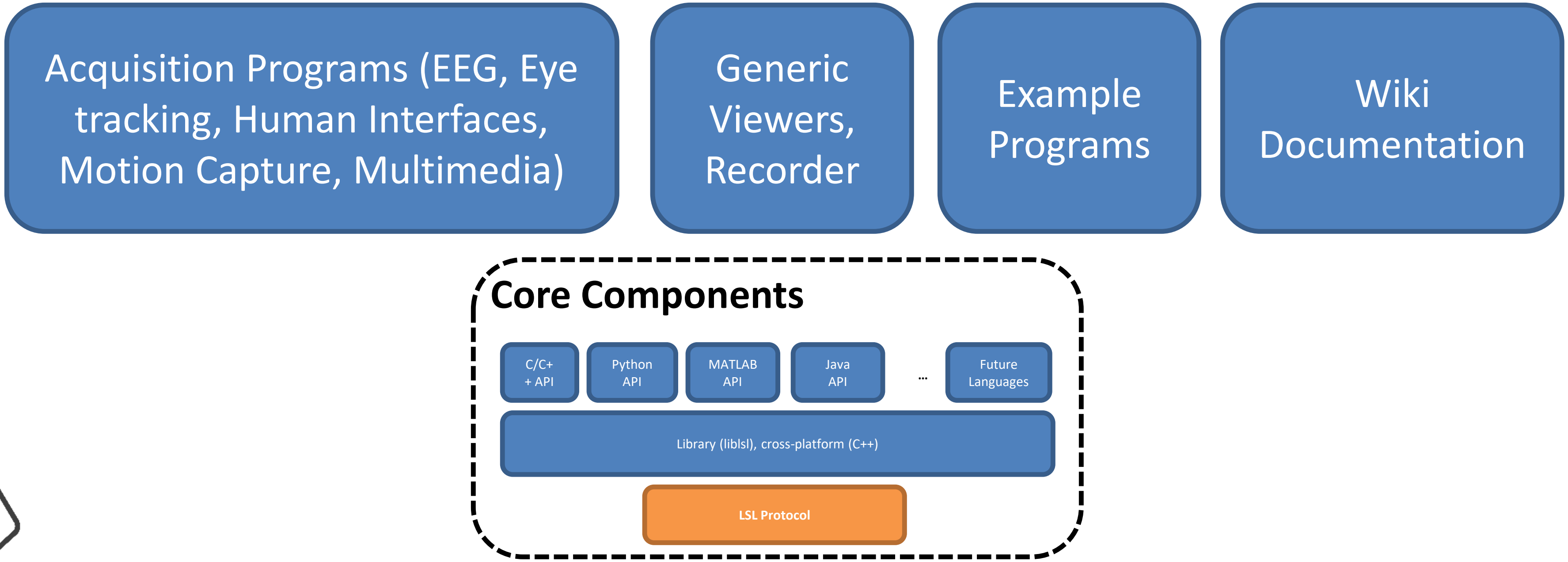


Name	Type	Size	Date modified
KinectMocap.exe	Application	27 KB	1/23/2013 5:04 PM
liblsl32.dll	Application extension	723 KB	1/23/2013 5:04 PM
Microsoft.Kinect.dll	Application extension	112 KB	1/23/2013 5:04 PM

Folder structure of a simple application that supports LSL

The LSL Software Distribution

- The larger distribution includes Documentation, User Guides, Example Programs, Acquisition Programs, Generic Tools
- Everything open source (mostly MIT-licensed)



Acquisition Programs (EEG, Eye tracking, Human Interfaces, Motion Capture, Multimedia)

Generic Viewers, Recorder

Example Programs

Wiki Documentation

Core Components

C/C+ + API

Python API

MATLAB API

Java API

…

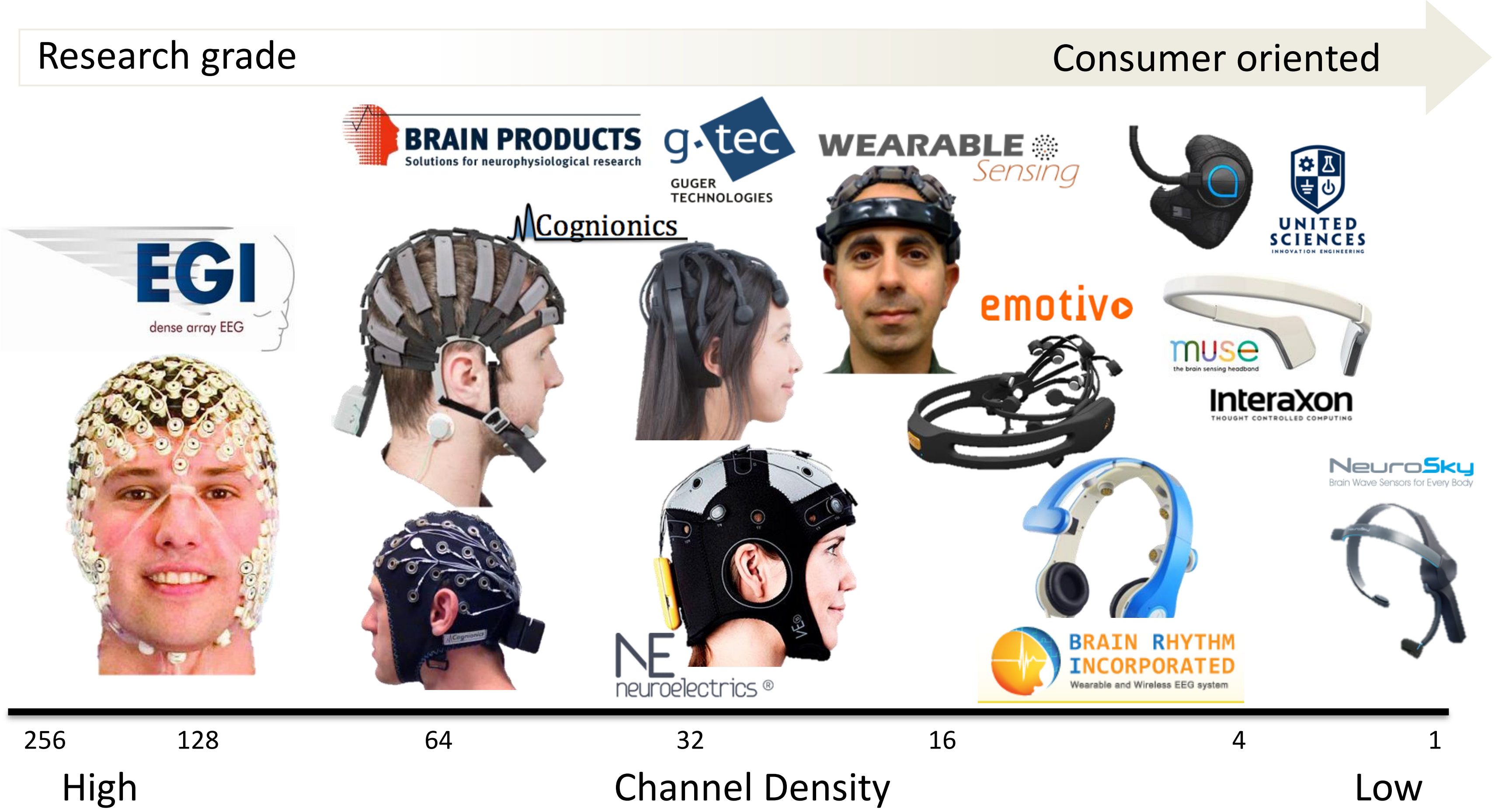
Future Languages

Library (liblsl), cross-platform (C++)

LSL Protocol

Some EEG Solutions Supported by LSL

LSL supports 30+ EEG systems and over 20 other device classes



Research grade

Consumer oriented

256 128 64 32 16 4 1

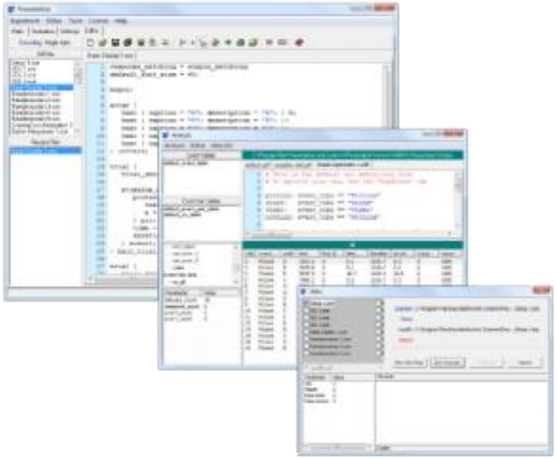
High Channel Density Low

BRAIN PRODUCTS
Solutions for neurophysiological research
g.tec
GUGER TECHNOLOGIES
WEARABLE Sensing
UNITED SCIENCES
INNOVATION ENGINEERING
EGI
dense array EEG
Cognionics
emotiv
muse
the brain sensing headband
InteraXon
THOUGHT CONTROLLED COMPUTING
NeuroSky
Brain Wave Sensors for Every Body
NE
neuroelectrics ®
BRAIN RHYTHM INCORPORATED
Wearable and Wireless EEG system

