uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería de Sistemas Audiovisuales
2017-2018

Trabajo Fin de Grado

"Detección de caracteres mediante accelerometría en dispositivos Android"

Autora:

Ana del Carmen Cañas Cañas

Tutor:

Mario Muñoz Organero

Leganés, Septiembre de 2018

RESÚMEN

El objetivo principal de este Trabajo Fin de Grado es la detección de caracteres, en concreto de las vocales del abecedario, a partir de señales de acelerometría en dispositivos Android. Para ello, se desarrollará un software en Android Studio que permitirá extraer muestras de datos generadas por un acelerómtero, para luego procesarlas y realizar una serie de operaciones matemáticas que lleven a la correcta identificación de la vocal a ser reconocida. El software será desarrollado previamente en Matlab, pues éste es uno de los programas más versátiles que existen para la realización de cálculos matemáticos.

AGRADECIMIENTOS

Deseo expresar mi agradecimiento a mi familia, mi pareja, mis amigos y mi tutor, ya que sin el apoyo y la motivación que me han brindado la realización de este trabajo no habría sido posible. Dedico, pues, mi obra a todos ellos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

En la primera sección de este documento se indican los motivos por los cuales se ha escogido el tema "Detección de caracteres mediante acelerometría en dispositivos Android" y los objetivos marcados en el desarrollo del proyecto, así como su planificación y presupuesto aproximado. Por último, se resume brevemente cada uno de los capítulos que componen la obra.

1.1. Motivación

Durante los primeros años de carrera se aprende a programar en Java, Matlab y C. El software desarrollado a lo largo de este periodo sirve para el alumno entender el fundamento del lenguaje que se está usando, pero no tiene una aplicación tangible.

Sin embargo, la situación va cambiando conforme se avanza de curso. Sin ir más lejos, en una de las asignaturas del último año se hace algo tan práctico y tangible como diseñar una aplicación para dispositivos Android, ya que cualquier persona, tenga o no conocimientos de programación, va a poder darle un uso real tras su ejecución. A estas alturas del grado, el trabajo realizado genera no sólo el aprendizaje de conocimientos, si no también mejoras en la calidad de vida de otras personas, motivo por el cual la motivación del alumno aumenta exponencialmente. Ahora se quieren crear tecnologías que rompan con lo establecido o, como mínimo, que lo mejoren, y las fuentes de inspiración son proyectos tales como "PhonePoint Pen: Using Mobile Phones to Write in Air", escrito por Sandip Agrawal y otros investigadores de la Universidad Duke (Estados Unidos de América) [1], cuyo objetivo es convertir los movimientos de un smarthphone en caracteres del alfabeto. Se hablará más profundamente de este artículo en el siguiente capítulo, pues es la mayor de las motivaciones para la realización de este proyecto.

1.2. Objetivos

El objetivo de este proyecto es desarrollar un algoritmo sencillo para el reconocimiento de vocales a partir de señales de acelerometría, así como la implementación y validación de dichos algoritmos en Matlab para su posterior migración a un dispositivo móvil con sistema operativo Android. Se partirá de los datos del acelerómetro de un dispositivo móvil que un usuario moverá trazando sobre una superficie plana la escritura de una vocal. En base a características básicas como la localización relativa y amplitudes de máximos y mínimos de la señal de aceleración lineal en crudo o bien la estimación de la posición a partir de ella, se tendrán que definir características manuales que permitan el reconocimiento de las 5 vocales.

Para la consecución de este objetivo, son necesarios los siguientes hitos intermedios:

- Desarrollo del algoritmo.
- Implementación en Matlab.
- Migración del software a Android Studio.
- Completar la aplicación en Android Studio.

1.3. Planificación y desarrollo

El presente trabajo comenzó a desarrollarse a comienzos de abril de 2018 a raíz de fijarse los objetivos, momento en el cual se procedió a investigar acerca de la tecnología que iba a ser utilizada.

Pasados treinta días, se empezó a desarrollar el algoritmo, cosa que, a partir de junio, fue haciéndose al mismo tiempo que éste se implementaba.

Un mes más tarde, se comenzó a trabajar con Android Studio, pero nunca se dejó de programar en Matlab, pues el algoritmo tuvo que ser modificado varias veces a lo largo del proyecto.

Por último, se dispuso a redactar la memoria con un margen de tiempo razonable.

A continuación, se presenta en una tabla la consecución de las tareas mencionadas anteriormente:

	abr/18	mayo/18	jun/18	jul/18	ago/18	sept/18
Trabajo de investigación						
Desarrollo del algoritmo						
Implementación en Matlab						
Migración a Android Studio						
Completar la aplicación en Android Studio						
Redacción de la memoria						

Tabla 1. Desarrollo del proyecto

1.4. Presupuesto

La realización de este proyecto supone un coste económico que resulta de la suma de los costes personales y materiales.

❖ Costes personales

El coste total asociado al trabajo de un participante del proyecto viene dado por la siguiente fórmula:

$$C_{participante} = t_t * C_h$$

Donde:

- t_t: tiempo total empleado (horas)
- C_h: coste horario (euros)

El trabajo del alumno se ha estimado en una media de 4 horas diarias, durante los días lectivos de los seis meses en los que se ha desarrollado el trabajo. Considerando una media de 15 días de trabajo al mes, el tiempo total empleado por el alumno asciende a 360 horas. El coste horario del alumno se ha fijado en 8 euros, por lo que:

$$C_{alumno} = 360 * 8 = 2.880\ €$$

El tiempo dedicado a tutorías ha sido, aproximadamente, de una hora cada quince días, lo que supone un total de 12 horas empleadas por el tutor.
El coste unitario de estas horas es de 15 euros, por lo que:

$$C_{tutor} = 12 * 15 = 180\ €$$

Así pues, los costes personales serán la suma de todos los costes totales asociados al trabajo de los participantes:

$$C_{personales} = C_{alumno} + C_{tutor} = 2.880 + 180 = 3.060\ €$$

❖ Costes materiales

El coste total asociado a cada material usado en el proyecto viene dado por la siguiente fórmula:

$$C_{material} = C_m + \frac{t_u}{t_d}$$

Donde:

- c_m: coste real material (euros)
- t_u: tiempo uso material durante el proyecto (horas)
- t_d: tiempo depreciación material (horas)

Se han utilizado los siguientes materiales en el desarrollo de este trabajo:

❖ Ordenador:

$$C_{ordenador} = 2.700 * \frac{360}{8.760} = 110{,}9\ €$$

❖ Dispositivo móvil (iPhone):

$$C_{movil} = 900 * \frac{240}{10.920} = 19{,}8\ €$$

❖ Licencia software Matlab:

$$C_{matlab} = 69 * \frac{180}{8.760} = 1{,}4\ €$$

❖ Licencia Microsoft Office:

$$C_{office} = 149 * \frac{120}{8.760} = 2{,}04\ €$$

Por lo tanto, los costes materiales se obtienen sumando todos los costes totales asociados a cada material:

$$C_{materiales} = C_{ordenador} + C_{movil} + C_{matlab} + C_{office} =$$

$$= 110{,}9 + 19{,}8 + 1{,}4 + 2{,}04 = 134{,}14\ €$$

Por lo tanto, el coste económico total del proyecto es:

$$C_{proyecto} = C_{personales} + C_{materiales} = 3.060 + 134{,}14 = 3.194{,}14\ €$$

1.5.Estructura de la obra

El presente documento está formado por ocho capítulos, cuyo contenido se resume a continuación:

- Capítulo 2: Se presenta la situación actual de la tecnología utilizada en el trabajo, así como el análisis de ésta y la propuesta que se ha llevado a cabo en dicho contexto.

- Capítulo 3: Se analizan tanto los requisitos para la realización del proyecto como las soluciones existentes para cumplir con el objetivo de éste. Finalmente, se presenta la solución escogida de entre todas las expuestas.

- Capítulo 4: Se explica cómo se ha diseñado la solución por la que se ha optado en el capítulo anterior, no sin antes hacer un breve resumen de las herramientas que se han empleado para ello.

- Capítulo 5: Se detalla cómo se ha implementado el algoritmo descrito en el último apartado del capítulo anterior.

- Capítulo 6: Recoge las pruebas que se han realizado para verificar que la solución funciona correctamente.

- Capítulo 7: Se exponen las conclusiones a las que se han llegado teniendo en cuenta todo lo anterior.

- Capítulo 8: Se citan distintas acciones que podrían mejorar el contenido de este trabajo.

CAPÍTULO 2. ESTADO DEL ARTE

En este apartado se presenta la situación actual de la tecnología utilizada en el trabajo, así como el análisis de ésta y la propuesta que se ha llevado a cabo en dicho contexto.

2.1. Crítica al estado del arte

En la antigüedad, los equipamientos industriales eran manipulados exclusivamente por personas, por lo que los procesos de monitoreo eran 100% manuales. Esto no era del todo eficiente, pues era imposible que la mano de obra estuviese supervisando las máquinas constantemente y que su tiempo de reacción a cualquier problema fuese siempre inmediato. En este contexto, surgieron aplicaciones que pretendían reemplazar el trabajo manual. Éstas fueron haciéndose cada más precisas y sofisticadas, provocando así que los equipamientos electrónicos y las plantas industriales obtuviesen respuestas más rápidas y realistas.

Con el objetivo de mejorar los procesos de monitoreo usando sistemas autónomos, nació el concepto de sensor.

- Sensor:

  Un sensor es un dispositivo capaz de dar una respuesta eléctrica a estímulos físicos externos. Se trata de un elemento que traduce una magnitud real, tanto física (temperatura, longitud, aceleración, etc.) como química (nivel de azúcar en sangre, nivel de un gas peligroso que flota por el aire, etc.), en unidades que pueden ser interpretadas de forma sencilla por un dispositivo electrónico.

  Dichas magnitudes son traducidas por el sensor a una unidad eléctrica, generalmente a voltaje. Después de todo un proceso, ese voltaje es analizado por un sistema inteligente que da respuesta a la acción que se haya llevado a cabo sobre él. Para ello, se basará en datos que haya registrado con anterioridad.

Figura 1. Sensores reales. [2]

La utilización de sensores en la cadena de producción de artículos aumenta la calidad de dicho proceso, ya que con su uso se mejora la detección de fallos y desgastes en los equipamientos utilizados. Se reacciona a una velocidad antes inimaginable, provocándose así la reducción de costes relacionados con pérdidas de producción. Sin embargo, el uso de sensores no se limita a aplicaciones industriales, ya que éstos están presentes en la vida cotidiana de la mayor parte de la población del planeta tierra. Sin percatarse de ello, la mayoría de las personas los usan a diario, pues los dispositivos móviles contienen una gran cantidad de ellos. A continuación, se mencionan algunos de los sensores contenidos en los dispositivos móviles [3]:

- GPS (Global Positioning System – Sistema de posicionamiento global):

  Permite al usuario determinar su posición utilizando señales de radio emitidas por satélites. Se usa, principalmente, en aplicaciones de navegación tales como Google Maps.

- Giroscopio:

  Determina la orientación del dispositivo. Las aplicaciones que más lo usan son las de navegación, juegos y realidad aumentada.

- Magnetómetro:

  Mide la fuerza o dirección de una señal magnética. Suele ser utilizado como brújula en algunas aplicaciones.

- Sensor de luz:

  Identifica la intensidad luminosa del ambiente. Se usa para ajustar automáticamente el brillo de la pantalla del dispositivo móvil.

- Sensor de proximidad:

  Identifica la proximidad a la que un objeto está del dispositivo. Sirve para, entre otras cosas, bloquearlo automáticamente si éste es introducido en un bolsillo, bolso, etc.

- Acelerómetro:

Detecta el movimiento y el giro del dispositivo. Mide la aceleración en tres ejes, ya sea en metros partido por segundo al cuadrado (m/s^2) o en fuerza-G (g)

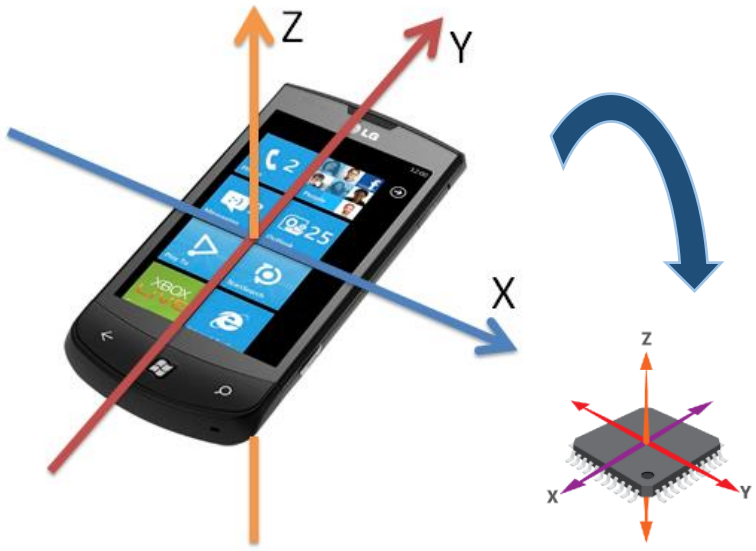

Figura 2. Representación ejes acelerómetro dispositivo móvil. [4] [5]

Es comúnmente utilizado para la rotación automática de la pantalla en el caso de cambiarse de una posición vertical a una horizontal, y viceversa.

Éste último es el sensor que se va a utilizar para la realización del proyecto, por lo que se va a proceder a estudiarlo con más detalle.

2.1.1. Acelerómetro

Un acelerómetro es un sensor que convierte aceleraciones (cambios de velocidad) en una señal eléctrica.

- Fundamento físico del acelerómetro:

Se trata de un sensor que está compuesto por dos partes fundamentales: una base, la cual está pegada al objeto del que se quiere medir la aceleración, unida a una masa por medio de un muelle. Cuando se mueve la base hacia arriba, la bola se queda atrás, estirándose así el resorte. Si se mide la distancia que el muelle se estira, se puede calcular la fuerza de gravedad.

Existen varios tipos, entre los que destacan los Piezoeléctricos, Piezorresistivos y Capacitivos.

❖ Acelerómetro Piezoeléctrico:

Está compuesto por un elemento piezoeléctrico, el cual se encuentra entre la base del acelerómetro y una masa sísmica en contacto directo con ambos [6].

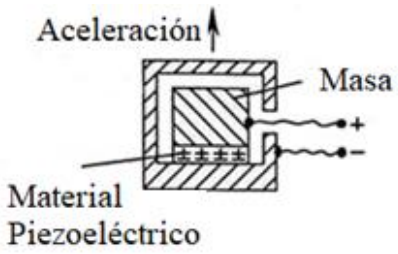

Figura 3. Fundamento físico del acelerómetro piezoeléctrico. [7]

Su funcionamiento obedece la segunda ley de Newton: cuando se produce un estímulo, la base del acelerómetro ejerce una fuerza en el material piezoeléctrico proporcional a la aceleración y a la masa sísmica. El elemento sensible genera una señal eléctrica a partir de la tensión causada, la cual se usa para determinar el valor de la aceleración.

Figura 4. Acelerómetro piezoeléctrico real. [8]

Este tipo de acelerómetro es utilizado comúnmente en locales que presentan vibraciones y choques.

❖ Acelerómetro Piezorresistivo:

Así como el piezoeléctrico, este tipo de acelerómetro también obedece la segunda Ley de Newton. Sin embargo, ambos se diferencian en el tipo de material utilizado y el tipo de magnitud eléctrica que varía en función de la aceleración [9].

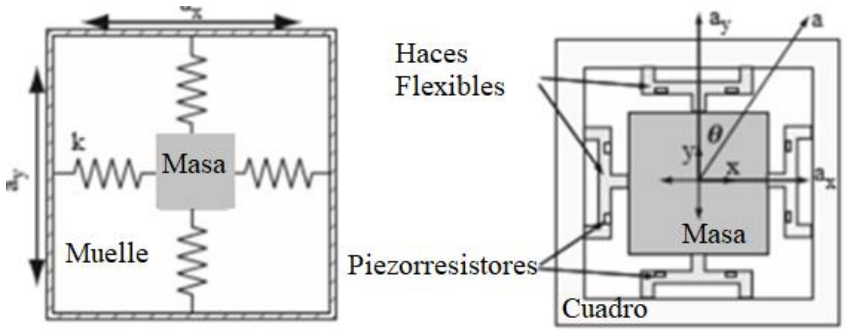

Figura 5. Fundamento físico del acelerómetro piezorresistivo. [10]

Este sensor está compuesto por un extensómetro hecho de material piezorresistivo, generalmente silicio, que, cuando es sometido a una fuerza, varía su resistencia eléctrica, la cual es utilizada para determinar la aceleración.

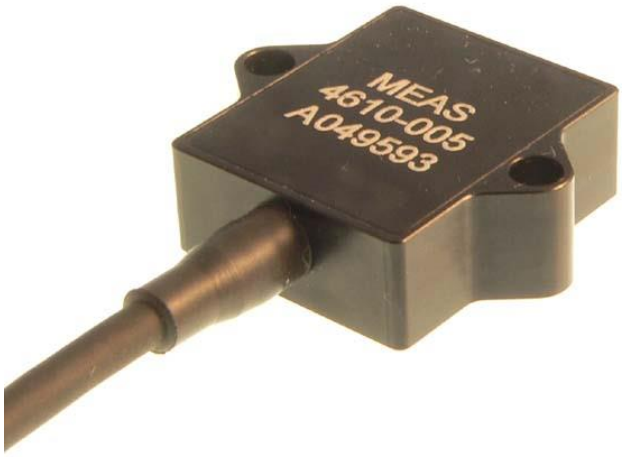

Figura 6. Acelerómetro piezorresistivo real. [11]

Su principal aplicación es la medición de choques. No se recomienda su uso para monitorizar vibraciones, ya que posee una baja sensibilidad.

❖ Acelerómetro Capacitivo:

Está compuesto por una serie de placas paralelas fijas, separadas por placas móviles que forman micro capacitores variables. Cuando es sometido a una aceleración, provoca el dislocamiento de las placas móviles que modifican el espacio entero y, consecuentemente, la variación de capacitancia [12]. Este cambio es detectado y transformado a través de algoritmos para determinar la aceleración.

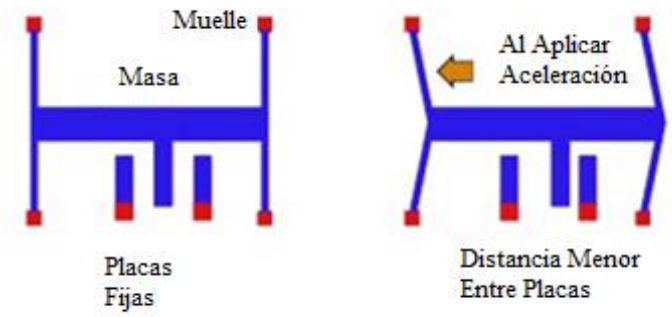

Figura 7. Fundamento físico del acelerómetro capacitivo. [13]

Dicho sensor, además de medir los valores de la aceleración dinámica, es capaz de determinar eventos estáticos, motivo por el cual presenta características prometedoras en la determinación de orientación y posición de objetos en dispositivos móviles y otras aplicaciones. Es, por lo tanto, el sensor comúnmente utilizado en dispositivos móviles.

Figura 8. Acelerómetro capacitivo real. [14]

Para utilizar este tipo de tecnología en smartphones, los fabricantes están creando sensores cada vez más compactos. La técnica MEMS (Micro Electro Mechanical System – Micro Sistemas electromecánicos) es bastante utilizada en la fabricación de algunos tipos de sensores. Los acelerómetros MEMS son dispositivos que se fabrican aplicando técnicas de microelectrónica, con la finalidad de crear estructuras mecánicas sensitivas de tamaño microscópico, las cuales están hechas de silicio y con los principios de funcionamiento de los acelerómetros capacitivos y piezorresistivos.

Figura 9. Ejemplo de MEMS. [15]

2.1.2. Aplicaciones acelerometría

Debido a su versatilidad, los acelerómetros suscitan el interés de muchos investigadores. Es el caso de Patrik Axelsson y Mikael Norrlöf, dos científicos suecos de la Universidad Linköping, que introdujeron en un brazo robótico un acelerómetro con seis grados de libertad (movimiento de dislocamiento a lo largo de los ejes x, y e z, y rotación de cada uno de ellos) para estimar su orientación y posición [16]. Para ello, realizaron dos experimentos. El primero consistió en dejar el robot parado y sólo mover su brazo. En concreto, lo cambiaron de posición 6 veces, para realizar así las respectivas mediciones de la aceleración en los ejes x, y e z. Esto lo hicieron para determinar sus valores inerciales y, de ese modo, definir su orientación. El segundo de los experimentos fue calcular el vector de la posición utilizando ecuaciones vectoriales, tomando como origen la base del robot y como final el acelerómetro. Dicho vector fue derivado dos veces en función del tiempo, dando como resultado el vector de la aceleración. Modificaron la posición del brazo con velocidad angular constante al fin de obtener los valores de aceleración experimentales. Finalmente, compararon estos resultados con los valores obtenidos en el método teórico. Se percataron de que sólo existía un grado de error entre ellos, por lo que el resultado del proyecto fue satisfactorio.

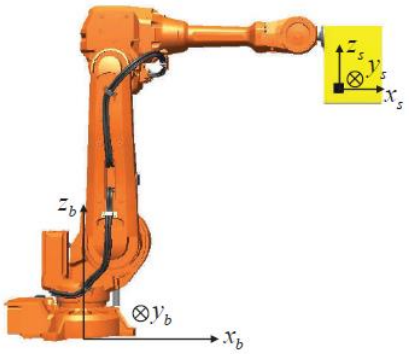

Figura 10. Robot con seis grados de libertad. [16]

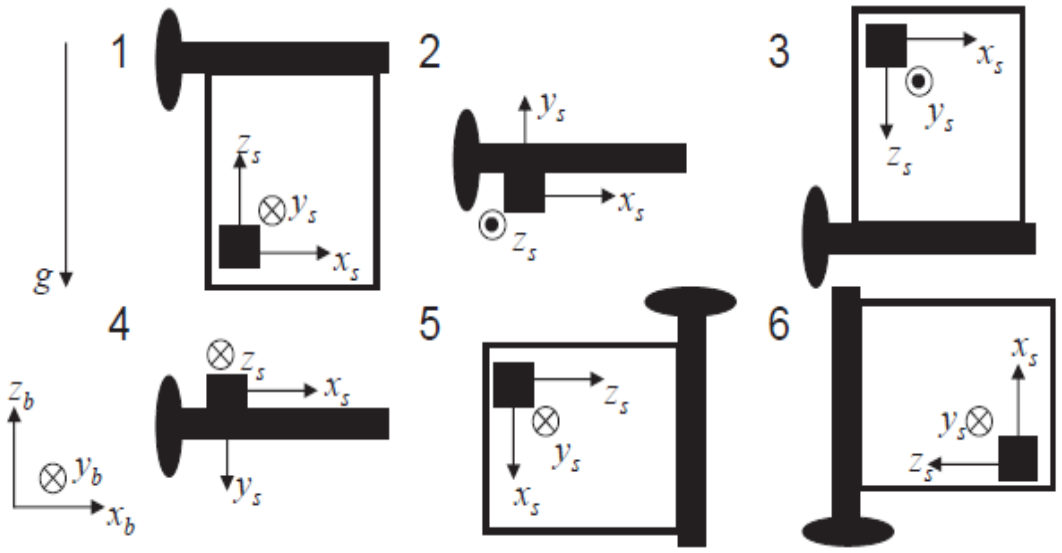

Figura 11. Posiciones del robot para mediciones de la aceleración. [16]

Por lo tanto, puede decirse que este método consigue determinar la posición del robot de una forma precisa. Sin embargo, el hecho de utilizar una velocidad constante es una limitación a la hora de estudiar movimientos más complejos, como son los implicados en este proyecto. Otro inconveniente es que el dispositivo móvil se disloca de tal forma que no es posible referenciar el origen de la posición definida en el propio aparato para la generación del vector posición, cosa que es necesaria para aplicar el método de Axelsson y Norrlöf.

