EEG algorithm SDK for OSX: Development Guide

February 1, 2016

NeuroSky
Brain-Computer Interface Technologies

The NeuroSky® product families consist of hardware and software components for simple integration of this biosensor technology into consumer and industrial end-applications. All products are designed and manufactured to meet consumer thresholds for quality, pricing, and feature sets. NeuroSky sets itself apart by providing building block component solutions that offer friendly synergies with related and complementary technological solutions.

Contents

About the OSX SDK

This document will guide you through the process of generating algorithm outputs from different NeuroSky Proprietary Mind Algorithms using **NeuroSky EEG Algorithm SDK for OSX** with EEG data collected by NeuroSky Biosensor System (e.g. TGAM module or MindWave Mobile Headset).

This development guide is intended for *OSX application developers* who are already familiar with standard OSX development using **Xcode** and Apple's OSX SDK. If you are not already familiar with developing for OSX, please first visit Apple's web site for instruction and tools to develop OSX applications.

Important: .

- Requires OS X 10.10+ or later

EEG Algorithm SDK for OSX Contents

- EEG Algorithm SDK for OSX: Development Guide (this document)
- EEG Algorithm SDK framework: AlgoSdk.framework
- Readme: Readme file
- Algo SDK Sample project

Application development

Introduction

We recommend developers to use our COMM SDK for OS X in their application. Comm SDK reduces the complexity of managing EEG Algorithm SDK connections and handles data stream parsing. With the help of our SDKs, the application could be as simple as passing the data received and parsed by the Comm SDK to specific function call(s) at Algo SDK. Specific EEG algorithm index would then be returned accordingly.

EEG Algorithm Sample Project

For collecting EEG data from NeuroSky Biosensor System (e.g. MindWave Mobile headset) with Mac device, please refer to our COMM SDK for OS X document.

Algo SDK Sample is an sample OS X application using Communication (Comm) SDK to connect to NeuroSky Biosensor System (e.g. MindWave Mobile headset) and Algorithm (Algo) SDK for algorithmic computation for specific NeuroSky Mind algorithm, including Attention, Meditation, Appreciation, Mental Effort and Familiarity.

1. On the Mac development machine, double click the Algo SDK Sample Xcode project (**"Algo SDK Sample.xcodeproj"**) to launch the project with Xcode
2. Update the project configuration in order to run the application on the Mac development machine
3. Select **Product** —> **Run** to build and run the "Algo SDK Sample" app
4. In the sample app:
   (a) realtime Attention and Meditation indices would be shown on the progress bar.
   (b) the eye blink indicator will flash (yellow color) when eye blink is detected.

Important: .

- The sample project requires XCode 6.0 or later.
- The sample project only demonstrates how to iterate with the EEG Algo SDK.
- **Comm SDK** enclosed in the sample project is **version 1.15**. Please make sure you are using the latest stable Comm SDK version and make proper changes on sample project if needed.
- It may not be completely compliant with Apple's guidelines for building deploy-able applications.

API Documentation

The **EEG Algorithm SDK API Reference** in this section contains descriptions of the classes and protocols available in the EEG Algorithm OSX API.