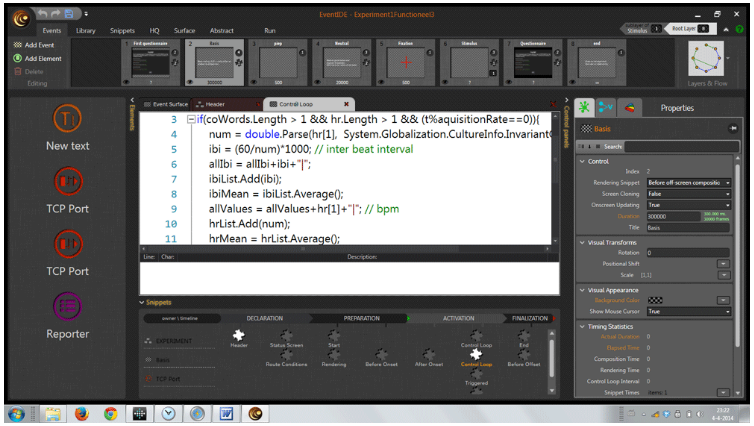
Some Other Device Types on LSL

- Eye Trackers
- Motion Capture
- Game Controllers
- Mice, Keyboards
- Serial Port
- Soundcards & (some) frame grabber cards
- Wearable EMG/ECG devices

Some LSL-Compatible Stimulus Presentation Software

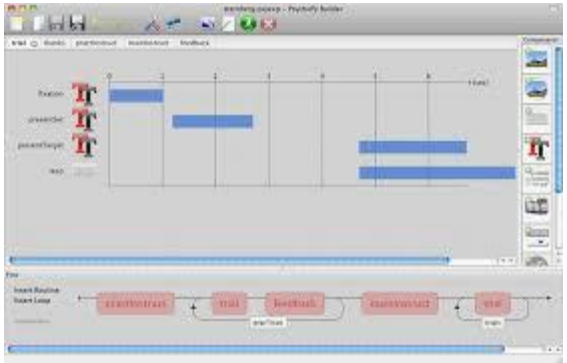


Presentation



```
if(coWords.Length > 1 && hr.Length > 1 && (t%aquisitionRate==0)){
    num = double.Parse(hr[1],  System.Globalization.CultureInfo.Invariant
    ibi = (60/num)*1000; // inter beat interval
    allIbi = allIbi+ibi+"|";
    ibiList.Add(ibi);
    ibiMean = ibiList.Average();
    allValues = allValues+hr[1]+"|"; // bpm
    hrList.Add(num);
    hrMean = hrList.Average();
```

EventIDE



PsychoPy

Unity (with plugin)

PsychToolbox

Currently-unmaintained integrations: SNAP, Unreal

Design Tradeoffs

- Designed for “lab-scale” recording operations:
  - Local: use VPN/broker/bridges to scale across the internet
  - Up to 20 streams per computer fine, 30-100 considered heavy load, likely needs high-end hardware beyond 100 streams (limited by # of USB ports, etc.)
  - Up to 10 computers involved per recording fine, >20 considered excessive, likely requires high-end networking equipment beyond 50 computers

- Designed for “human-scale” operating range:
  - Not a perfect fit for high-energy physics
  - Sub-milisecond time synchronization out of the box
  - Microsecond precision can only be achieved with user-supplied (e.g., GPS/PTP) time stamps
  - Latency <1ms, throughput up to 2MHz and 100MB/s (raw video)

3 Using LSL

(Quick Demo)

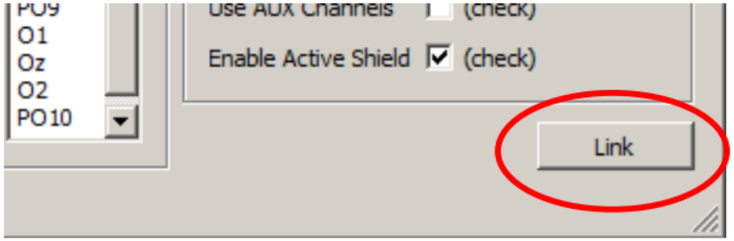
A Typical Experiment Setup with LSL

- “Record data from 2 devices while running a custom stimulus presentation script”

- Software needed for recording
  - Your experiment script (sends event markers)
  - Vendor A Application (e.g., sends EEG)
  - Vendor B Application (e.g., sends MoCap data)
  - Recording Program (LabRecorder)

A Typical Experimenter Workflow

1. Start EEG & MoCap apps, turn on LSL streaming if needed

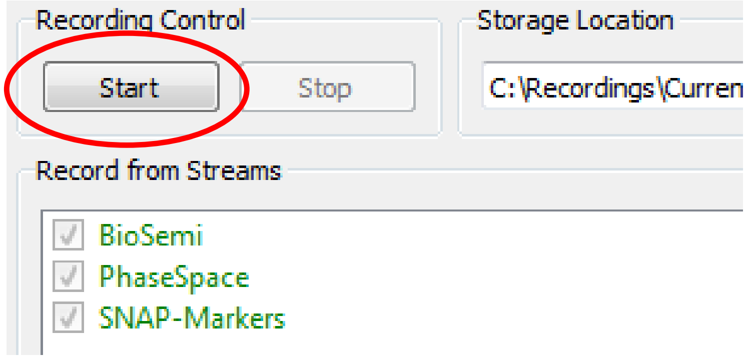


Use AUX Channels □ (check)

Enable Active Shield ☑ (check)

2. Start experiment script in ready mode

3. Open LabRecorder, confirm all LSL streams are there, and then click "Start"



Storage Location: C:\Recordings\Curren

Record from Streams:

☑ BioSemi
☑ PhaseSpace
☑ SNAP-Markers

Coding with LSL: Event Markers

```python
import random
import time

from pylsl import StreamInfo, StreamOutlet

# declare your marker stream information
info = StreamInfo('MyMarkerStream', 'Markers', 1, 0, 'string', ‘myuniqueid2345')

# create an outlet, now the stream is visible
outlet = StreamOutlet(info)

while True:
    # send an event marker
    outlet.push_sample(["Some Event Marker"])
    # do something else
    time.sleep(random.random()*3)
```

Example Code for sending event markers over LSL (Python)

Coding with LSL: Sending Time Series

```
import time
from random import random as rand

from pylsl import StreamInfo, StreamOutlet

# create stream info
info = StreamInfo('BioSemi', 'EEG', 8, 100, 'float32', 'myuid34234')

# create an outlet
outlet = StreamOutlet(info)

while True:
    # make a new random 8-channel sample and send it
    mysample = [rand(), rand(), rand(), rand(), rand(), rand(), rand(), rand()]
    outlet.push_sample(mysample)
    # wait for a bit until we send the next sample
    time.sleep(0.01)
```

Example Code for sending a multi-channel time series over LSL (Python)

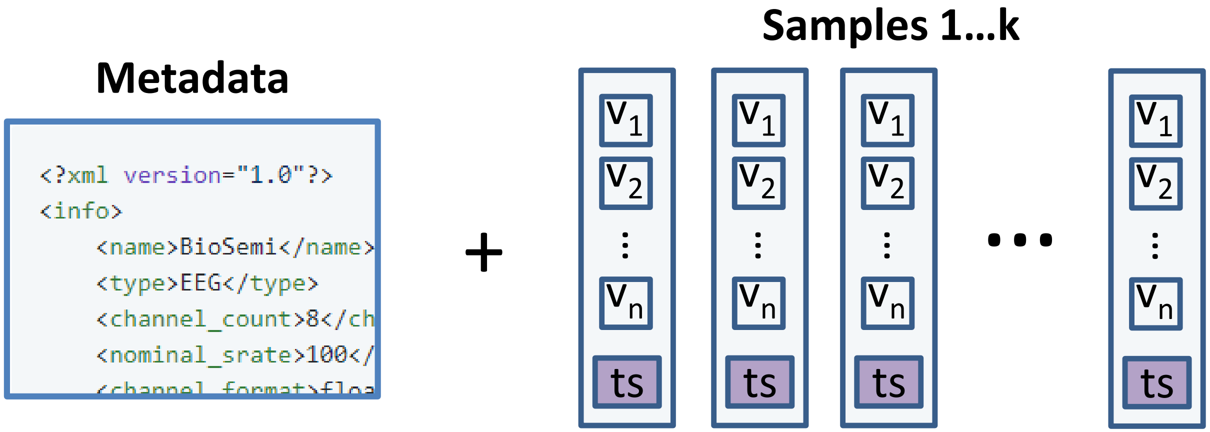
Coding with LSL: Receving Time Series

```python
from pylsl import StreamInlet, resolve_stream

# we wait until we find a stream with type EEG on the lab network... (or
more than one)
streams = resolve_stream('type', 'EEG')

# now that we have it, we create an inlet to read from it
inlet = StreamInlet(streams[0])

while True:
    # wait to get the next sample, also get its timestamp
    sample, timestamp = inlet.pull_sample()
    print(timestamp, sample)
```

