UNIVERSIDAD VERACRUZANA

FACULTAD DE ESTADÍSTICA E INFORMÁTICA

CÓDIGO DE LA APP “MYOTAMOPROTOTYPE”

ANDROID VERSIÓN 4.4

PRESENTA:
JORGE LUIS JÁCOME DOMÍNGUEZ

DIRECTOR:
M. C. C. ALFONSO SÁNCHEZ OREA

CODIRECTOR:
DR. RAFAEL ROJANO CÁCERES

XALAPA, VER. JUNIO DEL 2016

Índice

1 Ajustes para abrir el proyecto

La APP puede ser instalada en Equipos móviles con sistema operativo Android 4.4 (API 19) o superior. Sin embargo como se aprecia en la Figura 1, el proyecto del cual fue creada se compilo utilizando Android 6.0 (API 23). También fue agregada una referencia a “com.echo:holographlibrary:1.0” para compilar con el proyecto, pues esta librería adicional permitirá graficar en tiempo real el comportamiento de los sensores de la Pulsera MYO.

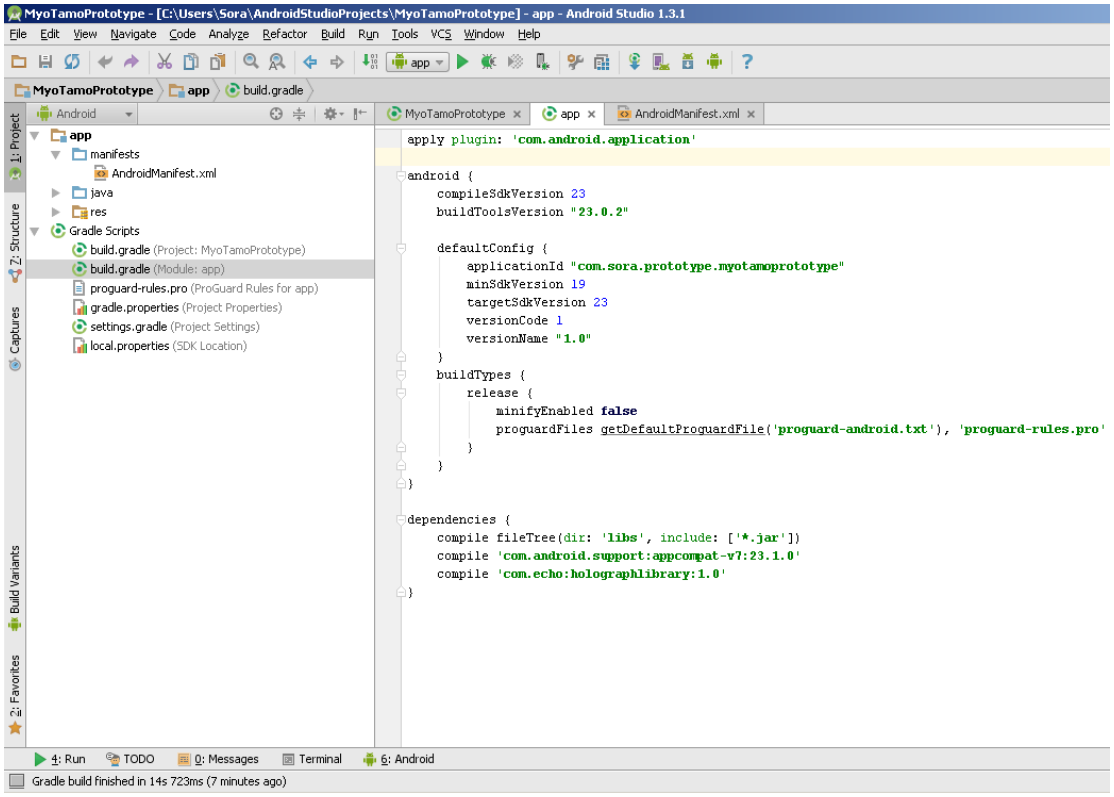

Figura 1

Como se aprecia en la Figura 2, se agregó una referencia al repositorio “https://github.com/powerje/mvn-repo/raw/master/”, como parte del anexo de la librería para graficar el comportamiento de la información de electromiografía de la Pulsera MYO.

