

ThinkGear SDK for OS X : Development Guide

Introduction

This guide will teach you how to use NeuroSky's **ThinkGear SDK for Mac** to write Mac applications that can utilize bio-signal data from NeuroSky's ThinkGear bio-sensors. This will enable your Mac apps to receive and use bio-signal data such as EEG acquired from NeuroSky's sensor hardware. The ThinkGear SDK is compatible with both the 32 and 64 bit versions of OS X.

This guide (and the entire **ThinkGear SDK for Mac**) is intended for programmers who are already familiar with standard Mac development using Xcode and Apple's Mac SDK. If you are not already familiar with developing for Mac, please first visit Apple's web site for instruction and tools to develop Mac apps.

If you are already familiar with creating typical Mac apps, then the next step is to make sure you have downloaded NeuroSky's **ThinkGear SDK for Mac** included with the Developer Tools for PC/Mac. Chances are, if you're reading this document, then you already have it.

Future revisions of the Mac OSX ThinkGear SDK will be updated to update the beta Mental Effort and Familiarity measurements.

ThinkGear SDK for Mac Contents

- ThinkGear SDK for Mac: Development Guide (this document mac_development_guide.pdf)
- ThinkGear.framework library
- ThinkGearTouch example project for Mac

You'll find the "ThinkGear.framework" in the lib/ folder, and the "HelloEEG" in the Sample Project/ folder within the ThinkGear SDK for Mac folder.

Supported ThinkGear Hardware

The ThinkGear SDK for Mac must be used with a ThinkGear-compatible hardware sensor device. The following ThinkGear-compatible hardware devices are currently supported:

- MindWave Mobile for use with an Mac device's built in Bluetooth

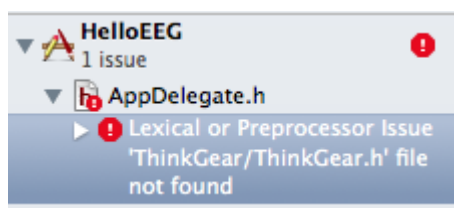
Before using any Mac application that uses the ThinkGear SDK for Mac OSX, make sure you have paired the ThinkGear sensor hardware to your Mac by carefully following the instructions in the User Manual that came with the corresponding ThinkGear hardware device.

Your First Project: HelloEEG

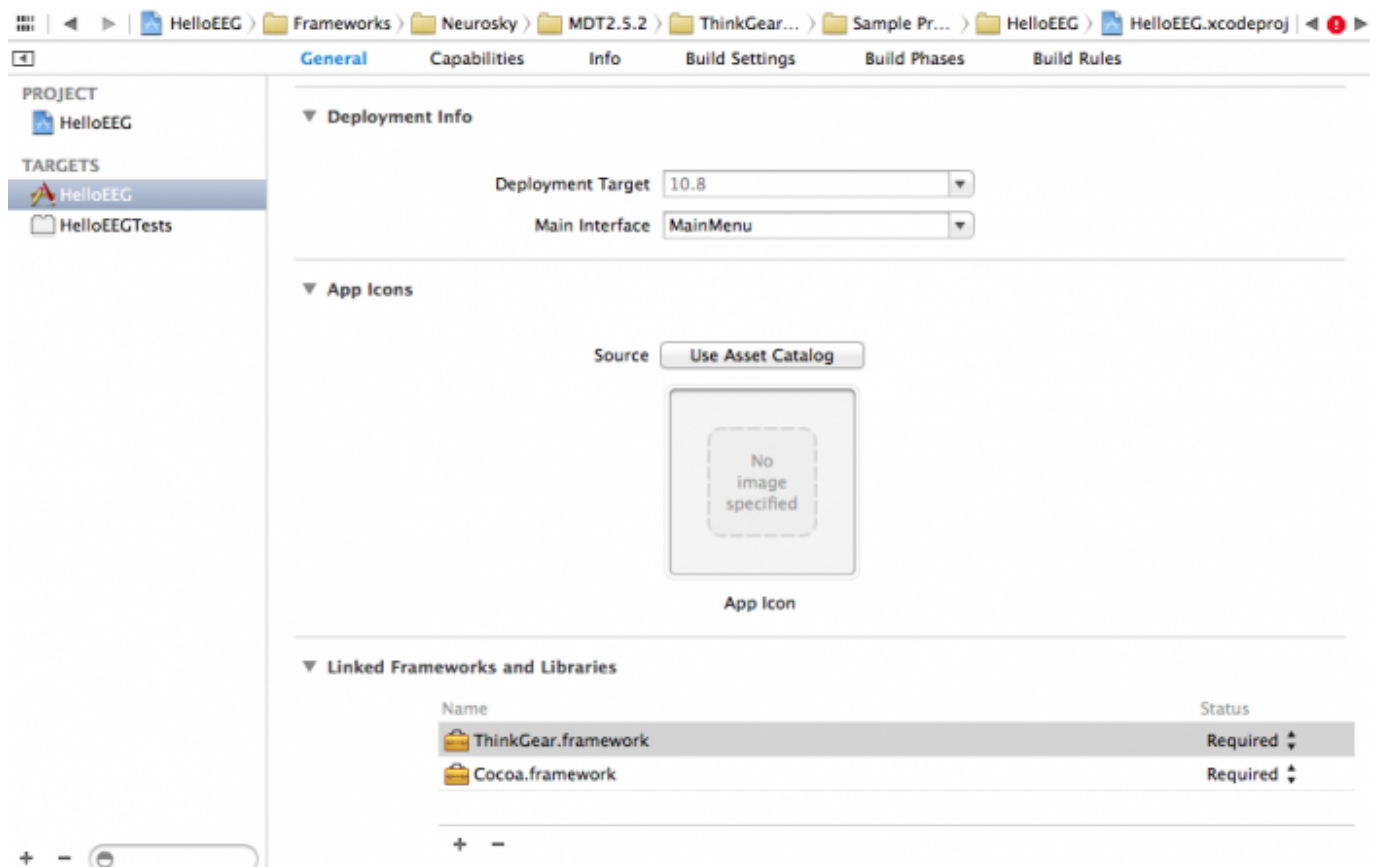
HelloEEG is a sample project we've included in the **ThinkGear SDK for Mac** that demonstrates how to setup, connect, and handle data to a ThinkGear device. Add the project to your Xcode environment by following these steps:

