Anexo 1: Código Java

```java
package javier_luna.fatiga_eeg;

// Añadimos las librerías necesarias para nuestro código
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Location;
import android.location.LocationListener;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.Vibrator;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.Toast;
import com.neurosky.thinkgear.TGDevice;
import com.neurosky.thinkgear.TGRawMulti;
import java.util.Timer;
import java.util.TimerTask;
import com.neurosky.theta;

@SuppressLint("ValidFragment")

public class MainActivity extends Activity implements
SensorEventListener{
    private Sensor mAccelerometer;
    BluetoothAdapter bluetoothAdapter;
// Declaramos el adaptador de bluetooth y utilizo la llamada de la
librería de Neurosky
    com.neurosky.thinkgear.TGDevice tgDevice;
```

```
// Declaro las variables necesarias para el código y las inicializo
    int pestaneos = 0;
    int Pestaneosprimerminuto = 0;
    int fatigado = 4;
    int primer_minuto=0;
    int dormido=1;
    int Radius=6371000;
    int theta=0;
    int thetadormido1=240; // Valor mínimo para ondas theta
    int thetadormido1=420; // Valor máximo para ondas theta
    int dormidoporondas=0;
    float x,latitud,longitud,longitud1,latitud1,velocidad;
// La variable fatigado la igualo a 4 porque es el valor que marcan
los especialistas entre un pestañeo normal y otro fatigado. La
variable Radius es el valor del radio de la tierra
    //Variables interficie de usuario
    Button btnConectar;
    Button btnSalir;
    TextView lblEstado;
    TextView txtConcentracion;
    TextView txtPestaneos_primer_minuto;
    TextView txtPestaneos;
    TextView txtRawdata;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Se obtiene un adaptador bluetooth
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        //Se comprueba si el bluetooth está disponible
        if (bluetoothAdapter == null) {
            //Indicar al usuario que el bluetooth no esta disponible
            Toast.makeText(this, "Bluetooth no disponible",
Toast.LENGTH_LONG).show();
            finish();//Terminar el programa
        } else {
            //Crear un nuevo dispositivo con el adaptador bluetooth y
el modo manual
            tgDevice = new TGDevice(bluetoothAdapter, handler);
        }
```

```
        //Se obtiene la referencia a los objetos de la interfaz
grafica
             btnConectar = (Button) findViewById(R.id.btnConectar);
            btnSalir = (Button) findViewById(R.id.btnSalir);
            lblEstado = (TextView) findViewById(R.id.lblEstado);
            txtConcentracion = (TextView)
findViewById(R.id.txtConcentracion);
            txtPestaneos_primer_minuto = (TextView)
findViewById(R.id.txtPestaneos_primer_minuto);
            txtPestaneos = (TextView) findViewById(R.id.txtPestaneos);
            txtdormido = (TextView) findViewById(R.id.txtdormido);
            txtacel = (TextView) findViewById(R.id.txtacel);
            txtvelocidad= (TextView) findViewById(R.id.txtvelocidad);
               //Acciones
            btnConectar.setOnClickListener(new View.OnClickListener()
//Se define lo que ocurrirá al presionar el botón
 {
            @Override
            public void onClick(View v) {
                conectar();
            }
        });
            @Override
            public void onCheckedChanged(CompoundButton arg0, boolean
isChecked) {
                tgDevice.stop();
                tgDevice.close();
                onResume();
                onStart();
                timerTask.run();
        });
       // Botón de salida (final de la aplicación)
      //Desconectamos lo que se ha utilizado en la aplicación
       btnSalir.setOnClickListener(new View.OnClickListener() {
             @Override
             public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_MAIN);
                intent.addCategory(Intent.CATEGORY_HOME);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                onStop();
```

```
                finish();
                timerTask.cancel();
        }
    });
}
// Creo la clase para el GPS y tomo los datos de mi posición acual
public class Miposicion implements LocationListener {
        @Override
        public void onLocationChanged(Location location) {
            location.getLatitude();
            location.getLongitude();
            latitud= (float) location.getLatitude();
            longitud=(float)location.getLongitude();
        }
        @Override