Example Code for receiving a multi-channel time series over LSL (Python)

Some Facts Worth Knowing

- LSL doesn’t reorder samples – the data you get out on the other side is always in-order
- LSL doesn’t spuriously drop or lose samples (unless the network connection is interrupted for a long time, default 5 min.)
- For LSL, it’s all just samples: one program can send whole chunks at a time, and the other side can read it sample-by-sample, or vice versa



Samples 1…k

Metadata

```
<?xml version="1.0"?>
<info>
    <name>BioSemi</name>
    <type>EEG</type>
    <channel_count>8</ch
    <nominal_srate>100</
    <channel_format>floa
```

- When a program first starts reading from a stream, it will begin reading from the stream’s next submitted sample onward (e.g., from sample #10053 on)

Some Facts Worth Knowing

- You can add any amount of meta-data to a stream, and for posterity's sake, you *should:*

```
info = StreamInfo('BioSemi', 'EEG', 8, 100, 'float32', 'myuid2424')

# add some meta-data (follow the spec at https://github.com/sccn/xdf/wiki/Meta-Data)
info.desc().append_child("reference").append_child_value("label", "Nasion")

# add some more meta-data
channels = info.desc().append_child("channels")
for c in ["C3", "C4", "Cz", "FPz", "POz", "CPz", "O1", "O2"]:
    chan = channels.append_child("channel")
    chan.append_child_value("name", c)
    chan.append_child_value("unit", "microvolts")
    chan.append_child_value("type", "EEG")
```

- For best compatibility, LSL apps should adhere to the meta-data conventions set forth by the XDF (Extensible Data Format) project, which can be found at: https://github.com/sccn/xdf/wiki/Meta-Data

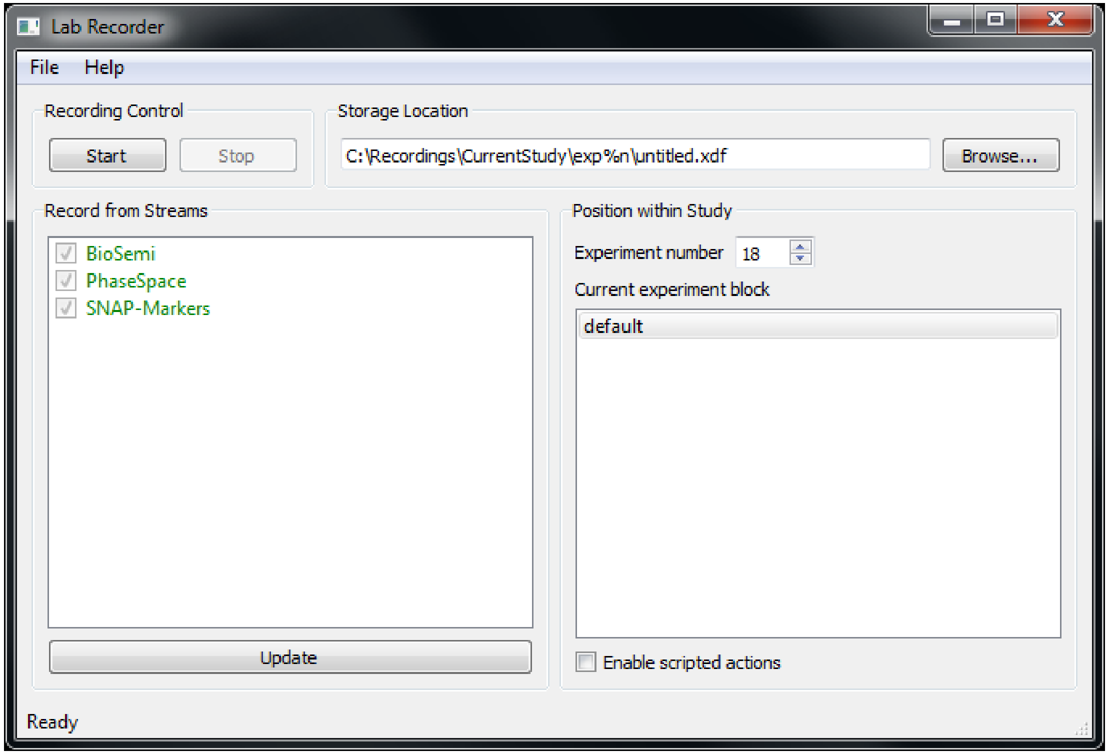
Fault Tolerance with Capital "F"

- Can turn off/on individual devices while recording continues; real-time processing can wait, ignore & warn, or throw error if desired
- Can unplug (and replace) network equipment while recording continues (data is buffered up to several minutes)
- Can restart computers with multiple devices while recording continues
- Can hot-swap computers (and devices under some circumstances) while recording continues
- Can have second backup recording machine
- **Caveat:** Need unique device/source IDs to handle duplicate streams (e.g., serial numbers or custom-assigned numbers)

```
info = StreamInfo('BioSemi', 'EEG', 8, 100, 'float32', 'myuid34234')
```

Ideally unique to your device/data source

Useful Tools: LabRecorder



Storage Location: C:\Recordings\CurrentStudy\exp%n\untitled.xdf

Record from Streams:

- ☑ BioSemi
- ☑ PhaseSpace
- ☑ SNAP-Markers

Position within Study

Experiment number: 18

Current experiment block: default

□ Enable scripted actions

Ready

The LabRecorder can record any number of LSL streams simultaneously into a single file (XDF)

Useful Tools: Viewers

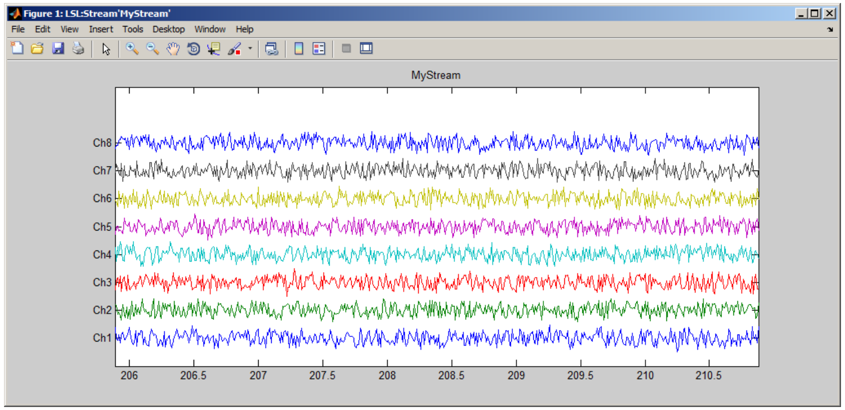
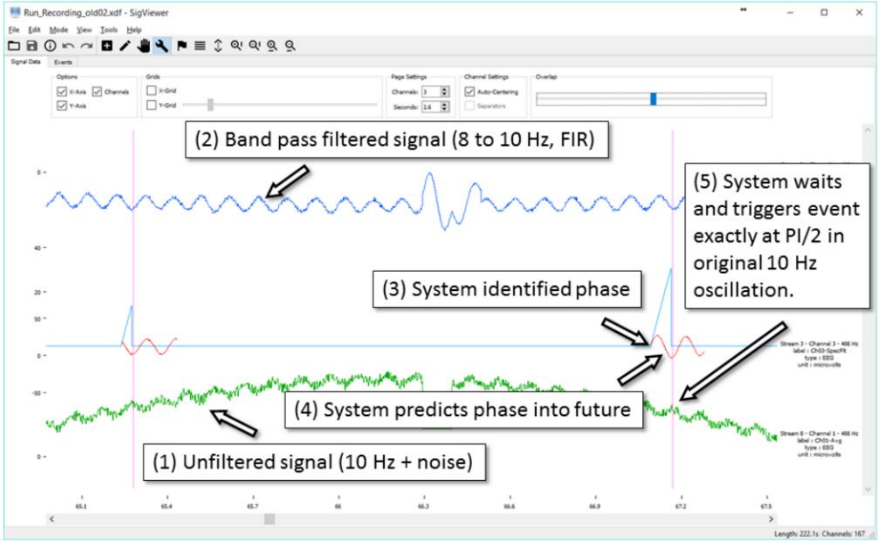


Figure 1: LSL:Stream'MyStream'
File Edit View Insert Tools Desktop Window Help
MyStream
Ch8
Ch7
Ch6
Ch5
Ch4
Ch3
Ch2
Ch1
206
206.5
207
207.5
208
208.5
209
209.5
210
210.5

MATLAB Viewer (included)


Run_Recording_old02.xdf - SigViewer
File Edit Mode View Tools Help
Signal Data
Events

(2) Band pass filtered signal (8 to 10 Hz, FIR)

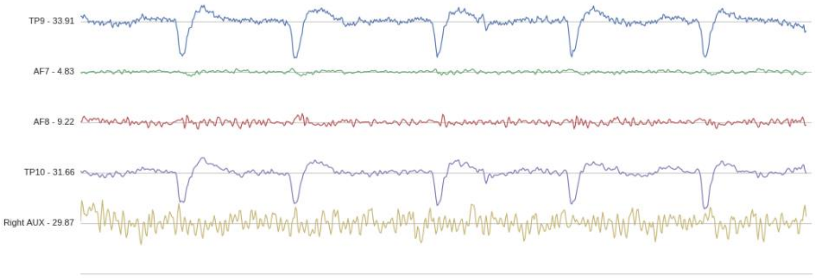
(5) System waits and triggers event exactly at PI/2 in original 10 Hz oscillation.