1. Browse in the TG-SDK to select the Sample Projects/HelloEEG directory
2. Double-click the HelloEEG.xcodeproj file
3. select **Product** → **Run** to compile, link and start HelloEEG in the Xcode emulator.

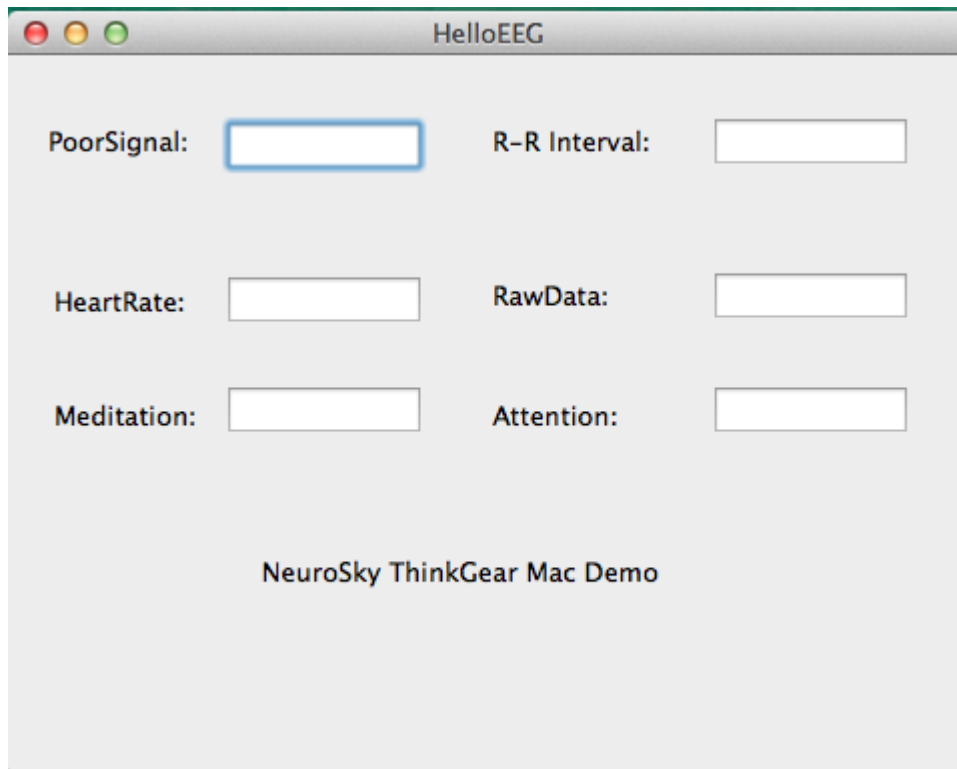
Note: If HelloEEG does not compile and you see the following error, try deleting and re-adding ThinkGear.framework under Frameworks and Libraries from the menu shown below.



