MiDro – The Mind-activated AnDroid Application
Capstone Senior Design Project (2014)

Authors: Steven Fisher, Andrew Rodriguez, Timothy Su
Advisors: Dr. Dario Pompili
Graduate Student Mentor: Harirahasudhan Viswanathan

Abstract:

The human body can be a peripheral for every-day technology use through Bluetooth electroencephalography (EEG), which wirelessly tracks electrical and muscular activity in the brain. This electrical activity can be allocated to commands that can span a host of different applications such as changing a room’s temperature, turning lights off and on, or even direct application to phones.

The focus of our project is to make a body-phone interface application for Android that utilizes the versatility of the human face and its various recognizable features. Through cerebral electrical impulses, an EEG device, and Bluetooth, we will associate differing, deliberate facial gestures and actions with certain phone events.

Using the Android Studio IDE allowed us to work in a practical Java environment to establish this connection between the EEG headset and a phone. The application runs strictly in the background for user convenience – the user will not have to dabble in convoluted UI or have to learn much to be able to use a new device they were previously unfamiliar with.

Certain commands include:

- Blinks
- Eye movements
- Eye positioning

The applications of this concerns questions of convenience more than it does rehabilitative applications, but it still has possibilities for safety: if a consumer is experiencing a seizure, for instance, wearing the headset could trigger an emergency signal to 911 without need of any communication in the least bit.

This project is meant to serve a gateway into the world of using EEG technology to utilize other technology, creating a true hands-free and mostly voice-free alternative to much modern technology. Using a peripheral that people spend most of their waking life using means not having to learn a new set of controls, and offers a new level of practicality of interacting with handheld devices.
Introduction/Overview:

Using reliable and portable technology opens up the opportunity for developers to make an interface between operating systems and the human body. Our team uses the NeuroSky Mindwave Mobile headset to gather raw EEG data and allows us to create primitive mind-interactive applications for both Android and iOS. Being able to use the human body as a controller introduces a new use for the human body – a universal platform that can interact with various devices formed for infinitely different purposes.

Normally, the human brain is observed in relation to externalities (stimuli such as light, sound, and other queues), but our methods observes internalities (intent of eye movement, number of blinks) and make them applicable as options of control. For instance, a deliberate action such as a set number of blinks, say three blinks, can open a certain bookmarked website. Five blinks may display your current GPS location. Seven blinks may make a phone call to an emergency contact of your choice. Eye movements up and down could trigger applications such as flipping a coin.

Instead of relying on pre-mediated artifact removal methods that came pre-installed into the NeuroSky Mindwave Mobile headset, we were able to set our own thresholds, allowing full control over the triggers of the phone functions.

Approach/Methods/Results:

Planning

At first, the native NeuroSky Mindwave Mobile software and independently created applications were used to determine different statuses of brain activity. Through observing the spectrograms and live EEG feeds, we achieved a rough idea of what commands we were able to use with the limited EEG technology at hand. The spectrograms revealed how the headset reacted to facial impulses and other noise-related signals. The rest was a trial-and-error method through coding thresholds.

Spectrograms

Each length of testing time was about half a minute with a one second refresh rate (this meant about 30 samples per screenshot). In the figures below, the brightness of a rectangle corresponds to the activity in that frequency band.

The spectrograms gave some idea as to what situations using the headset would be inappropriate for. For example, in Figure 1 (below), eating showed an extraordinary amount of noise. This meant the headset is virtually impossible to use while chewing due to the rapid facial muscle movement as well as unreliable sensor movement.

The lower region (Delta, Theta waves) is filled with frequency bands associated with less-intense brain activity such as rest or sleep. The higher frequencies (Beta, Gamma) correspond to a more intense brain activity. Figure 2 demonstrates the spectrogram of a person walking in silence while constantly observing the environment. Note the high density of lower-frequency waves, showing relaxation.

This is a constant rate, the lowest possible impulse that the EEG can read. Any higher impulse than these would denote an activity in eye motion.
The spectrograms also gave ideas towards similar facial structures that may have confused the program. Due to the simplicity of the one-channel EEG technology, we could not differentiate between a left-eye movement and a right-eye movement (See: Figures 3 and 4).

And, to demonstrate high brain activity, Figure 5 demonstrates the recordings of a person blinking quickly.
Testing through these trials led to the next step: Gathering the raw EEG data from the headset and creating our own thresholds for our purposes.

**Raw EEG Data and Arduino Studio**

By this point, we had about four commands we could work with: Eye blink x3, Eye blink x5, Eye blink x7, Eye movement up.