(3) System identified phase

(4) System predicts phase into future

(1) Unfiltered signal (10 Hz + noise)

SigViewer (offline)

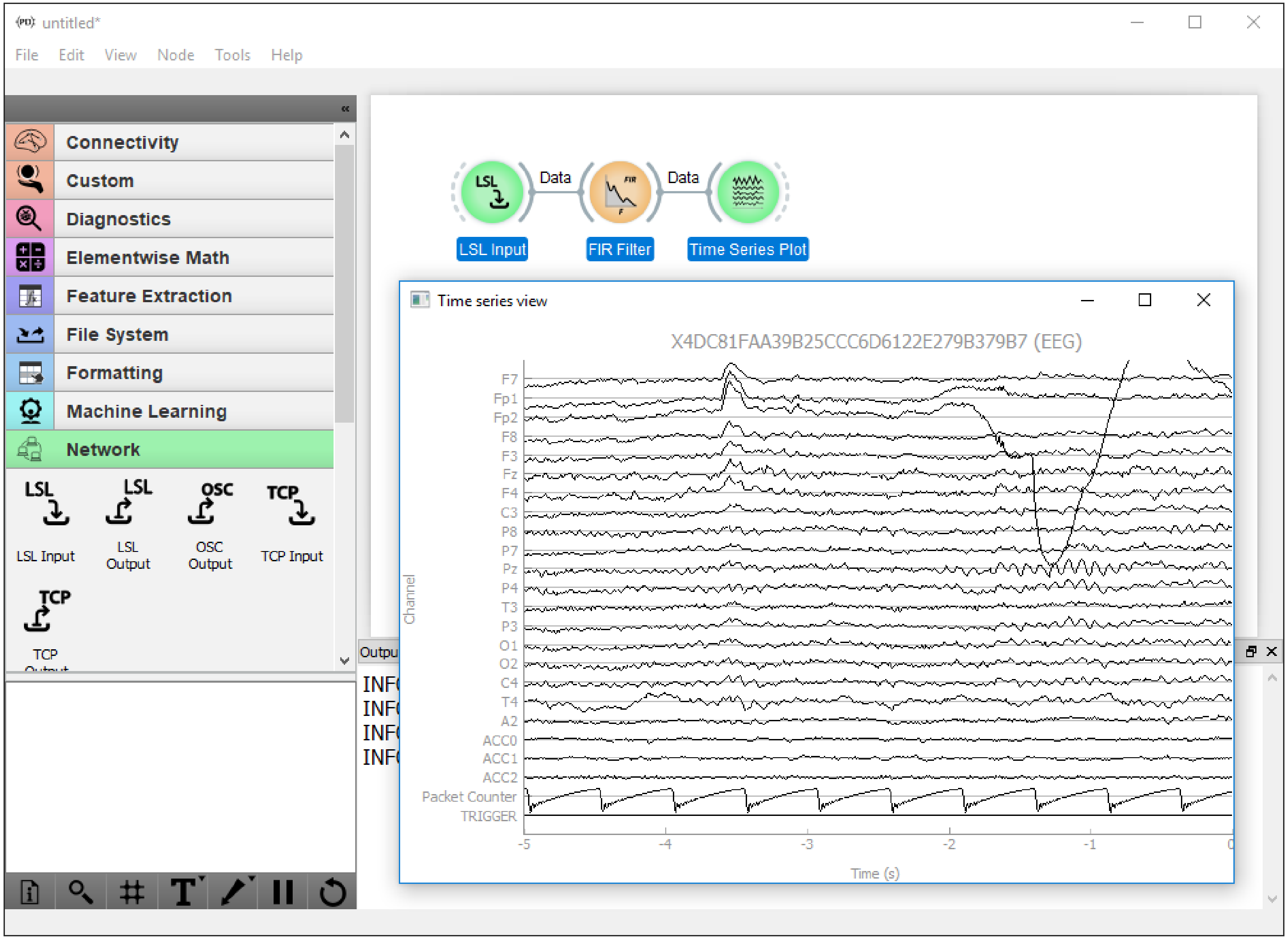


MuseLSL Viewer

?

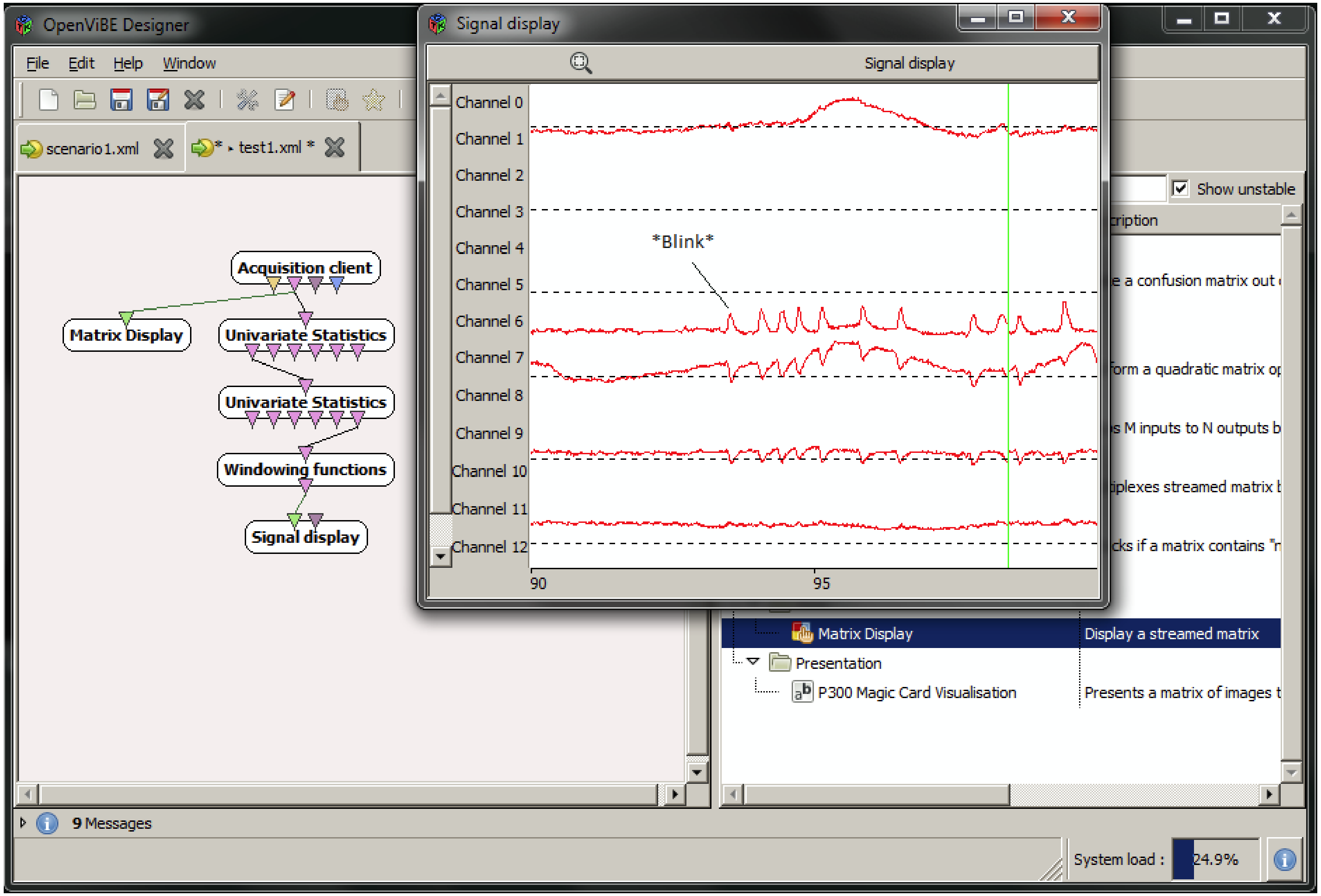
Your Viewer Here?