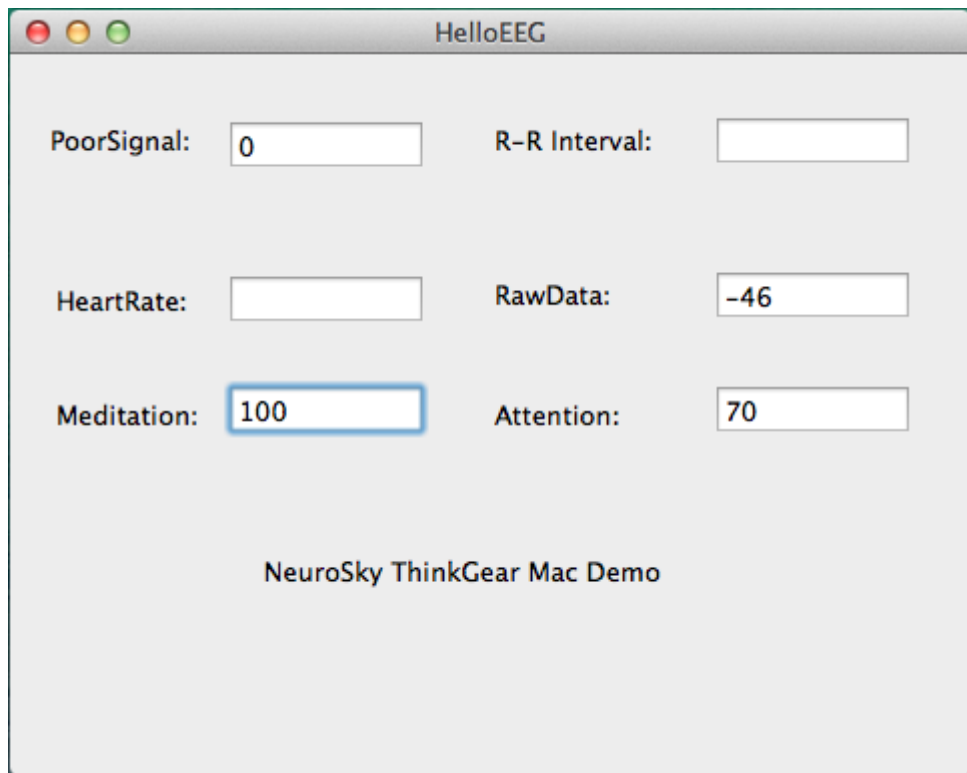
1. Click **ThinkGear.framework** and then click the - button.
2. Click the + button and then click **Add Other**
3. Find and add **ThinkGear.framework** in **TG-SDK** → **TG-SDK For Mac** → **lib** → **ThinkGear.framework**



If HelloEEG is run without connecting the ThinkGear-enabled headset, the application should look like this:



Once the headset is connected, run the application once again. The application should now look like this:



If the Poor Signal data reads 0, Meditation and Attention data should be fluctuating every second. If Poor Signal reads something other than 0, re-adjust the headset accordingly until Poor Signal data goes down to 0.

At this point, you should be able to browse the code, make modifications, compile, and deploy the app to your device or emulator just like any typical OS X application.

Developing Your Own ThinkGear-enabled Apps for Mac

For most applications, using the ThinkGear Mac API is recommended. It reduces the complexity of managing ThinkGear accessory connections and handles parsing of the data stream from these ThinkGear accessories. To make an application, all you need to do is to import a library, and assign a delegate object to which accessory event notifications will be dispatched.

Some limitations of the ThinkGear for Mac API include:

