**UNIVERSIDAD AUTÓNOMA DE MADRID**

**ESCUELA POLITÉCNICA SUPERIOR**



# PROYECTO FIN DE CARRERA

## APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES SECUENCIALES BAJO ANDROID

**Pablo Molinero Merino**

**Febrero 2014**

# APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES SECUENCIALES BAJO ANDROID

**AUTOR: Pablo Molinero Merino**
**TUTOR: Fernando Barbero Díaz**
**PONENTE: Eduardo Boemo Scalvinoni**

**DSLab**
**Dpto. de Tecnología Electrónica y Comunicaciones**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Febrero de 2014**

## *Resumen*

En este proyecto se desarrolla una aplicación móvil destinada a los alumnos de la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior (Universidad Autónoma de Madrid), que permite solucionar diez problemas de diseño de máquinas de estado tipo Moore y tipo Mealy, además de incorporar un tutorial dedicado a máquinas de Moore.

El objetivo principal es mejorar el formato de las guías de problemas de la asignatura, añadiendo corrección automática y ayudas en los ejercicios, además de facilitar la compartición de soluciones por correo electrónico.

En primer lugar se realiza un estudio del arte en el que se describen a groso modo los diferentes sistemas operativos móviles, con más detalle en Android. También se realiza una búsqueda de aplicaciones en el mercado que tenga alguna similitud con la que se desarrolla en este proyecto.

Seguidamente, se describe y justifica cómo va a ser el diseño de la aplicación, para después detallar el desarrollo de la misma.

Finalmente se describe cómo se ofrece la aplicación a los alumnos, añadiendo algunos resultados a corto plazo.



## *Palabras clave*

Aplicación, Android, Moore, Mealy, circuitos secuenciales, digitales, FSM, máquina de estados finitos, móvil, tableta.

## *Abstract*

In this project, we develop a mobile application for students of Digital Electronic Circuits subject in Escuela Politécnica Superior (Universidad Autónoma de Madrid). The application allows to solve ten problems of Moore and Mealy finite state machines, as well as include a Moore state machine tutorial.

The main objective is to improve the exercise guide format of the subject, adding automatic correction and tips in exercises, in addition to facilitate sharing solutions by mail.

First, a study of the art in which we described roughly the different mobile operative systems is done. A search of mobile applications in market with some similitude to the one developed in this project is also done.

After, we describe and justify how the application design will be, and then detail the development of it.

Finally, we describe how the application is provided to students, adding some results in the short run.



## *Key words*

Application, Android, Moore, Mealy, sequential circuits, digital, FSM, finite state machine, mobile, tablet.

## *Agradecimientos*

En primer lugar y más especial, quiero dar las gracias a mis padres, a mi hermana y a toda mi familia por todo lo que me han ayudado y aportado para llegar hasta aquí.

A Cristina, pilar muy importante en mi vida. Por lo que hemos vivido y por lo que nos queda por vivir juntos.

Agradecer a Eduardo por darme la oportunidad de hacer este proyecto.

Tampoco olvidarme de mi gran compañero de viaje Pedro.

Gracias a todos mis amigos y a toda persona que, de alguna manera, han aportado algo en mi vida.

# ÍNDICE DE CONTENIDOS

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1 Introducción

## *1.1 Motivación*

Las guías de problemas de la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid se entregan en papel tal y como se hacía en los años 70. Adicionalmente se publican en formato PDF, pero el resultado final sigue siendo una hoja de papel.

Estas guías de problemas no traen sugerencias, soluciones parciales, ni tampoco corrigen los ejercicios. Por lo tanto, podemos cuestionarnos si existe un nuevo formato que mejore el actual.

Una posible respuesta la podemos encontrar en los teléfonos y tabletas inteligentes. La alta penetración de estos dispositivos entre los estudiantes ofrece una plataforma ideal para desarrollar aplicaciones educativas. Una posibilidad es mejorar el formato actual de las guías de problemas, creando una aplicación que las ofrezca y agregue las siguientes características:

- Incorporación de herramientas de corrección.
- Incorporación de ayudas parciales.
- Posibilidad de enviar por mail los resultados al profesor/a o a compañeros/as.
- Ofrecer tutoriales sobre temas concretos.