Useful Tools: Real-Time Processing



NeuroPype Academic Edition

Useful Tools: Real-Time Processing



OpenViBE

Useful Tools: Command-Line Utils

- LSL comes with small utilities out of the box
- Can quickly diagnose network issues etc.
- E.g., `FindAllStreams`, `ReceiveData`, `SendData`, `ReceiveStringMarkers`, `SendStringMarkers`
- Generally available for all platforms

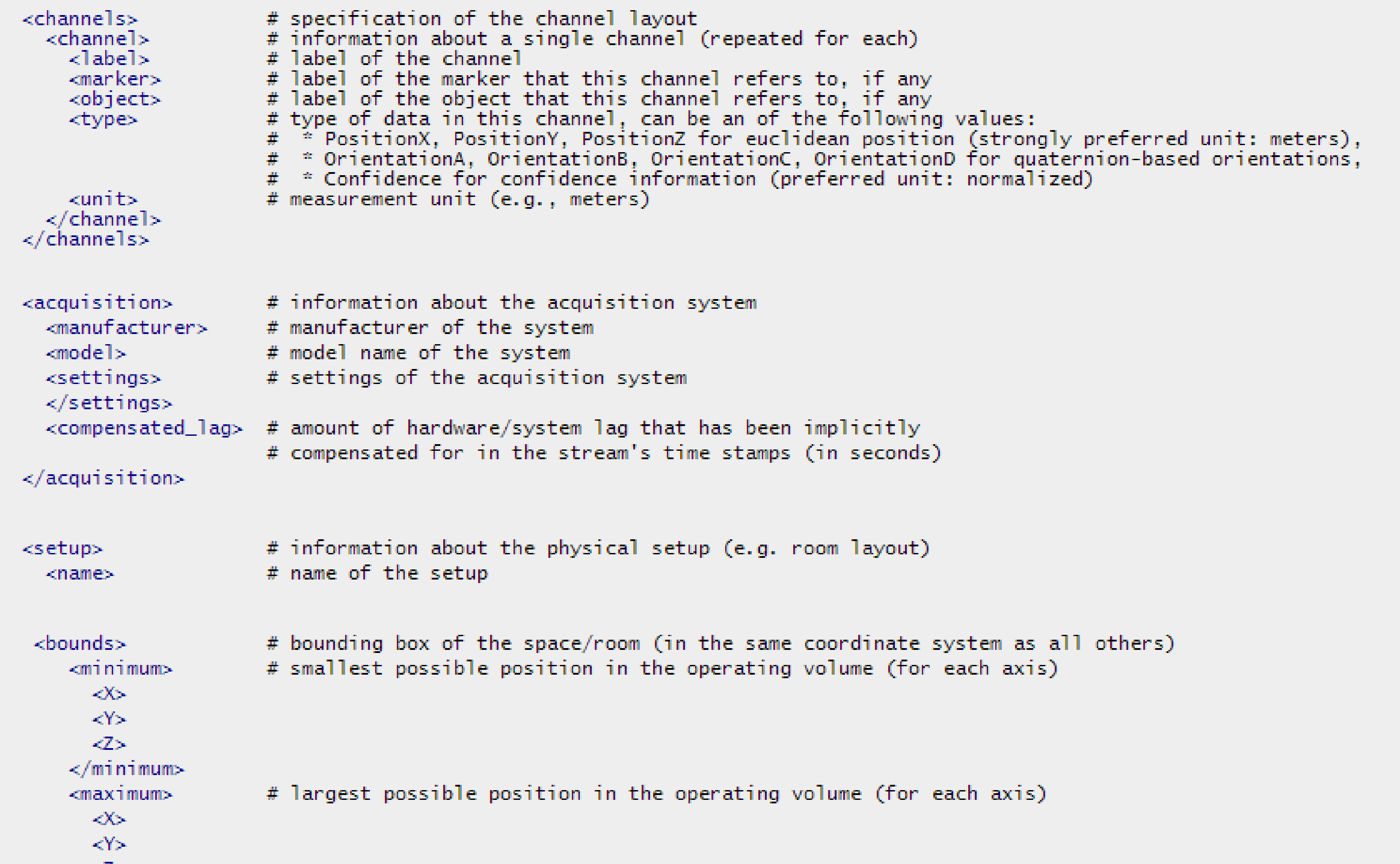
5 The XDF File Format

XDF File Format

- Developed with Clemens Brunner (Graz Univ.)
- Independent of LSL, but supports full feature set (and comes with importers for MATLAB, EEGLAB, BCILAB, MoBILAB, Python)
- Very simple (ca. 100 LoC parser) modern container file format supporting:
  - Any number of streams, time-synched
  - Extensible meta-data per stream with core subset specified online (https://github.com/sccn/xdf)

XDF Extensible Meta-Data

- A portion of the MoCap meta-data specs:



```
<channels>           # specification of the channel layout
  <channel>          # information about a single channel (repeated for each)
    <label>          # label of the channel
    <marker>         # label of the marker that this channel refers to, if any
    <object>         # label of the object that this channel refers to, if any
    <type>           # type of data in this channel, can be an of the following values:
                     #  * PositionX, PositionY, PositionZ for euclidean position (strongly preferred unit: meters),
                     #  * OrientationA, OrientationB, OrientationC, OrientationD for quaternion-based orientations,
                     #  * Confidence for confidence information (preferred unit: normalized)
    <unit>           # measurement unit (e.g., meters)
  </channel>
</channels>


<acquisition>        # information about the acquisition system
  <manufacturer>     # manufacturer of the system
  <model>            # model name of the system
  <settings>         # settings of the acquisition system
  </settings>
  <compensated_lag>  # amount of hardware/system lag that has been implicitly
                     # compensated for in the stream's time stamps (in seconds)
</acquisition>


<setup>              # information about the physical setup (e.g. room layout)
  <name>             # name of the setup


 <bounds>            # bounding box of the space/room (in the same coordinate system as all others)
    <minimum>        # smallest possible position in the operating volume (for each axis)
      <X>
      <Y>
      <Z>
    </minimum>
    <maximum>        # largest possible position in the operating volume (for each axis)
      <X>
      <Y>
```

The “X” in XDF

- No single lab can specify meta-data across full range of relevant data modalities (EEG, fMRI, MoCap, Gaze, Video, …)
- No time to wait for a working group to form and come up with a major consensus on a specification
- Extensible part of the XDF specification is hosted on the web, is grown incrementally by reviewed/invited contributions with very low friction (wiki)
- Private/vendor-specific extensions are permitted in parallel (given some care with naming)
- Can still be summarized into revisions of a more traditional paper standard

Attuned Container Format

- ANSI standard based on XDF 1.0, aimed at industry use (ANSI/CTA-2060-2017)

ANSI/CTA Standard

Standard for Consumer EEG File Format
(Attuned Container Format)