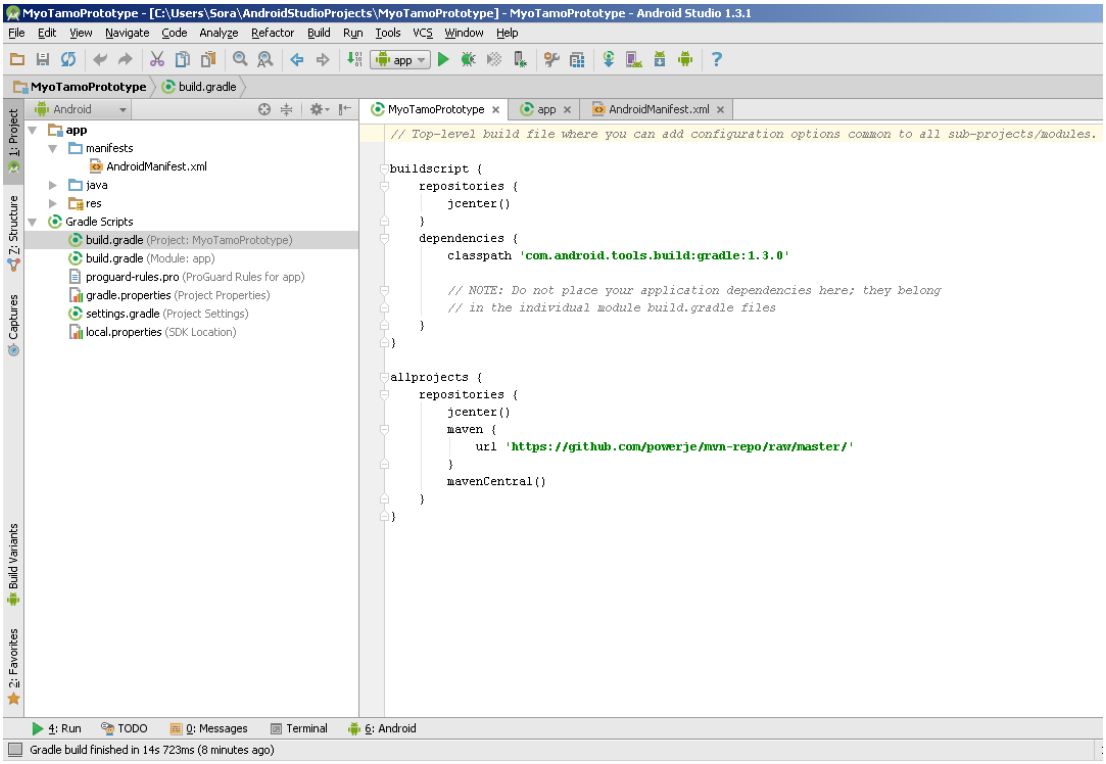

Figura 2

Para el acceso Bluetooth y acceso a la memoria del equipo móvil, como se ve en la Figura 3, se anexan los permisos:

- <uses-permission android:name="android.permission.BLUETOOTH" />
- <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
- <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