Otros investigadores que se interesaron por la acelerometría fueron Wen-Hong Zhu y Tom Lamarche (Agencia Espacial Canadiense) [17]. Éstos realizaron conjuntamente un trabajo cuyo objetivo era calcular la velocidad de una masa que se movía a lo largo de un único eje. Para ello, comenzaron usando sensores de velocidad, pero se percataron de que los errores de medición en altas frecuencias eran abismales. Intentaron enmendar dicho error aplicando filtros. Lo consiguieron, pero a cambio generaron un nuevo error. Tras aplicarse el filtro, siempre se produce un desfase de -90º, por lo que el tiempo de reacción para realizar un movimiento aumenta. Fue entonces cuando tuvieron la idea sustituir el sensor de velocidad por un acelerómetro, pues la velocidad siempre está desfasada -90º con respecto a la aceleración, así que, usándolo, lo que realmente se estarían obteniendo serían valores de velocidad. Ahora se plantea otro problema, y es que el acelerómetro posee un margen de error muy grande en bajas frecuencias. La solución más acertada fue la de contar con la ayuda de otro sensor, tal como el codificador lineal. Combinando ambos sensores, se terminó por enmendar el error en bajas frecuencias, obteniéndose, finalmente, el valor real de la velocidad.

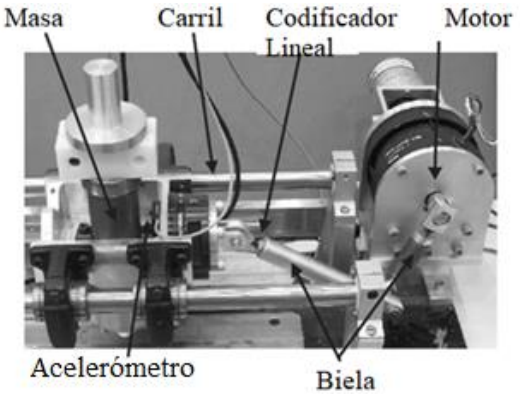

Figura 12. Experimento de acelerómetro con codificador lineal. [17]

Al utilizar el acelerómetro como un codificador, se observó una notable disminución de errores en la posición de los equipamientos industriales. Sin embargo, para ello fue necesario aplicar teoría de filtros, transformadas de Laplace y métodos de control de procesos para relacionar el acelerómetro con el codificador lineal. Esto podrá ser eficiente para un uso industrial, pero a la hora de implementar dichos algoritmos en Android Studio la eficiencia del programa va a disminuir notablemente, ya que éste se hace muchísimo menos robusto. Otro inconveniente es que se tiene que hacer uso de otro sensor aparte del acelerómetro. Realmente es una solución óptima, pero ésta no entra en el contexto de este trabajo fin de grado, pues sólo con leer el título se percibe que está limitado al uso de acelerometría.

Aunque los proyectos descritos hasta ahora presenten estimaciones de movimientos más simples (movimientos con velocidad constante o rectilíneos) y aplicaciones puramente mecánicas, también están siendo desarrolladas aplicaciones de accelerometría para dispositivos móviles. Ejemplo de ello es la aplicación "PhonePoint Pen", desarrollada por varios estudiantes de la Universidad de Duke [1] con el objetivo de mejorar la forma de comunicación entre las personas. Ésta permite al usuario usar el dispositivo móvil a modo de bolígrafo, para así escribir un mensaje corto en dos dimensiones. Los datos provenientes del acelerómetro son exportados a Matlab, donde se integran los valores de la aceleración en función del tiempo, hallándose así los valores de la posición.

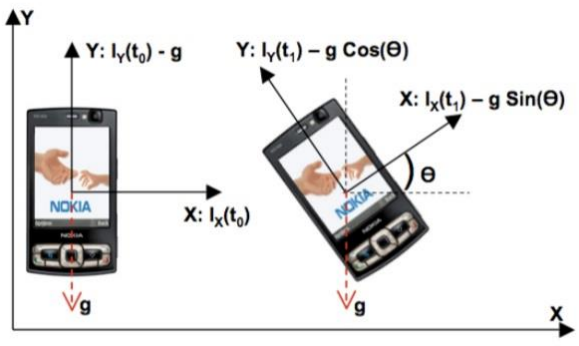

Figura 13. Dispositivo móvil como bolígrafo. [1]

Pero no todo lo que reluce es oro. Dicha aplicación presenta problemas relacionados con el tiempo requerido para escribir una letra, pues necesita de pausas en posiciones específicas para igualar a cero los errores acumulados provenientes del acelerómetro. Cuando son integrados esos fallos, aumenta más todavía la imprecisión de los valores de la posición, lo que hace difícil el reconocimiento eficiente de los caracteres. Otro problema que presenta es que no está focalizada en Android Studio, ya que es necesario ejecutar el programa de Matlab para ver los resultados obtenidos. Por lo tanto, no es viable que un usuario sin conocimientos de ingeniería descargue dicha aplicación y le dé un uso real.

2.2. Propuesta

El uso del acelerómetro ha presentado resultados positivos para la determinación de la posición, aunque éste genere errores naturales en algunas aplicaciones. Como se mostró en el proyecto “PhonePoint Pen”, existen problemas tanto en la toma como en el manejo de datos. De esta forma, nace la necesidad de traducir los movimientos realizados por un dispositivo móvil en caracteres con un tiempo de respuesta corto y eficiente, tanto en la adquisición de datos como en la determinación de los resultados. No interesa mostrar al usuario la forma exacta de la vocal que ha diseñado, si no informarle de si lo ha hecho bien o no. Por ello, no es necesario calcular con exactitud la posición, pero sí encontrar un método matemático que, partiendo de la aceleración, llegue a identificar los tipos de caracteres que se están estudiando. Dichos caracteres poseen movimientos previsibles y con un patrón definido, por lo que el recurso de entrenamiento puede ser obviado, ya que sólo es necesario identificar formas concretas y no un abanico de posibilidades. Además, un abordaje más simple y comprensible de esta tecnología puede traer beneficios a los Ingenieros de Telecomunicaciones, ya que sólo habrían de centrarse en perfeccionar este tipo de aplicación.

CAPÍTULO 3. ANÁLISIS

El propósito de este capítulo es analizar tanto los requisitos para la realización del proyecto como las soluciones existentes para cumplir con el objetivo de éste. Finalmente, se presenta la solución escogida de entre todas las expuestas.

3.1. Requisitos

Cada vez que una persona coge un bolígrafo y escribe una letra en un papel, está haciendo una cosa tan simple como trazar las líneas de una figura específica en una superficie plana. Dichas líneas no son más que sucesiones continuas e indefinidas de puntos. En definitiva, diseñar una letra implica mover un objeto para que éste vaya generando coordenadas en un espacio bidimensional.

Pues bien, lo que se pretende es que el acelerómetro que está integrado en el dispositivo móvil, el cual se moverá implícitamente al moverse el dispositivo, haga de bolígrafo. Dicha herramienta se desplazará en el espacio donde se esté utilizando, provocando así una aceleración en tres dimensiones.

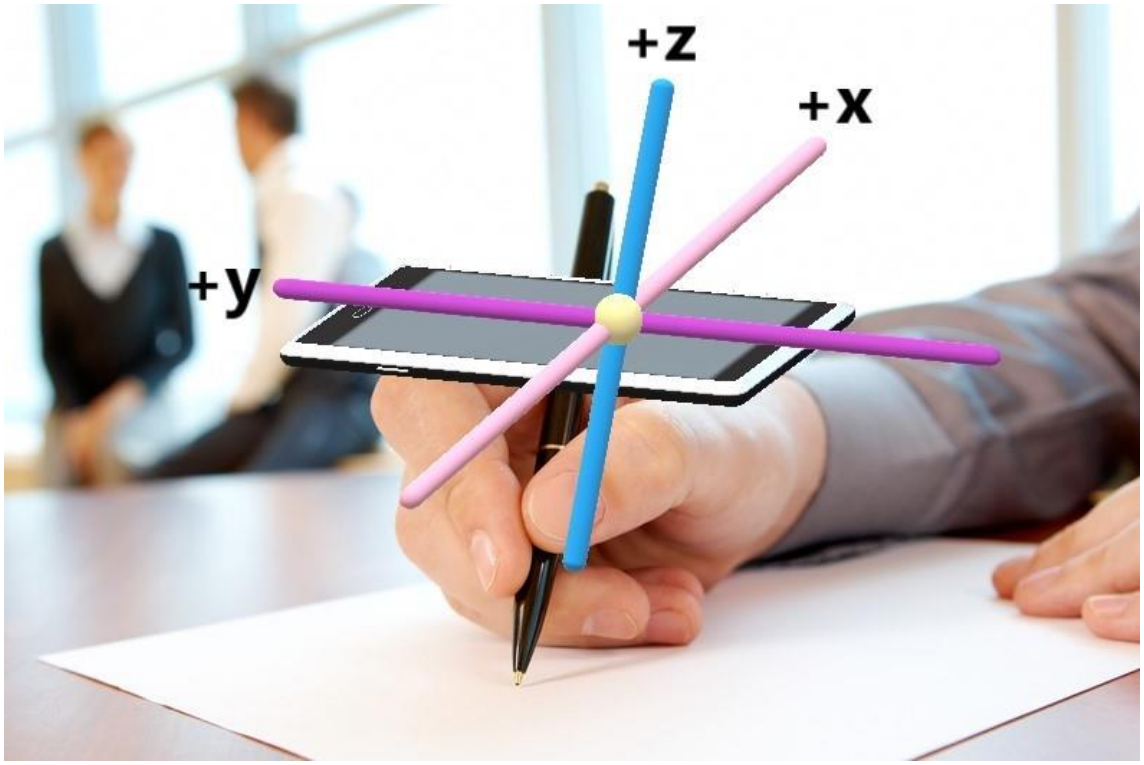

Figura 14. Relación dispositivo-bolígrafo. [18]

Al pretender simular el movimiento de una persona escribiendo en un en una superficie bidimensional como es el papel, la información que interesa está en los ejes “x” e “y”, siendo los valores del eje “z” descartables.

Después de aclarar qué puntos del espacio interesan, es preciso resaltar cuál es la trayectoria que éstos han de seguir para ser identificados como una de las cinco vocales que se pretenden reconocer. Las formas concretas que se quieren identificar son las siguientes:

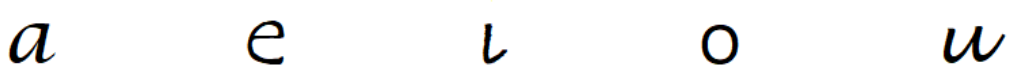

Figura 15. Forma concreta de las vocales objetivo.

Además, la dirección en la que éstas han de ser diseñadas viene dada por las formas más comunes de escritura:

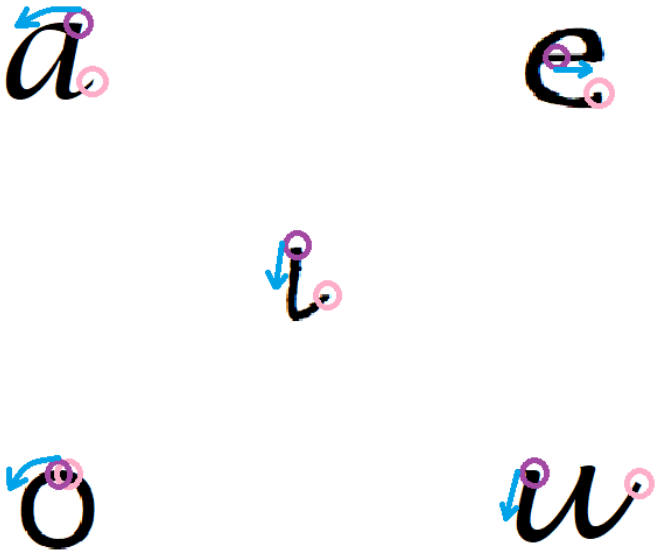

Figura 16. Modo de diseñar las vocales objetivo.

Siendo:

- Punto de origen: O
- Punto final: O
- Dirección: →

Dicho esto, lo único que resta es analizar los métodos matemáticos que puedan llevar a una óptima solución del problema planteado en este proyecto, no sin antes hacer unas aclaraciones acerca de la herramienta que se va a usar para obtener las muestras con las que se van a trabajar. Como ya se ha mencionado anteriormente, el acelerómetro puede devolver los valores de la aceleración metros partido por segundo al cuadrado (m/s^2) o en fuerza-G (g). En este proyecto interesa trabajar con el segundo tipo de medida, así que se deberá utilizar una aplicación que devuelva los valores en fueras g o, en su defecto, hacer la conversión si son tomadas de la otra forma. Al usar el acelerómetro, se obtendrán las medidas de la aceleración en tres ejes en cada instante de tiempo, en función de la frecuencia de muestreo. Como se pretende usar la aplicación con muestras provenientes de cualquier acelerómetro de cualquier sistema operativo, se tiene que escoger una frecuencia de muestreo que pueda ser seleccionada en todos ellos. El tiempo estimado de la colecta de muestras está comprendido entre 4 y 10 segundos, puesto es el tiempo estimado de escribir una letra.

3.2. Soluciones

A partir de aquí, se tendrá que plantear qué hacer con los datos obtenidos tras el uso del acelerómetro para poder conseguir diferenciar cada una de las vocales. Dichos datos son valores de aceleraciones, por lo que la primera solución que se plantea es calcular el módulo de su aceleración instantánea. Como alternativa, se propone hallar la posición del dispositivo a partir de los datos del acelerómetro en un sistema cartesiano de dos dimensiones. Reduciendo el ámbito a un solo eje, el procedimiento para hallar la posición puede ser visto como una doble integración:

$$p(t) = \iint_0^t a(t)dt\,dt$$

Se van a manejar funciones discretas y finitas, por lo que la solución más eficiente es aplicar alguno de los métodos de integración numérica.

A continuación, se describen con más detalle ambas soluciones.

3.2.1. Aceleración instantánea

Se define aceleración instantánea como el límite de la aceleración media cuando el intervalo de tiempo considerado tiende a cero, o lo que es lo mismo, como la derivada de la velocidad respecto al tiempo [19]:

$$\vec{a} = \lim_{\Delta t \to 0} \vec{a}_m = \lim_{\Delta t \to 0} \frac{\Delta \vec{v}}{\Delta t} = \frac{d\vec{v}}{dt}$$

Por lo tanto, el vector de la aceleración instantánea en dos dimensiones coordenadas cartesianas sería:

$$\vec{a} = a_x\vec{\imath} + a_y\vec{\jmath} = \left(\lim_{\Delta t \to 0} \frac{\Delta v_x}{\Delta t}\right)\vec{\imath} + \left(\lim_{\Delta t \to 0} \frac{\Delta v_y}{\Delta t}\right)\vec{\jmath} = \frac{dv_x}{dt}\vec{\imath} + \frac{dv_y}{dt}\vec{\jmath}$$

Cuyo módulo se expresa mediante la siguiente fórmula:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2}$$

Si se aplicase dicho cálculo a, por ejemplo, los valores de la aceleración de cinco muestras, cada una de ellas correspondiente a una de las cinco vocales, se obtendría un resultado tal que así:

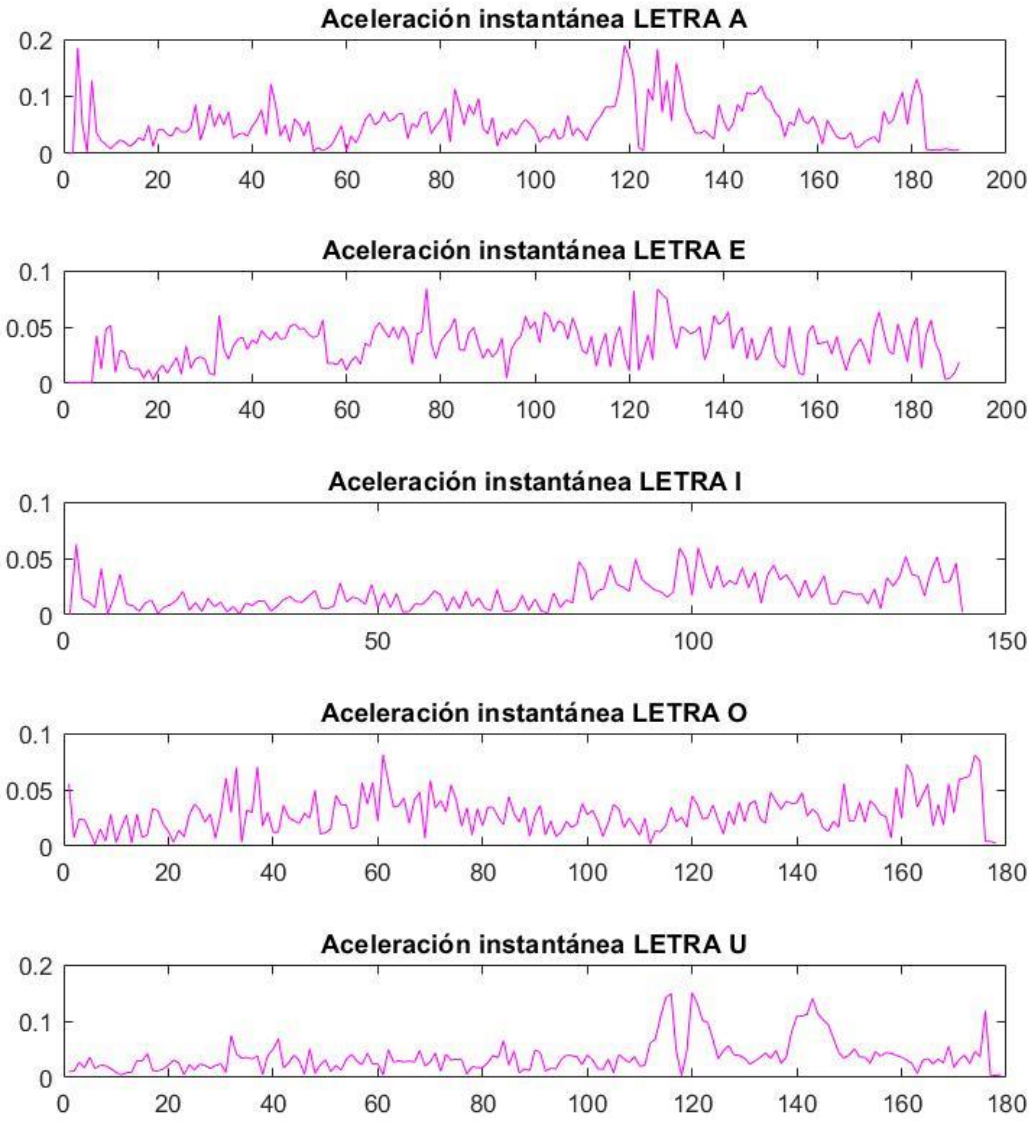

Figura 17. Aceleraciones instantáneas de diferentes vocales.

Tras observar estas gráficas, no es intuitivo pensar en cálculos que consigan diferenciar cada una de estas formas de onda. Sin embargo, se podría pensar en algo más complejo como calcular un número razonable de máximos y mínimos para cada uno de ellos, con la finalidad de llevar dichos datos a las Apps de entrenamiento de los programas a ser utilizados.

3.2.2. Métodos integración numérica

Se aplican los métodos de integración numérica a una función con el objetivo de calcular el valor aproximado de la integral definida.

Es necesario hacer esto cuando:

- La función que se quiere integrar es tan compleja que es imposible aplicar los métodos algebraicos.
- No existe la posibilidad de generar una función que defina el problema correctamente.
- La función que se quiere integrar es discreta

Básicamente, un método de integración numérica consiste en la subdivisión de intervalos dentro del intervalo de integración, utilizando figuras geométricas con áreas conocidas y cuyas sumas se aproximan al resultado real.

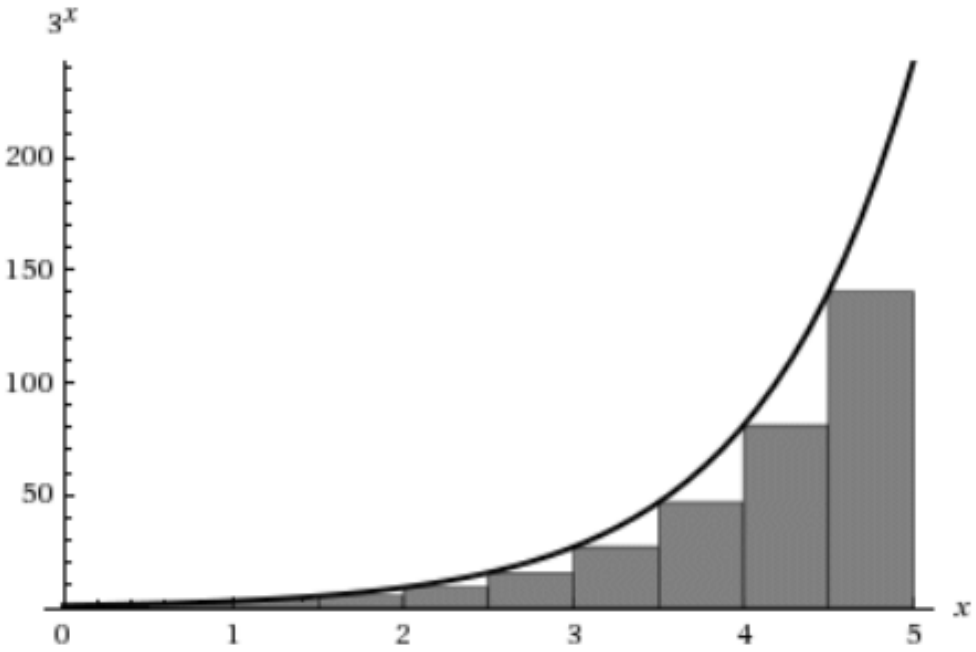

Figura 18. Ejemplo integración numérica.

Como bien se puede observar, existe un error asociado, el cual es directamente proporcional al espaciamiento entre las muestras, o lo que es lo mismo, al número de rectángulos.

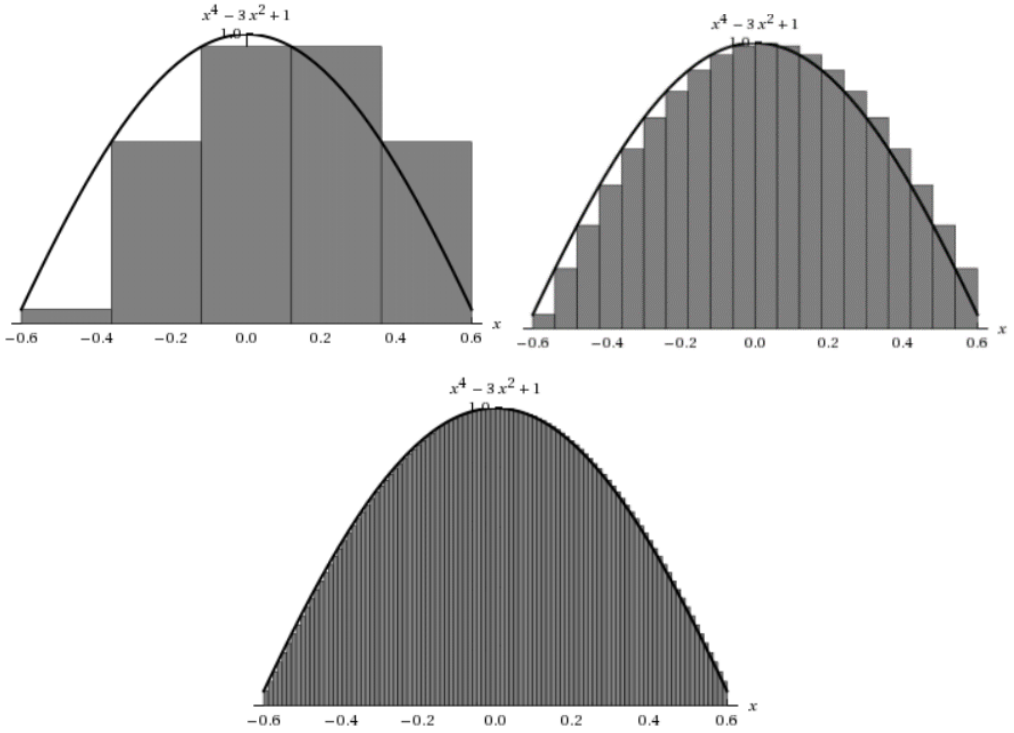

Figura 19. Diferencia del número de rectángulos.