- Can only communicate with one attached ThinkGear-enabled accessory

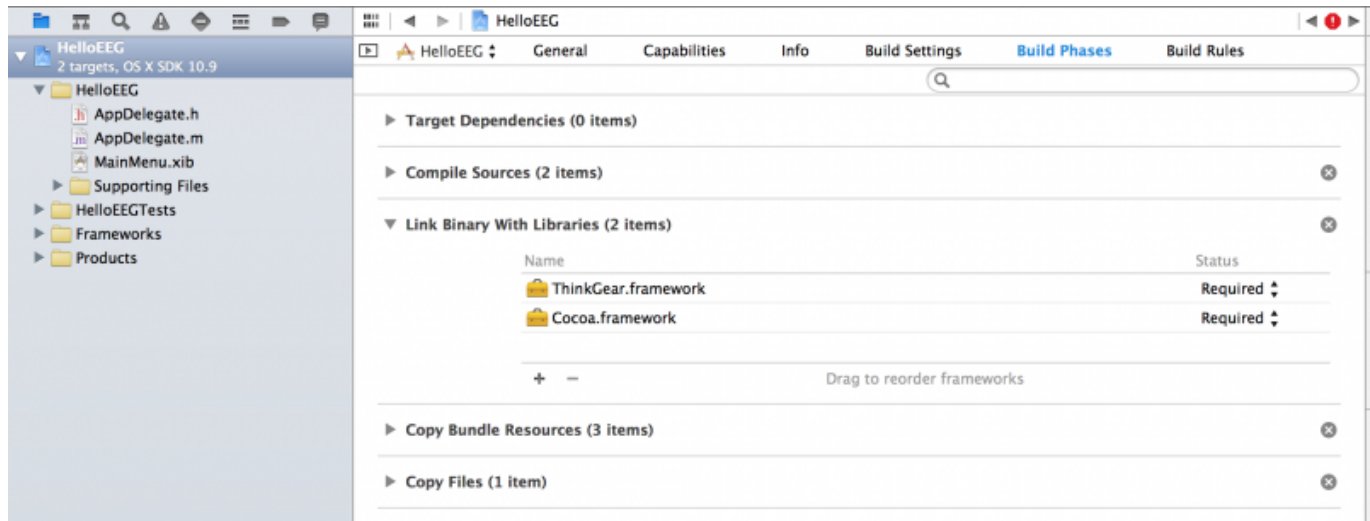
Configuring Your Environment

Add `ThinkGear.framework` to your project.

1. Navigate to your project settings

2. Select your target
3. Select Build Phases
4. Expand **Link Binary With Libraries**
5. Click on + and select ThinkGear . framework and click **Add**

Your project window should now look similar to this:



App Sandbox

Creating ThinkGear Object

A ThinkGear object is used to manage a single connection to a single ThinkGear hardware device.

- Import following files into your header files

```
#import "ThinkGear/ThinkGear.h"
```

- Declare a ThinkGearObjc object in the header(.h) file

```
@interface AppDelegate : NSObject <UIApplicationDelegate, ThinkGearDelegate>
{
    ThinkGearObjC *thinkGear;

    NSString *devicePortName;
    NSString *deviceNameToSearch;
}

@property (assign) IBOutlet NSWindow *window;

...
@end
```

- Initialize the thinkGear in your implementation(.m) file

```
// initialize the thinkgear object
thinkGear = [[ThinkGearObjC alloc] init];
[thinkGear setDelegate:self];

deviceNameToSearch = @"MindWaveMobile";
```

Connecting to ThinkGear Hardware Device

Simple add the following code to your implementation(.m) file:

```
if([thinkGear connected] == NO) {

    NSArray *devContents = [[NSFileManager defaultManager]
contentsOfDirectoryAtPath:@"/dev" error:nil];
    NSPredicate *ttyPredicate = [NSPredicate predicateWithFormat:@"SELF
startswith 'tty.'"];
    NSArray *deviceList = [devContents filteredArrayUsingPredicate:
ttyPredicate];

    devicePortName = @"";
    BOOL found = NO;

    if(deviceNameToSearch.length > 0){
        for(id device in deviceList){
            NSString *temp = (NSString *)device;
            NSLog(@"%@", temp);
            if([temp rangeOfString:deviceNameToSearch].location !=
NSNotFound){
                if([deviceNameToSearch isEqualToString:@"MindWave"]){
                    if([temp rangeOfString:@"MindWaveMobile"].location
== NSNotFound){
                        found = YES;
                        devicePortName = [@" /dev/"
stringByAppendingString:temp];
                        break;
                    }
                }else {
                    found = YES;
                    devicePortName = [@" /dev/" stringByAppendingString:
temp];
                    break;
                }
            }
        }
    }
}
```

```
    if (found) {  
        if(devicePortName.length > 0){  
            [thinkGear ConnectTo:devicePortName];  
        }  
    }  
}
```

Handling Data Receipt

To receive the message from ThinkGear SDK, the `dataReceived:` method must be implemented. In the header (.h) file, add the following method definition:

```
- (void)dataReceived:(NSDictionary *)data;
```

And in the implementation (.m) file, implement the method.

```
- (void)dataReceived:(NSDictionary *)data {  
    if([data objectForKey:@"poorSignal"]){  
        ...  
    }  
  
    if([data objectForKey:@"rawData"]){  
        ...  
    }  
  
    if([data objectForKey:@"eSenseMeditation"]){  
        ...  
    }  
  
    if([data objectForKey:@"eSenseAttention"]){  
        ...  
    }  
}
```

ThinkGear Data Types

Poor Signal

This integer value provides an indication of how good or how poor the bio-signal is at the sensor. This value is typically output by all ThinkGear hardware devices once per second.