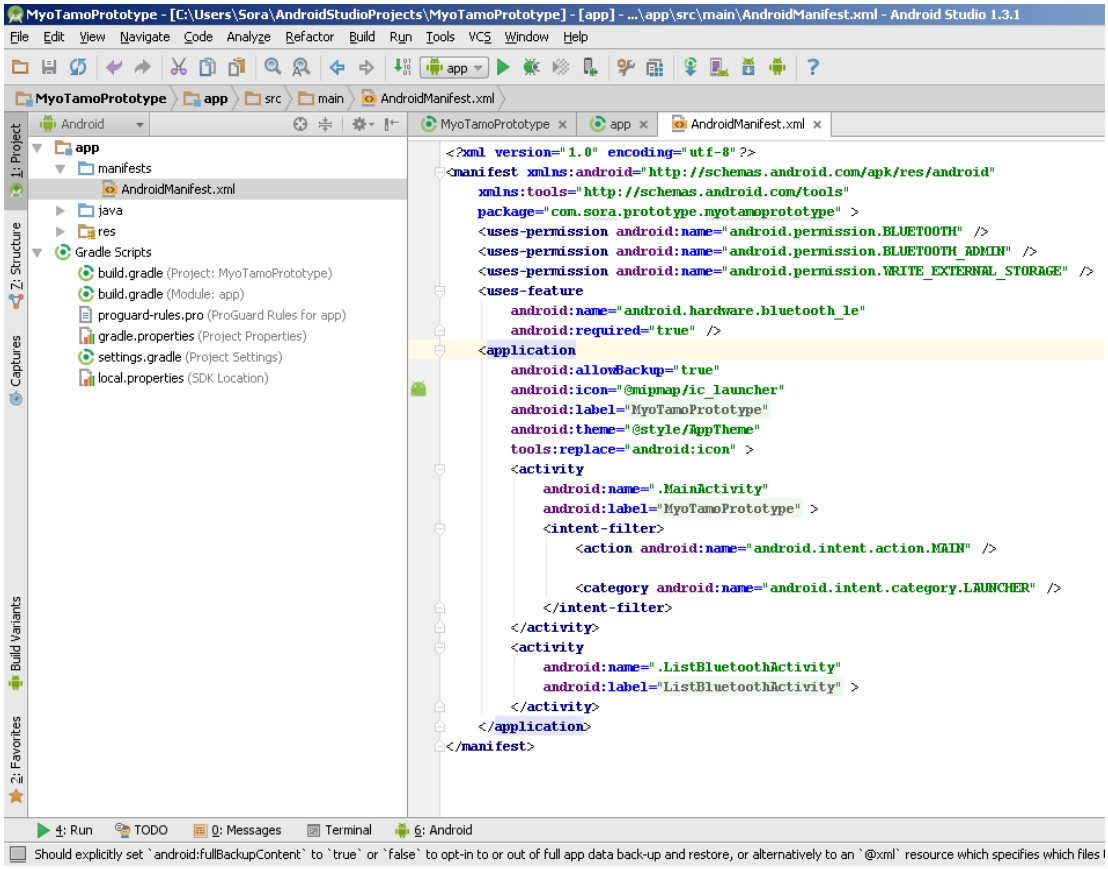

Figura 3

2 Partes importantes del código

2.1 Inicialización de componentes

Es importante destacar que el iniciar la APP se crea un “GestureManager” o Administrador de señas, que recibe como parámetro un “GestureFileReader” (Ver Figura 4) siendo este último la parte de importante, pues desde aquí se carga lo siguiente:

- Las palabras del diccionario de palabras que empleara el predictor de texto, agregando que este predictor se encontrara alojado dentro del “GestureManager”.
- Los pesos y bias para la creación de las RNA (Red o Redes Neuronales Artificiales), definiendo los valores de estas así como sus capas.

```
@Override // Al inicializar la Actividad
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initializeInterface();
    gestureManager = new GestureManager(new GestureFileReader());
}
```

Figura 4

2.2 Graficas de datos EMG

Para ir a la sección de código donde se gráfica, al igual que para ir al uso de Administrador de señas, como se muestra en la Figura 5 se debe ir a la clase “MyoGattCallback” y buscar el método “onCharacteristicChanged”.

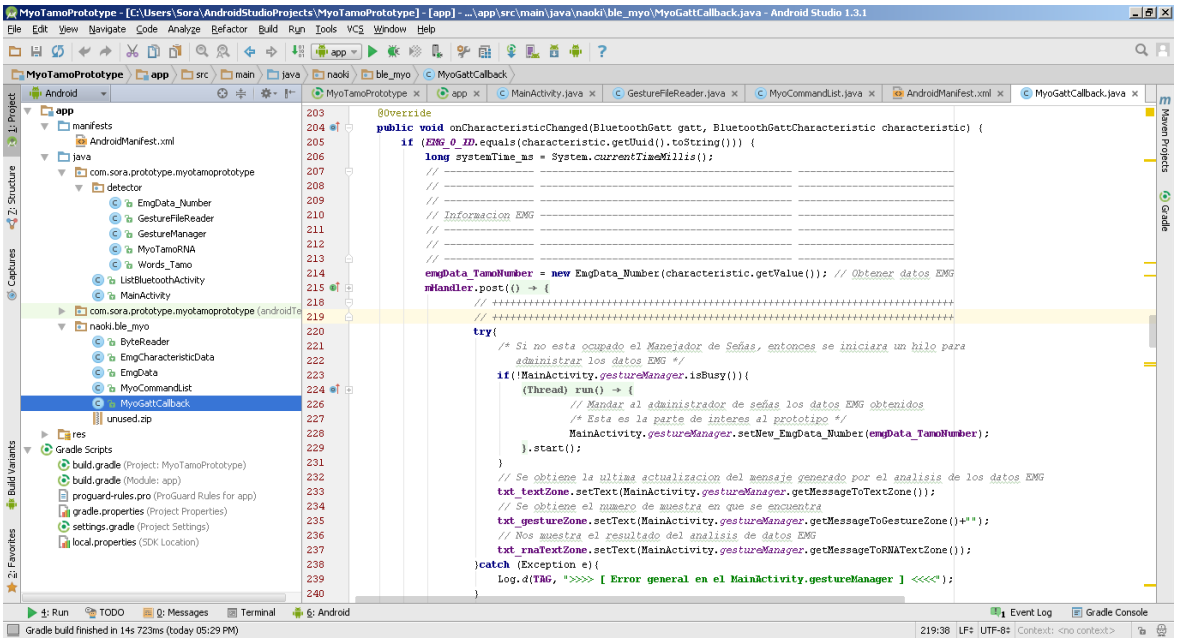

Figura 5

Estando en el método antes mencionado, como muestra la Figura 6, es a partir de esta parte que se encuentra el código correspondiente a la creación y actualización de los valores de las gráficas.