ANSI/CTA-2060

November 2017

ANSI

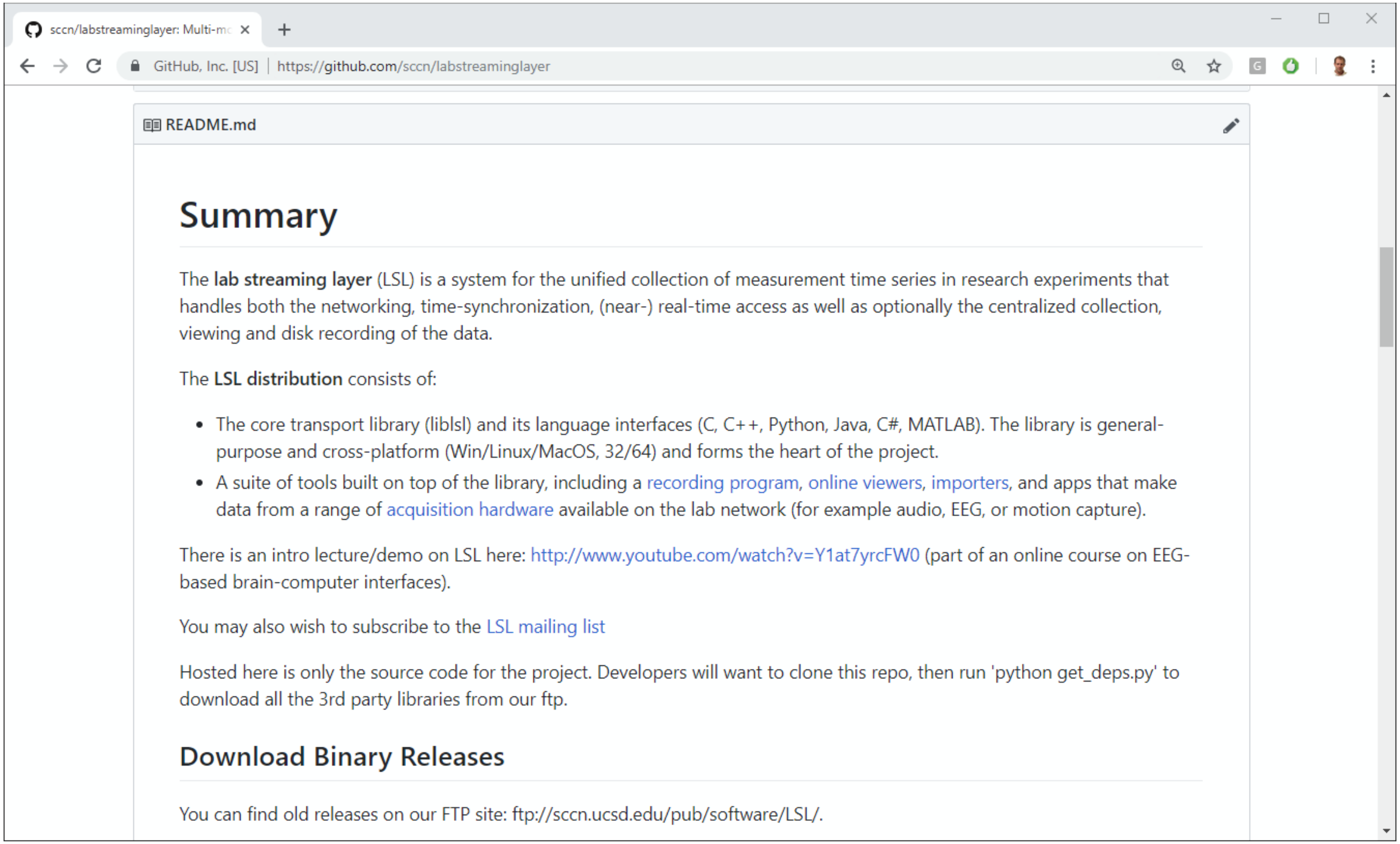
Consumer
Technology
Association

Co-developed in 2017 by Intheon, Wearable Sensing, InteraXon, ARL, and others

4 The LSL Ecosystem

Places to Go
(to Learn More, Get Help, …)

GitHub Home Page of LSL


sccn/labstreaminglayer: Multi-mc
GitHub, Inc. [US] | https://github.com/sccn/labstreaminglayer
README.md

Summary

The **lab streaming layer** (LSL) is a system for the unified collection of measurement time series in research experiments that handles both the networking, time-synchronization, (near-) real-time access as well as optionally the centralized collection, viewing and disk recording of the data.

The **LSL distribution** consists of:

- The core transport library (liblsl) and its language interfaces (C, C++, Python, Java, C#, MATLAB). The library is general-purpose and cross-platform (Win/Linux/MacOS, 32/64) and forms the heart of the project.
- A suite of tools built on top of the library, including a recording program, online viewers, importers, and apps that make data from a range of acquisition hardware available on the lab network (for example audio, EEG, or motion capture).

There is an intro lecture/demo on LSL here: http://www.youtube.com/watch?v=Y1at7yrcFW0 (part of an online course on EEG-based brain-computer interfaces).

You may also wish to subscribe to the LSL mailing list

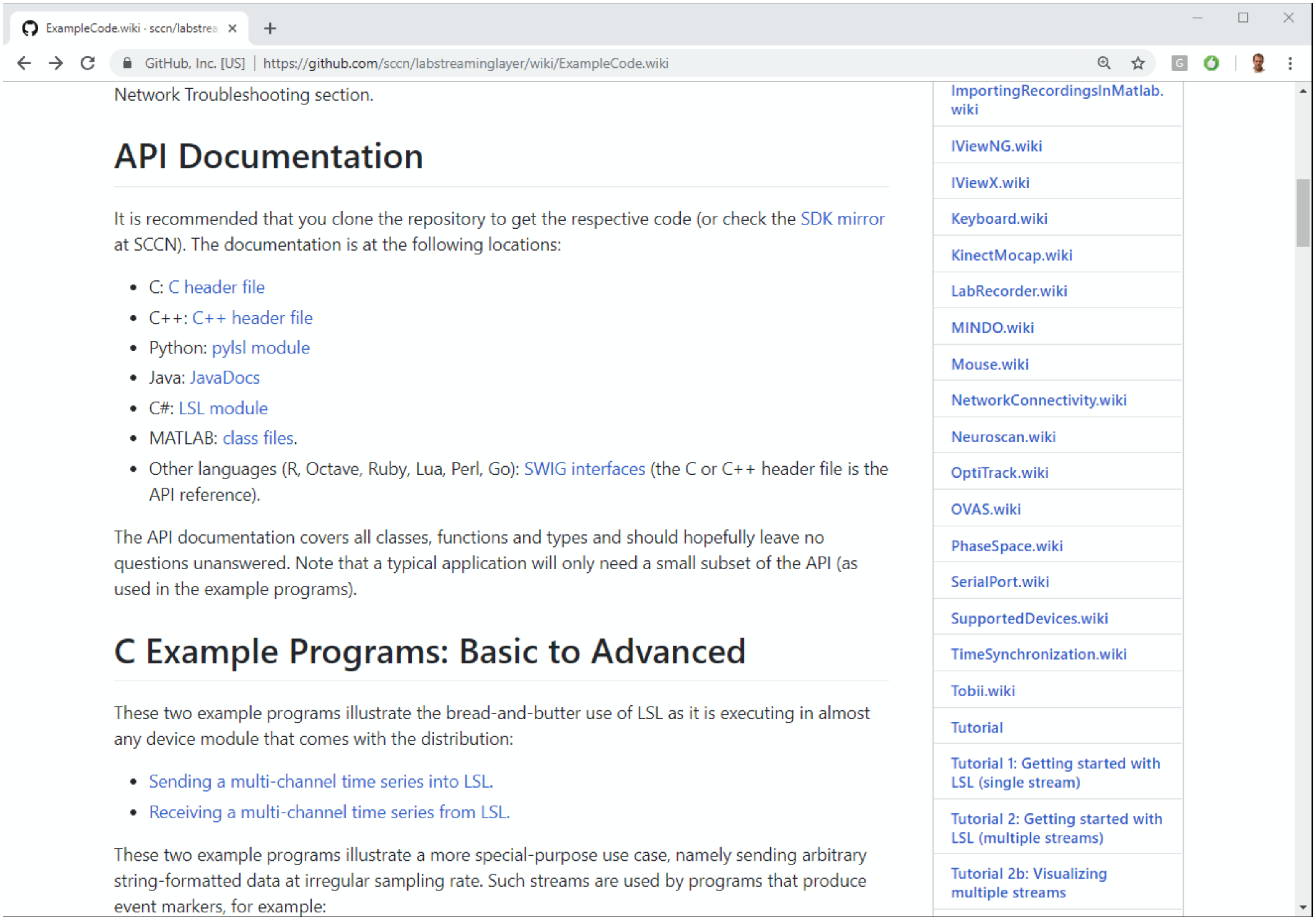
Hosted here is only the source code for the project. Developers will want to clone this repo, then run 'python get_deps.py' to download all the 3rd party libraries from our ftp.

Download Binary Releases

You can find old releases on our FTP site: ftp://sccn.ucsd.edu/pub/software/LSL/.

https://github.com/sccn/labstreaminglayer
https://github.com/labstreaminglayer

LSL Wiki (also on GitHub)


ExampleCode.wiki · sccn/labstrea
GitHub, Inc. [US]
https://github.com/sccn/labstreaminglayer/wiki/ExampleCode.wiki

Network Troubleshooting section.

API Documentation

It is recommended that you clone the repository to get the respective code (or check the SDK mirror at SCCN). The documentation is at the following locations:

- C: C header file
- C++: C++ header file
- Python: pylsl module
- Java: JavaDocs
- C#: LSL module
- MATLAB: class files.
- Other languages (R, Octave, Ruby, Lua, Perl, Go): SWIG interfaces (the C or C++ header file is the API reference).

The API documentation covers all classes, functions and types and should hopefully leave no questions unanswered. Note that a typical application will only need a small subset of the API (as used in the example programs).