Data Types

```
/* EEG data signal quality definitions */
typedef NS_ENUM(NSInteger, NskAlgoSignalQuality) {
    NskAlgoSignalQualityGood,           /* Signal quality is in good level */
    NskAlgoSignalQualityMedium,         /* Signal quality is in medium level */
    NskAlgoSignalQualityPoor,           /* Signal quality is in poor level */
    NskAlgoSignalQualityNotDetected     /* Sensor signal is not detected */
};

/* SDK state definitions */
typedef NS_ENUM(NSInteger, NskAlgoState) {
    NskAlgoStateInited = 1,             /* Algo SDK initialized */
    NskAlgoStateRunning,                /* Algo SDK is performing analysis (i.e. startProcess() 
invoked) */
    NskAlgoStateCollectingBaselineData, /* Algo SDK is collecting baseline data [RESERVED] */
    NskAlgoStateStop,                   /* Algo SDK stops data analysis/baseline collection 
[RESERVED] */
    NskAlgoStatePause,                  /* Algo SDK pauses data analysis */
    NskAlgoStateUninited,               /* Algo SDK is uninitialized */
    NskAlgoStateAnalysingBulkData       /* Algo SDK is analysing a bulk of EEG data [RESERVED] */
};

/* SDK state change reason definitions */
typedef NS_ENUM(NSInteger, NskAlgoReason) {
    NskAlgoReasonConfigChanged = 1,      /* RESERVED: SDK configuration changed */
    NskAlgoReasonUserProfileChanged,     /* RESERVED: Active user profile has been changed */
    NskAlgoReasonUserTrigger,            /* User triggers */
    NskAlgoReasonBaselineExpired,        /* RESERVED: Baseline expired */
    NskAlgoReasonNoBaseline,             /* RESERVED: No baseline data collected yet */
    NskAlgoReasonSignalQuality           /* Due to signal quality */
};

/* EEG algorithm type definitions */
typedef NS_ENUM(NSInteger, NskAlgoEegType) {
    NskAlgoEegTypeAtt   = 0x008,          /* Attention */
    NskAlgoEegTypeMed   = 0x010,          /* Meditation */
    NskAlgoEegTypeBlink = 0x080,          /* Eye Blink Detection */
    NskAlgoEegTypeBP    = 0x800           /* EEG Bandpower */
};

/* EEG data type definitions (data from COMM SDK) */
typedef NS_ENUM(NSInteger, NskAlgoDataType) {
    NskAlgoDataTypeEEG,                  /* Raw EEG data */
    NskAlgoDataTypeAtt,                  /* Attention data */
    NskAlgoDataTypeMed,                 /* Meditation data */
    NskAlgoDataTypePQ,                  /* Poor signal quality data */
    NskAlgoDataTypeBulkEEG,             /* Bulk EEG data (must be multiple of 512, i.e. Ns of
continuous GOOD EEG data */
};
```

SDK Delegate Methods

stateChanged

EEG Algo SDK state change notification delegate method

```
// Required
- (void) stateChanged: (NskAlgoState)state reason:(NskAlgoReason)reason;
```

Note: .

- Developer will always need to check with the SDK state and perform proper GUI handling

attAlgoIndex

Attention Algorithm index notification delegate method

```
// Required
- (void) attAlgoIndex: (NSNumber*)att_index;
```

Note: .

- Attention algorithm has a fixed output interval of 1 second
- Attention algorithm index in range of 0 - 100 where higher the attention index, higher the attention level

medAlgoIndex

Meditation Algorithm index notification delegate method

```
// Required
- (void) medAlgoIndex: (NSNumber*)med_index;
```

Note: .

- Meditation algorithm has a fixed output interval of 1 second
- Meditation algorithm index in range of 0 - 100 where higher the meditation index, higher the meditation level

eyeBlinkDetect

Eye blink detection notification delegate method

```
// Optional
- (void) eyeBlinkDetect: (NSNumber*)strength;
```

Note: .

- **eyeBlinkDetect** will be invoked when eye blink is detected from the provided EEG data

bpAlgoIndex

EEG Bandpower delegate method

```
// Optional
- (void) bpAlgoIndex: (NSNumber*)delta theta:(NSNumber*)theta alpha:(NSNumber*)alpha
beta:(NSNumber*)beta gamma:(NSNumber*)gamma;
```

Note: .

- EEG bandpower values (in dB) have a fixed output interval of 1 second

signalQuality

EEG data signal quality notification delegate method

```
// Optional
- (void) signalQuality: (NskAlgoSignalQuality)signalQuality;
```

Note: .