Por lo tanto, cuanto menor es la distancia entre las muestras, más preciso es el resultado. Sin embargo, los algoritmos han de ser aplicados más veces, por lo que el tiempo de ejecución es mayor.

A continuación, se explican los métodos de integración numérica más utilizados.

3.2.2.1. Suma de Riemman

Es el método más simple de todos. Se define por la suma de rectángulos contenidos en un intervalo [a,b] con una base constante de tamaño "h" [20], definida por:

$$h = \frac{b-a}{n}$$

Donde "n" es el número de muestras y "a" es la altura correspondiente al valor instantáneo de la muestra f(x).

Su algoritmo está definido por:

$$I = \int f(x)dx \approx \sum_{k=1}^{n} f(t_k)(x_k - x_{k-1}) = \sum_{k=1}^{n} f(t_k)h$$

Gráficamente:

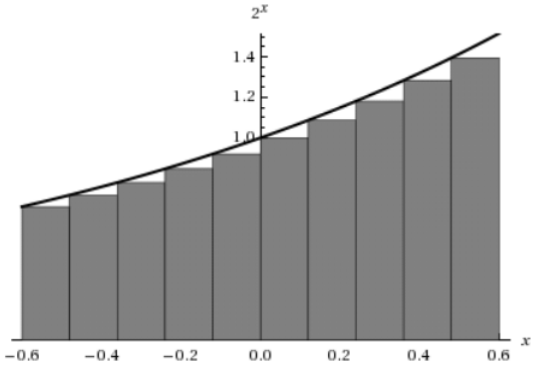

Figura 20. Ejemplo Suma de Riemman.

3.2.2.2. Regla del trapecio

Es una aproximación similar a la anterior, con la diferencia de que se utiliza una recta como polinomio interpolador en lugar de un rectángulo. Para este método, se usa un trapecio con bases correspondientes a los valores de las funciones $f(t_k)$ y $f(t_{k-1})$, y su altura está descrita por el intervalo de muestreo h [21].

Su algoritmo se define por:

$$I = \int_a^b f(x)dx \approx \sum_{k=1}^{n} \frac{(f(t_k) - f(t_{k-1})}{2} (x_k - x_{k-1}) = \sum_{k=1}^{n} \frac{(f(t_k) - f(t_{k-1})}{2} h$$

Gráficamente:

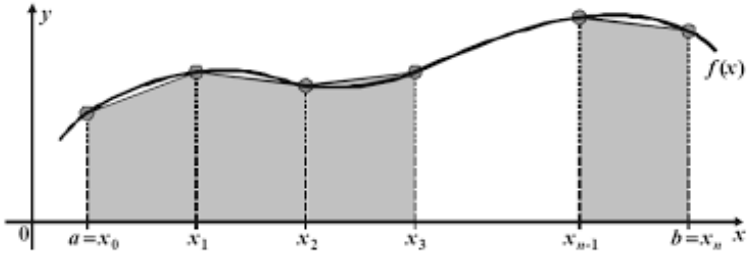

Figura 21. Ejemplo Regla del Trapecio. [22]

3.2.2.3. Regla de Simpson 1/3

Este algoritmo requiere la utilización de tres puntos. Su polinomio interpolador es una función de segundo grado (parábola) [21].

Su algoritmo se define por:

$$I = \int_a^b f(x)dx \approx \sum_{k=1}^{n} \frac{h}{3}[f(t_{k-1}) + 4f(t_k) + f(t_{k+1})]$$

En el caso de la función ser par, la ecuación puede ser reescrita de la siguiente manera:

$$I = \int_a^b f(x)dx \approx \sum_{k=1}^{n} \frac{h}{3}[f(t_{2k-2}) + 4f(t_{2k}) + f(t_{2k+2})]$$

Gráficamente:

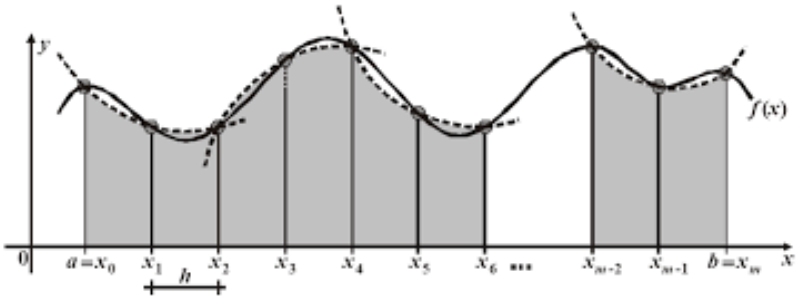

Figura 22. Ejemplo regla 1/3 de Simpson. [22]

3.2.2.4. Cuadratura Gaussiana

Éste se diferencia de los algoritmos anteriores en que el intervalo entre muestras no es constante. Viene motivado por la necesidad de compensar los errores provenientes del espaciamiento constante y de atender las particularidades de cada función. La solución será ideal en el caso de que se utilice un polinomio de grado menor o igual a 2n+1, para n muestras [21].

Su algoritmo se define por:

$$I = \int f(x)dx \approx \sum_{k=1}^{n} f(t_k)w_k$$

Siendo w_k el tamaño variable del intervalo entre muestras.

Gráficamente:

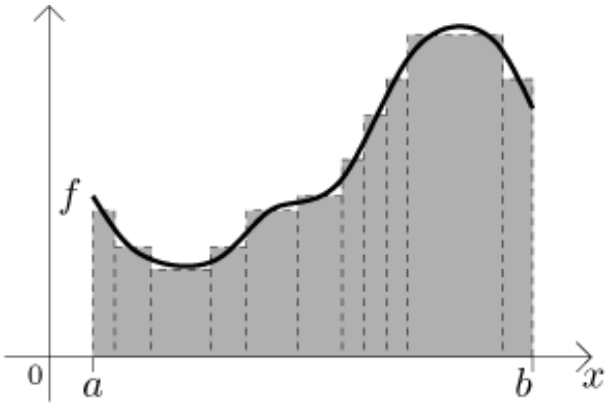

Figura 23. Ejemplo Cuadratura Gaussiana. [23]

3.3. Solución propuesta.

La primera opción que se descartó fue la de trabajar con la aceleración instantánea, ya que escoger ese camino implicaba el uso de Apps de entrenamiento, cosa que sería un acierto si se tuviera como objetivo la detección de vocales en todas las formas y tamaños posibles, pero no es el caso. Como interesa identificar única y exclusivamente las formas geométricas mostradas al inicio de este capítulo, es más eficiente reducir las formas de onda iniciales a algo más básico y, a partir de ahí, diferenciarlas con métodos matemáticos lo más simples posibles, ya que su implementación resultará más robusta. Se escoge, por lo tanto, calcular la posición por medio de integración numérica.

Se pretende integrar dos veces los valores de la aceleración con el objetivo de obtener resultados diferentes para cada tipo de vocal, ya que así se puede extraer características de manera intuitiva y sencilla. Por ello, no es necesario que el resultado sea lo más parecido a los valores de las posiciones reales, así que se va a aplicar el método más simple de todos: suma de Riemman. De nuevo, se ha escogido, de entre todas las alternativas que solucionan el problema, la más robusta, pues en ambos casos se gana rapidez en la ejecución sin perderse la calidad de la detección.

CAPÍTULO 4. DISEÑO

En esta parte de la memoria se explica cómo se ha diseñado la solución escogida en el capítulo anterior, no sin antes hacer un breve resumen de las herramientas que se han empleado para ello.

4.1. Herramientas utilizadas

Para resolver el objetivo principal de este proyecto ha sido necesario desarrollar un algoritmo, el cual ha sido implementado, en primer lugar, en Matlab. Después se ha migrado dicho código a Android Studio. Como ha sido imposible disponer de un dispositivo Android para la realización del proyecto, se han tenido que extraer los datos del acelerómetro de un iPhone mediante la aplicación "Accelerometer", para posteriormente exportarlos a Matlab y Android Studio.

4.1.1 Accelerometer

"Accelerometer" es una aplicación del sistema operativo iOS que permite medir la aceleración en los tres ejes. Mide la fuerza g, la cual está basada en la aceleración que produciría la gravedad de la Tierra en un objeto cualquiera.

La aplicación posee las siguientes funcionalidades:

- Grafica en tiempo real los valores de la fuerza g en los tres ejes en función del tiempo.
- Muestra el valor actual de la muestra en cada eje y sus respectivos máximos y mínimos hasta ese momento.
- Se puede seleccionar la frecuencia de muestreo (de 1 a 30 Hz)
- Permite exportar los datos en formato csv, JSON o XML
- Permite despreciar el efecto de la gravedad.
- Dispone de un botón que puede pulsarse tanto cuando se quiera empezar a medir como cuando se quiera parar.

En este proyecto se ha usado la versión 2.1.0. Se han hecho todas las pruebas despreciando el efecto de la gravedad y con una frecuencia de muestreo igual a 30 Hz. El motivo de haber tomado esta decisión es que dicha frecuencia está incluida en todos los rangos de frecuencias posibles de todas las aplicaciones para medir datos del acelerómetro hechas para el sistema operativo Android, por lo que el resultado del software desarrollado para el proyecto será el mismo independientemente de en qué dispositivo se tomen las muestras.

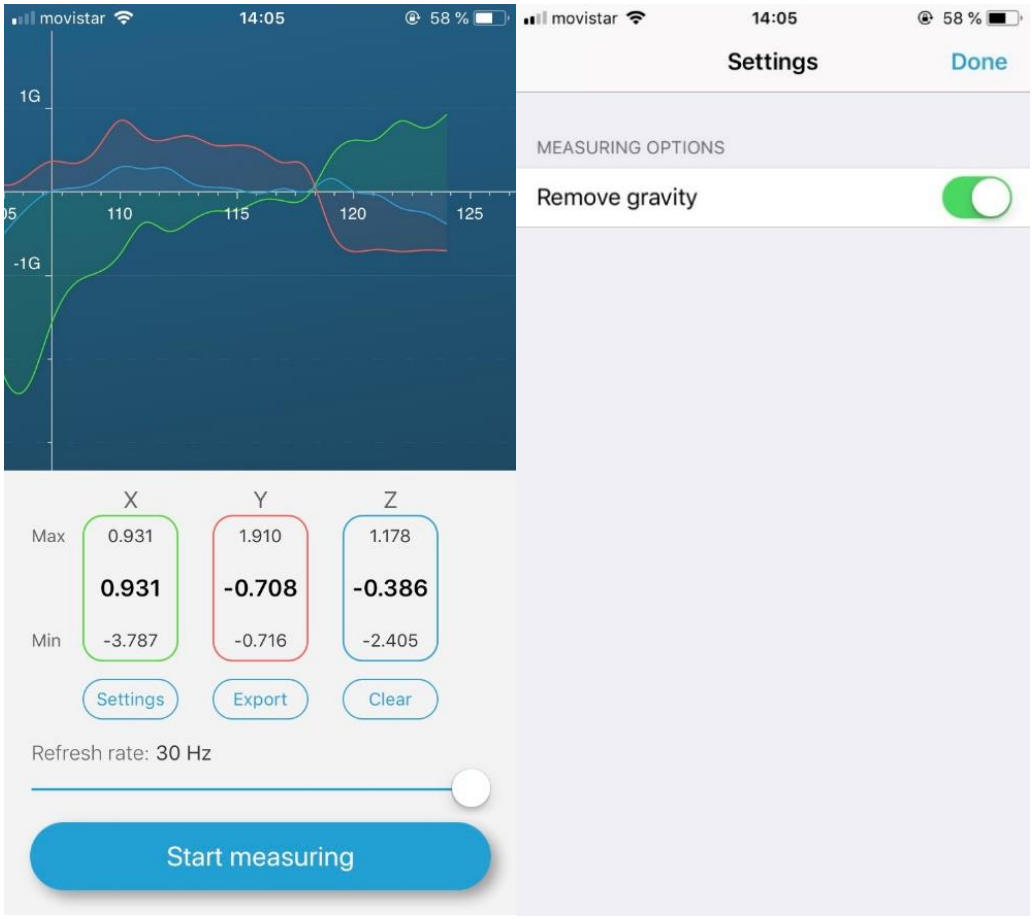

Figura 24. Interfaz gráfica aplicación Accelerometer para cambiar parámetros y opciones.

A la hora de exportar los resultados, éstos siempre han sido mandados como archivos csv. Si se hubiese hecho la aplicación directamente en Adroid Studio, la decisión más acertada hubiera sido exportarlos en formato JSON o XML, ya que es más común trabajar con este tipo de datos en Android y, por lo tanto, existe mucha más información acerca de cómo implementarlo. Sin embargo, había que contar con que primero se tenía que desenvolver el programa en Matlab, en donde es mucho más sencillo leer los datos si éstos tienen un formato csv. En resumen, se tomó esta decisión porque el tipo de datos más óptimo para ser manejado tanto en Matlab como en Android Studio es csv.

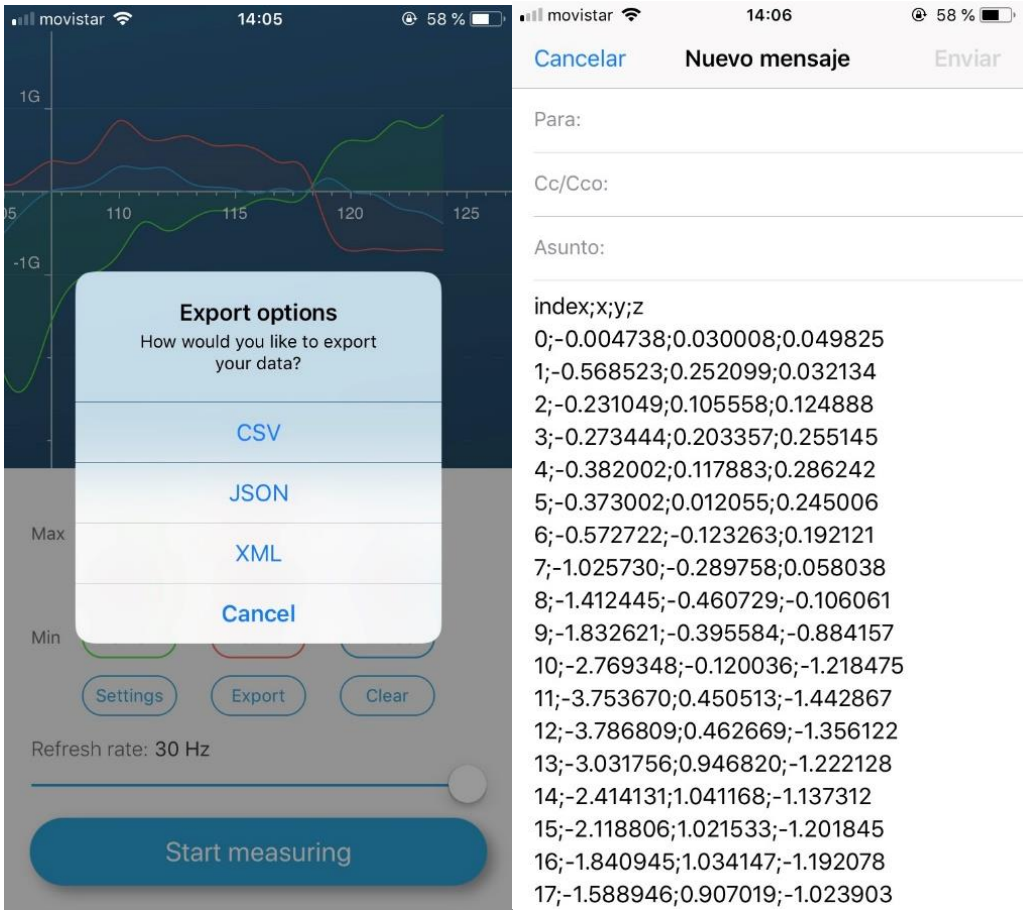

Figura 25. Interfaz gráfica aplicación accelerometer para exportar los archivos.

4.1.2 MatLab

MatLab es un entorno informático de análisis numérico y representación gráfica de fácil manejo. Se puede considerar, por otro lado, que MatLab es una calculadora totalmente equipada, aunque, en realidad, es mucho más versátil que cualquier calculadora para hacer cálculos matemáticos. Se trata de una plataforma para el desarrollo de aplicaciones y para la resolución de problemas en múltiples áreas de aplicación. Entre sus utilidades, se encuentran:

- Cálculo matricial y Algebra lineal [24].
- Polinomios e interpolación.
- Regresión y ajuste de funciones.
- Ecuaciones diferenciales ordinarias.
- Integración.
- Funciones y gráficos en dos y tres dimensiones [25]

Los datos de la versión usada están contenidos en la siguiente imagen:

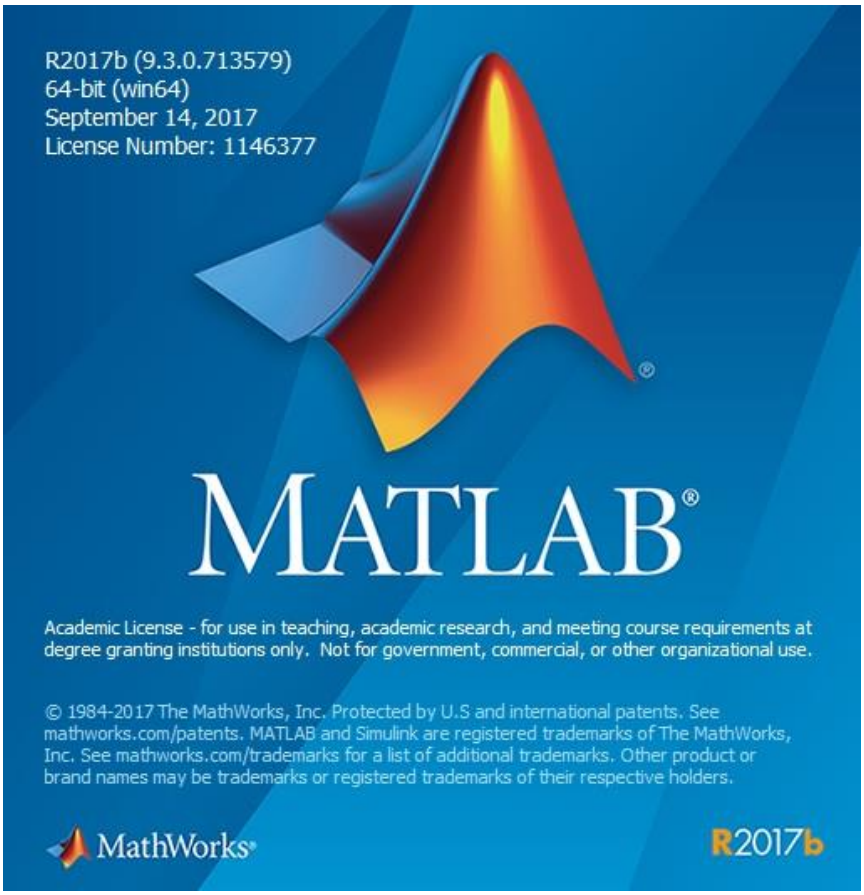

Figura 26. Imagen generada tras abrir el programa Matlab en ordenador personal.

4.1.3. Android Studio

Android Studio [26] es el entorno de desarrollo integrado (IDE) oficial para el desenvolvimiento de aplicaciones para Android, el cual se basa en IntelliJ IDEA.

En primer lugar, se ha de saber que las aplicaciones Android están escritas en el lenguaje de programación orientado a objetos Java. El SDK de Android tiene una serie de herramientas que permiten compilar el código y meterlo en un fichero APK (paquete Android), el cual será el instalador. Cada aplicación será un usuario diferente dentro de Android como sistema operativo basado en un sistema Linux multiusuario.

Una aplicación Android está formada por

- Activities: Representa una pantalla independiente con una interfaz de usuario.

- Services: Es un componente que corre en segundo plano haciendo operaciones de larga duración o trabajos en procesos remotos.

- ❖ Content Provider: Permite gestionar un conjunto de datos de la aplicación para compartir.

- ❖ Brodcast Receiver: Es un componente que permite responder a anuncios broadcast del sistema.

El fichero de manifiesto de la aplicación describe qué componentes forman la aplicación y cómo interactúan, así como metadatos sobre la aplicación (ejemplo: requisitos hardware/software).

Con respecto a la interfaz de usuario (UI), se ha de tener claro los siguientes conceptos a la hora de su diseño:

- ❖ View: Es la clase base de los elementos que forman la UI (controles, widgets), todos los elementos de UI derivan de esta clase.

- ❖ View Goups: Son extensiones de la clase View que contienen múltiples hijos View (controles compuestos formados por varios View anidados)

- ❖ Activities: Representan una pantalla o una ventana de la aplicación, Para construir un UI asignamos un View a una Activity.

- ❖ Fragments: Encapsula fragmentos de una interfaz gráfica.

Cuando se arranca una Activity, la pantalla está vacía, por lo que se va a tener que crear el UI llamando a setContentView en el método onCreate.

Una vez entendido el fundamento de Android Studio, es imprescindible saber cómo son sus proyectos.

Cualquier proyecto en Android Studio va a contener siempre uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen:

- Módulos de Apps para Android.

- Módulos de bibliotecas.

- Módulos de Google App Engine.

Por defecto, se muestran los archivos de dicho proyecto en la vista de proyectos de Android, tal y como se puede ver a continuación:

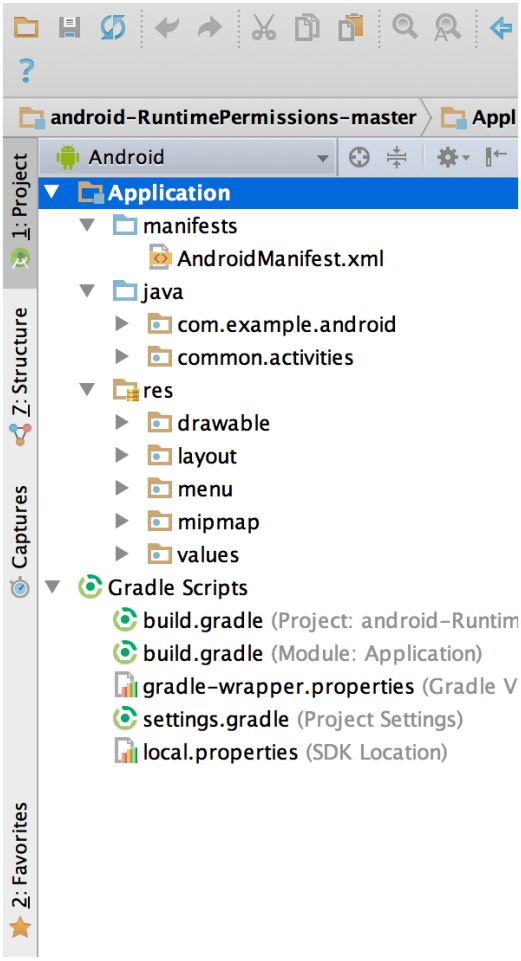

Figura 27. Archivos del proyecto en la vista de Android.

El hecho de que esta vista se organice en módulos proporciona un rápido acceso a los archivos de origen del proyecto.

Todos los archivos de compilación son visibles en el nivel superior de Secuencias de comando Gradle, y cada módulo de la aplicación contiene las siguientes carpetas:

- Manifest: Contiene el archivo AndroidManifest.xml
- Java: Contiene los archivos de código fuente de Java, incluido el código de prueba Junit.
- Res: Contiene todos los recursos, como diseños XML, cadenas de UI e imágenes de mapa de bits.