This is an extremely important value for any app using ThinkGear sensor hardware to always read, understand, and handle. The value is stored in the key of “poorSignal”.

Raw Data

This data type supplies the raw sample values acquired at the bio-sensor. The sampling rate (and therefore output rate), possible range of values, and interpretations of those values (conversion from raw units to volt) for this data type are dependent on the hardware characteristics of the ThinkGear hardware device performing the sampling. You must refer to the documented development specs of each type of ThinkGear hardware that your app will support for details.

As an example, the majority of ThinkGear devices sample at 512Hz, with a possible value range of -32768 to 32767.

As another example, to convert TGAT-based EEG sensor values (such as TGAT, TGAM, MindWave, MindWave Mobile) to voltage values, use the following conversion:

```
(rawValue * (1.8/4096)) / 2000
```

Attention

This int value reports the current eSense™ Attention meter of the user, which indicates the intensity of a user's level of mental “focus” or “attention”, such as that which occurs during intense concentration and directed (but stable) mental activity. Its value ranges from 0 to 100. Distractions, wandering thoughts, lack of focus, or anxiety may lower the Attention meter levels. See [eSense Meters](#) below for details about interpreting eSense levels in general.

The value is stored in the key of “eSenseAttention”. It is typically output once a second.

Meditation

This unsigned one-byte value reports the current eSense™ Meditation meter of the user, which indicates the level of a user's mental “calmness” or “relaxation”. Its value ranges from 0 to 100. Note that Meditation is a measure of a person's **mental** levels, not **physical** levels, so simply relaxing all the muscles of the body may not immediately result in a heightened Meditation level. However, for most people in most normal circumstances, relaxing the body often helps the mind to relax as well. Meditation is related to reduced activity by the active mental processes in the brain, and it has long been an observed effect that closing one's eyes turns off the mental activities which process images from the eyes, so closing the eyes is often an effective method for increasing the Meditation meter level. Distractions, wandering thoughts, anxiety, agitation, and sensory stimuli may lower the Meditation meter levels. See [eSense Meters](#) below for details about interpreting eSense™ levels in general.

The value is stored in the key of "eSenseMeditaion". It is typically output once a second.

eSense Meters

For all the different types of eSense™ (i.e. Attention, Meditation), the meter value is reported on a relative eSense™ scale of 1 to 100. On this scale, a value between 40 to 60 at any given moment in time is considered "neutral", and is similar in notion to "baselines" that are established in conventional EEG measurement techniques (though the method for determining a ThinkGear baseline is proprietary and may differ from conventional EEG). A value from 60 to 80 is considered "slightly elevated", and may be interpreted as levels being possibly higher than normal (levels of Attention or Meditation that may be higher than normal for a given person). Values from 80 to 100 are considered "elevated", meaning they are strongly indicative of heightened levels of that eSense™.

Similarly, on the other end of the scale, a value between 20 to 40 indicates "reduced" levels of the eSense™, while a value between 1 to 20 indicates "strongly lowered" levels of the eSense™. These levels may indicate states of distraction, agitation, or abnormality, according to the opposite of each eSense™.

BLINK

This int value reports the intensity of the user's most recent eye blink. Its value ranges from 1 to 255 and it is reported whenever an eye blink is detected. The value indicates the relative intensity of the blink, and has no units.

The Detection of Blinks must be enabled.

From:

<http://developer.neurosky.com/docs/> - **NeuroSky Developer - Docs**

Permanent link:

http://developer.neurosky.com/docs/doku.php?id=mac_development_guide

Last update: **2014/07/01 20:07**



Warnings and Disclaimer of Liability

THE ALGORITHMS MUST NOT BE USED FOR ANY ILLEGAL USE, OR AS COMPONENTS IN LIFE SUPPORT OR SAFETY DEVICES OR SYSTEMS, OR MILITARY OR NUCLEAR APPLICATIONS, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE ALGORITHMS COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. YOUR USE OF THE SOFTWARE DEVELOPMENT KIT, THE ALGORITHMS AND ANY OTHER NEUROSKY PRODUCTS OR SERVICES IS "AS-IS," AND NEUROSKY DOES NOT MAKE, AND HEREBY DISCLAIMS, ANY AND ALL OTHER EXPRESS AND IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL NEUROSKY BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING BUT

NOT LIMITED TO LOSS OF PROFITS OR INCOME, WHETHER OR NOT NEUROSKY HAD KNOWLEDGE, THAT SUCH DAMAGES MIGHT BE INCURRED.