- Signal Quality was measured and reported at a fixed output interval of 1 second
- SDK state will be changed from **RUNNING** to **PAUSE** when signal quality is poor or sensor off-head is detected. It would return to its previous state (e.g. **RUNNING**) when the signal quality returns to normal

SDK Utility Methods

sharedInstance

Developer could always use this method to get the EEG Algorithm instance throughout the app

```
/*
 * Return: EEG Algo instance
 */
+ (id) sharedInstance;
```

Example

```
NskAlgoSdk *nskAlgoInstance = [NskAlgoSdk sharedInstance];
```

getAlgoVersion

Get the specific EEG algorithm version

```
/*
 * Return: Specific EEG algorithm version
 */
- (NSString*) getAlgoVersion: (NskAlgoEegType)algoType;
```

Example

```
NSString *version = [NSString stringWithFormat:@"Attention Algo Ver.: %@", [[NskAlgoSdk
sharedInstance] getAlgoVersion:NskAlgoEegTypeAtt]];
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
                                         message:version
                                         delegate:nil
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
[alert show];
```

getSdkVersion

Get Algo SDK version

```
/*
 * Return: Algo SDK version
 */
- (NSString*) getSdkVersion;
```

Example

```
NSString *version = [NSString stringWithFormat:@"SDK Ver.: %@", [[NskAlgoSdk sharedInstance]
getSdkVersion]];
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
                                         message:version
                                         delegate:nil
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
[alert show];
```

setAlgorithmTypes

Select specific EEG algorithm(s)

```
/*
 * Return: 0 - Algo SDK is initialized successfully; Otherwise, something wrong on SDK
initialization (please contact with NeuroSky technical support)
 */
- (NSInteger) setAlgorithmTypes: (NskAlgoEegType)algoTypes;
```

Example 1 - Single algorithm

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Attention algorithms
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAtt];
```

Example 2 - Multiple algorithms

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Attention and Meditation algorithms
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAtt| NskAlgoEegTypeMed];
```

Note: .

- By invoking **setAlgorithmTypes:** with supported EEG algorithm(s), the SDK will always change back to **INITED**

dataStream

Feed-in the parsed EEG data from NeuroSky Biosensor System to the EEG Algo SDK

Important: .

- There are different data output giving out from NeuroSky Biosensor System.
- EEG Algo SDK handles only the following **4** data output for now. They are:
  - Poor Signal Quality
  - EEG Raw Data
  - Attention
  - Meditation

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) dataStream: (NskAlgoDataType)type data:(int16_t*)data length:(int16_t)length;
```

Handling realtime EEG data

```
-(void) onDataReceived:(NSInteger)datatype data:(int)data obj:(NSObject *)obj{
    switch (datatype) {

        case MindDataType_CODE_POOR_SIGNAL:
        {
            int16_t poor_signal[1];
            poor_signal[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypePQ data:poor_signal length:1];
        }
```

```
            break;

        case MindDataType_CODE_RAW:
        {
            int16_t eeg_data[1];
            eeg_data[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeEEG data:eeg_data length:1];
        }
            break;

        case MindDataType_CODE_ATTENTION:
        {
            int16_t attention[1];
            attention[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeAtt data:attention length:1];
        }
            break;

        case MindDataType_CODE_MEDITATION:
        {
            int16_t meditation[1];
            meditation[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeMed data:meditation length:1];
        }
            break;
        default:
            break;
    }
}
```

startProcess

Start analysing feed-in EEG data with selected EEG algorithm(s)

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) startProcess;
```

Example

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Attention algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAtt];
// start analysing EEG data
[nskAlgo startProcess];
```

Note: .

- SDK state will only change to **RUNNING** by invoking **startProcess**

pauseProcess

Pause analysing feed-in EEG data

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) pauseProcess;
```

Example

```
// User presses PAUSE button on app to pause the data analysis
- (IBAction)pausePress:(id)sender {
    [[NskAlgoSdk sharedInstance] pauseProcess];
}
```

Note: .