En este proyecto, se ha desarrollado la aplicación en Android sobre la versión 4.0.3 (IceCreamSandich), nivel de API 15

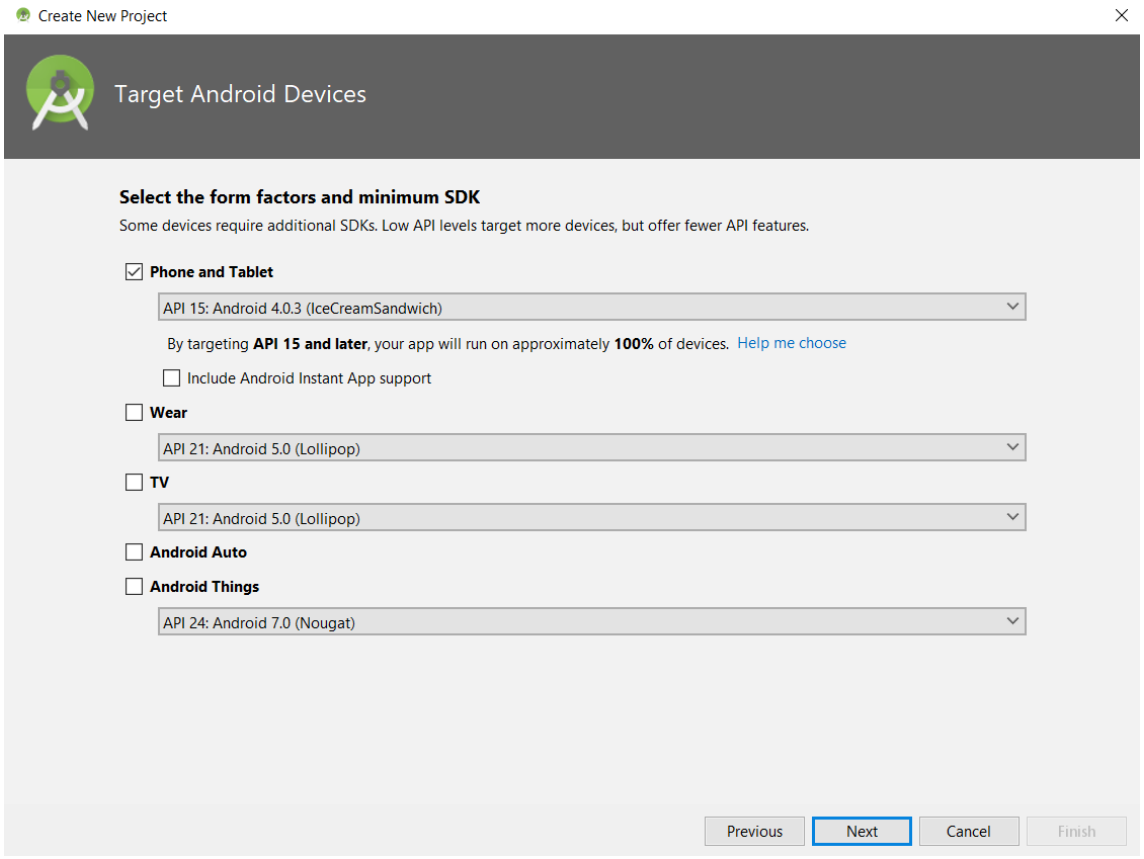

Figura 28. Creación del proyecto en Android Studio.

El emulador ha sido creado con las siguientes características:

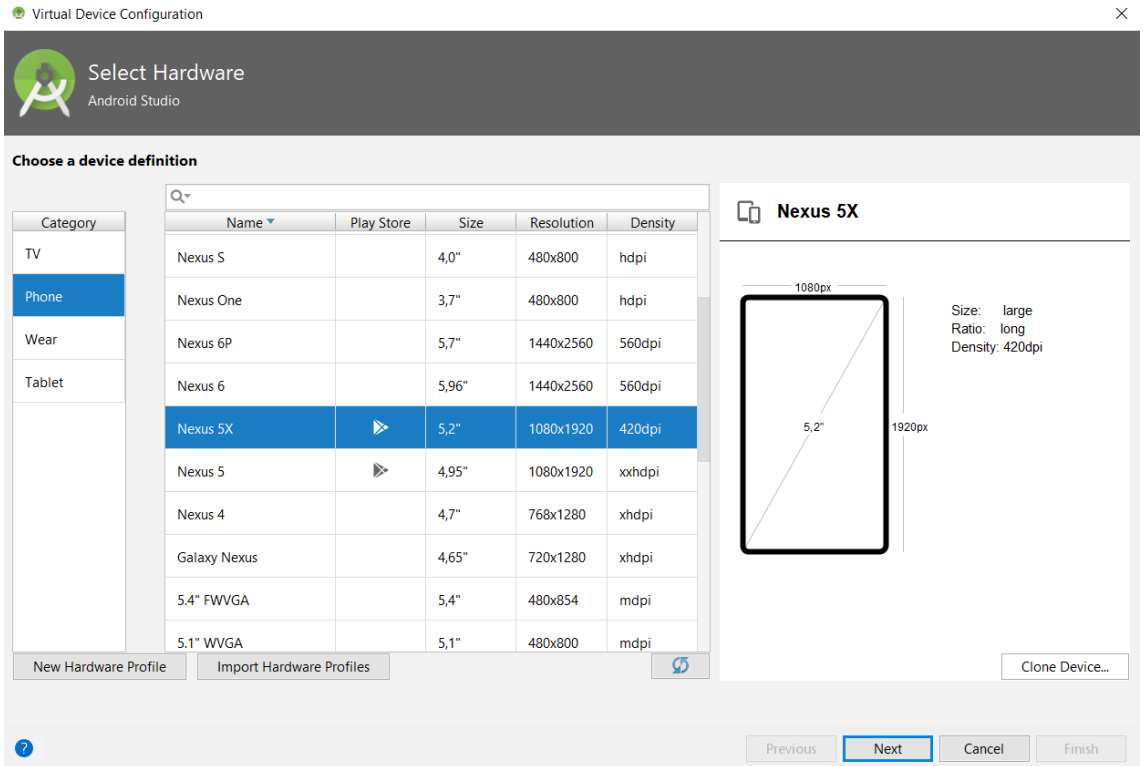

Figura 29. Paso 1 creación emulador usado en este proyecto.

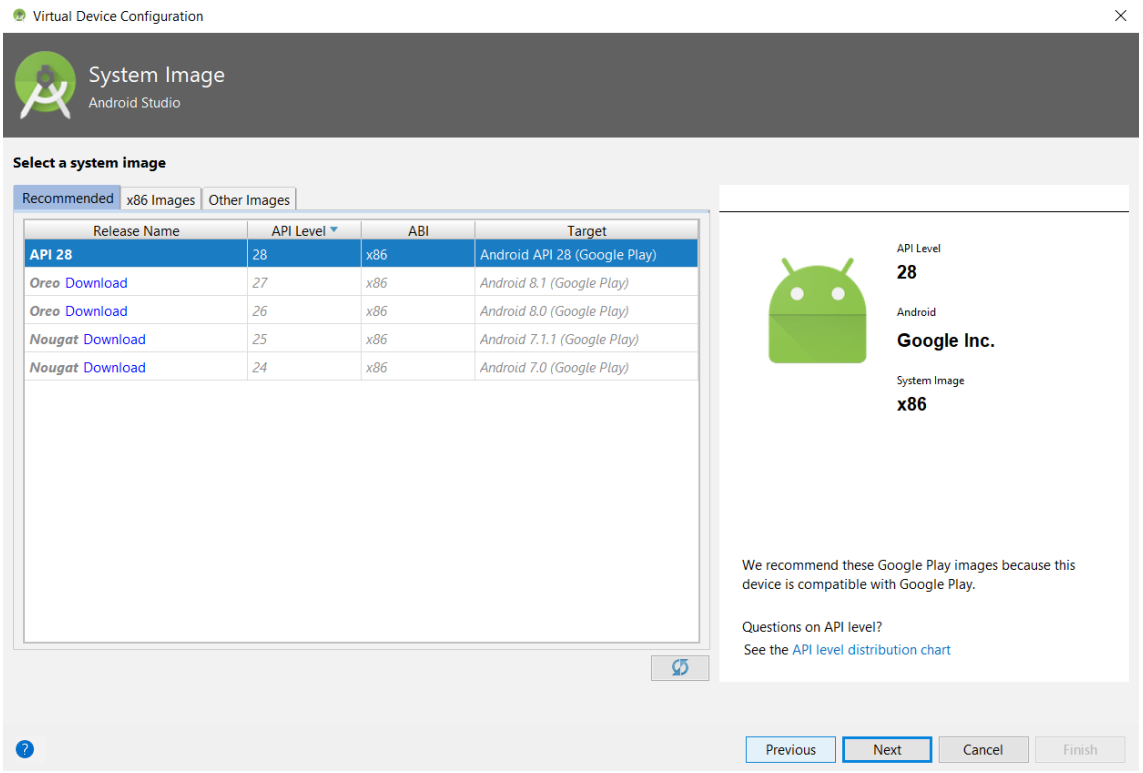

Figura 30. Paso 2 creación emulador usado en este proyecto.

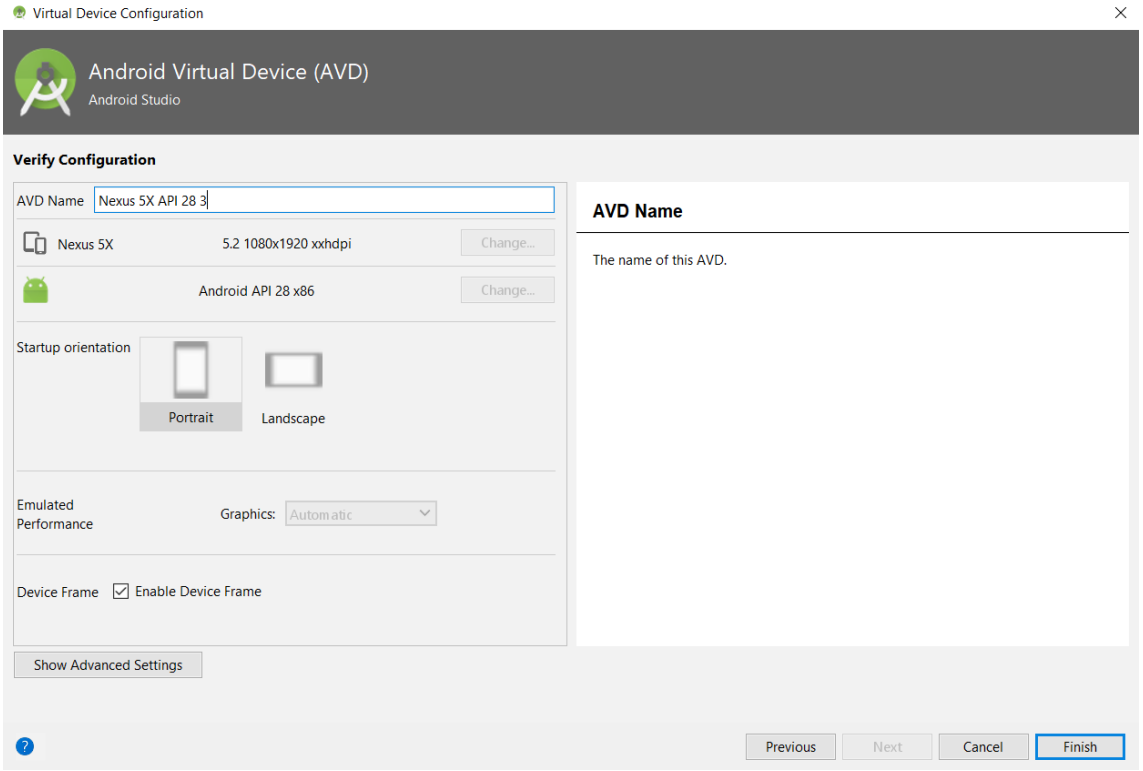

Figura 31. Paso 3 creación emulador usado en este proyecto.

4.2. Diseño de la solución

La idea inicial era diseñar una aplicación sencilla en Android Studio que hiciera lo siguiente:

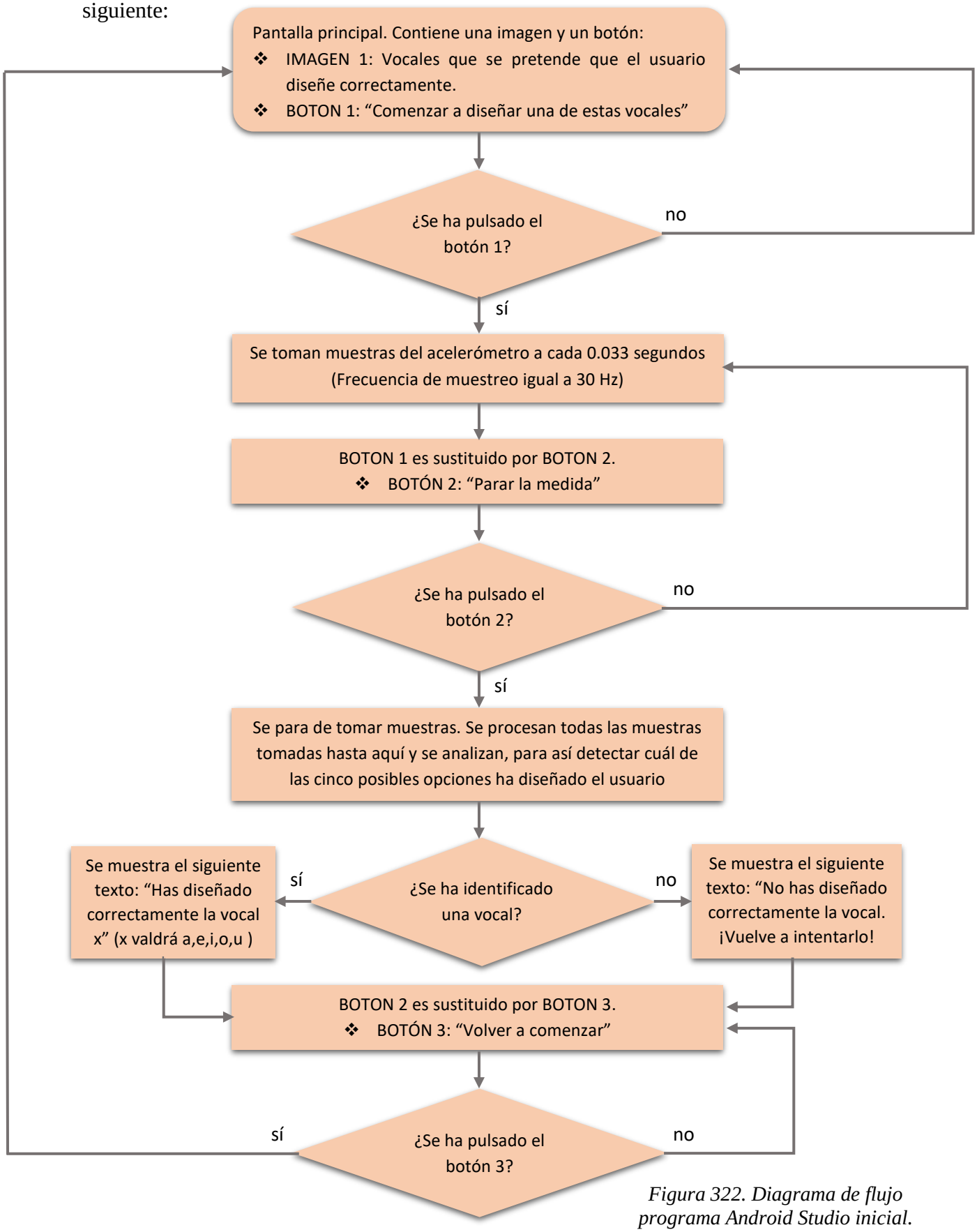

Figura 322. Diagrama de flujo programa Android Studio inicial.

Como bien se ha explicado anteriormente, fue imposible contar con un dispositivo Android para la realización de este proyecto, por lo que se adaptó la solución bajo dichas circunstancias. Lo que cambia es que se lee el archivo que contiene las muestras tomadas con un acelerómetro cualquiera, por lo que no se toman directamente del acelerómetro del dispositivo Android. Este archivo se ha obtenido usando el acelerómetro de un iPhone. Por lo tanto, la solución definitiva que se ha implementado posteriormente en Android Studio ha sido la siguiente:

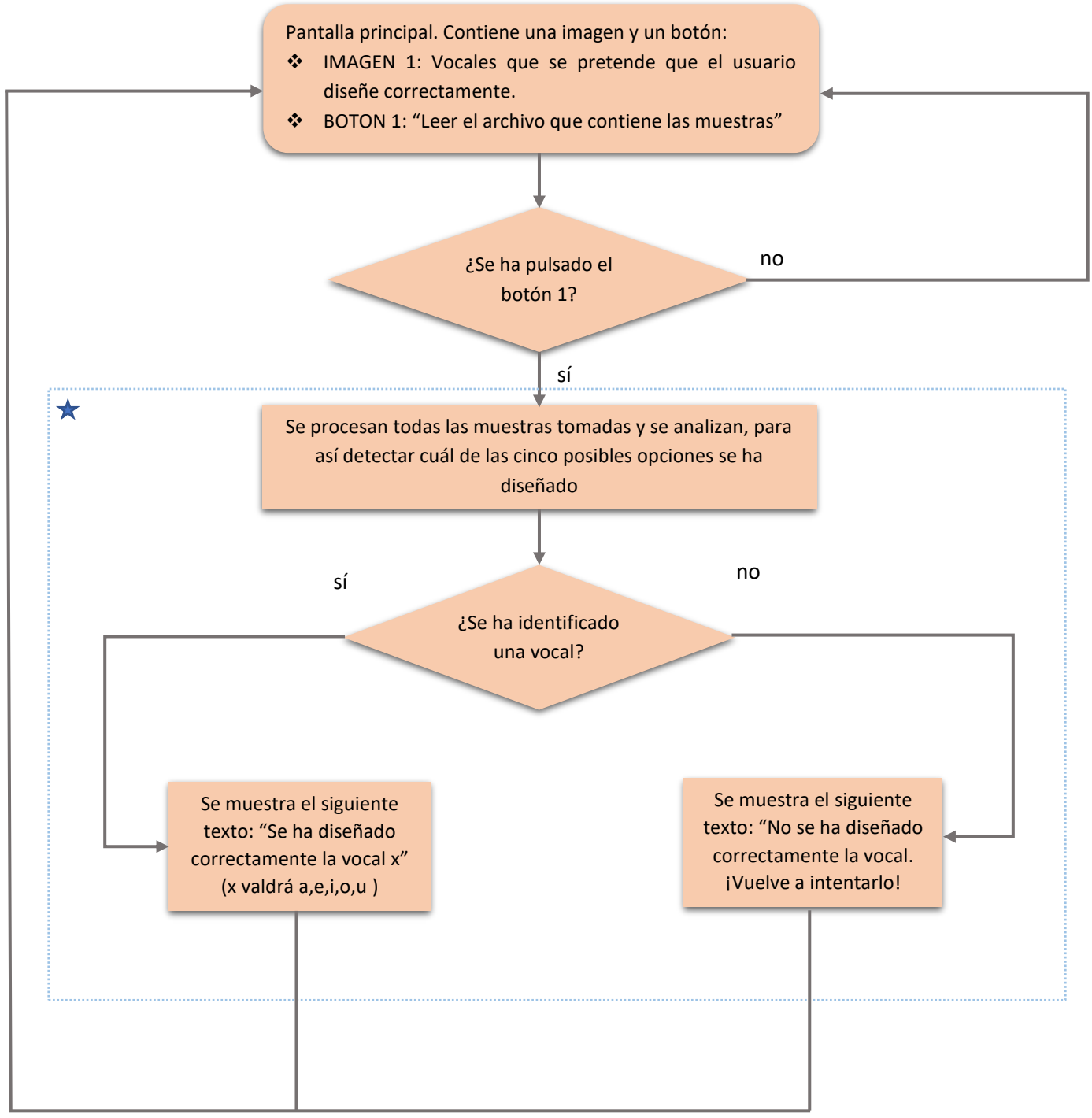

Figura 33. Diagrama de flujo programa Android Studio definitivo.

La parte recuadrada en azul (★) es la parte del algoritmo que se ha de programar en Matlab, para después ser traducido al lenguaje de programación usado en Android Studio. Por lo tanto, se va a hacer un único diagrama de flujo principal por cada parte de la solución, ya que el algoritmo será el mismo para ambos programas. Donde sí se va a esclarecer las diferencias entre uno y otro va a ser en el próximo capítulo, ya que se detallarán las implementaciones de éste. Se procede, pues, a mostrar la solución que se ha aplicado para desarrollar este proyecto.

- Parte 1:

  El siguiente diagrama de flujo representa el cálculo de los valores aproximados de la posición en función de los valores de la aceleración obtenidos con el acelerómetro.

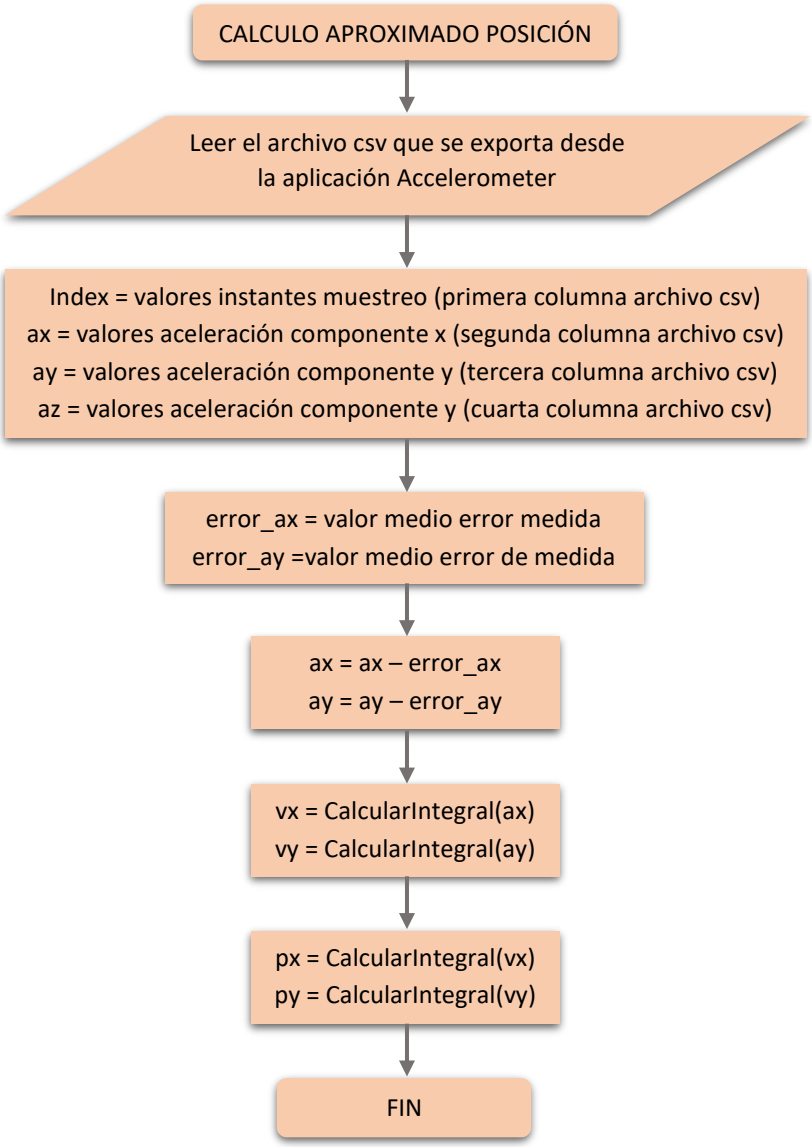

Figura 34. Diagrama de flujo del algoritmo para obtener aproximación posición.