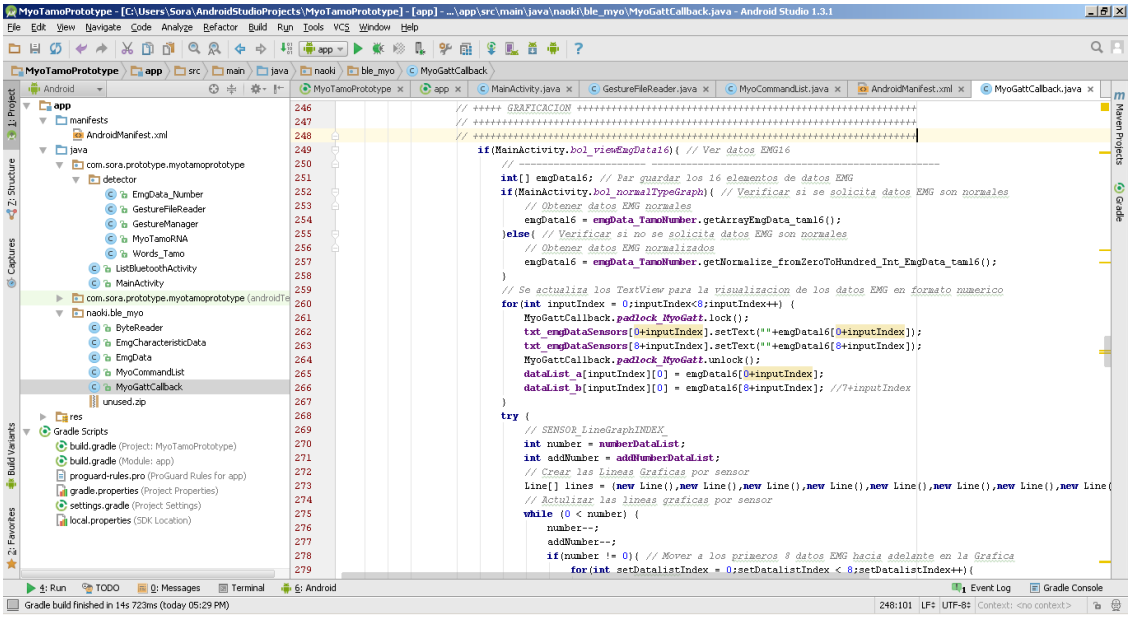

Figura 6

2.3 Administrador de señas

Para ir a la sección donde se comienza el uso del Administrador de Señas, al igual que para la sección donde se gráfica, como se ve en la Figura 7 se debe ir a la clase “MyoGattCallback” y buscar el método “onCharacteristicChanged”. Ahí se encontrara una parte donde se inicializa un hilo donde se llama al “gestureManager” o “Administrador de señas” desde la clase “MainActivity”, notara que se convoca al método “setNew_EmgData_Number” el cual recibe la data EMG que acaba de ser recibida, siendo esta parte la encargada de analizar la data EMG. Además, Tras la creación de dicho hilo vera como se actualizan los siguientes componentes:

- La zona de texto o “txt_textZone”.
- El contador de muestras o “txt_gestureZone”.
- La zona para el resultado del análisis de los datos EMG o “txt_rnaTextZone”.