- SDK state will change to **PAUSE**
- No **algoIndex** delegate method will not be invoked unless **startProcess** method is invoked again

stopProcess

Stop analysing feed-in EEG data

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) stopProcess;
```

Example

```
// User presses STOP button on app to stop the data analysis
- (IBAction)stopPress:(id)sender {
    [[NskAlgoSdk sharedInstance] stopProcess];
}
```

Note: .

- SDK state will change to **STOP**
- No **algoIndex** delegate method will be invoked unless **startProcess** method is invoked again

Applications

Application of Attention Algorithm

- Selecting Attention Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- Attention index will be returned every 1 second when Algo SDK state is **RUNNING**
- Attention index ranges from **0 to 100**. The higher the index, the higher the attention level

Note: .

- Attention has a fixed output interval of 1 second, i.e. one new Attention index every second

Application of Meditation Algorithm

- Selecting Meditation Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- Meditation index will be returned every 1 second when Algo SDK state is **RUNNING**
- Meditation index ranges from **0 to 100**. The higher the index, the higher the meditation level

Note: .

- Meditation has a fixed output interval of 1 second, i.e. one new Meditation index every second

Application of Eye Blink Detection

- Selecting Eye Blink Detection by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- Eye blink strength will be returned once eye blink is detected when Algo SDK state is **RUNNING**