Adicionalmente, la tecnología de teléfonos móviles y tabletas ofrece otras ventajas tales como:

- Posibilidad de modificación y redistribución automática a través de la estructura de Google Play.
- Eliminación de papel.
- Mayor protección intelectual frente a plagios.
- Trabajar en cualquier sitio con o sin iluminación.
- Acceso rápido a cualquier estudiante interesado, y en cualquier parte del mundo que tenga acceso al mercado de las aplicaciones móviles.

## *1.2 Objetivos*

El objetivo de este proyecto es desarrollar una aplicación móvil destinada a los alumnos de la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior (Universidad Autónoma de Madrid), que ofrezca diez problemas de circuitos digitales secuenciales, además de un tutorial dedicado a las máquinas de Moore (el tutorial sobre máquinas de Mealy puede entrar en futuras actualizaciones o en una nueva aplicación).

Los problemas tratarán concretamente en el diseño de máquinas de estado de Moore y Mealy.

La aplicación permitirá al usuario solucionar los problemas y la posibilidad de corregirlos.

Como complemento, podrán guardar, cargar y enviar ejercicios por correo para una mayor interacción alumno-alumno y alumno-profesor.

Tanto los problemas como el tutorial proporcionado ayudarán al alumno a comprender los siguientes conceptos:

- Máquinas de Moore y Mealy.
- Detección de secuencias.
- Aritmética serie.
- Solapamiento.
- Control industrial.

La aplicación se desarrollará para el sistema operativo Android. En el apartado 3.3 se justifica la elección de este sistema operativo. Será válida tanto para móviles como para tabletas.

La aplicación será ofrecida a los alumnos a través de la plataforma de Google Play, en la que podrán descargarla, valorar, opinar y proponer mejoras para un futuro.

No será necesaria la conexión a Internet para utilizar los contenidos de la aplicación.

## *1.3 Organización de la memoria*

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción, motivación y objetivos del proyecto.
- Capítulo 2: Estado del arte.
- Capítulo 3: Diseño de la aplicación
- Capítulo 4: Desarrollo de la aplicación
- Capítulo 5: Integración, pruebas y resultados.
- Capítulo 6: Conclusiones y trabajo futuro.

# 2 Estado del arte

## *2.1 Sistemas operativos móviles*

En este apartado se van a exponer de una manera simple los principales sistemas operativos móviles activos en la actualidad, con gran énfasis en Android, el sistema operativo elegido para realizar el proyecto.

Después se realizará un pequeño repaso del estado del arte de las aplicaciones educativas en el apartado de los circuitos secuenciales, máquinas de estado y resolución de ejercicios de electrónica digital.

### 2.1.1 Android

#### *2.1.1.1 Historia y características*

Android es un sistema operativo para dispositivos móviles basado en la versión 2.6 de Linux. Fue desarrollado inicialmente por Android Inc., empresa que fue comprada por Google en el año 2005.

En el año 2007 la Open Handset Alliance, un consorcio de varias empresas como Texas Instruments, Samsung Electronics o Intel entre otras, y creado para desarrollar estándares abiertos para dispositivos móviles, anunció su primer producto, el sistema operativo Android. En 2008, salió a la venta el primer dispositivo móvil que usaba Android, el HTC Dream.

Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto. Esto permite que los desarrolladores accedan al código para modificarlo y mejorarlo. Para publicar una aplicación propia en la tienda Google de aplicaciones Android, será necesario abonar una cantidad muy baja una sola vez y para toda la vida.

Android combina una serie de características que lo hacen diferente, y que son:

- **Plataforma abierta**: como ya se ha comentado, Android es una plataforma de código abierto, y por lo tanto cualquier desarrollador tiene acceso a él. Esto hace que aparezcan continuas mejoras y se construya un sistema robusto.

- **Arquitectura de componentes**: las partes de una aplicación pueden utilizarse en otras distintas, incluso se pueden sustituir componentes integrados por otros de tu propia creación. Esto hace que se active la creatividad entre los desarrolladores.