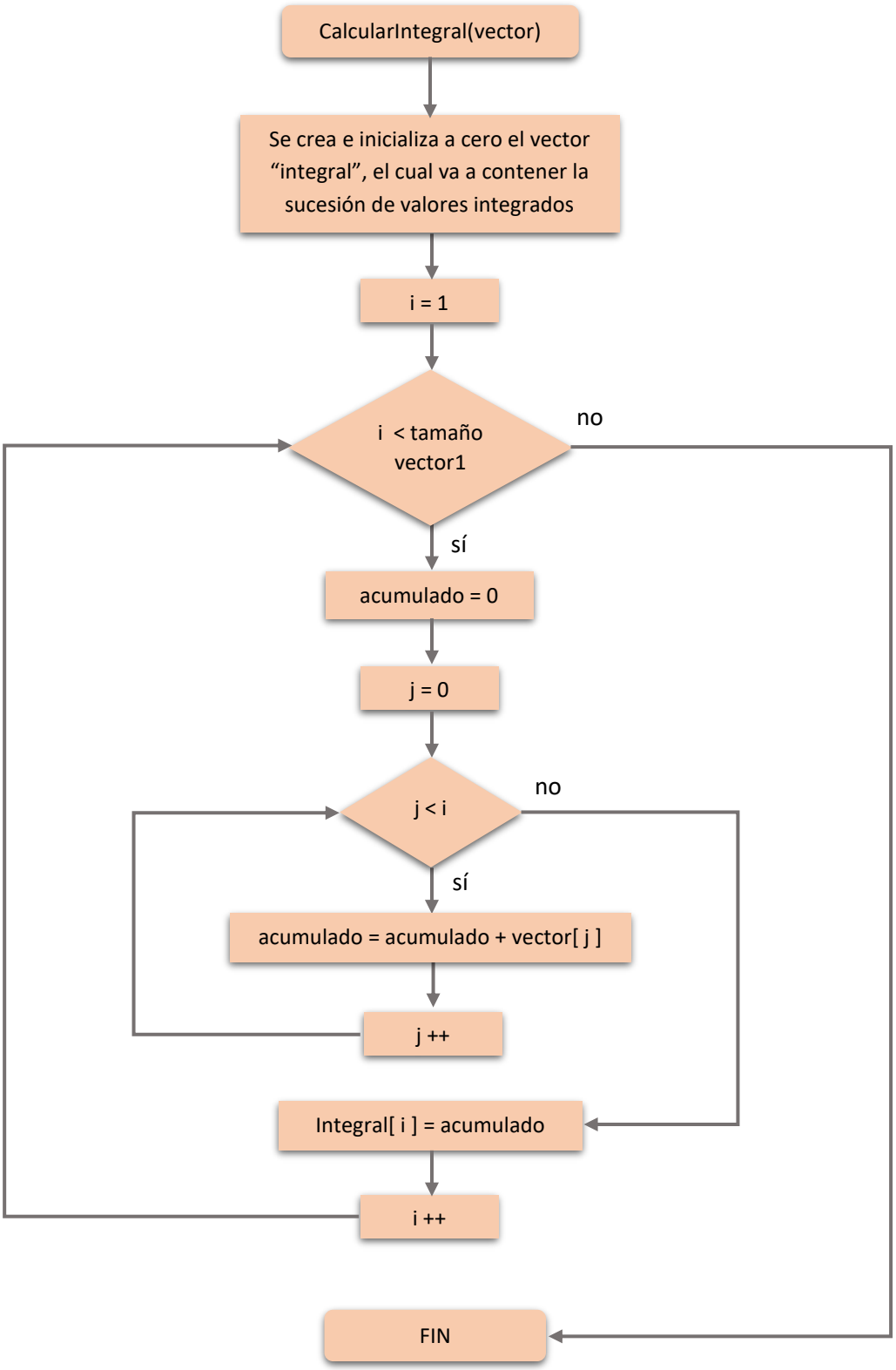

Figura 35. Diagrama de flujo método integración Riemman.

❖ Parte 2:

Si se ejecuta el algoritmo anterior con, por ejemplo, cinco muestras correspondientes a las cinco vocales, se obtiene algo parecido a esto:

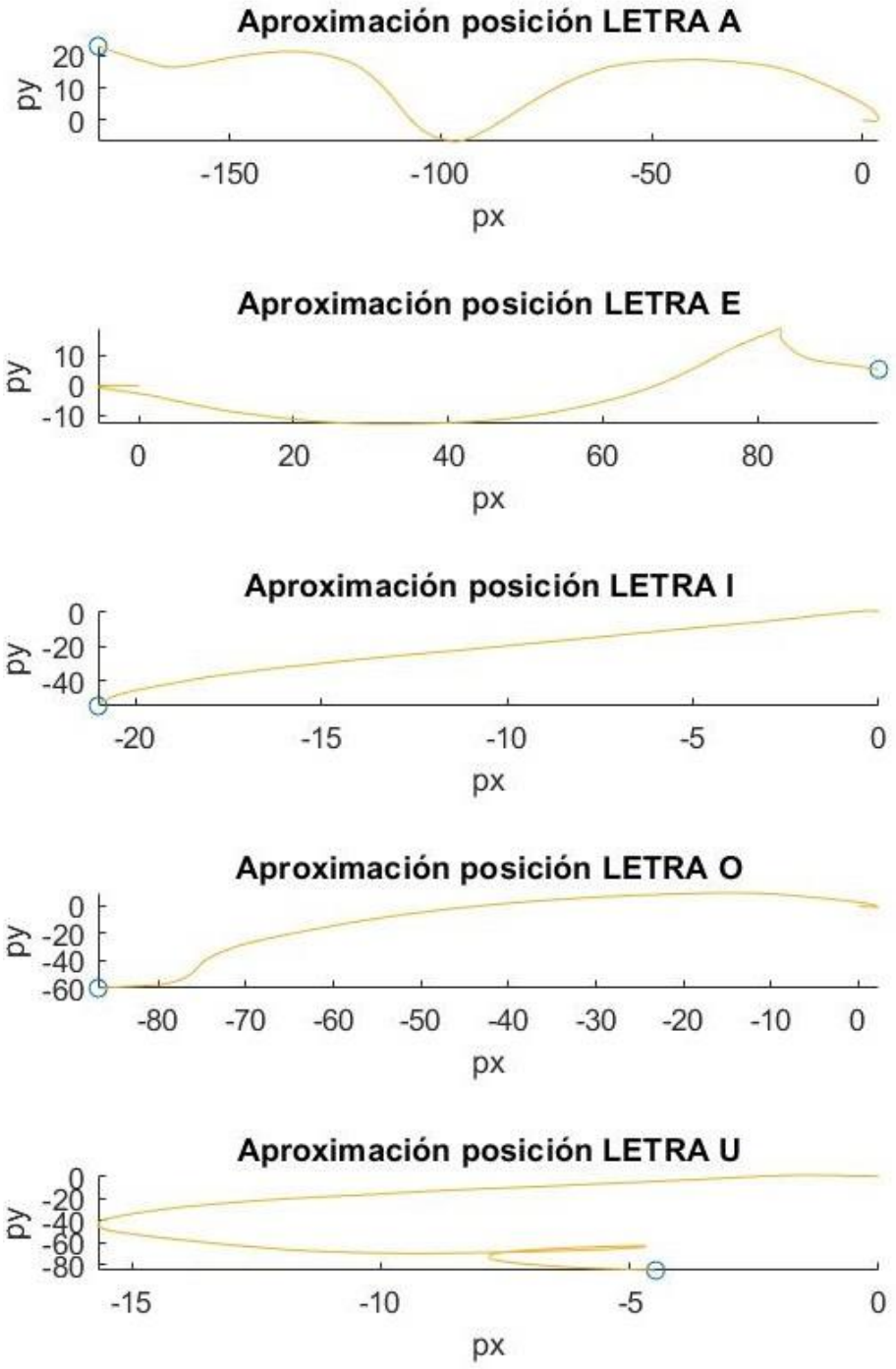

Figura 36. Valores aproximados posiciones de cada una de las vocales.

Tras observar dichas gráficas, se han extraído las características que diferencian una forma de onda de otra para proceder a la identificación de las vocales:

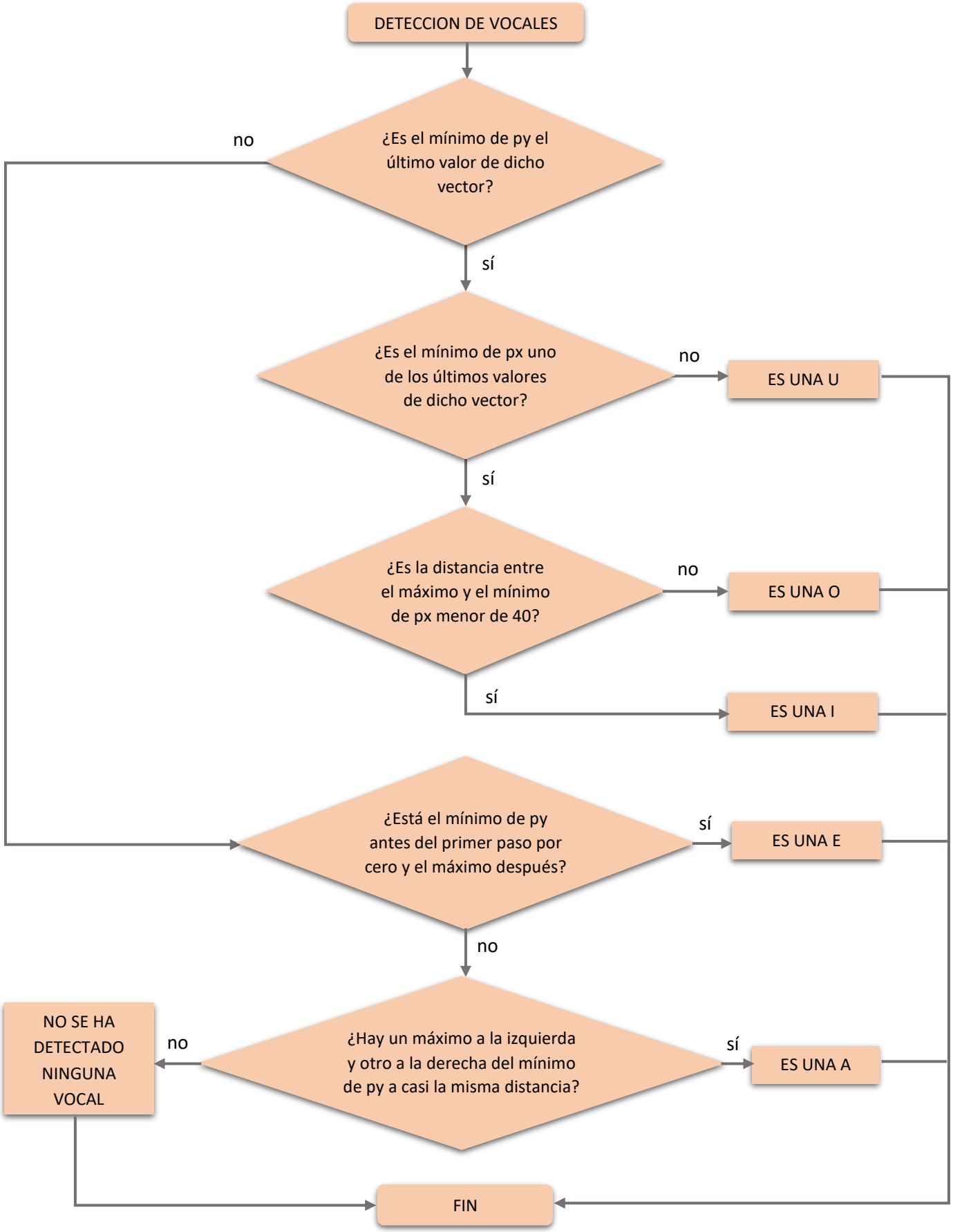

Figura 37. Diagrama de flujo reconocimiento vocales.

CAPÍTULO 5. IMPLEMENTACIÓN

El propósito de esta sección es detallar cómo se ha implementado el algoritmo descrito en el último apartado del capítulo anterior.

5.1. Implementación en Matlab

Se desarrolló primero el código en Matlab porque, como bien se mencionó en el capítulo anterior, éste dispone de herramientas de cálculo muy potentes, haciendo la búsqueda de soluciones más llevadera. El código descrito a continuación es la implementación de los diagramas de flujo de las figuras 34, 35y 37.

Antes de empezar, se han de exportar al directorio local de MatLab los archivos en formato csv obtenidos con la aplicación "Accelerometer".

El programa comienza leyendo dos archivos: el que contiene los valores tomados con el móvil en reposo (archivo_reposo.csv) y el que contiene los valores de la muestra a analizar (muestra_1.csv) [27].

```
archivoReposo = 'archivo_reposo.csv';
archivoMuestra = 'muestra_1.csv';
```

Ahora, para cada archivo, se van a crear 4 vectores: el primero contendrá los valores de "index", el segundo los de "x". el tercero los de "y" y el último los de "z".

```
delimitador = ';';
[i_r, ax_r, ay_r, az_r] =
textread(archivoReposo,'%f%f%f%f','delimiter',delimitador);
[i, ax, ay, az] =
textread(archivoMuestra,'%f%f%f%f','delimiter',delimitador);
```

Como bien se ha mencionado anteriormente, los únicos valores que interesan son los de los ejes "x" e "y", por lo que sólo se van a manipular sus correspondientes vectores.

Dicho esto, se va a calcular la media del vector que contiene los valores cuando el dispositivo móvil está en reposo, para así obtener el valor aproximado del error de medida. A cada uno de los valores de las aceleraciones se le va a restar dicho error.

```
err_ax = median(ax_r);
err_ay = median(ay_r);
ax = ax - err_ax;
ay = ay - err_ay;
```

Con estos últimos vectores, se va a proceder a implementar el método de integración numérica por el que se optó en la fase de análisis. Éste se va a aplicar por separado a los valores de "x" y a los valores "y".

```
%CÁLCULO APROXIMADO DE LA VELOCIDAD (COMPONENTE X)

vx = zeros(length(ax),1);

for i=2:length(ax)
    acumx = 0;
    for j=1:i
        acumx = acumx + ax(j);
    end
    vx(i) = acumx;
end

% CÁLCULO APROXIMADO DE LA VELOCIDAD (COMPONENTE Y)

vy = zeros(length(ay),1);

for i=2:length(ay)
    acumy = 0;
    for j=1:i
        acumy = acumy + ay(j);
    end
    vy(i) = acumy;
end
```

Una vez obtenidos los valores aproximados de la velocidad (*vx*, *vy*), se procede a aplicarles el mismo método para así obtener los valores aproximados de la posición (*px*, *py*)

```
%CÁLCULO APROXIMADO DE LA POSICIÓN (COMPONENTE X)

px = zeros(length(vx),1);

for i=2:length(vx)
    acumx = 0;
    for j=1:i
        acumx = acumx + vx(j);
    end
    px(i) = acumx;
end

%CÁLCULO APROXIMADO DE LA POSICIÓN (COMPONENTE Y)

py = zeros(length(vy),1);

for i=2:length(vy)
    acumy = 0;
    for j=1:i
        acumy = acumy + vy(j);
    end
    py(i) = acumy;
end
```

Aquí acabaría el primer bloque del programa, el cual corresponde al tercero y al cuarto de los diagramas de flujo expuestos en el capítulo anterior. A partir de aquí comienza el reconocimiento de las cinco vocales (quinto diagrama).

Lo primero que se hace es crear un vector de cinco posiciones llamado "vocal", el cual se inicializa a cero. Cada una de las posiciones de dicho vector corresponde a una vocal, siendo dicha correspondencia en orden alfabético. De ese modo, la primera posición del vector corresponde a la vocal "a", mientras que la última corresponde a la vocal "u". Cuando se haya detectado una de las cinco vocales, el valor de la posición correspondiente de "vocal" pasará de ser 0 a ser 1. Por lo tanto, sólo hay dos estados posibles: se ha identificado la vocal, donde vector[i] será igual a uno, o no se ha identificado la vocal, donde vector[i] será igual a cero.

```
vocal = zeros(1,5);

% Si al final del programa vocal[1] == 1, se ha detectado una a
% Si al final del programa vocal[2] == 1, se ha detectado una e
% Si al final del programa vocal[3] == 1, se ha detectado una i
% Si al final del programa vocal[4] == 1, se ha detectado una o
% Si al final del programa vocal[5] == 1, se ha detectado una u
% Si al final del programa todos los valores de este vector son
%igual a cero, no se ha detectado ninguna vocal
```

A continuación, se van a guardar por separado los valores positivos y los negativos, respetando los instantes de tiempo en los que éstos se dieron.

```
valores_negativos = zeros(1,length(py));
valores_positivos = zeros(1,length(py));

for i=2:length(py)
    if(py(i)<0)
        valores_negativos(i) = py(i); %vector valores negativos
        valores_positivos(i) = 0;
    else
        valores_positivos(i) = py(i); %vector valores positivos
        valores_negativos(i) = 0;
    end
end
```

Se calcula el máximo y el mínimo de los vectores de las posiciones tanto del eje x como del el eje y.

```
minimo_py = min(py);
maximo_py = max(py);

maximo_px = max(px);
minimo_px = min(px);
```

Se guarda el índice del vector py en el que se tiene el menor de los valores de las posiciones en el eje y.

```
for i=1:length(py)
     if (minimo_py == py(i))
         indice_minimo_py = i; %posición del vector py en la que está
                               %su valor mínimo
     end
end
```

Como bien se explicó en la parte de desarrollo, una de las primeras características que se puede extraer tras observar las gráficas de la aproximación de las posiciones de cada una de vocales, es que las letras “i”, “o” y “u” se diferencian de “a” y “e” en que el mínimo siempre es el valor de la última muestra, por lo que se va a hacer una primera condición de clasificación. Dentro de ella, se va diferenciar de manera sencilla cada una de estas tres vocales.

```
if(indice_minimo_py == length(py))
%Si el mínimo de py está en la última posición del vector, se trata
%de una de estas tres vocales: i,o,u.


%%%%%%%%%%%%%%%%%%%%%% COMIENZO DEL CÓDIGO IOU %%%%%%%%%%%%%%%%%%%%%%

valor_x_en_minimo_py = px(indice_minimo_py);

if(round(valor_x_en_minimo_py) == round(minimo_px))
%Si el valor de x en el instante de tiempo en el que está el valor
%mínimo de py coincide con el mínimo de px, se trata de una de estas
%dos vocales: i,o.

   distancia_max_min_px = maximo_px - minimo_px;
   %Se calcula la distancia entre el máximo y el mínimo de px

   if (distancia_max_min_px < 40)

       vocal(3) = 1; %SE HA IDENTIFICADO LA VOCAL I

   else

       vocal(4) = 1; %SE HA IDENTIFICADO LA VOCAL O

   end
   %El rango de valores del eje x cuando se diseña la letra o es,
   %como mínimo, el doble que cuando se diseña la i, y su tamaño es
   %siempre superior a 40.

else

        vocal(5) = 1; %SE HA IDENTIFICADO LA VOCAL U

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%% FIN DEL CÓDIGO IOU %%%%%%%%%%%%%%%%%%%%%%%%


else
%Si el índice correspondiente al valor mínimo de py NO coincide con
%el último índice de py, se trata de una de estas dos vocales: a,e.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aquí va el código AE (el cual se va a mostrar a continuación) %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


end
```

En la última parte del anterior fragmento de código se ha indicado que cuando el mínimo de py no sea el valor de la última muestra se trataría de una "a" o una "e", en cuyo caso habría que diferenciar ambas. Esto se hace a continuación:

```
if(indice_minimo_py == length(py))
%Si el mínimo de py está en la última posición del vector, se trata
%de una de estas tres vocales: i,o,u.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% Aquí va el código IOU %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


else

%Si el índice correspondiente al valor mínimo de py NO coincide con
%el último índice de py, se trata de una de estas dos vocales: a,e.


%%%%%%%%%%%%%%%%%%%%%%%% COMIENZO DEL CÓDIGO AE %%%%%%%%%%%%%%%%%%%%%%%%


%PRIMERA PARTE: DIFERENCIAR LETRA E

py_enteros = round(py);
%Se guardan en py_enteros los valores de py redondeados al entero más
%cercano.

paso_por_cero = 0;
%Se crea una variable que va a incrementarse cada vez que la función
%de los valores de la posición en y pase por cero.

%Se va a recorrer el vector de los valores enteros de py en busca de
%las veces que dicha función pasa por cero.
for i=2:length(py_enteros)
```

```
    if(py_enteros(i) == 0 && py_enteros(i-1) ~= 0)
    %Si el valor actual es cero y el anterior es diferente de cero
    %significa que hay un paso por cero.

        paso_por_cero = paso_por_cero + 1;

        if(paso_por_cero == 1)
        %Si estamos ante el primer paso por cero, se va a guardar
        %el valor del instante de tiempo en el que éste se produce.
            instante_primer_paso_por_cero = i;
        end

    end

end

intervalo_1 = py(1:instante_primer_paso_por_cero);
%Se guardan en intervalo_1 los valores de py comprendidos entre el
%comienzo de dicho vector y el instante en el que se produce el
%primer paso por cero.

intervalo_2 = py(instante_primer_paso_por_cero:length(py));
%Se guardan en intervalo_2 los valores positivos de py comprendidos
%entre el instante en el que se produce el primer paso por cero y el
%final de éste.

minimo_intervalo_1 = min(intervalo_1);
%Se calcula el valor mínimo de intervalo_1

maximo_intervalo_2 = max(intervalo_2);
%Se calcula el valor máximo de intervalo_2

if (minimo_intervalo_1 == minimo_py && maximo_intervalo_2 ==
maximo_py)
%Si el valor mínimo de py está en el intervalo_1 y el valor máximo de
%py está en el intervalo_2, se trata de la vocal e.

    vocal(2) = 1; %SE IDENTIFICA LA VOCAL E

end


%SEGUNDA PARTE: DIFERENCIAR LETRA A

intervalo1_positivos = valores_positivos(1:indice_minimo_py);
%Se guardan en intervalo1_positivos los valores positivos de py
%comprendidos entre el comienzo de dicho vector y el instante en el
%que se encuentra su valor mínimo.

intervalo2_positivos =
valores_positivos(indice_minimo_py:length(py));
%Se guardan en intervalo2_positivos los valores positivos de py
%comprenidos entre el instante en el que se encuentra el valor mínimo
%de dicho vector y el final de éste.

inicio = round(length(intervalo1_positivos)*(1/4));
%se define dónde va a comenzar el intervalo que se va a definir a
%continuación (intervalo1_positivos_acotado)

intervalo1_positivos_acotado =
intervalo1_positivos(inicio:length(intervalo1_positivos));
```

```
fin = round(length(intervalo2_positivos)*(3/4));
%se define dónde va a terminar el intervalo que se va a definir a
%continuación (intervalo2_positivos_acotado)

intervalo2_positivos_acotado = intervalo2_positivos(1:fin);

maximo_valores_positivos_intervalo1 =
max(intervalo1_positivos_acotado);
%Se calcula el valor máximo de py en el intervalo 1 (sin tener en
%cuenta los valores del comienzo, en concreto los comprendidos entre
%0 y ¼ del tamaño del vector intrvalo1_positivos)

maximo_valores_positivos_intervalo2 =
max(intervalo2_positivos_acotado);
%Se calcula el valor máximo de py en el intervalo 2 (sin tener en
%cuenta los valores del final, en concreto los comprendidos entre 3/4
%del tamaño del vector intervalo2_positivos y el final de dicho
%vector)

for i=1:length(intervalo1_positivos)

    if (maximo_valores_positivos_intervalo1 ==
    intervalo1_positivos(i))
           posicion_maximo_valores_positivos_intervalo1 = i;
           %Se guarda la posición en la que se encuentra el máximo de
           %los valores positivos de py del intervalo 1 (sin tener en
           %cuenta los valores del comienzo, en concreto los
           %comprendidos entre 0 y 1/4 del tamaño del vector
           %intrvalo1_positivos)
    end

end


for i=1:length(intervalo2_positivos)

    if (maximo_valores_positivos_intervalo2 ==
    intervalo2_positivos(i))
        posicion_maximo_valores_positivos_intervalo2 =
        indice_minimo_py - 1 + i;
        %Se guarda la posición en la que se encuentra  el máximo de
        %los valores positivos de py del intervalo 2 (sin tener en
        %cuenta los valores del final, en concreto los comprendidos
        %entre 3/4 del tamaño del vector intervalo2_positivos y el
        %final de dicho vector)
    end

end

distancia_max_int_1_min_py = abs(indice_minimo_py -
posicion_maximo_valores_positivos_intervalo1);
%Se calcula la distancia entre la posición donde se encuentra el
%mínimo de py y la posición donde se encuentra el máximo de py del
%intervalo 1 (sin tener en cuenta los valores del comienzo, en
%concreto los comprendidos entre 0 y 1/4 del tamaño del vector
%intrvalo1_positivos)

distancia_max_int_2_min_py =
abs(posicion_maximo_valores_positivos_intervalo2 - indice_minimo_py);
%Se calcula la distancia entre la posición donde se encuentra el
%mínimo de py y la posición donde se encuentra el máximo de py del
```

```
%intervalo 2 (sin tener en cuenta los valores del final, en concreto
%los comprendidos entre 3/4 del tamaño del vector
%intervalo2_positivos y el final de dicho vector)

distancia_maximos = distancia_max_int_1_min_py +
distancia_max_int_2_min_py;
%Se calcula la diferencia entre las dos distancias calculadas
%anteriormente.

valor_pico_pico_1 = maximo_valores_positivos_intervalo1 +
(abs(minimo_py));
%Se calcula la distancia entre el máximo valor de py en el intervalo
%1 acotado y el mínimo de py.

valor_pico_pico_2 = maximo_valores_positivos_intervalo2 +
(abs(minimo_py));
%Se calcula la distancia entre el máximo valor de py en el intervalo
%2 acotado y el mínimo de py

if(valor_pico_pico_1 > 20 && valor_pico_pico_2 > 20  &&
distancia_maximos < 100 && distancia_maximos > 70)

    vocal(1) = 1; %SE HA IDENTIFICADO LA VOCAL A

end
%Si:
%  1)Hay un máximo a la izquierda del mínimo de py (intervalo1) y el
%  valor de pico a pico es mayor de 20
%  2)Hay un máximo a la derecha del mínimo de py (intervalo2) y el
%  valor de pico a pico es mayor de 20
%  3)La distancia entre ambos máximos está acotada entre 70 y 100
%Se trata de la vocal a


%%%%%%%%%%%%%%%%%%%%%%%%% FIN DEL CÓDIGO AE %%%%%%%%%%%%%%%%%%%%%%%%%


end
```

5.2. Implementación en Android Studio

Una vez se ha obtenido el resultado deseado en Matlab, se procede a la traducción del código a Java, ya que es el lenguaje nativo de Android Studio. Tras ello, se completa el código con el fin de diseñar una interfaz gráfica que facilite el uso de la aplicación.