        public void onStatusChanged(String provider, int status,
Bundle extras) {
        }
        @Override
        public void onProviderEnabled(String provider) {
        }
        @Override
        public void onProviderDisabled(String provider) {

        }
    }
        //Defino los parametros del temporizador
        Timer timer = new Timer();
        private TimerTask timerTask;
{
    timerTask = new TimerTask() {
        @Override
        public void run() {
            //Llamo de nuevo al GPS para que tome la posición en este
momento
            class Miposicion1 implements LocationListener {

                @Override
                public void onLocationChanged(Location location) {
                    location.getLatitude();
                    location.getLongitude();
```

```
                         latitud1 = (float) location.getLatitude();
                         longitud1 = (float) location.getLongitude();
                     }
                     @Override
                     public void onStatusChanged(String provider, int
status, Bundle extras) {
                     }
                     @Override
                     public void onProviderEnabled(String provider) {
                     }
                     @Override
                     public void onProviderDisabled(String provider) {
                     }
                 }
                 //Una vez conseguido el pestañeo típico del conductor le
restamos el valor del descenso de pestañeos típicos ante la fatiga por
conducción y decidimos si está fatigado o no en función del resultado
                 if (pestaneos <= Pestaneosprimerminuto - fatigado) )&&(
concentracion<=50) {
                     Vibrator v = (Vibrator)
getSystemService(getApplicationContext().VIBRATOR_SERVICE);
                     v.vibrate(5000);
                 }
                 //Si el valor obtenido es inferior al de una persona no
fatigada el terminal vibrará durante 5 segundos

                 if (theta >= thetadormido1)&&( theta <= thetadormido2) {
                     Vibrator v = (Vibrator)
getSystemService(getApplicationContext().VIBRATOR_SERVICE);
                     v.vibrate(5000);
                     Dormidoporondas=1;
                 }
                if (theta >= thetadormido1)&&( theta <= thetadormido2) {
                     Dormidoporondas=0;
                 //Si el valor obtenido está entre los valores reconocidos
para las ondas tipo theta. Esto nos indica que está en fase de
fatigarse, por lo que el terminal vibrará durante 5 segundos
                 if (pestaneos == 0) {
                     Vibrator v = (Vibrator)
getSystemService(getApplicationContext().VIBRATOR_SERVICE);
                     v.vibrate(5000);
```

```
                dormido = 0;
            }
            if (pestaneos >= 1) {
                dormido = 1;
            }
            //Fórmula del Haversine para determinar distancia en
metros desde las coordenadas GPS
            double lat1 = latitud / 1E6;
            double lat2 = latitud1 / 1E6;
            double lon1 = longitud / 1E6;
            double lon2 = longitud1 / 1E6;
            double dLat = Math.toRadians((lat2 - lat1));
            double dLon = Math.toRadians(lon2 - lon1);
            double a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *
Math.sin(dLon / 2) * Math.sin(dLon / 2);
            double c = 2 * Math.asin(Math.sqrt(a));
            //El resultado se multiplica por el Radio de la tierra y
por 60 (minutos) y se divide por 1000 para obtener Km/h
            velocidad = (float)((Radius * c) * 60) / 1000;
                        //Pongo el contador de pestañeos a 0 para
volver a empezar, cuento uno a la variable primer_minuto y pongo las
coordenadas para el siguiente ciclo
            pestaneos = 0;
            primer_minuto++;
            longitud1 = longitud;
            latitud1 = latitud;
        };
    //Defino el tiempo a temporizar
        timer.schedule(timerTask,60000,60000);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        return id == R.id.action_settings ||
super.onOptionsItemSelected(item);
```

```
    }

//A partir de este punto defino los casos y utilizo el del pestañeo
para poder contar y definir mis datos para calcular la fatiga

public void conectar() {
        if (tgDevice.getState() != TGDevice.STATE_CONNECTING &&
tgDevice.getState() != TGDevice.STATE_CONNECTED);
    }
    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:
                            lblEstado.setText("Dispositivo en
reposo");
                            break;
                        case TGDevice.STATE_CONNECTING:
                            lblEstado.setText("Conectando...");
                            break;
                        case TGDevice.STATE_CONNECTED:
                            tgDevice.start();
                            lblEstado.setText("Conectado");
                            break;
                        case TGDevice.STATE_NOT_FOUND:
                            lblEstado.setText("Dispositivo no
encontrado");
                            break;
                        case TGDevice.STATE_NOT_PAIRED:
                            lblEstado.setText("Dispositivo no
vinculado");
                            break;
                        case TGDevice.STATE_DISCONNECTED:
                            lblEstado.setText("Desconectado");
                    }
                    break;
                case TGDevice.MSG_POOR_SIGNAL:
                    break;
```

```
                    case TGDevice.MSG_ATTENTION:

txtConcentracion.setText(String.valueOf(msg.arg1));
                        break;
                    case TGDevice.MSG_BLINK:
                        //Con cada pestañeo sumo 1 a la variable pestañeo
y la ploteo en pantalla
                        pestaneos++;
                        txtPestaneos.setText(String.valueOf(pestaneos));
                        //En este bucle if defino si el usuario se ha
quedado                           dormido sin pestañear. Si los
pestañeos son >0 mandará el                        mensaje que está
dormido
                        if(primer_minuto==0) {
                            Pestaneosprimerminuto++;
txtPestaneos_primer_minuto.setText(String.valueOf(Pestaneosprimerminut
o));
                        if (dormido<=0) {
                            txtdormido.setText("NO");
                        }
                        if (dormido>=1){
                            txtdormido.setText("SI");
                         }
                        txtvelocidad.setText(String.valueOf(velocidad));
                       //Depende del valor del entero dormidoporondas, el
texto reflejado en la pantalla variará
                       if (dormidoporondas==1) {
                           txttheta.setText("Duerme");
                                     }
                      if (dormidoporondas==1) {
                          txttheta.setText("No duerme");
                      }
                   }
                        break;
                   case TGDevice.MSG_LOW_BATTERY:
                        Toast.makeText(getApplicationContext(), "¡Bateria
       baja!", Toast.LENGTH_LONG).show();
                        break;
                  case TGDevice.MSG_THETA:
                        theta++;
                        break;
```

```
            }
        }
    };
//Defino el acelerometro y los límites de aviso
    @Override
    protected void onResume(){
        super.onResume();
        SensorManager sm= (SensorManager)
getSystemService(SENSOR_SERVICE);
        List<Sensor> sensors=
sm.getSensorList(Sensor.TYPE_ACCELEROMETER);
        if (sensors.size()>0){

sm.registerListener(this,sensors.get(0),SensorManager.SENSOR_DELAY_GAM
E);
            Toast.makeText(this,"Dispositivo con
acelerómetro",Toast.LENGTH_SHORT);
        }
        if (sensors.size()==0){
            Toast.makeText(this,"Dispositivo sin
acelerómetro",Toast.LENGTH_SHORT);
        }
    }
    protected void onPause(){
        SensorManager mSensorManager=(SensorManager)
getSystemService(SENSOR_SERVICE);
        mSensorManager.unregisterListener(this, mAccelerometer);
        super.onPause();
    }
    protected void onStop(){
        SensorManager mSensorManager=(SensorManager)
getSystemService(SENSOR_SERVICE);
        mSensorManager.unregisterListener(this, mAccelerometer);
        super.onStop();
    }
    @Override
    public void onSensorChanged(SensorEvent event) {
        x= event.values[SensorManager.AXIS_X];
        txtacel.setText(String.valueOf(x));
        if (x>=10){
            Vibrator v = (Vibrator)
```

```
getSystemService(getApplicationContext().VIBRATOR_SERVICE);
            v.vibrate(3000);
        }
        if (x<=-10){
            Vibrator v = (Vibrator)
getSystemService(getApplicationContext().VIBRATOR_SERVICE);
            v.vibrate(3000);
        }
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
}
```