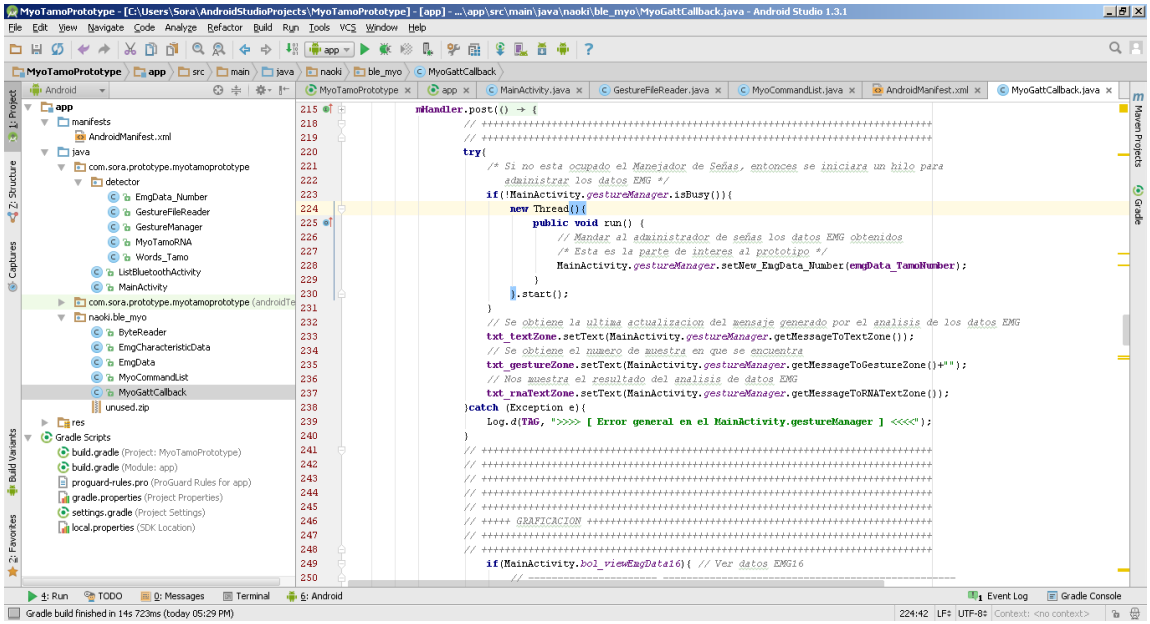

Figura 7

El método “setNew_EmgData_Number” del “GestureManager” tiene dos partes clave como se expone en la Figura 8, la primer parte es aquella que lleva el control del número de muestras de datos EMG recogidos (método “goHandGestureManager” que recibe los datos EMG), y la segunda parte es aquella lleva el análisis y traducción de la seña (método “translationHandGestureDetect”) junto a la administración del resultado (método “adminTranslation” que recibe una traducción).

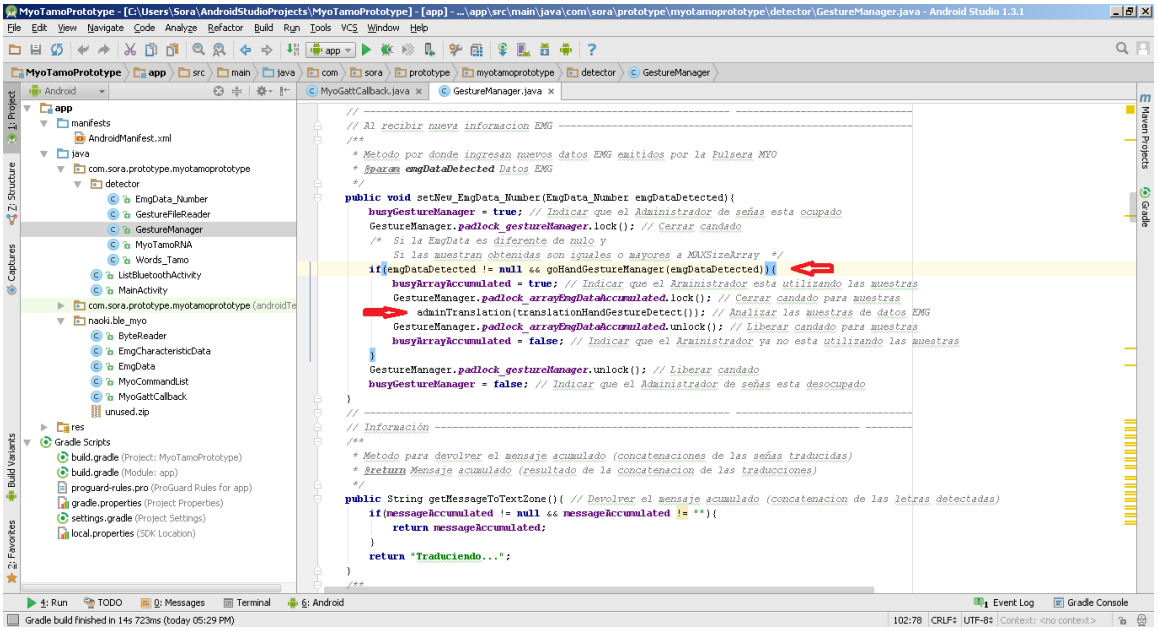

Figura 8

2.3.1 Acumulación de muestras

La acumulación de muestras es administrada como se observa en la Figura 9 por el método “goHandGestureManager”, el cual recibe los datos EMG de entrada.