Using this and the NeuroSky Mindwave Mobile Source Developer’s Kit, we were able to access the raw EEG data from the headset to apply our own thresholds and develop an interactive set of facial options to trigger certain phone events. The code is directly below, followed by screenshot examples of the application’s display when it’s not running in the background:

**Code**

All of the code was created and compiled using Android Studio. A few of the ideas extracted from online sources, which are listed in the bibliography:

```java
package com.example.helloapp.app;

import android.support.v7.app.ActionBarActivity; import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;
import com.neurosky.thinkgear.TGDevice; import android.bluetooth.BluetoothAdapter; import 
android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity {
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    tv = (TextView)findViewById(R.id.textView1);
tv.setText("\nAndroid version: " + Integer.valueOf(android.os.Build.VERSION.SDK) + "\n");
btAdapter = BluetoothAdapter.getDefaultAdapter();
if(btAdapter == null) {
    // Alert user that Bluetooth is not available
    Toast.makeText(this, "Bluetooth not available", Toast.LENGTH_LONG).show();
    finish();
    return;
}else {
    /* create the TGDevice */
tgDevice = new TGDevice(btAdapter, handler);
}
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

@Override
public void onDestroy() {
    tgDevice.close();
    super.onDestroy();
}
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:
                        tv.append("Connecting...
");
                        break;
                    case TGDevice.STATE_CONNECTED:
                        tv.append("Connected.
");
                        tgDevice.start();
                        break;
                    case TGDevice.STATE_NOT_FOUND:
                        tv.append("Can't find
");
                        break;
                    case TGDevice.STATE_NOT_PAIRED:
                        tv.append("not paired
");
                        break;
                    case TGDevice.STATE_DISCONNECTED:
                        tv.append("Disconnected mang
");
                        break;
                }
                break;
            case TGDevice.MSG_POOR_SIGNAL:
                //signal = msg.arg1;
                tv.append("PoorSignal: " + msg.arg1 + 
"");
                break;
            case TGDevice.MSG_RAW_DATA:
                //TODO if raw dips below steady state, down deflection, but I need stready
                state
                int raw1 = msg.arg1;
                //tv.append("Got raw: " + msg.arg1 + 
"");

                //TODO if raw1 dips down below thresholdVertical, eye up has ended.
                // if only one crest, eye up -v^-
        }
    }
}
if two crest, eye down -^v-
if(recordingUp){
    recordIndex++;
    if(recordIndex>1023)
    {
        //It's been 2sec worth of samples
        //If this statement reached, only up motion detected. trigger up event
        //debug: print entire data block
        tv.append("2sec record complete "+"\n");
        recordingUp =false;
        //recordIndex=0;
    }
    else
    {
        //raw_2sec[recordIndex]=raw1;
        if(raw1>thresholdBlink){
            //if magnitude > thresholdBlink, eyebrow muscle used.
            // Therefore, has to be blink
            tv.append("Blink threshold crossed. "+raw1+"\n");
            numBlinks++; //changed to increment
            crest=true;
            recordingUp =false;
            recordingBlink=true;
        }
    }
}
else if(recordingBlink)
{
    recordIndex++;
    if(recordIndex>1023)
    {
        //stop counting and determine how many blinks occured
        //TODO trigger proper event handler
        recordingBlink=false;
        tv.append("There were "+numBlinks/2+"captured."+"\n");
        switch (numBlinks){
            //case 0: tv.append("zero blinks")
            case 4: tv.append(+numBlinks/2+ "blinks - Voice activation")
                //SPEECH COMMAND
                // Build the intent
                Intent voiceIntent = new Intent(Intent.ACTION_VOICE_COMMAND);
// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(voiceIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe)
{
    startActivity(voiceIntent);
} 
break;

case 8: tv.append(+numBlinks/2+ "blinks - Opening Webpage")
//WEBPAGE LAUNCH
// Build the intent
Uri page= Uri.parse("http://www.rutgers.edu");
Intent webIntent= new Intent(Intent.ACTION_VIEW, page);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(webIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe)
{
    startActivity(webIntent);
} 
break;

case 10: tv.append(+numBlinks/2+ "blinks - Navigate Home...")
//SHOW HOME
// Build the intent
Uri location = Uri.parse("geo:0,0?q=home");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe)
{
    startActivity(mapIntent);
}
break;
default: tv.append("No action Triggered")
}
else
{
    //TODO record each independent crest
    //TODO keep count of last crest. if long, increase capture length