C Example Programs: Basic to Advanced

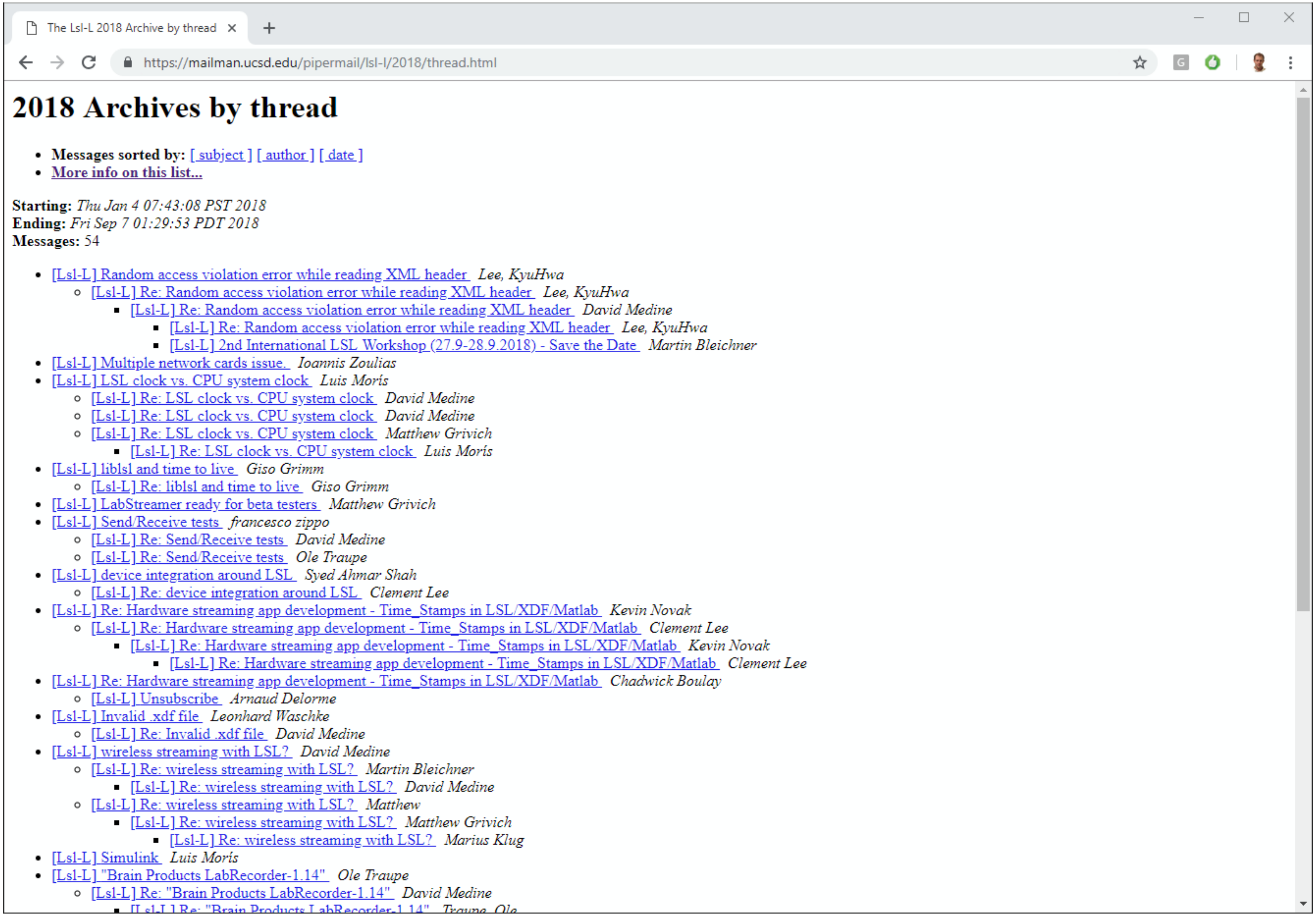
These two example programs illustrate the bread-and-butter use of LSL as it is executing in almost any device module that comes with the distribution:

- Sending a multi-channel time series into LSL.
- Receiving a multi-channel time series from LSL.

These two example programs illustrate a more special-purpose use case, namely sending arbitrary string-formatted data at irregular sampling rate. Such streams are used by programs that produce event markers, for example:

- ImportingRecordingsInMatlab.wiki
- IViewNG.wiki
- IViewX.wiki
- Keyboard.wiki
- KinectMocap.wiki
- LabRecorder.wiki
- MINDO.wiki
- Mouse.wiki
- NetworkConnectivity.wiki
- Neuroscan.wiki
- OptiTrack.wiki
- OVAS.wiki
- PhaseSpace.wiki
- SerialPort.wiki
- SupportedDevices.wiki
- TimeSynchronization.wiki
- Tobii.wiki
- Tutorial
- Tutorial 1: Getting started with LSL (single stream)
- Tutorial 2: Getting started with LSL (multiple streams)
- Tutorial 2b: Visualizing multiple streams

LSL Mailing List & Slack Channel


The Lsl-L 2018 Archive by thread
https://mailman.ucsd.edu/pipermail/lsl-l/2018/thread.html

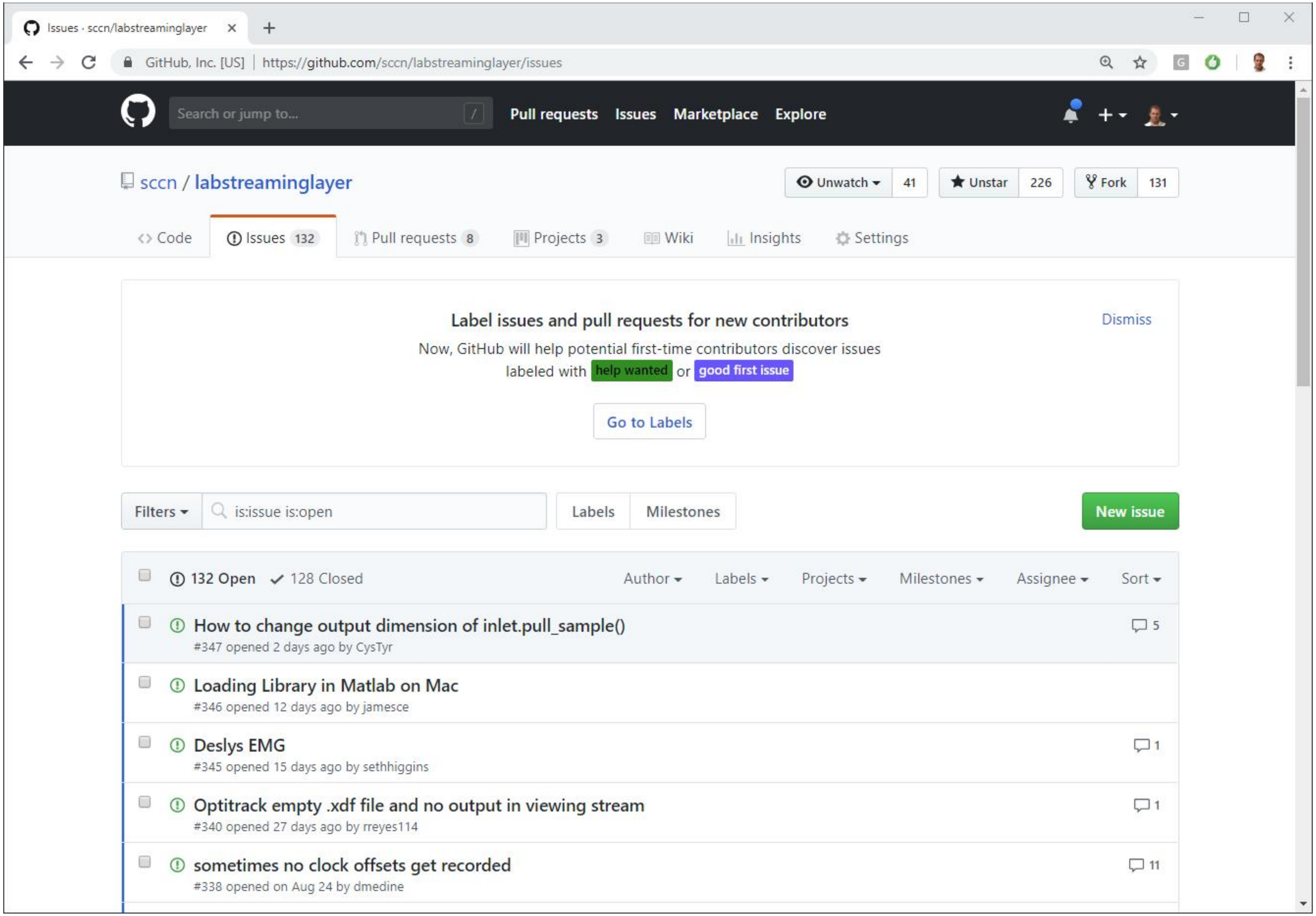
2018 Archives by thread

- **Messages sorted by:** [subject] [author] [date]
- **More info on this list...**

Starting: *Thu Jan 4 07:43:08 PST 2018*
Ending: *Fri Sep 7 01:29:53 PDT 2018*
Messages: 54