5.2.1. Migración del código de Matlab a Android Studio

Se fue traduciendo línea a línea el código de Matlab, ya que se trata de dos lenguajes de programación completamente diferentes. No se va a detallar todo el código, pues la funcionalidad es la misma que la descrita en el apartado anterior. Lo que sí se va a proceder es a comentar las nuevas aportaciones.

Antes de comenzar, se va a mostrar los archivos del proyecto en la vista de Android, ya que puede ayudar a localizar lo más relevante [28]:

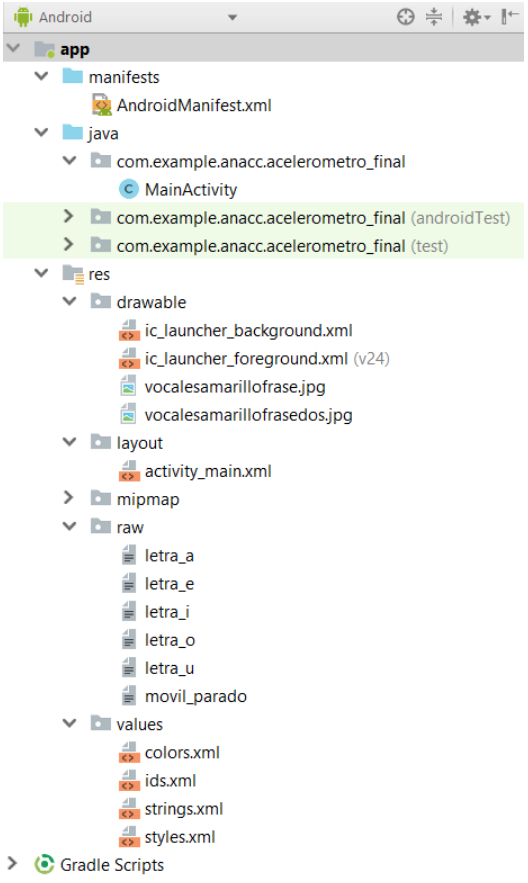

Figura 37. Vista del proyecto creado en Android Studio.

Todo el código equivalente al mostrado en el apartado 5.1 va a implementarse en el método “DeteccionVocales”, el cual ha sido programado dentro del código de la actividad, lugar donde se implementa la lógica de la aplicación.

Para que quede más claro, se muestra a continuación la parte más significativa de "MainActivity.java"

```
//AQUÍ COMIENZA MainActivity.java
package com.example.anacc.acelerometro_final;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    int [] vocal = new int [5];
    float [] vx;
    float [] vy;
    float [] px;
    float [] py;
    Button readFile;
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        readFile=
            (Button)findViewById(R.id.readFile);
        readFile.setOnClickListener(this);
        textView = (TextView)findViewById(R.id.textView);
    }

    @Override
    public void onClick(View view) {
        try {
            DeteccionVocales(view);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void DeteccionVocales(View view) throws IOException {
       /////////////// AQUÍ VA EL CÓDIGO DE DeteccionVocales ////////////
    }

    /*****************************************************************
     ********* AQUÍ VA EL RESTO DEL CÓDIGO DE LA ACTIVIDAD *********
     *****************************************************************/

}//AQUÍ TERMINA MainActivity.java
```

A continuación, se muestra el trozo de dicha función en donde se carga el primer archivo, ya que la forma de parsear datos dista mucho de un programa a otro [15]:

```
public void DeteccionVocales(View view) throws IOException {

    List<String> index = new ArrayList<String>();
    List<String> xaceleration = new ArrayList<String>();
    List<String> yaceleration = new ArrayList<String>();
    String linea;

    InputStream archivo =
            this.getResources().openRawResource(R.raw.letra_a);
    BufferedReader reader =
            new BufferedReader(new InputStreamReader(archivo));

    if (archivo != null) {

        while ((linea = reader.readLine()) != null) {
            index.add(linea.split(";")[0]);
            xaceleration.add(linea.split(";")[1]);
            yaceleration.add(linea.split(";")[2]);
        }

    }
    archivo.close();


    String datos1[] =
            index.toArray(new String[index.size()]);
    String datos2[] =
            xaceleration.toArray(new String[xaceleration.size()]);
    String datos3[] =
            yaceleration.toArray(new String[yaceleration.size()]);


    float[] valores_x = new float[datos2.length];
    float[] valores_y = new float[datos3.length];


    for (int i = 0; i < datos2.length; i++) {
           valores_x[i] = Float.parseFloat(datos2[i]);
    }

    for (int i = 0; i < datos3.length; i++) {
           valores_y[i] = Float.parseFloat(datos3[i]);
    }


   /*****************************************************************
    ******* AQUÍ VA EL RESTO DEL CÓDIGO DE DeteccionVocales *******
    *****************************************************************/

}
```

Como se usa el mismo algoritmo que el implementado en Matlab, va a hacerse el parseo tanto con el archivo que contiene la letra a ser identificada, como con el archivo que contiene los datos del dispositivo parado.

El hecho de que Matlab trabaje con vectores hace que el manejo de éstos sea mucho más fácil que en otros programas, pues dispone de una librería mucho más completa. Al no darse el caso en Android Studio, se han tenido que diseñar algunas funciones para cálculos básicos en la actividad de "MainActivity.java"

- Cálculo del valor mínimo de un vector:

```
public float valorMinimo(float[] vector) {

    float minimo = 0;

    for (int i = 0; i < vector.length; i++){
        if (minimo > vector[i])
            minimo = vector[i];
    }

    return minimo;

}
```

- Cáluclo del valor máximo de un vector:

```
public float valorMaximo(float[] vector) {

    float maximo = 0;

    for (int i = 0; i < vector.length; i++) {
        if (maximo < vector[i])
            maximo = vector[i];
    }

    return maximo;

}
```

- Cálculo del valor medio de un vector:

```
public float valorMedio (float[] vector){

    float promedio = 0;

    for (int i=0; i < vector.length; i++) {
        promedio = promedio + vector[i];
    }

    promedio = promedio/vector.length;

    return promedio;

}
```

- Hacer un vector con los valores enteros:

```
public int [] vectorEnteros (float[] vector) {

    int[] enteros = new int[vector.length];

    for (int i = 0; i < py1.length; i++) {
        enteros[i] = Math.round(py1[i]);
    }

    return enteros;

}
```

- Extraer los valores de una parte acotada del vector:

```
public float[] extraerIntervalo (float[] vector,int inicio, int fin){

    int tamano_intervalo = (fin-inicio)+1;
    float[] intervalo = new float[tamano_intervalo];
    int i=inicio;
    int j=0;

    while(i<=fin) {

        if (i < vector.length) {
            intervalo[j] = vector[i];
        }

        i++;
        j++;

    }

   return intervalo;

}
```

A continuación, se dejarán ver algunas partes del código de la función "DeteccionVocales" donde se ha tenido que hacer uso de estas funciones, para así demostrar su relevancia en el programa.

En primer lugar, se mostrará una de las primeras partes del código, en la cual se calcula el valor medio de los vectores que contienen los valores obtenidos con el dispositivo parado (valores_x_error y valores_y_error), para así restar dichos valores a los de la medida correspondiente a la vocal que se pretende reconocer (en este caso, los correspondientes al archivo "letra_a" que se lee al comienzo de la función):

```
float valor_medio_x_error = valorMedio(valores_x_error);
float valor_medio_y_error = valorMedio(valores_y_error);


for (int i = 0; i < valores_x.length; i++) {
    valores_x[i] = valores_x[i] - valor_medio_x_error;
}

for (int i = 0; i < valores_y.length; i++) {
    valores_y[i] = valores_y[i] - valor_medio_y_error;
}
```

También se utiliza una de esas funciones cuando se pretende calcular el máximo y el mínimo de los vectores de las posiciones tanto del eje x como del eje y.

```
float  minim_py = valorMinimo(py);
float  maxim_py = valorMaximo(py);

float  minim_px = valorMinimo(px);
float  maxim_px = valorMaximo(px);
```

En la primera parte del código correspondiente a la identificación de la vocal “e”, se aplican las funciones restantes:

```
Int [] py_enteros = vectorEnteros(py);
//Se hace el equivalente a round(py) de Matlab

int paso_por_cero = 0;

int tiempo_primer_paso_por_cero = 0;

for (int i = 1; i < py_enteros.length; i++) {

    if (py_enteros[i] == 0) {
        if (py_enteros[i - 1] != 0)
            paso_por_cero = paso_por_cero + 1;
        if (paso_por_cero == 1)
            tiempo_primer_paso_por_cero = i;
    }
}

float [] intervalo_1 =
      extraerIntervalo(py,1,tiempo_primer_paso_por_cero-1);
/*Se guardan en intervalo_1 los valores de py comprendidos entre el
comienzo de dicho vector y el instante en el que se produce el
primer paso por cero*/

float [] intervalo_2 =
      extraerIntervalo(py,tiempo_primer_paso_por_cero,(py.length)-1);
/*Se guardan en intervalo_2 los valores de py comprendidos entre el
el instante en el que se produce el primer paso por cero y el final
de éste*/
```

5.2.2. Aplicación final en Android

Se ha desarrollado la aplicación tal y como se ha descrito en el segundo diagrama de flujo del apartado 4.2 (figura 33). Cuando ésta se ejecuta con el emulador, cuyas características han sido detalladas en el el apartado 4.1.3, se ve lo siguiente:

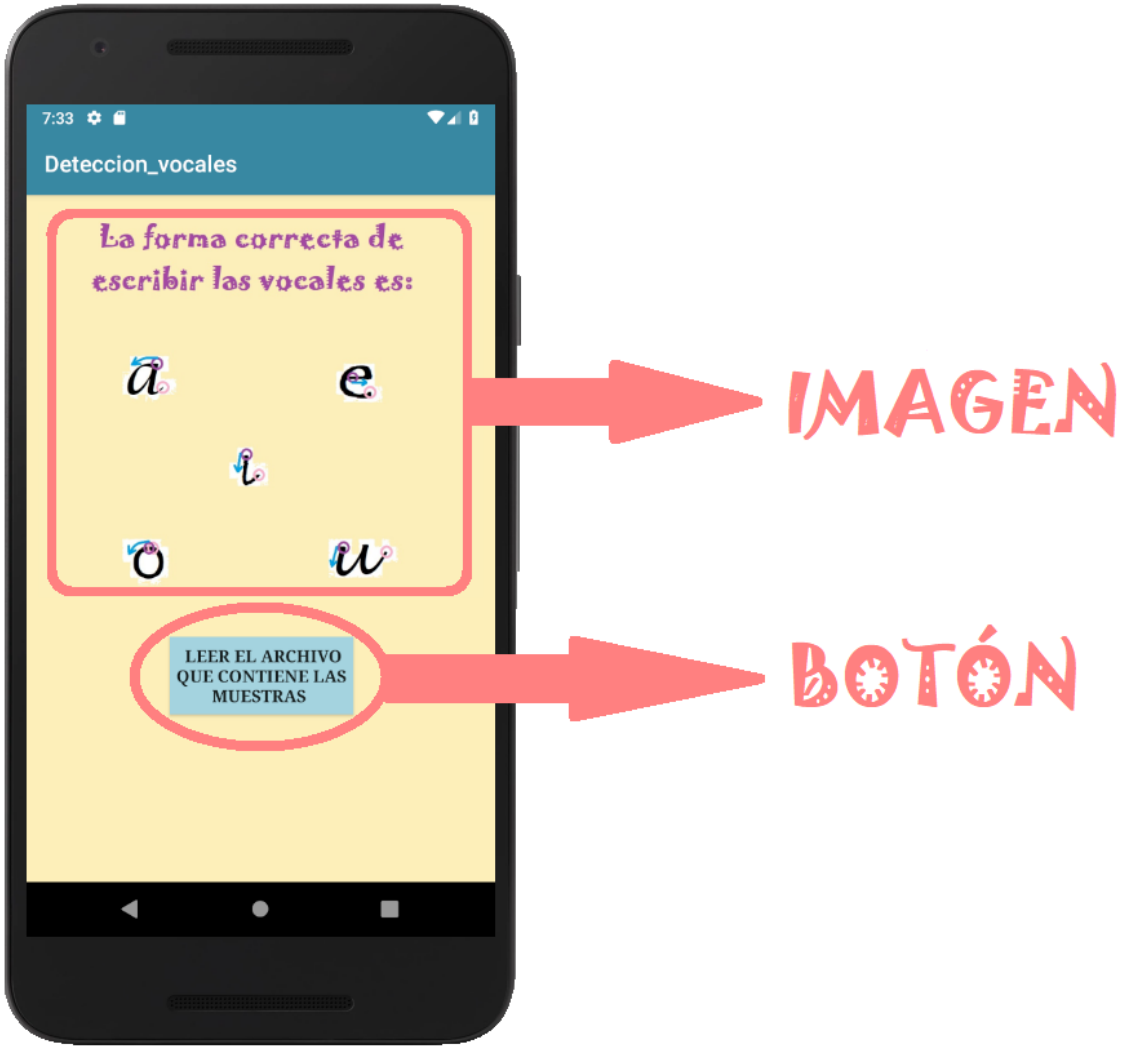

Figura 38. Pantalla principal aplicación.

Se trata de la pantalla principal, que a su vez es la única que hay en la aplicación. Se ha marcado dónde están situadas la imagen y el botón, por si sólo con el print no quedara claro.

Siguiendo con el código descrito en el apartado anterior, se ha ejecutado el programa habiendo leído el archivo “letra_a”, por lo que aparece el siguiente texto tras pulsarse el botón (en realidad no es que el texto aparezca por arte de magia…ya existía un textView desde el arranque de la aplicación, solo que éste estaba vacío hasta el momento de pulsarse el botón, momento en el cual cambia de valor):

Figura 39. Pantalla principal aplicación tras pulsarse el botón.

La aplicación continúa en funcionamiento hasta que se pulse el botón del móvil que retrocede de pantalla, pero esto tiene que ver con el software del sistema operativo Android y no con el software de esta aplicación.

A continuación, se van a mostrar las partes del proyecto Android más relevantes para el funcionamiento de la aplicación:

- **activity_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorAccent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="396dp"
        android:layout_height="337dp"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/vocalesamarillofrasedos" />


    <Button
        android:id="@+id/button"
        android:layout_width="161dp"
        android:layout_height="68dp"
        android:layout_marginEnd="124dp"
        android:layout_marginRight="124dp"
        android:layout_marginTop="388dp"
        android:background="@color/Azul"
        android:onClick="CargaDatos"-
        android:text=" LEER EL ARCHIVO QUE CONTIENE LAS MUESTRAS "
        android:textSize="15sp"
        android:textStyle="bold"
        android:typeface="serif"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="313dp"
        android:layout_height="114dp"
        android:text=" "
        android:textAlignment="center"
        android:textColor="#FFE08AAD"
        android:textSize="25sp"
        android:textStyle="bold"
        android:typeface="serif"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0" />

</android.support.constraint.ConstraintLayout>
```

❖ **strings.xml**

```
<resources>

    <string name="app_name">Deteccion_vocales</string>

</resources>
```

❖ **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.anacc.acelerometro_final">

    <application

        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

    </application>

</manifest>
```

❖ **MainActivity.java**

```
public void DeteccionVocales(View view) throws IOException {


/*****************************************************************
 ******* AQUÍ VA EL PRINCIPIO DEL CÓDIGO DE DeteccionVocales *******
 *****************************************************************/


      String identificador_vocal= "NO SE HA DETECTADO NINGUNA VOCAL,
                                   ¡INTÉNTALO DE NUEVO!";

      if(vocal[0]==1)
          identificador_vocal = "SE HA DISEÑADO CORRECTAMENTE LA
                                 VOCAL A";

      if(vocal[1]==1)
          identificador_vocal = "SE HA DISEÑADO CORRECTAMENTE LA
                                 VOCAL E";

      if (vocal[2]==1)
          identificador_vocal = "SE HA DISEÑADO CORRECTAMENTE LA
                                 VOCAL I";

      if (vocal[3]==1)
          identificador_vocal = "SE HA DISEÑADO CORRECTAMENTE LA
                                 VOCAL O";

      if (vocal[4]==1)
          identificador_vocal = "SE HA DISEÑADO CORRECTAMENTE LA
                                 VOCAL U";


      textView.setText(identificador_vocal);

      textView = (TextView)findViewById(R.id.textView);


}//Fin del método DeteccionVocales
```

CAPÍTULO 6. RESULTADOS

En el presente capítulo se muestran las pruebas que se han realizado para verificar que la solución funciona correctamente. Se van a realizar varias pruebas por cada tipo de vocal, por lo que se va a seccionar el capítulo en función del tipo. Dentro de cada sección, se van a poder observar:

- Datos de las muestras tomadas
- Resultado de aplicarles a dichas muestras una doble integral (usando el método de Riemman)
- Valores de las variables más relevantes de la parte del código correspondiente a la vocal que se está queriendo identificar tras ser éste ejecutado (ya sea en Matlab o en Android Studio)
- Resultado de ejecutar con el emulador la aplicación con los datos de dichas muestras.

Finalmente, se hará un análisis global de todas ellas.

6.1. Pruebas diseñando vocal "a"

❖ Toma de muestras con aplicación "Accelerometer"

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción "Remove gravity" activada	Formato del archivo exportado
Primera prueba	30 Hz	209	sí	csv
Segunda prueba	30 Hz	184	sí	csv
Tercera prueba	30 Hz	208	sí	csv

Tabla 2. Datos toma de muestras vocal a.

❖ Valores aproximados posición ejecutando el código de Matlab

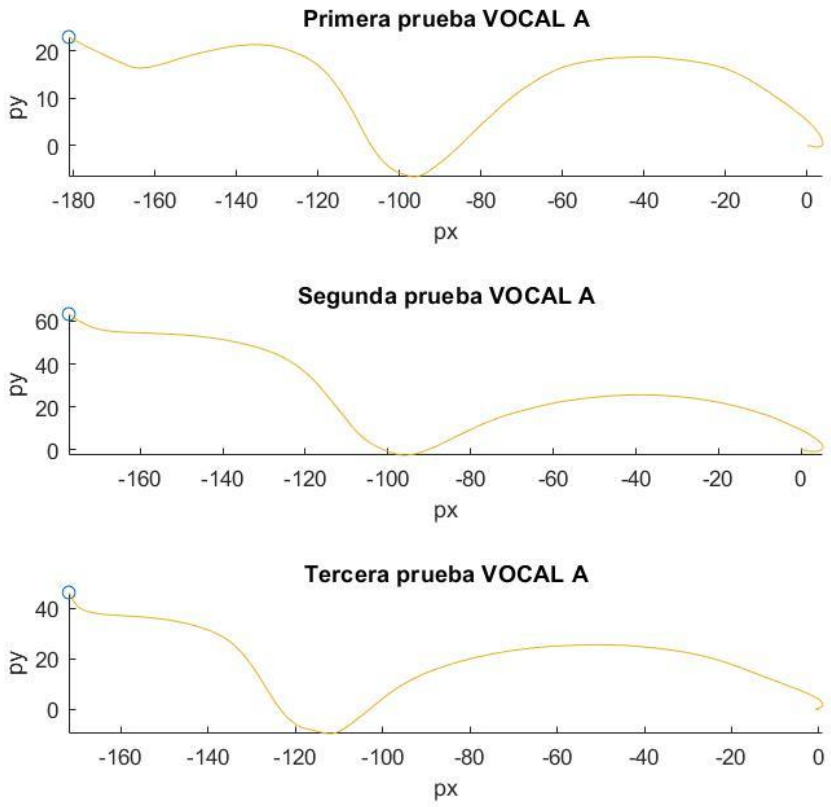

Figura 40. Valores aproximados posición vocal a

❖ Identificación vocal

	Mínimo de py	Valor de la última muestra de py	Valor de pico a pico min. py –> max. int1	Valor de pico a pico min. py –> max. int2	Distancia entre máximos int1 e int2	vocal =
Primera prueba	-6.6059	22.9127	24.0411	27.9194	82	1 0 0 0 0
Segunda prueba	-2.6664	63.3072	28.0858	48.8451	75	1 0 0 0 0
Tercera prueba	-9.5753	46.2373	34.6818	38.3228	85	1 0 0 0 0

Tabla 3. Resultados ejecución código con muestras vocal a

❖ Resultado Android Studio para las tres pruebas

Figura 41. Resultado ejecución aplicación con muestras vocal a.

6.2. Pruebas diseñando vocal “e”

❖ Toma de la muestra

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción “Remove gravity” activada	Formato del archivo exportado
Primera prueba	30 Hz	189	sí	csv
Segunda prueba	30 Hz	190	sí	csv
Tercera prueba	30 Hz	202	sí	csv

Tabla 4. Datos toma de muestras vocal e.

❖ Valores aproximados posición

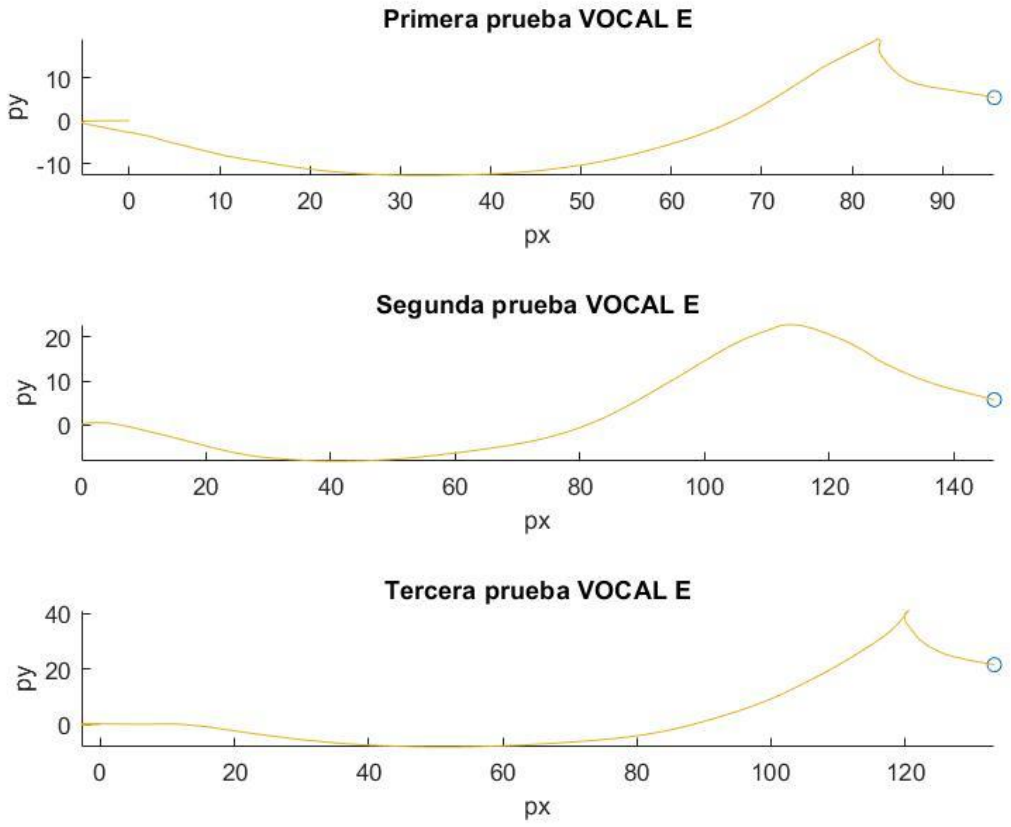

Figura 42. Valores aproximados posición vocal e.