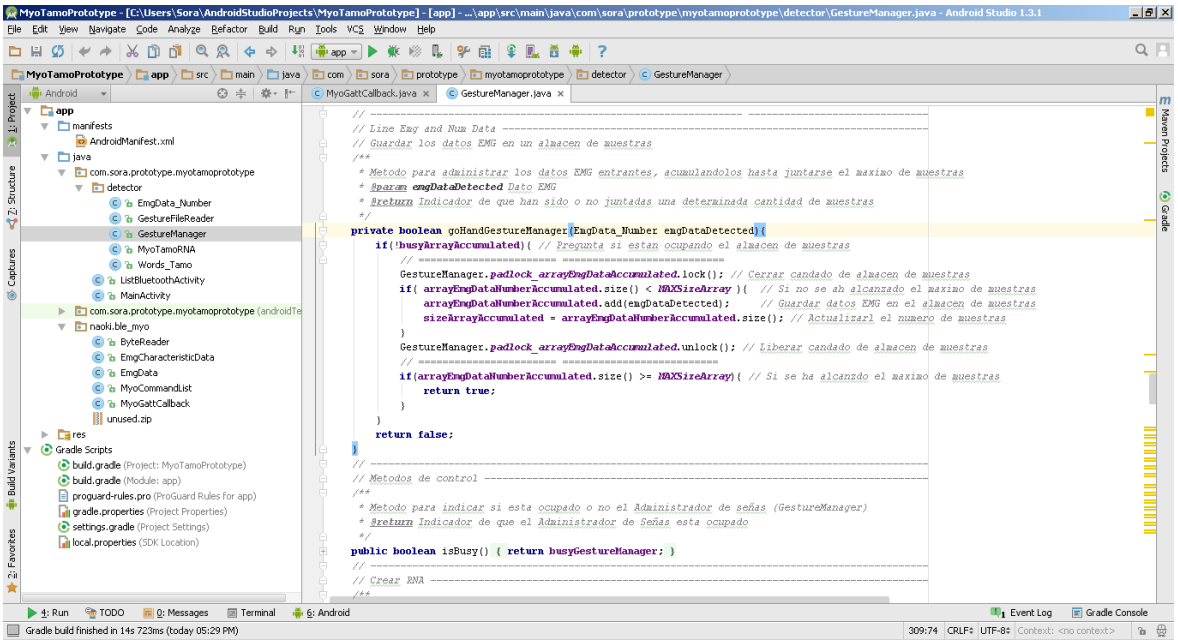

Figura 9

2.3.2 Análisis de los datos EMG

Las muestras de datos EMG se analizan hasta alcanzar el límite de muestras definido en el código (siendo estas un total de 25 lo equivalente a 1 segundo de muestreo). Como se aprecia en la Figura 10 en el método “translationHandGestureDetect”, el análisis de las muestras viene acompañada de la aplicación de las RNA (Red o Redes Neuronales Artificiales), cuyos resultados interactúan entre sí para definir si en verdad ha sido detectada una seña y emitir la traducción de esta.

