Android Development Guide for ThinkGear

Features

- Develop Android applications that utilize ThinkGear technology
- Downloadable ThinkGear-enabled sample Android project with full sample code

Introduction

Thanks to the availability of the MindWave Mobile, developers can now create Android applications that can sense users' brainwaves. This development guide will walk you through the process of creating a MindWave-capable Android application.

This guide is written for programmers who are familiar with Android development on Eclipse. More information on how to develop on Android can be found at http://developer.android.com.

SDK Bugs and Issues

The SDK currently has no known bugs nor issues.

If you encounter any bugs or issues, please visit http://support.neurosky.com, or contact support@neurosky.com.

Supported Hardware

The ThinkGear Android API supports the following hardware:

- MindSet
- MindWave Mobile
- ThinkCap 1.0
- TGAP DS SDK
- EGO
- MindBand

MindWave Mobile

The MindWave Mobile utilizes Bluetooth to connect to an Android device.

Usage

1. Open the Settings app on the Android device
2. Navigate to **Wireless and network** and enable Bluetooth if not already enabled
3. Go to **Bluetooth settings**
4. Power on the MindWave Mobile
5. `MindWave Mobile` will show up in the list of devices
6. Touch `MindWave Mobile` and pairing will complete automatically
   (a) If prompted for a passkey, enter in '0000'

Note: Consult the MindWave Mobile User guide for pairing details.

Broadcast data

Data is sent from the MindWave Mobile with the following information:

- Poor signal value (1Hz)
- eSense Attention (1Hz)
- eSense Meditation (1Hz)
- EEG power bands (1Hz)
- Raw EEG data (512Hz)
- Blink (When a blink is detected)

Using the ThinkGear API

For most applications, using the ThinkGear Android 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 a brainwave-sensing application, all you need to do is to import a library, add the requisite setup and teardown functions, and create a handler object to which accessory event notifications will be dispatched.

Some limitations of the ThinkGear Android API include:

- Can only communicate with one attached ThinkGear-enabled accessory

The Android API Reference contains descriptions of the classes and protocols available in the ThinkGear Android API.

The ThinkGear Android SDK also includes the `HelloEEG` sample project (contained in `src/`), which is a simple Android application that displays the data coming from a MindWave Mobile headset.

Configuring Your Environment

1. Add the `ThinkGear.jar` file to your project in the `lib` folder. If the `lib` folder does not exist, create it. Then right-click on `ThinkGear.jar` in the package explorer inspector and select **Build Path** » **Add to build path**.
2. Then import the following classes into your application actvity:

```
import com.neurosky.thinkgear.TGData;
import com.neurosky.thinkgear.TGDevice;
import com.android.bluetooth.BluetoothAdapter;
import com.android.bluetooth.BluetoothDevice;
import com.android.util.Log;
```

In order for your application to access the Bluetooth API's, your application must declare the `BLUETOOTH` permission. Declare the Bluetooth permission in your application manifest file.

```
<manifest ... >
  <uses-permission android:name="android.permission.BLUETOOTH" />
  ...
</manifest>
```

Setting Up the TGDevice

Declare a TGDevice and a BluetoothAdapter instance in your activity class

```
public class HelloEEGActivity extends Activity {
//...
TGDevice tgDevice;
BluetoothAdapter btAdapter;
//...
```

Initialize `tgDevice` and `btAdapter` in the onCreate() method

```
public void onCreate(Bundle savedInstanceState) {
//...
    btAdapter = BluetoothAdapter.getDefaultAdapter();
    if(btAdapter != null) {
        tgDevice = new TGDevice(btAdapter, handler);
    }
//...
}
```

Handling Data Receipt

The TGDevice will communicate with the application through messages send to a handler function. Add the following code to your application class:

```
private final Handler handler = new Handler() {
  @Override
  public void handleMessage(Message msg) {
    switch (msg.what) {
      case TGDevice.MSG_STATE_CHANGE:
        switch (msg.arg1) {
          case TGDevice.STATE_IDLE:
          break;
```

```
            case TGDevice.STATE_CONNECTING:
            break;
            case TGDevice.STATE_CONNECTED:
              device.start();
            break;
            case TGDevice.STATE_DISCONNECTED:
            break;
            case TGDevice.STATE_NOT_FOUND:
            case TGDevice.STATE_NOT_PAIRD:
            default:
            break;
          }
        break;
        case TGDevice.POOR_SIGNAL:
            Log.v("HelloEEG", "PoorSignal: " + msg.arg1);
        case TGDevice.MSG_ATTENTION:
            Log.v("HelloEEG", "Attention: " + msg.arg1);
          break;
        case TGDevice.MSG_RAW_DATA:
            int rawValue = msg.arg1;
          break;
        case TGDevice.MSG_EEG_POWER:
            TGEegPower ep = (TGEegPower)msg.arg1;
            Log.v("HelloEEG", "Delta: " + ep.delta);
        default:
          break;
        }
    }
};
```

The following table details each message type:

Message	Description	Data
MSG_STATE_CHANGE	The state of the TGDevice has changed	STATE messages stored in the `arg1` field of the message object
MSG_POOR_SIGNAL	Signal quality status data	The poor signal status from the headset is stored in the `arg1` field of the message object
MSG_ATTENTION	Attention level data	The attention level is stored in the `arg1` field of the message object
MSG_MEDITATION	Meditation level data	The meditation level is stored in the `arg1` field of the message object
MSG_BLINK	Strength of detected blink	The blink strength is stored in the `arg1` field of the message object
MSG_RAW_DATA	Raw EEG data	The raw EEG value is stored as an int in the `arg1` field of the message object
MSG_EEG_POWER	EEG powers data	The EEG powers are passed in as TGEegPower object in the `obj` field of the message object
MSG_RAW_MULTI	Multi-channel raw data	The multi-channel raw data is passed in as a TGRawMulti object in the `obj` field of the message object
MSG_HEART_RATE	Heart rate data	The heart rate data is passed in as an int in the `arg1` field of the message object

TGDevice States

State	Description
STATE_IDLE	Initial state of the TGDevice. Not connected to a headset
STATE_CONNECTING	Attempting a connection to the headset
STATE_CONNECTED	A valid device hand been found and data is being received
STATE_DISCONNECTED	The connection to the device is lost
STATE_NOT_FOUND	Could not connect to headset
STATE_NOT_PAIRED	A valid headset could not be found

Starting the Data Stream

Connect to a headset by calling the tgDevice's `connect` method as follows

```
tgDevice.connect(true);
```

The tgDevice will search through the paired Bluetooth devices and connect to the first known ThinkGear compatible device. Setting the parameter to true or false will enable or disable raw EEG output.

After successfully connecting to a ThinkGear device, the tgDevice will send a "BT_STATE_CONNECTED" message. To start receiving data, call the tgDevice's `start` method.

```
tgDevice.start();
```

Close the connection by calling the `close` method

```
tgDevice.close();
```

Further Considerations

- The application should not expect there to be a ThinkGear accessory attached to the Android-based device on startup. As such, it should handle that case accordingly (e.g. by displaying a static splash screen prompting the user to connect a ThinkGear accessory).

References

- http://developer.android.com/guide/topics/wireless/bluetooth.html

Corporate Address
NeuroSky, Inc.
125 S. Market St., Ste. 900
San Jose, CA 95113
United States
(408) 600-0129

Questions/Support: http://support.neurosky.com
or email: support@neurosky.com

Community Forum: http://developer.neurosky.com/forum

NeuroSky
Brain-Computer Interface Technologies