❖ Identificación vocal

	Valor de la última muestra de py	Minimo de py	Minimo de intervalo_1	Máximo de py	Máximo de intetrvalo_2	vocal =
Primera prueba	5.3918	-12.6991	-12.6991	19.0441	19.0441	0 1 0 0 0
Segunda prueba	5.6982	-8.3087	-8.3087	22.8508	22.8508	0 1 0 0 0
Tercera prueba	21.4937	-8.0137	-8.0137	41.0231	41.0231	0 1 0 0 0

Tabla 5. Resultados ejecución código con muestras vocal e.

❖ Resultado Android Studio para las tres pruebas

Figura 43. Resultado ejecución aplicación con muestras vocal e.

6.3. Pruebas diseñando vocal “i”

❖ Toma de la muestra

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción “Remove gravity” activada	Formato del archivo exportado
Primera prueba	30 Hz	145	sí	csv
Segunda prueba	30 Hz	137	sí	csv
Tercera prueba	30 Hz	143	sí	csv

Tabla 6. Datos toma de muestras vocal i.

❖ Valores aproximados posición

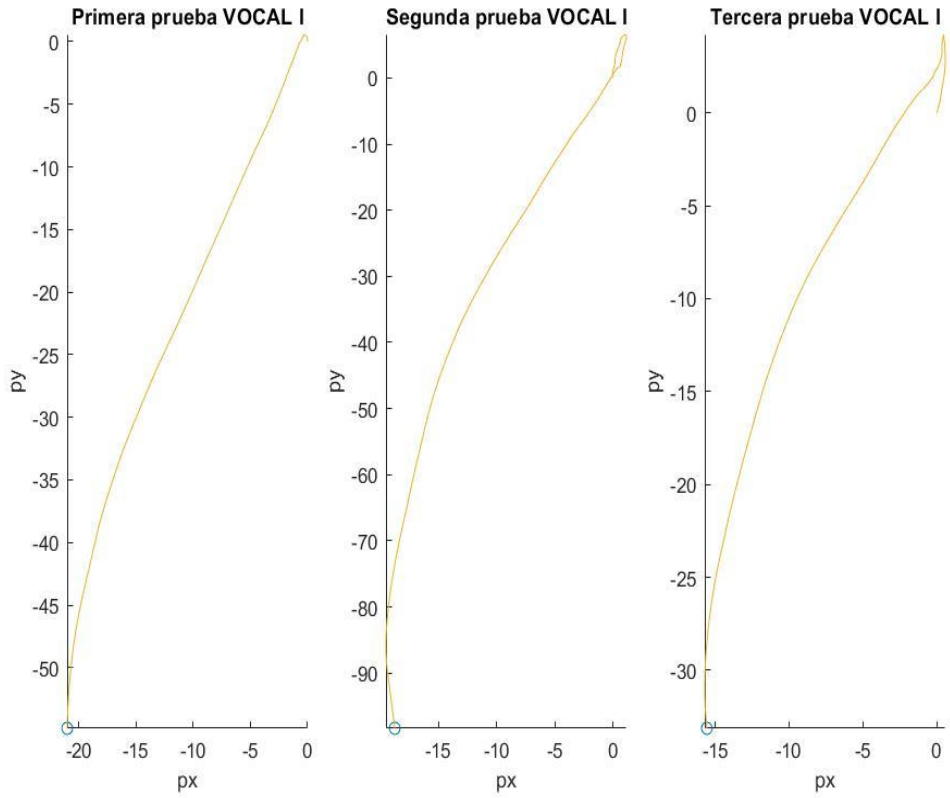

Figura 44. Valores aproximados posición vocal i.

❖ Identificación vocal

	Mínimo de py	Valor de la última muestra de py	Mínimo de px	Valor de la última muestra de py	Distancia entre máximo y minimo de px	vocal =
Primera prueba	-54.8342	-54.8342	-21.0060	-21.0060	21.0131	0 0 1 0 0
Segunda prueba	-98.3471	-98.3471	-19.4945	-18.7430	20.6351	0 0 1 0 0
Tercera prueba	-33.1268	-33.1268	-15.7034	-15.5624	16.2769	0 0 1 0 0

Tabla 7. Resultados ejecución código con muestras vocal i.

❖ Resultado Android Studio para las tres pruebas

Figura 45. Resultado ejecución aplicación con muestras vocal i.

6.4. Pruebas diseñando vocal “o”

❖ Toma de la muestra

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción “Remove gravity” activada	Formato del archivo exportado
Primera prueba	30 Hz	190	sí	csv
Segunda prueba	30 Hz	180	sí	csv
Tercera prueba	30 Hz	178	sí	csv

Tabla 8. Datos toma de muestras vocal o.

❖ Valores aproximados posición

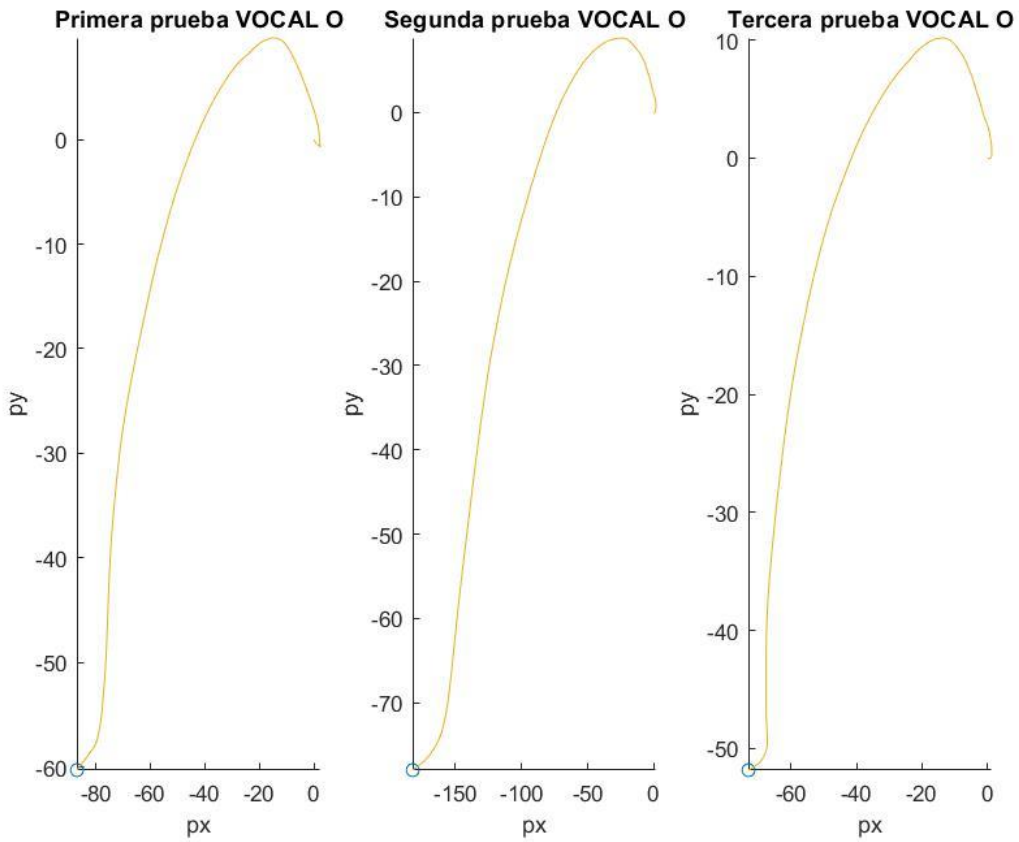

Figura 46. Valores aproximados posición vocal o.

❖ Identificación vocal

	Mínimo de py	Valor de la última muestra de py	Mínimo de px	Valor de la última muestra de py	Distancia entre máximo y minimo de px	vocal =
Primera prueba	-105.2930	-105.2930	-145.8311	-145.8311	147.9555	0 0 0 1 0
Segunda prueba	-78.0007	-78.0007	-182.2778	-182.2778	183.7846	0 0 0 1 0
Tercera prueba	-86.4710	-86.4710	-168.7009	-168.7009	170.4166	0 0 0 1 0

Tabla 9. Resultados ejecución código con muestras vocal o.

❖ Resultado Android Studio para las tres pruebas

Figura 47. Resultado ejecución aplicación con muestras vocal o.

6.5. Pruebas diseñando vocal "u"

❖ Toma de la muestra

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción "Remove gravity" activada	Formato del archivo exportado
Primera prueba	30 Hz	175	sí	csv
Segunda prueba	30 Hz	179	sí	csv
Tercera prueba	30 Hz	175	sí	csv

Tabla 10. Datos toma de muestras vocal u.

❖ Valores aproximados posición

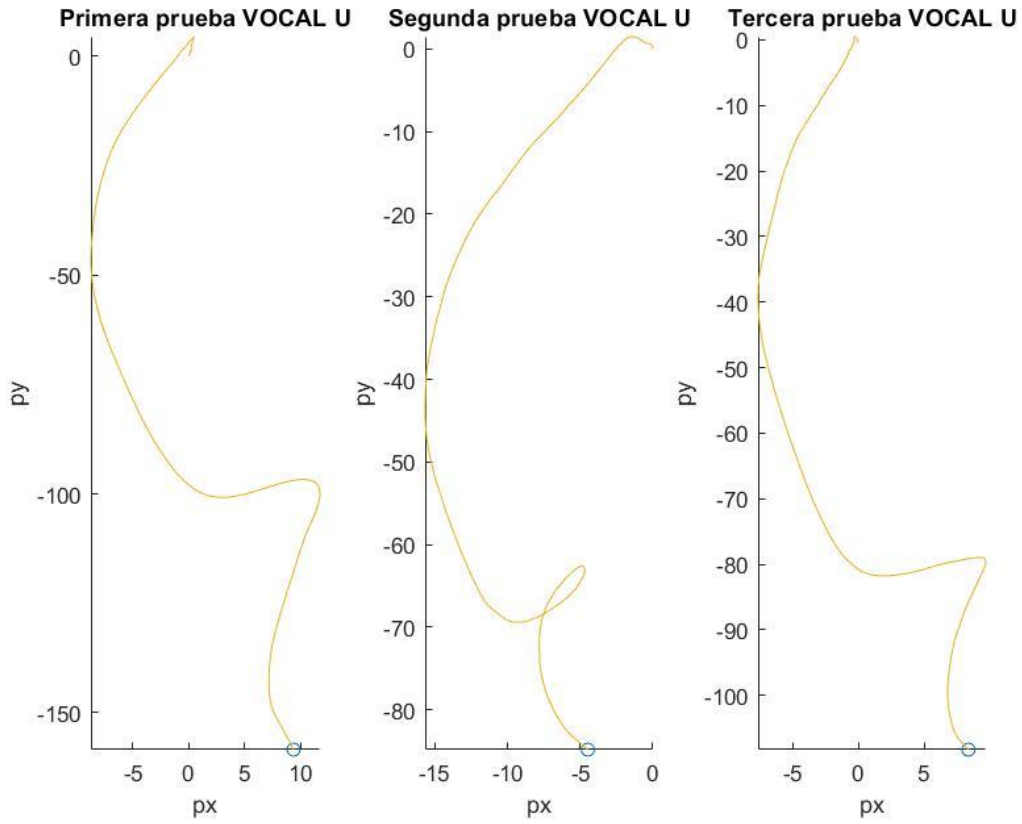

Figura 48. Valores aproximados posición vocal u.

❖ Identificación vocal

	Mínimo de py	Valor de la última muestra de py	Mínimo de px	Valor de la última muestra de py	vocal =
Primera prueba	-158.4553	-158.4553	-8.7342	9.4018	0 0 0 0 1
Segunda prueba	-84.8188	-84.8188	-15.6752	-4.4799	0 0 0 0 1
Tercera prueba	-108.2808	-108.2808	-7.6300	8.3496	0 0 0 0 1

Tabla 11. Resultados ejecución código con muestras vocal u.

❖ Resultado Android Studio para las tres pruebas

Figura 49. Resultado ejecución aplicación con muestras vocal u.

6.6. Pruebas diseñando otros caracteres

❖ Toma de la muestra

	Frecuencia de muestreo (en Hercios)	Número de muestras	Opción "Remove gravity" activada	Formato del archivo exportado
Primera prueba	30 Hz	177	sí	csv
Segunda prueba	30 Hz	253	sí	csv
Tercera prueba	30 Hz	265	sí	csv

Tabla 12. Datos toma de muestras otros caracteres.

❖ Valores aproximados posición

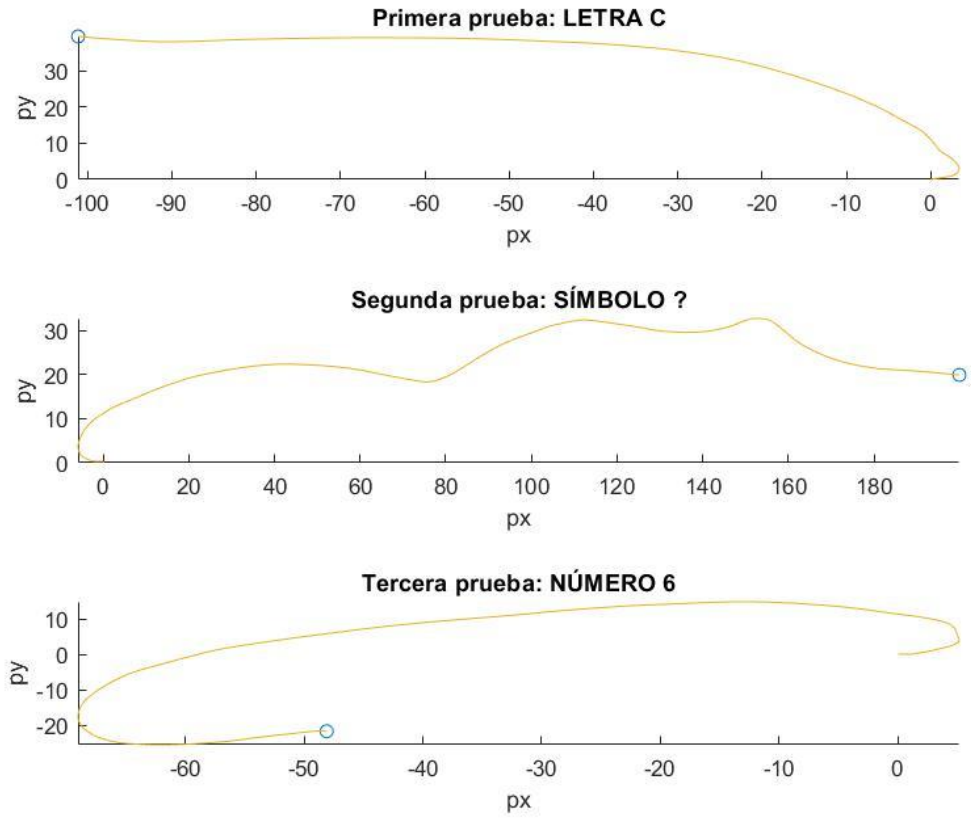

Figura 50. Valores aproximados posición otros caracteres

❖ Identificación vocal

	Mínimo de py	Valor de la última muestra de py	Mínimo de px	Valor de la última muestra de py	vocal =
Primera prueba	-0.0483	-39.6280	-101.1275	-101.1275	0 0 0 0 0
Segunda prueba	-0.0031	19.9040	-5.8583	199.9259	0 0 0 0 0
Tercera prueba	-25.4403	-21.6357	-69.0244	-48.0969	0 0 0 0 0

Tabla 13. Resultados ejecución código con muestras de otros caracteres.

❖ Resultado Android Studio para las tres pruebas

Figura 51. Resultado ejecución aplicación con muestras de otros caracteres.

6.7. Análisis global

Se han realizado más de 50 pruebas por cada tipo de vocal, de las cuales se han mostrado sólo tres, siendo éstas las más significativas, o sea, las que más diferencias presentan unas de las otras. El motivo de no colocarlas todas es que ocuparía casi todo el espacio de este documento. Se produciría así una redundancia, ya que dicha información no aportaría nada nuevo, pues entre las muestras de una misma vocal apenas existen diferencias.

Después de esta aclaración, se puede afirmar que los resultados indican que el programa reconoce de forma eficiente cada una de las cinco vocales, siendo el espacio en memoria y el tiempo de ejecución de dicho programa mucho menor que si se hubieran aplicado las otras soluciones propuestas al principio de la memoria, ya que el algoritmo aplicado es el más sencillo posible.

La posición no ha sido el resultado que se buscaba en un principio, ya que en ese caso las formas de ondas obtenidas serían las formas de las vocales en sí. Esto ha ocurrido debido a imprecisiones del aparato de medida y la simplicidad del método de integración numérica. Sin embargo, lo que interesaba realmente era obtener formas de onda que se diferenciaran unas de las otras visualmente para luego aplicarles métodos matemáticos sencillos.

Así pues, se ha cumplido el objetivo marcado al principio del documento.

CAPÍTULO 7. CONCLUSIONES

El Reconocimiento de vocales a partir del movimiento del dispositivo móvil resultó satisfactorio. Sin embargo, no fue posible realizar las medidas con el acelerómetro de un dispositivo Android al no existir recursos suficientes para adquirirlo. De esa forma, el proyecto tuvo que ser adaptado, tomándose los valores de aceleración con el dispositivo móvil personal, cuyo sistema operativo es iOS. y exportándose estos datos a Android Studio. Aun con estos inconvenientes, la aplicación cumple con el objetivo de este proyecto.

La principal característica de este trabajo es la sencillez, tanto en la elección del método de integración numérica para obtener los valores aproximados de la posición, el cual realiza los cálculos necesarios sin exigir una gran carga computacional, como en la manera de definirse los patrones característicos de cada una de las formas de onda obtenidas. Aun con el error que introduce el acelerómetro, la utilización de estos métodos básicos fue suficiente para realizarse la distinción entre las cinco vocales. En resumen, fue necesario analizar el resultado de la integración de cada tipo de vocal para luego generar una secuencia de código consistente que consiguiese el objetivo marcado al comienzo de esta memoria.

Durante las pruebas iniciales fue posible identificar los patrones que distinguían las vocales con pocas muestras. Como se obtuvo un resultado óptimo, se procedió a implementar los algoritmos en Matlab y en Android Studio sin ninguna dificultad, pues ya se había programado en ellos durante la graduación. Lo que sí llevó más tiempo fue la traducción de un lenguaje a otro, cosa que no se había hecho antes. Más adelante, se hicieron una cantidad abismal de pruebas para testar la eficiencia del código. Fue entonces cuando se percató de que unas pocas muestras no fueron identificas como la vocal que les correspondía, por lo que se tuvieron que perfeccionar algunos detalles del algoritmo, siendo esto un problema, ya que hubo que reescribir el código tanto en Matlab como en Android Studio, lo que provocó un atraso en el proyecto.

La utilización de filtros después de aplicar los métodos de integración para disminuir el error introducido por éstos no fue considerada. Aunque con su uso se obtendrían valores de posición más precisos, no es relevante en este proyecto, pues sólo se necesita diferenciar cinco tipos de caracteres diferentes. En el caso de tener que diferenciarse, por ejemplo, todas las letras del abecedario, sí que sería necesario su uso, pues existe mucha más probabilidad de algunos resultados ser casi idénticos entre sí.

Habiendo aclarado estos puntos, es relevante destacar las utilidades de este proyecto:

- La más interesante es el aprendizaje de caligrafía. Lo ideal sería que los padres se descargasen esta aplicación para usarla con sus hijos pequeños, ya que éstos podrían practicar la escritura intentando diseñar las vocales de la forma más correcta posible, cosa que ayudaría en su formación.

- La segunda y no menos importante, es con fines académicos. Podría plantearse como futura asignatura de los grados en Ingeniería de Telecomunicaciones, ya que el alumno no sólo desenvolvería habilidades para programar en ciertos lenguajes, sino que también aprendería a traducir de unos a otros, lo que los prepararía mejor para enfrentarse al mercado laboral.

Como se podrá deducir, con la realización de este proyecto se adquirió la capacidad de traducir lenguajes de programación, ya que los alumnos no se enfrentan a este reto en ningún momento de la carrera. Otra de las cosas que se aprendió fue a adaptarse circunstancias negativas. En vista a no poder obtener las herramientas necesarias para el desarrollo de la aplicación, se tuvo que actuar con rapidez para encontrar una solución y rehacer el esquema del proyecto. Todo ello enriqueció bastante a nivel profesional.

CAPÍTULO 8. TRABAJOS FUTUROS

A continuación, se citan distintas acciones que podrían mejorar el contenido de este trabajo:

- La primera cosa que se recomienda hacer si se pretende seguir la línea de investigación de este proyecto es adquirir un dispositivo Android. Después de esto, solo resta modificar el código en Android Studio para tomar los datos directamente del acelerómetro y, finalmente, testar la aplicación no sólo en el emulador, sino también en el mismo dispositivo. Esta mejora no supone ningún esfuerzo, pues el algoritmo que toma los datos directamente del sensor ya está implementado en todas las formas posibles. Cualquier persona que haya programado en Android Studio podrá realizar esta mejora en cuestión de días.

- La utilización de otros sensores, tales como el GPS o el giroscopio, es una buena alternativa para auxiliar al acelerómetro en el reconocimiento de caracteres. Aunque el error de posicionamiento de un GPS aún sea de aproximadamente 4,9 metros para la mayoría de los smartphones [29], esta tecnología va a ir mejorando con el paso del tiempo, por lo que su uso podría proporcionar mayor precisión de monitoreo en aplicaciones como la desarrollada en este trabajo fin de grado.

- Aplicar otros métodos de integración numérica en el caso de querer aumentar el número de caracteres específicos a reconocer, o aplicar métodos de entrenamiento con los valores de aceleración instantánea en el caso de querer reconocer todas las variaciones posibles que existen a la hora de diseñar una vocal, fueron dos soluciones descartadas en este proyecto, pero sería interesante su estudio en el caso de modificarse los objetivos iniciales.