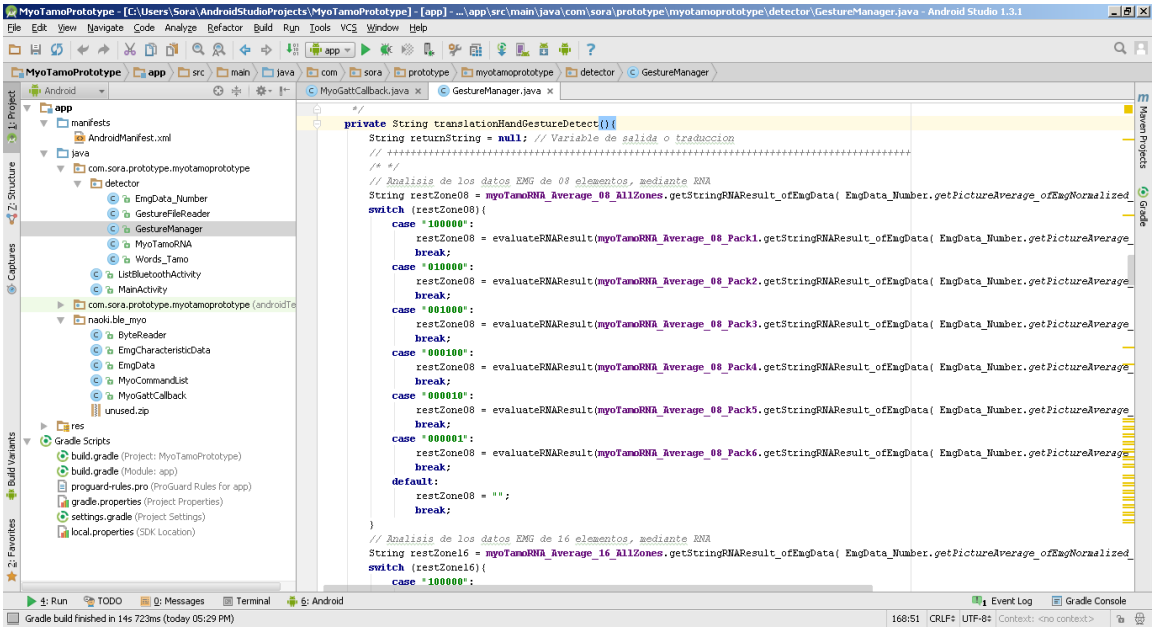

Figura 10

2.3.3 Predictor de texto

El predictor de texto se acción dentro del administrador de traducciones, como se ve en la Figura 11 es dentro del método “adminTranslation” que se aplica la predicción de texto, el cual se encuentra implementado mediante el método “fixedUltimateWord” de la variable “wordsTamo” y que recibe de entrada la concatenación del mensaje actual más la nueva traducción.

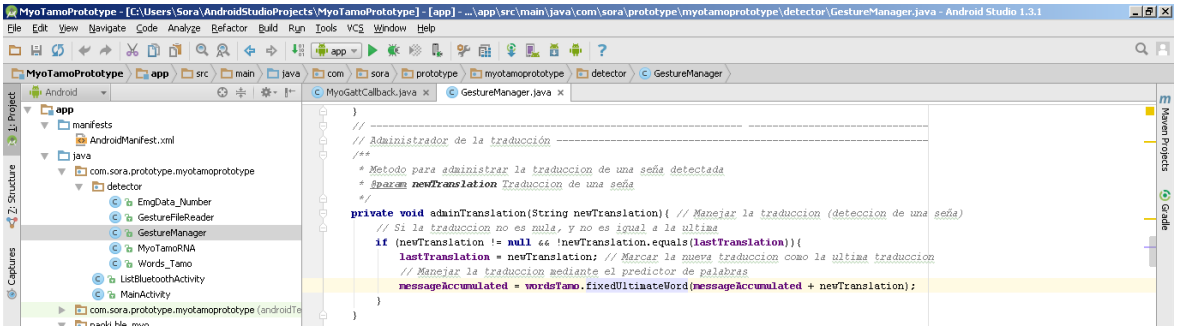

Figura 11

3 Código comentado para obtener máximos de datos EMG y guardar patrones de muestras de datos EMG

Se observan en la Figura 12 dos flechas rojas dirigidas al inicio de dos partes de código comentando, donde:

- La primer parte corresponde a código para definir los valores EMG máximos alcanzados por una persona, cuyo resultado aparecerá en consola.
- La segunda parte corresponde a código necesario para guardar muestras, patrones, etc., de datos EMG dentro de archivos separados por coma (archivos ".csv").

Cabe mencionar que este código se encuentra dentro del método "translationHandGestureDetect", misma sección encargada del análisis de las muestras de datos EMG.