- [Lsl-L] Random access violation error while reading XML header *Lee, KyuHwa*
  - [Lsl-L] Re: Random access violation error while reading XML header *Lee, KyuHwa*
    - [Lsl-L] Re: Random access violation error while reading XML header *David Medine*
      - [Lsl-L] Re: Random access violation error while reading XML header *Lee, KyuHwa*
      - [Lsl-L] 2nd International LSL Workshop (27.9-28.9.2018) - Save the Date *Martin Bleichner*
- [Lsl-L] Multiple network cards issue. *Ioannis Zoulias*
- [Lsl-L] LSL clock vs. CPU system clock *Luis Moris*
  - [Lsl-L] Re: LSL clock vs. CPU system clock *David Medine*
  - [Lsl-L] Re: LSL clock vs. CPU system clock *David Medine*
  - [Lsl-L] Re: LSL clock vs. CPU system clock *Matthew Grivich*
    - [Lsl-L] Re: LSL clock vs. CPU system clock *Luis Moris*
- [Lsl-L] liblsl and time to live *Giso Grimm*
  - [Lsl-L] Re: liblsl and time to live *Giso Grimm*
- [Lsl-L] LabStreamer ready for beta testers *Matthew Grivich*
- [Lsl-L] Send/Receive tests *francesco zippo*
  - [Lsl-L] Re: Send/Receive tests *David Medine*
  - [Lsl-L] Re: Send/Receive tests *Ole Traupe*
- [Lsl-L] device integration around LSL *Syed Ahmar Shah*
  - [Lsl-L] Re: device integration around LSL *Clement Lee*
- [Lsl-L] Re: Hardware streaming app development - Time_Stamps in LSL/XDF/Matlab *Kevin Novak*
  - [Lsl-L] Re: Hardware streaming app development - Time_Stamps in LSL/XDF/Matlab *Clement Lee*
    - [Lsl-L] Re: Hardware streaming app development - Time_Stamps in LSL/XDF/Matlab *Kevin Novak*
      - [Lsl-L] Re: Hardware streaming app development - Time_Stamps in LSL/XDF/Matlab *Clement Lee*
- [Lsl-L] Re: Hardware streaming app development - Time_Stamps in LSL/XDF/Matlab *Chadwick Boulay*
  - [Lsl-L] Unsubscribe *Arnaud Delorme*
- [Lsl-L] Invalid .xdf file *Leonhard Waschke*
  - [Lsl-L] Re: Invalid .xdf file *David Medine*
- [Lsl-L] wireless streaming with LSL? *David Medine*
  - [Lsl-L] Re: wireless streaming with LSL? *Martin Bleichner*
    - [Lsl-L] Re: wireless streaming with LSL? *David Medine*
  - [Lsl-L] Re: wireless streaming with LSL? *Matthew*
    - [Lsl-L] Re: wireless streaming with LSL? *Matthew Grivich*
      - [Lsl-L] Re: wireless streaming with LSL? *Marius Klug*
- [Lsl-L] Simulink *Luis Moris*
- [Lsl-L] "Brain Products LabRecorder-1.14" *Ole Traupe*
  - [Lsl-L] Re: "Brain Products LabRecorder-1.14" *David Medine*
    - [Lsl-L] Re: "Brain Products LabRecorder-1.14" *Traupe, Ole*

GitHub Issues (Bug Tracker)



Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with help wanted or good first issue

□ 132 Open ✓ 128 Closed Author Labels Projects Milestones Assignee Sort

□ How to change output dimension of inlet.pull_sample() 5
#347 opened 2 days ago by CysTyr

□ Loading Library in Matlab on Mac
#346 opened 12 days ago by jamesce

□ Deslys EMG 1
#345 opened 15 days ago by sethhiggins

□ Optitrack empty .xdf file and no output in viewing stream 1
#340 opened 27 days ago by rreyes114

□ sometimes no clock offsets get recorded 11
#338 opened on Aug 24 by dmedine

LSL Support from Industry

mBrainTrain

WE DON'T WAIT FOR THE FUTURE. WE LIVE IT.

BRAIN PRODUCTS

Solutions for neurophysiological research

NE

neuroelectrics

EBNEURO

Cognionics

ant neuro

inspiring technology

OPENBCI

neuro behavioral systems

intheon

EMOTIV

InteraXon

WEARABLE Sensing

*Incomplete List

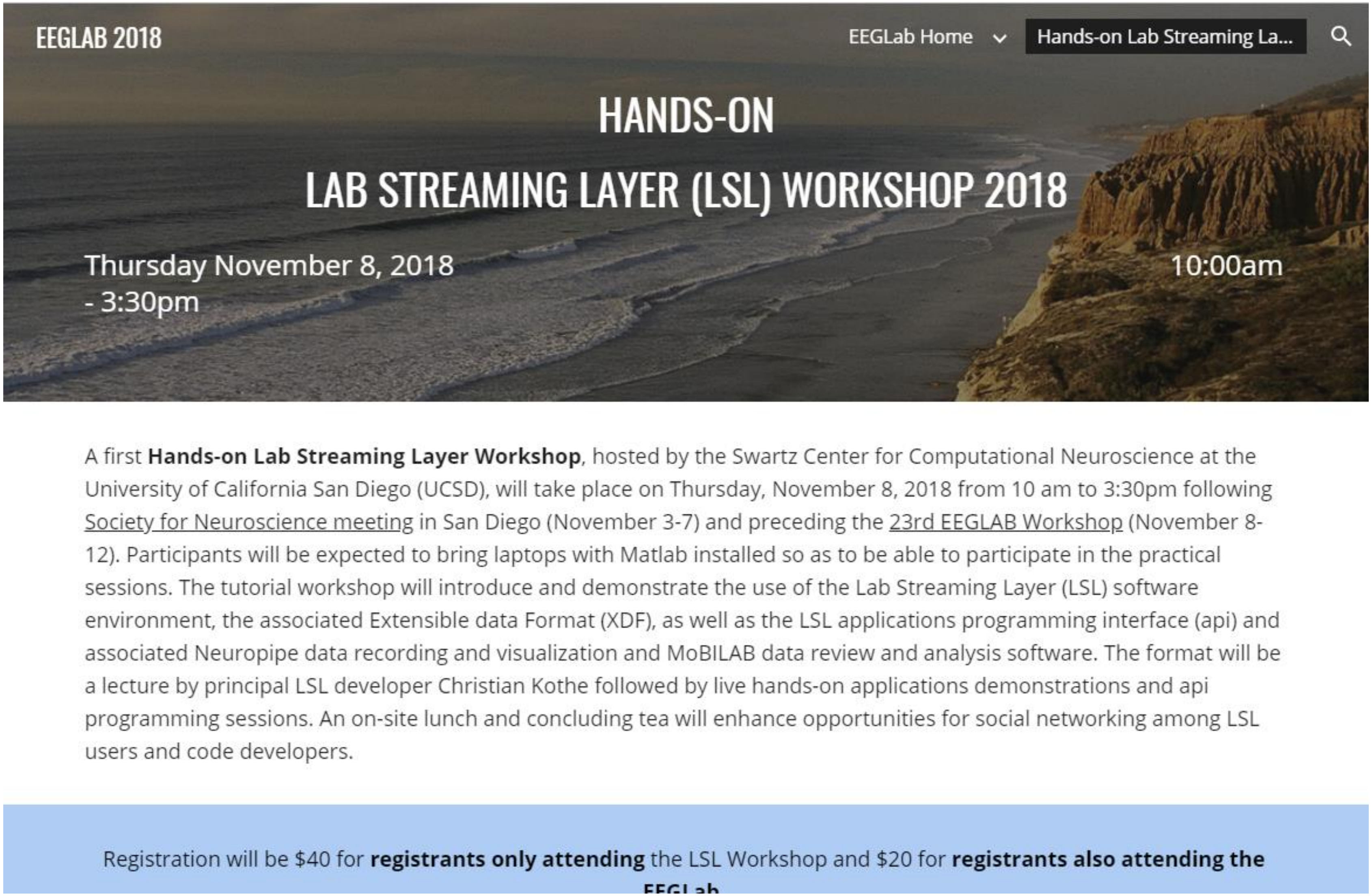
Places to Meet (Hack Devices, Socialize, etc)

IEEE SMC

IEEE SMC Budapest Hackathon 2016

IEEE SMC Hackathons San Diego 2016, Budapest 2016, Banff 2017, Miyazaki 2018

Workshops in San Diego



HANDS-ON

LAB STREAMING LAYER (LSL) WORKSHOP 2018

Thursday November 8, 2018 10:00am - 3:30pm

A first **Hands-on Lab Streaming Layer Workshop**, hosted by the Swartz Center for Computational Neuroscience at the University of California San Diego (UCSD), will take place on Thursday, November 8, 2018 from 10 am to 3:30pm following Society for Neuroscience meeting in San Diego (November 3-7) and preceding the 23rd EEGLAB Workshop (November 8-12). Participants will be expected to bring laptops with Matlab installed so as to be able to participate in the practical sessions. The tutorial workshop will introduce and demonstrate the use of the Lab Streaming Layer (LSL) software environment, the associated Extensible data Format (XDF), as well as the LSL applications programming interface (api) and associated Neuropipe data recording and visualization and MoBILAB data review and analysis software. The format will be a lecture by principal LSL developer Christian Kothe followed by live hands-on applications demonstrations and api programming sessions. An on-site lunch and concluding tea will enhance opportunities for social networking among LSL users and code developers.

Registration will be $40 for **registrants only attending** the LSL Workshop and $20 for **registrants also attending the EEGLab**

Nov 8th, 2018 at UCSD

Workshops in Germany

SEPTEMBER 27-28, 2018

Second International Workshop on LSL

HANSE-WISSENSCHAFTSKOLLEG
DELMENHORST, GERMANY

Enjoy the workshop! ☺

Next: Q&A until 3pm