ANEXO 1. ACRÓNIMOS

API - Application Programming Interface

APK - Android Package Kit

Csv - Comma-separated values

GPS- Global Positioning System

IDE - Integrated development environment

iOS – iPhone Operating System

JSON - JavaScript Object Notation

MEMS - Micro Electro Mechanical System

NDK - Native Development Kit

XML - Extensible Markup Langu

BIBLIOGRAFÍA

[1] S. Agrawal **et al.**, "PhonePoint Pen: Using Mobile Phones to Write in Air", Systems & Networking Research Group Illinois, agosto de 2008, **[En línea]. Disponible en:** https://synrg.csl.illinois.edu/papers/phonepen.pdf [Último acceso: 15/08/2018]

[2] Ravi, "Different Types of Sensors", *'Eletronics Hub',* 08 de noviembre de2017, **[En línea]. Disponible en:** https://www.electronicshub.org/different-types-sensors/. [Último acceso: 11/06/2018]

[3] "Sensors/Encoders", *Automation Direct.com*, sin fecha, **[En línea]. Disponible en:** https://www.automationdirect.com/adc/Overview/Catalog/Sensors_-z-_Encoders [Último acceso: 11/06/2018]

[4] J. Ramirez, "Interacción en XNA para Windows Phone: Usando el Acelerómetro", *Jorge Ramirez*, 14 de mayo de 2012, **[En línea]. Disponible en:** https://jramirezms.wordpress.com/2012/05/14/interaccion-en-xna-para-windows-phone-usando-el-acelerometro/, [Último acceso: 11/06/2018]

[5] L. Díaz, "Usando los sensores del terminal: acelerómetro y sensor de orientación", *DIWO*, 03 de junio de 2015 **[En línea]. Disponible en:** http://diwo.bq.com/usando-los-sensores-del-terminal-acelerometro-y-sensor-de-orientacion/, [Último acceso: 11/06/2018]

[6] J. Schnellinger, "The Principles of Piezoelectric Accelerometers", *Sensor Online*, 01 de abril de 2004, **[En línea]. Disponible en:** https://www.sensorsmag.com/components/principles-piezoelectric-accelerometers [Último acceso: 11/06/2018]

[7] "Tipos mais comuns de acelerômetros", *Seara da Ciência*, sin fecha, **[En línea]. Disponible en:** http://www.searadaciencia.ufc.br/tintim/tecnologia/acelerometro/acelerometro01.htm, [Último acceso: 11/06/2018]

[8] "Acelerómetros piezoeléctricos", *Sensing*, sin fecha, **[En línea]. Disponible en:** http://www.sensores-de-medida.es/sensing_sl/SENSORES-Y-TRANSDUCTORES_35/Aceler%C3%B3metros---Sensores-de-aceleraci%C3%B3n_49/Aceler%C3%B3metros-piezoel%C3%A9ctricos_50/, [Último acceso: 11/06/2018]

[9] C. Aszkler, "The Principles of Acceleration, Shock, and Vibration Sensors", *Sensors Online*, 01 de mayo de 2005, **[En línea]. Disponible en:** https://www.sensorsmag.com/components/principles-acceleration-shock-and-vibration-sensors, [Último acceso: 11/06/2018]

[10] "Question: 10.3 Piezoresistive accelerometer. The basic structure of a Piezoresistive accelerometer", *Cheggy Study*, sin fecha, **[En línea]. Disponible en:** https://www.chegg.com/homework-help/questions-and-answers/103-piezoresistive-

accelerometer-basic-structure-piezoresistive-accel-erometer-shown-figur-q23358746, [Último acceso: 19/06/2018]

[11] "Acelerómetros piezoresistivos", *Sensing* , sin fecha, **[En línea]. Disponible en:** http://www.sensores-de-medida.es/sensing_sl/SENSORES-Y-ZRANSDUCTORES_35/Aceler%C3%B3metros---Sensores-de-aceleraci%C3%B3n_49/Aceler%C3%B3metros-piezoresistivos_59/, [Último acceso: 19/06/2018]

[12] Engineerguy, "How a Smartphone Knows Up from Down (accelerometer)", *Youtube*, 22 de mayo de 2012, **[Video en línea]. Disponible en:** https://www.youtube.com/watch?v=KZVgKu6v808, [Último acceso: 20/06/2018]

[13] F. Mecafenix, "Acelerómetro sensor de movimiento o vibración", *Ingeniería Mecafenix,* 30 de octubre de 2017, **[En línea]. Disponible en:** http://www.ingmecafenix.com/automatizacion/acelerometro/, [Último acceso: 19/06/2018]

[14] "Acelerómetros capacitivos", *Sensing,* sin fecha, **[En línea]. Disponible en:** http://www.sensores-de-medida.es/sensing_sl/SENSORES-Y-TRANSDUCTORES_35/Aceler%C3%B3metros---Sensores-de-aceleraci%C3%B3n_49/Aceler%C3%B3metros-capacitivos_80/, [Último acceso: 19/06/2018]

[15] "High Performance MEMS Accelerometers", *Tronics Group,* sin fecha, **[En línea]. Disponible en:** https://www.tronicsgroup.com/AXO-High-Performance-MEMS-Accelerometers, [Último acceso: 19/06/2018]

[16] P.Axelsson **y** M. Norrlöf, " Method to Estimate the Position and Orientation of a Triaxial Accelerometer Mounted to an Industrial Manipulator", *IFAC*, vol. 45, n.°22, pp. 283-288, septiembre de 2012. **[En línea]. Disponible en:** https://www.sciencedirect.com/science/article/pii/S1474667016336242. [Último acceso: 20/07/2018]

[17] W. Zhu **y** T. Lamarche, " Position Tracking Control with Velocity from Accelerometer and Encoder", IFAC, 2008. *IFAC*, vol. 41, n.°2, pp. 14828-14833, julio de 2008. **[En línea]. Disponible en:** https://www.sciencedirect.com/science/article/pii/S1474667016413753, [Último acceso: 20/07/2018]

[18] D. Pilo, "De la pluma de ganso al bolígrafo", *Inspirulina,* 24 de abril de 2013, **[En línea]. Disponible en:** https://www.inspirulina.com/de-la-pluma-de-ganso-al-boligrafo.html , [Último acceso: 19/06/2018]

[19] J.L. Fernández , "Aceleración Instantánea", Fisicalab, sin fecha, [En línea]. Disponible en: https://www.fisicalab.com/apartado/aceleracion-instantanea#contenidos, [Último acceso: 20/07/2018]

[20] A. Palagia, "Somas de Riemman e Integração Numérica", *Universidade Federal de Pelotas,* julio de 2012, **[En línea]. Disponible en:** https://wp.ufpel.edu.br/nucleomatceng/files/2012/07/Somas-de-Riemann-e-integra%C3%A7%C3%A3o-num%C3%A9rica_.pdf, [Último acceso: 23/07/2018]

[21] R. Biloti, "Integração Numérica", *Instituto Militar de Engenharia,* 2018, **[En línea]. Disponible en:** http://www.ime.unicamp.br/~biloti/an/quad.pdf, [Último acceso: 23/07/2018]

[22] S. Pilling, "IV-Integração Numérica", *Univap*, sin fecha, **[En línea]. Disponible en:** https://www1.univap.br/spilling/CN/CN_Capt6.pdf, [Último acceso: 23/07/2018]

[23] "Riemman Integral", *Katedra Matematiky,* sin fecha, **En línea]. Disponible en:** http://math.feld.cvut.cz/mt/txtd/1/txe3da1a.htm, [Último acceso: 23/07/2018]

[24]"Mathematics", *Mathworks,* sin fecha, **[En línea]. Disponible en:** https://es.mathworks.com/help/matlab/mathematics.html, [Último acceso: 23/07/2018]

[25]"Graphics", *Mathworks,* sin fecha, **[En línea]. Disponible en:** https://es.mathworks.com/help/matlab/graphics.html, [Último acceso: 23/07/2018]

[26] "Conoce Android Studio", *Android Developers,* 25 de abril de 2018, **[En línea]. Disponible en:** https://developer.android.com/studio/intro/?hl=es-419, [Último acceso: 19/06/2018]

[27]"Data Import and Analysis", *Mathworks,* sin fecha, **[En línea]. Disponible en:** https://es.mathworks.com/help/matlab/data-import-and-analysis.html, [Último acceso: 23/07/2018]

[28] Fran Garcia, "23. Manejo de Ficheros en Android. (Programación Android Studio tutorial español)", *Youtube,* 18 de octubre de 2015, **[Video en línea]. Disponible en:** https://www.youtube.com/watch?v=lbBosCrFE7Y, [Último acceso: 23/08/2018]

[29] "What is GPS?, Available", *GPS.GOV,* sin fecha, **[En línea]. Disponible en:** https://www.gps.gov/systems/gps/performance/accuracy/ [Último acceso: 23/08/2018]

RESUMEN EN INGLÉS

A continuación, se incluye el documento que contiene el resumen en inglés. Se numerarán las páginas desde cero, pues es como si se tratase de un archivo pdf diferente a éste.

Esta página no está incluida en el índice, ya que se ha creado a modo de aclaración.

uc3m | Universidad Carlos III de Madrid

University Degree in Audiovisual System Engineering

2017-2018

Bachelor Tesis

“Detection of characters by accelerometry in Android devices”

Author:

Ana del Carmen Cañas Cañas

Tutor:

Mario Muñoz Organero

Leganés, September 2018

1. INTRODUCTION

During the first years of the degree, students learned to program in Java, Matlab and C. The software developed during this period served to understand the foundation of the language that was being used, but they did not have a tangible application.

However, the situation changed dramatically as time passed. Without going any further, during one of the subjects of the last course something as practical and tangible as designing an application for Android devices was made, since anyone, whether or not they had programming knowledge, could be able to make a real use after its execution. At this point, this work carried out not only generated knowledge about this subject, but also improvements in the quality of life of other people, and that was the moment it was born the desire to create, to discover more about that world. Specifically, the beginning of reading about the sensors of mobile devices, because during the execution of the project mentioned above the use of the GPS sensor aroused a remarkable interest. The finding that drew particular attention was "PhonePoint Pen: Using Mobile Phones to Write in Air", an article written by Sandip Agrawal and other researchers from Duke University (United States of America), where they develop a project whose objective was to turn the movements of a smartphone in alphabet characters. After reading it, there was an interest in developing this project.

The objective of this project is to develop a simple algorithm for the recognition of vowels from accelerometry signals, as well as the implementation and validation of these algorithms in Matlab for later migration to a mobile device with the Android operating system. It will be based on the accelerometer data of a mobile device that a user will move by drawing on a flat surface the writing of a vowel. Based on basic characteristics such as the relative location and maximum and minimum amplitudes of the raw linear acceleration signal or the estimation of the position from it, manual characteristics that allow the recognition of the 5 vowels will be defined. In order to execute this project, material and personal costs were included, the former is 134,14 € and the latter 3060 €. Which means that the total amount necessary is 3194,14 €.

2. STATE OF ART

Many years ago, industrial equipment was handled exclusively by people, so the monitoring processes were 100% manual. This is not efficient since it is impossible for human beings constantly supervise the machines and react in time to any problem. To solve it, applications were created in order to replace the manual work. They were

becoming more precise and sophisticated, thus causing electronic equipment and industrial plants to obtain faster and more realistic responses.

In order to improve the monitoring processes using autonomous systems, the sensor concept was born, which is a device capable of giving an electrical response to external physical stimuli. It is an element that translates a real quantity, both physical (temperature, length, acceleration, etc.) and chemical (level of sugar in the blood, level of a dangerous gas that floats through the air, etc.), in units that can be interpreted simply by an electronic device.

The use of sensors in the manufacturing chain increases the quality of this process since its use improves the detection of failures and wear in the equipment used. It reacts at a speed previously unimaginable, therefore causing the reduction of costs related to production losses. However, the use of sensors is not limited to industrial applications, they are present in the daily life of the majority of the world population. Without realizing it, most people use them daily, since mobile devices contain a lot of them.

An accelerometer is one of these sensors, it converts accelerations and changes of speed, into an electrical signal. Accelerations can be measured in meters per second squared (m/s^2) or in G-Force (g). There are many kinds of this device, among which Piezoelectric, Piezoresistive, and Capacitive stand out.

Due to their characteristics, accelerometers provoke the interest of researchers seeking to improve movement systems. This is the case of Patrik Axelsson and Mikael Norrlöf, two Swedish scientists from Linköping University, who introduced an accelerometer into a robotic arm with six degrees of freedom (movement of displacement along the x, y and, and rotation of each of them) to estimate their orientation and position. For this, they conducted two experiments. The first consisted of leaving the robot stopped and only moving its arm. Specifically, they changed position 6 times, to make the respective measurements of the acceleration in the x, y, and z-axes. They did this to determine their inertial values and define their orientation. The second experiment calculated the position's vector using vector equations, taking as origin the base of the robot and as end the accelerometer. This vector was derived twice as a function of time, resulting in the acceleration's vector. They modified the arm position with constant angular velocity in order to obtain the experimental acceleration values. Finally, they compared these results with the values obtained in the theoretical method. They realized that there was only one degree of error between them, so the result of the project was satisfactory.

Other researchers who were interested in accelerometry were Wen-Hong Zhu and Tom Lamarche (Canadian Space Agency). They carried out a work whose the objective was

to calculate the speed of a mass that moved along a single axis. For this, they started using speed sensors, but they realized that measurement errors at high frequencies were abysmal. They tried to correct this error by applying filters. They succeeded, but in return, they generated another error. After applying the filter, there was always a lag of -90º, so the reaction time to make a movement increased. Then they had the idea to replace the speed sensor with an accelerometer because the speed is always out of phase -90º with respect to acceleration, so, using it, what they would really be obtaining would be speed values. Now another problem arises, the accelerometer has a very large margin of error in low frequencies. Then they thought it is necessary to use another sensor, such as the linear encoder to correct it. Combining both, the error in low frequencies was reduced, obtaining, finally, the real value of the speed.

Although the projects described so far present estimates of simpler movements (movements with constant or rectilinear velocities) and purely mechanical applications, applications for mobile devices are also being developed. An example of this is the application developed by students at Duke University with the aim of improving the way of communication between people. This allows the user to use the mobile device like a pen, in order to write a short message in two dimensions. The data from the accelerometer are exported to Matlab, where the acceleration values are integrated as a function of time, thus finding the values of the position.

But this application presents problems related to the time required to write a letter because it needs pauses in specific positions to equal zero accumulated errors from the accelerometer. When such failures are integrated, the imprecision of the position values further increases, making efficient character recognition difficult. Another problem is that it is not focused on Android Studio since it is necessary to run the program made in Matlab to check the results obtained. Therefore, it is not feasible for a user without engineering knowledge to create that application and give it an actual use.

The use of the accelerometer has presented positive results for the determination of the position, although this one generates natural errors and particular characteristics in some applications. As shown in the project "PhonePoint Pen", there are problems in both the taking and the handling of data. Therefore, there is a need to translate the movements made by a smartphone into characters with a short and efficient response time, both in data acquisition and in determining results. Showing the user the exact form of the vowel it has written is not the scope of this project, neither informing whether it has done it well or not. Therefore, it is not interesting to calculate the position accurately, neither to find a mathematical method that, starting from the acceleration, can identify the types of characters that are being studied. These characters have predictable and standardized movements so that the training resource can be avoided since it is only necessary to identify concrete forms, and not to identify a range of possibilities.

In addition to this, a simpler and more understandable approach to this technology can bring benefits to Telecommunications Engineers or Computer Engineers, which would only focus on perfecting this type of application.

3. ANALYSIS

Each time a person picks up a pen and writes a letter on a piece of paper, he is doing something as simple as drawing the lines of a specific figure on a flat surface. These lines are nothing more than continuous and indefinite successions of points. In short, designing a letter involves moving an object so that it generates coordinates in a two-dimensional space.

Well, what is intended is that the accelerometer that is integrated into the mobile device, which will move implicitly when the device moves, act as a ballpoint pen. This tool will move in the space where it is being used, thus causing an acceleration in three dimensions.

When trying to simulate the movement of a person writing on a two-dimensional surface such as paper, the information of interest is on the axes "x" and "y", the values of the "z" axis being disposable.

After clarifying which points of space are of interest, it is necessary to highlight the trajectory that they must follow to be identified as one of the five vowels that they intend to recognize.

That said, all that remains is to analyze the mathematical methods that can lead to an optimal solution to the problem that arises in this project, but not before making some clarifications about the tool that will be used to obtain the samples with which they go to work. As already mentioned above, the accelerometer can measure accelerations in meters per second squared (m/s^2) or in G-Force (g). In this project, we want to work with the second type of measure, so we must use an application that returns the values in G or, failing that, makes the conversion if they were taken other way. When using the accelerometer, the acceleration measurements will be obtained in three axes at each instant of time, depending on the sampling frequency. Since it is intended to use the program with all types of accelerometers of mobile devices, it is necessary to choose a sampling frequency that can be selected in all of them. The estimated time of the sample collection is between 4 and 10 seconds since it is usually the time to write a vowel.

From here, it is considered what to do with data obtained after using the accelerometer to differentiate each of the vowels. These data are values of accelerations, so the first solution that arises is to calculate the module of its instantaneous acceleration. As an alternative, it is proposed to find the position of the device from the accelerometer data in a two-dimensional Cartesian system. By reducing the scope to a single axis, the procedure to find the position can be seen as a double integration. Discrete and finite functions will be handled, so the most efficient solution is to apply some of the numerical integration methods.

The numerical integration methods are applied to a function in order to calculate the approximate value of the definite integral.

Working with instant acceleration was the first option disregarded, since choosing that path involved the use of training apps, which would be a success if the aim was to detect vowels in all possible shapes and sizes, but is not the case. Since it is interesting to identify the geometric shapes shown in previous chapters only, it is more efficient to see the pattern they follow in order to differentiate them, since their implementation will be less robust without implying the loss of precision. It is chosen, therefore, to calculate the position by means of numerical integration.

It is intended to integrate the values of acceleration twice in order to obtain different results for each type of vowel, in order to extract features in an intuitive and simple way. Therefore, it is not necessary for the result to be as close as possible to the values of the real positions, so the simplest method of all is going to be applied: the sum of Riemman. Again, it has been chosen, among all the alternatives that solve the problem, the least robust.

4. DESING

To solve the main objective of this project it has been necessary to develop an algorithm, which has been implemented, first, in Matlab. After that code has been migrated to Android Studio. As it has been impossible to have an Android device for the realization of the project, it was necessary to extract the accelerometer data from an iPhone, through the application "Accelerometer", to later export them to Matlab and Android Studio.

Accelerometer collected acceleration data from the mobile device and created a .cvs file that was sent to Matlab. This file contained the sampling index and the acceleration present in the three axes (x, y, and z), these values were integrated once to obtain the velocity, and twice to position, then they were plotted in order to extract the

characteristics that differentiate one waveform from another to proceed to the identification of the vowels.

The Matlab code was translated line by line, in order to run using Android studio simulator since these are two completely different programming languages.

It was impossible to have an Android device to perform this project, so the solution was adopted under these circumstances. What changes is that the file containing the samples taken with any accelerometer is read, so they are not taken directly from the Android device's accelerometer. This file has been obtained using the accelerometer of an iPhone.

5. IMPLEMENTATION

As well mentioned above, the code was first developed in Matlab and then taken to Android Studio.

Before starting, files in .csv format obtained with the "Accelerometer" application must be exported to the local MatLab directory.

The program starts by reading two files: the one that contains the values taken with the mobile at rest (file_reposo.csv) and the one that contains the values of the sample to analyze (muestra_1.csv).

Now, for each file, 4 vectors are going to be created: the first one will contain the values of "index", the second one those of "x". the third those of "and" and the last those of "z".

As has been mentioned previously, the only values that are of interest are those of the "x" and "y" axes, so that only their corresponding vectors will be manipulated.

That said, the average of the vector containing the values when the mobile device is at rest is calculated, in order to obtain the approximate value of the measurement error. Each error value will be subtracted from each of the acceleration values.

With these last vectors, we will proceed to implement the numerical integration method by which we opted in the analysis phase. This will be applied separately to the values of "x" and the values of "y".

Once obtained the approximate values of the velocity (vx, vy), we proceed to apply the same method to obtain the approximate values of the position (px and py).

The first thing that is done is to create a vector of five positions called "vocal", which is initialized with zero. Each of the positions of the vector corresponds to a vowel, being the correspondence in alphabetical order. Thus, the first position of the vector corresponds to the vowel "a", while the last one corresponds to the vowel "u". When one of the five vowels have been detected, the value of the corresponding position of "vowel" will change from being 0 to being 1. Therefore, there are only two possible states: the vowel has been identified, where vector [i] will be equal to one, or the vowel has not been identified, where vector [i] will be equal to zero.

One of the first characteristics that can be extracted after observing the graphs of the approximation of the positions of each of the vowels is that the letters "i", "o" and "u" differ from "a" and "e" in which the minimum is always the value of the last sample, so that is the first classification condition. Within it, a simple way will be used to differentiate each one of these three vowels.

- If the value of x at the time instant in which is the minimum value of py does not coincide with the minimum of px, it is the vowel “u”.

- If the value of x at the time instant in which is the minimum value of py coincides with the minimum of px, and the distance between the maximum value of py and the minimum value of px is lower than 40, it is the vowel “i”.

- If the value of x at the time instant in which is the minimum value of py coincides with the minimum of px, and the distance between the maximum value of py and the minimum value of px is higher than 40, it is the vowel “o”.

When the minimum of py is not the value of the last sample, it would be an "a" or an "e", in which case the two would have to be differentiated.

- If the minimum of py is before the first pass by zero and the maximum is after, it is an “e”.

- If there is a maximum to the left and another to the right of the minimum of py and they are almost at the same distance, it is an “a”.

- If none of these criteria is detected, it is not a vowel.

This code was implemented using Android Studio and an application with graphics interface shows how the letters should be written, in order to recognize the movement.

6. RESULTS

It was performed more than 50 tests for each vowel, of which three of each of them have been shown, these being the most significant, in other words, the ones with the most differences among the others. The reason for not placing all of them is that it would occupy almost all the space of the document, which would cause unnecessary repetition since it does not bring anything new, because the rest of the samples show small differences.

After this clarification, it can be affirmed that the results indicate that the program efficiently recognizes each of the five vowels, the space in memory and the execution time of this program being much less than if the other proposed solutions had been applied at the beginning of the memory, since the applied algorithm is the simplest possible. Thus, the objective set at the beginning of the document has been achieved.

The position has not been the result that was sought in the beginning, since in that case the waveforms obtained would be the same as the letters. This has happened due to inaccuracies of the measuring device and the simplicity of the numerical integration method. However, the goal was to obtain waveforms that were visually different from each other and then apply simple mathematical methods, so that the objective marker at the beginning of the document has been fulfilled.

7. CONCLUSIONS

The Recognition of vowels from the movement of the mobile device was satisfactory. However, it was not possible to perform the measurements with the accelerometer of an Android device, as there were not enough resources to acquire it. In this way, the project had to be adapted, taking the values of acceleration with the personal mobile device, whose operating system is iOs. and exporting this data to Android Studio. Even with these drawbacks, the application meets the objective of this project.

The main characteristic of this work is the simplicity, both in the choice of the numerical integration method to obtain the approximate values of the position, which performs the necessary calculations without requiring a large computational load, and in the way of defining the characteristic patterns of each of the vectors obtained. Even with the error introduced by the accelerometer, the use of these basic methods was enough to make the distinction between the five vowels. In short, it was necessary to analyze the result of the integration of each type of vowel and then generate a consistent code sequence that achieved the target marked at the beginning of this report.

During the initial tests it was possible to identify the patterns that distinguished the vowels with few samples. As an optimal result was obtained, we proceeded to implement the algorithms in Matlab and in Android Studio without any difficulty, since it had already been programmed in them during the graduation. What did take longer was the translation from one language to another, as this had not been done before. Later, a huge number of tests were done to test the efficiency of the code. It was then when he realized that a few samples were not identified as the corresponding vowel, so they had to refine some details of the algorithm, this being a problem, since the code had to be rewritten in both Matlab and Android Studio, which caused a delay in the project.

The use of filters after applying the integration methods to reduce the error introduced by them was not considered. Although with its use more precise position values are obtained, it is not relevant in this project, since only five different types of characters need to be differentiated. In the case of having to differentiate, for example, all the letters of the alphabet, it would be necessary, since there is much more probability of some results being almost identical to each other.

Having clarified these points, it is important to highlight the utilities of this project

The most interesting is the learning of calligraphy. Ideally, parents should download this application for use with their young children, as they could practice writing trying to design the vowels in the most correct way possible, which would help in their training.

The second and not least, is for academic purposes. It could be considered as a future subject of the degrees in Telecommunications Engineering, since the student would not only develop skills to program in certain languages, but also learn to translate from one to another, which would prepare students better to face the labor market.

As it can be deduced, with the realization of this project the ability to translate programming languages was acquired, since the students do not face this challenge at any moment of the race. Another thing that was learned was to adapt to the circumstances. In

view of not being able to obtain the necessary tools for the development of the application, we had to act quickly to find a solution and redo the scheme of the project. All this enriched quite a professional level.

8. FUTURE WORKS

The first thing that is recommended to do if you intend to follow the research line of this project is to acquire an Android device. After this, it only remains to modify the code in Android Studio to take the data directly from the accelerometer and, finally, to test the application not only in the emulator but also in the same device. This improvement does not involve any effort since the algorithm that takes the data directly from the sensor is already implemented in all possible ways. Anyone who has programmed in Android Studio could make this improvement in a matter of days.

The use of other sensors, such as GPS or gyroscope, could help the accelerometer in character recognition. Although the positioning error of a GPS is still approximately 4.9 meters for most smartphones, this technology will improve with the passage of time, so its use could provide greater monitoring accuracy in applications such as developed in this final degree project.

Apply other methods of numerical integration in case you want to increase the number of specific characters to recognize or apply training methods with the values of instantaneous acceleration in case you want to recognize all the possible variations that exist when designing a vowel

l, were two solutions discarded in this project, but it would be interesting to study them in the case of modifying the initial objectives.