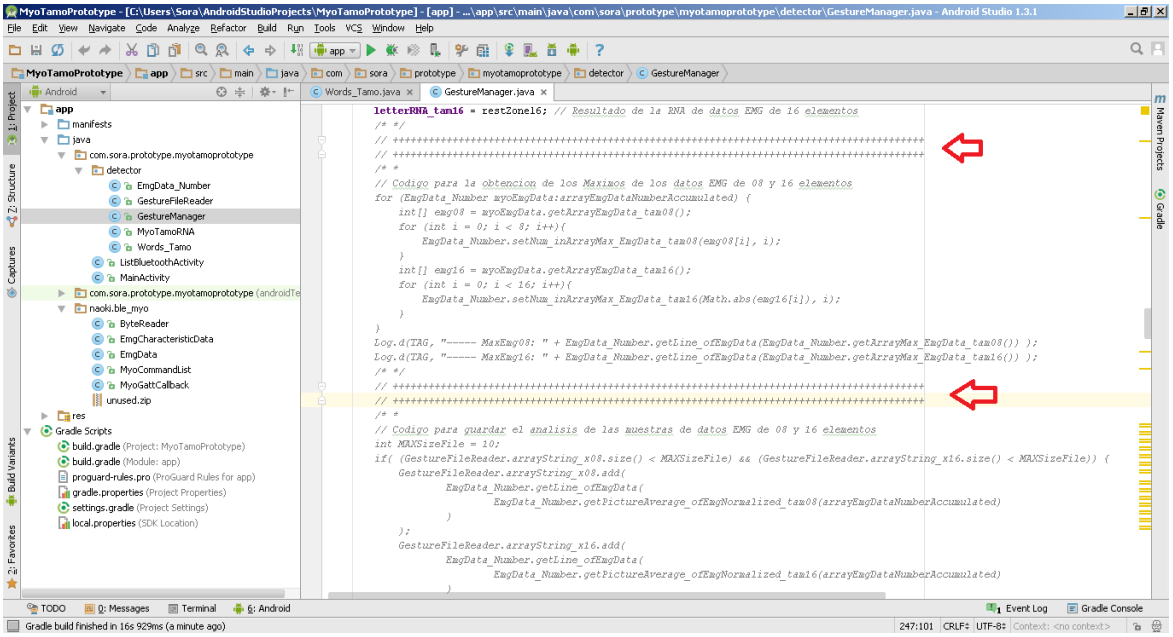

Figura 12

4 Archivos para script de MatLab

Para el entrenamiento de las Redes Neuronales Artificiales (RNA) se fabricaron 2 scripts (para MatLab) que generan los archivos con los pesos y bias, los cuales solo deben ser copiados dentro de la carpeta "MyoTamoPrototype" del equipo móvil para que las RNA de la APP se inicialicen con esos datos. Como se aprecia en la Figura 13, los script tiene el nombre de "myoClasificadorRNA_Experimental_x08.m" y "myoClasificadorRNA_Experimental_x16.m" y se alimentan con los patrones de muestras de datos EMG almacenados en archivos ".csv" en "..\PatronesEMG_x08\archivo.csv" y "..\PatronesEMG_x16\archivo.csv". Agregando que estas Redes Neuronales se diferencian por el tipo de patrones EMG con los cuales trabajan, siendo estos los datos EMG de 08 elementos y los datos EMG de 16 elementos.

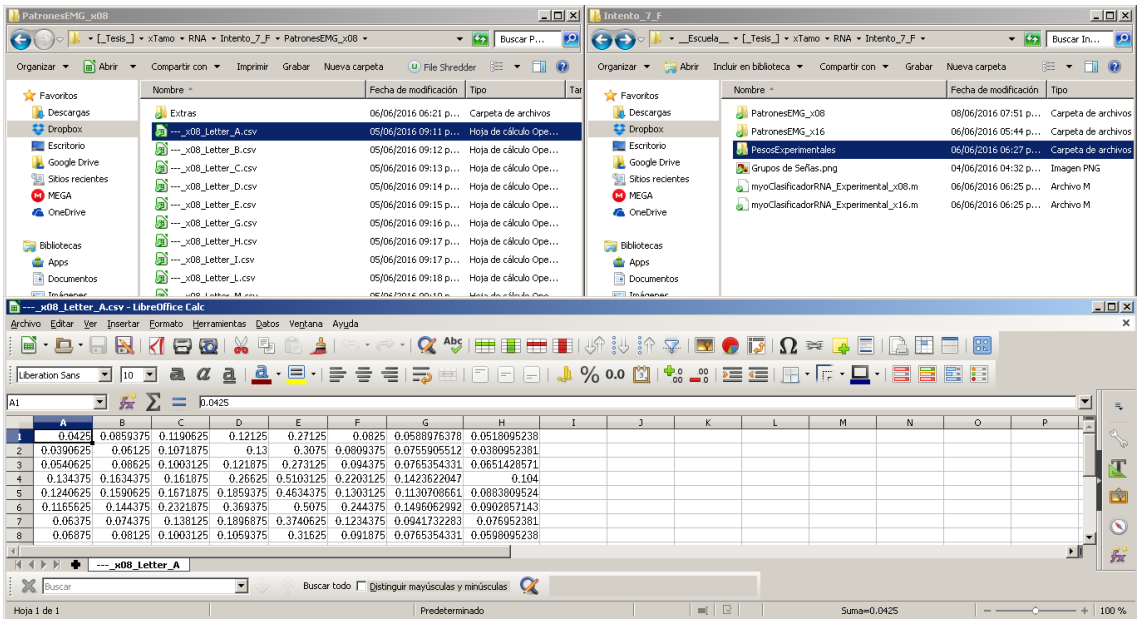
PatronesEMG_x08
Intento_7_F
---_x08_Letter_A.csv - LibreOffice Calc
---_x08_Letter_A

Figura 13