    if(crest){
        if(raw1<thresholdCrestLow){
            //direction change
            crest^=true;
            numBlinks++;
        }
    }
    else{
        if(raw1>thresholdCrestHigh){
            crest^=true;
            numBlinks++;
        }
    }
    //TODO introduce crest/trough indicator
}
else if(raw1>thresholdBlink)
{
    recordingBlink=true;
    recordIndex=0;
    tv.append("Blink entrance"+raw1+"\n");
}
else if(raw1> thresholdVertical)// && raw1< thresholdBlink)//up/down movement only
{
    recordingUp =true;
    recordIndex=0;
    tv.append("Up entrance"+raw1+"\n");
}
//if(raw1>300)
//tv.append("Got big raw: " + msg.arg1 + "\n");
break;
    case TGDevice.MSG_HEART_RATE:
        tv.append("Heart rate: " + msg.arg1 + 
        break;
    case TGDevice.MSG_ATTENTION:
        //att = msg.arg1;
        //tv.append("Attention: " + msg.arg1 + 
        //Log.v("HelloA", "Attention: " + att + 
        break;
    case TGDevice.MSG_MEDITATION:
        break;
    case TGDevice.MSG_BLINK:
        tv.append("Blink: " + msg.arg1 + 
        break;
    case TGDevice.MSG_RAW_COUNT:
        //tv.append("Raw Count: " + msg.arg1 + 
        break;
    case TGDevice.MSG_LOW_BATTERY:
        Toast.makeText(getApplicationContext(), "Low battery!",
        Toast.LENGTH_SHORT).show();
        break;
    case TGDevice.MSG_RAW_MULTI:
        //TGRawMulti rawM = (TGRawMulti)msg.obj;
        //tv.append("Raw1: " + rawM.ch1 + 
        //Raw2: " + rawM.ch2);
        //case TGDevice.MSG_RAW_DATA:
        //int rawValue = msg.arg1;
        //break;
    case TGDevice.MSG_EEG_POWER:
        //TGEegPower ep = (TGEegPower)msg.obj;
        //Log.v("HelloEEG", "Delta: " + ep.delta);
        //tv.append("EEGpowerdelta: " + ep.delta + 
        default:
            break;
        }
    }
}

public void doStuff(View view) {
    if(tgDevice.getState() != TGDevice.STATE_CONNECTING && tgDevice.getState() !=
    TGDevice.STATE_CONNECTED)
        tgDevice.connect(rawEnabled);
    //tgDevice.ena
}
Cost/Sustainability Analysis:

Sophisticated electroencephalography technology is not cheap. For multi-channel EEGs, the range can easily get into the thousands, which is likely not something an average consumer is willing to spend to be able to use brain-to-phone technology.

Funding, as one can imagine, was an issue. The official price quotes are tabulated in Table 1 and are sorted by number of channels, in increasing order, offered by each device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Channels</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeuroSky MindWave</td>
<td>1</td>
<td>$130, including software</td>
</tr>
<tr>
<td>Advanced Brain Monitoring</td>
<td>3</td>
<td>$5,000 + $6,000 for software</td>
</tr>
<tr>
<td>BioRadio Lab Course + Software</td>
<td>8</td>
<td>$10,887, including software</td>
</tr>
<tr>
<td>Advanced Brain Monitoring</td>
<td>9</td>
<td>$9,950 + $8,500 for software</td>
</tr>
<tr>
<td>Cognionics DryCap</td>
<td>12</td>
<td>$12,230</td>
</tr>
<tr>
<td>Emotiv</td>
<td>14</td>
<td>$750</td>
</tr>
<tr>
<td>V-amp BrainVision</td>
<td>16</td>
<td>$13,930</td>
</tr>
</tbody>
</table>

Table 1 – Comparison of EEG headsets of varying quality that were considered in early stages of the project

In conclusion, many of the sophisticated EEG manufacturers do not sell at a price that would interest the average consumer. Preparations involving gel or replacing nodes to use such devices were also considered too inconvenient.

We decided on NeuroSky MindWave because of its simplicity – anyone can put it on in a matter of seconds. Developing applications for the inexpensive, yet portable, device could begin to introduce a way for the average consumer to begin dabbling in the world of neuro-controlling. An increased interest may mean more manufacturing geared towards the average consumer and, hopefully, a better market opportunity for EEG manufacturers.
Conclusion/Summary:

The field of neuro-controlling is vast and difficult to apply practically, but it has applications in a market of people who wish for more convenience. Toying with this field, however, reveals that it is extremely possible to use rapidly-improving EEG technology to add a level of interactivity between the human mind and modern devices. It is an alternative to the augmented reality market and offers interaction with an immediate world rather than a virtual one.

We only hope that in the future, companies could look forward to improving on mobile EEG technology and make smaller, more reliable, and more slightly headsets that may soon accompany our smart phones and other devices. Until then, we can only truly rely on headset devices and the Android and iOS platforms.

Bibliography:


[6] - http://advancedbrainmonitoring.com/ The Advanced Brain Monitoring website. While it offered a good range of channels, the software was too costly to justify.

[7] - http://www.cognionics.com/ Cognionics was the second closest contender being that it was the only other company with a dry-application option.