Note: .

- No baseline data collection will be needed

Application of EEG Bandpower Computation

- Selecting EEG bandpower by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- EEG bandpower values (in dB) will be returned every 1 second when Algo SDK state is **RUNNING**

SDK Operations

Pause and Resume

- Assuming SDK is in **RUNNING** state
- Pausing EEG algorithm data analysis by invoking **pauseProcess:** method
- Resuming EEG algorithm data analysis by invoking **startProcess:** method

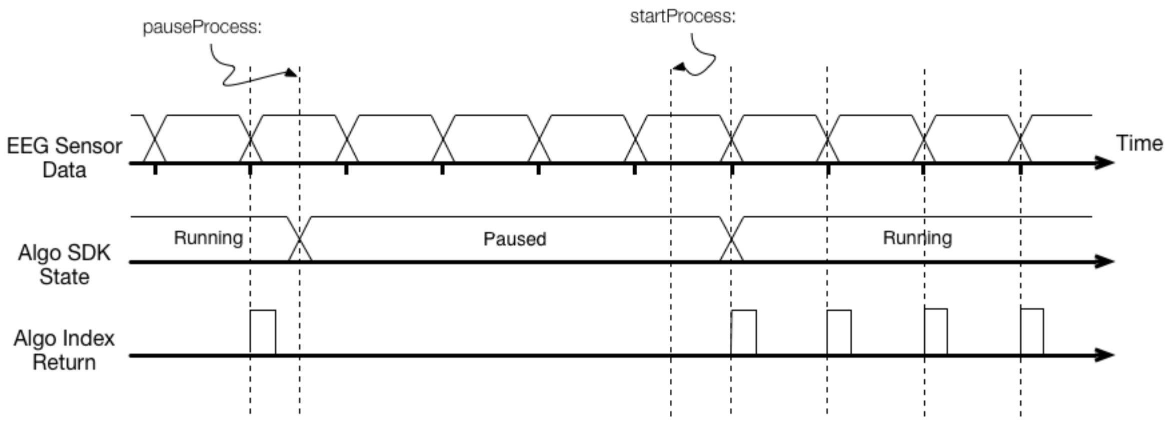

Figure 3.1: Time diagram on Pause/Resume SDK

Note: .

- There will be no effect on **pauseProcess:** when previous SDK state is not **RUNNING**

Stop and Start

- Assuming SDK is in **RUNNING** state
- Stopping EEG algorithm data analysis by invoking **stopProcess:** method
- Restart EEG algorithm data analysis by invoking **startProcess:** method

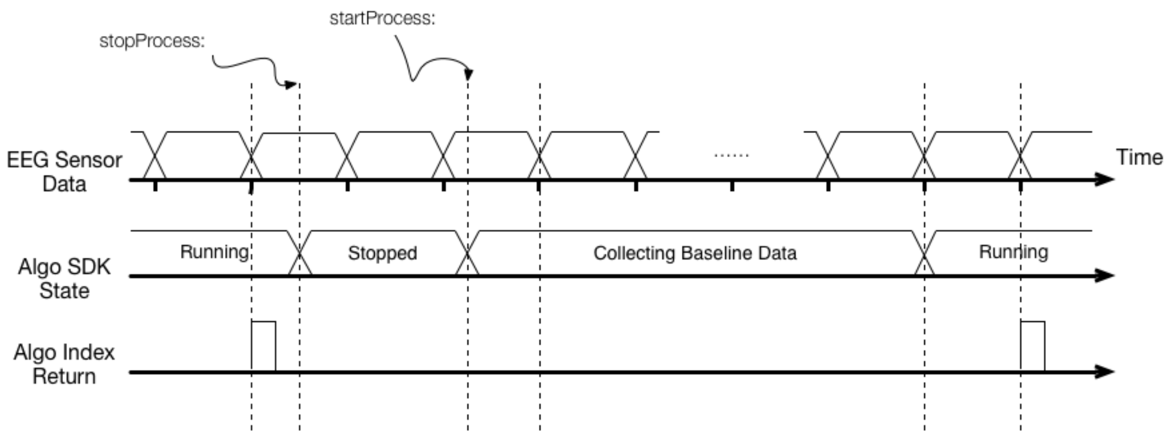

Figure 3.2: Time diagram on Stop/Start SDK

Note: .

- There will be no effect on **stopProcess:** when previous SDK state is not **RUNNING**

Customize Algorithm Output Interval

- Algorithm output interval can be configured at any time once SDK has been initialized
- The new configured output interval will become effective based on last index returned

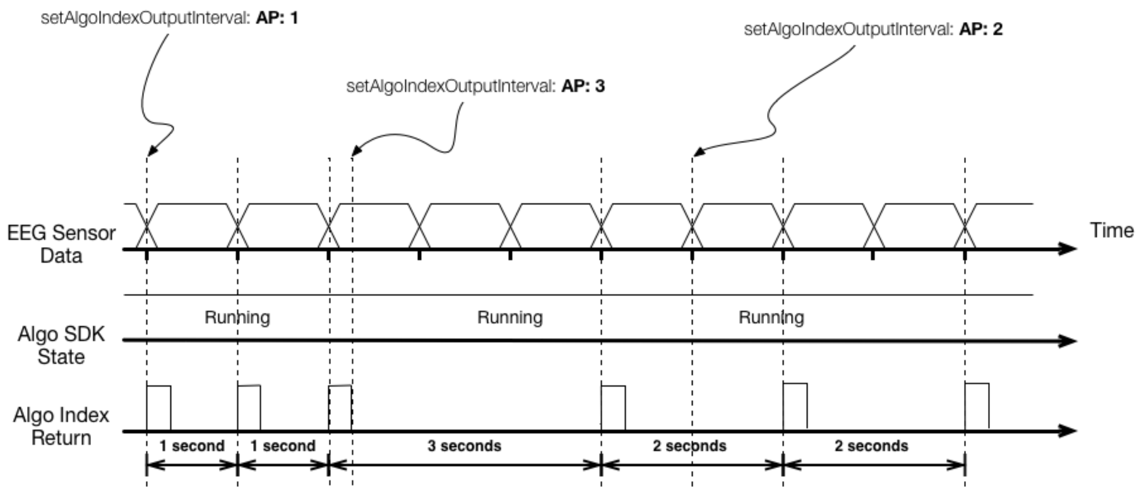

Figure 3.3: Time diagram on configuring algorithm output interval

Note: .

- Different algorithm may have different **minimum/default** output interval

Frequently Asked Questions