- **Filosofía de equipo siempre conectado a Internet.**

- **Servicios incorporados:** poseen una gran cantidad de éstos, como por ejemplo localización GPS o localización basada en torres de telefonía móvil (AGPS), reconocimiento de voz, navegador, bases de datos…

- **Aceptable nivel de seguridad:** los programas se encuentran aislados unos de otros gracias al concepto de ejecución dentro de una caja que incorpora la máquina virtual, y que se ha heredado de Linux. Además, cada aplicación dispone de una serie de permisos que limitan su campo de actuación.

- **Gráficos y sonido de alta calidad:** gráficos vectoriales 2D, gráficos 3D basados en OpenGL, códec estándar comunes como H.264, MP3, AAC…

- **Portabilidad asegurada:** el lenguaje de programación de las aplicaciones es principalmente Java, lo que nos asegura que podrán ser ejecutadas en gran variedad de dispositivos tanto presentes como futuros, gracias al concepto de máquina virtual.

    - **Concepto de máquina virtual:** Java es un lenguaje que primero se compila obteniendo un código intermedio llamado "bytecode" (que no es el lenguaje máquina que entiende la plataforma donde se va a ejecutar el programa) y después se "ejecuta" en una máquina virtual Java.

      Esta máquina virtual es la que traduce ese código intermedio al lenguaje que entiende la plataforma (ARM, x86…). Por lo tanto es necesario que en cada plataforma exista una máquina virtual Java para ejecutar el programa. La mayoría de plataformas disponen de una máquina virtual Java.

      La ventaja de esto es que cualquier programa Java ya compilado se puede ejecutar en cualquier plataforma que disponga de dicha máquina virtual. Esto produce la citada portabilidad asegurada.

      Android utiliza su propia máquina virtual llamada **Dalvik**, creada por ellos y basada en la máquina virtual de Java, que está optimizada para dispositivos móviles.

#### *2.1.1.2 Arquitectura*

La arquitectura de Android está basada en capas de software libre. Se explica de una forma sencilla cada capa junto a un gráfico explicativo.

- **Núcleo de Linux:** proporciona una capa de abstracción para los elementos hardware que tienen que utilizar las aplicaciones. Esto permite que se pueda acceder a esos componentes sin necesidad de conocer el modelo o características en cada teléfono. Para cada elemento hardware del teléfono existe un controlador dentro del Kernel que permite utilizarlo desde el software.

- **Bibliotecas:** bibliotecas nativas de Android, escritas en C o C++ y compiladas para la arquitectura específica hardware del teléfono, tarea que normalmente realiza el fabricante, que también las instala en el dispositivo. Proporciona funcionalidad a las aplicaciones, para tareas que se realizan con frecuencia.

- **Runtime de Android:** no se considera una capa en sí mismo, ya que contiene las bibliotecas esenciales de Android. Éstas contienen las propias bibliotecas de Java y otras específicas de Android.

En el runtime de Android se encuentra la comentada anteriormente máquina virtual Dalvik, propia de Android, que no recibe los archivos compilados java ".class". Al compilar los archivos programados en Java, en el SDK de Android se crean unos archivos de formato específico llamado ".dex" (Dalvik EXecutable). Son archivos más compactos que los ".class", preparados para una mayor optimización. Además, Dalvik está basada en registros y no en pila para guardar los datos, lo que requiere de menos instrucciones y además produce ejecuciones más rápidas.

- **Entorno de aplicación:** en esta capa se encuentran las clases y servicios que utilizan las aplicaciones para realizar sus funciones. Estas clases utilizan las bibliotecas y la máquina virtual Dalvik para trabajar.

- **Aplicaciones:** finalmente ésta es la capa superior en la que aparecen las aplicaciones. Está la aplicación Inicio, que es la que permite ejecutar las demás aplicaciones, las nativas (escritas en C o C++), las administradas (escritas en Java) o los Widgets, que son pequeñas aplicaciones que operan en una pequeña región de la pantalla de la aplicación Inicio.

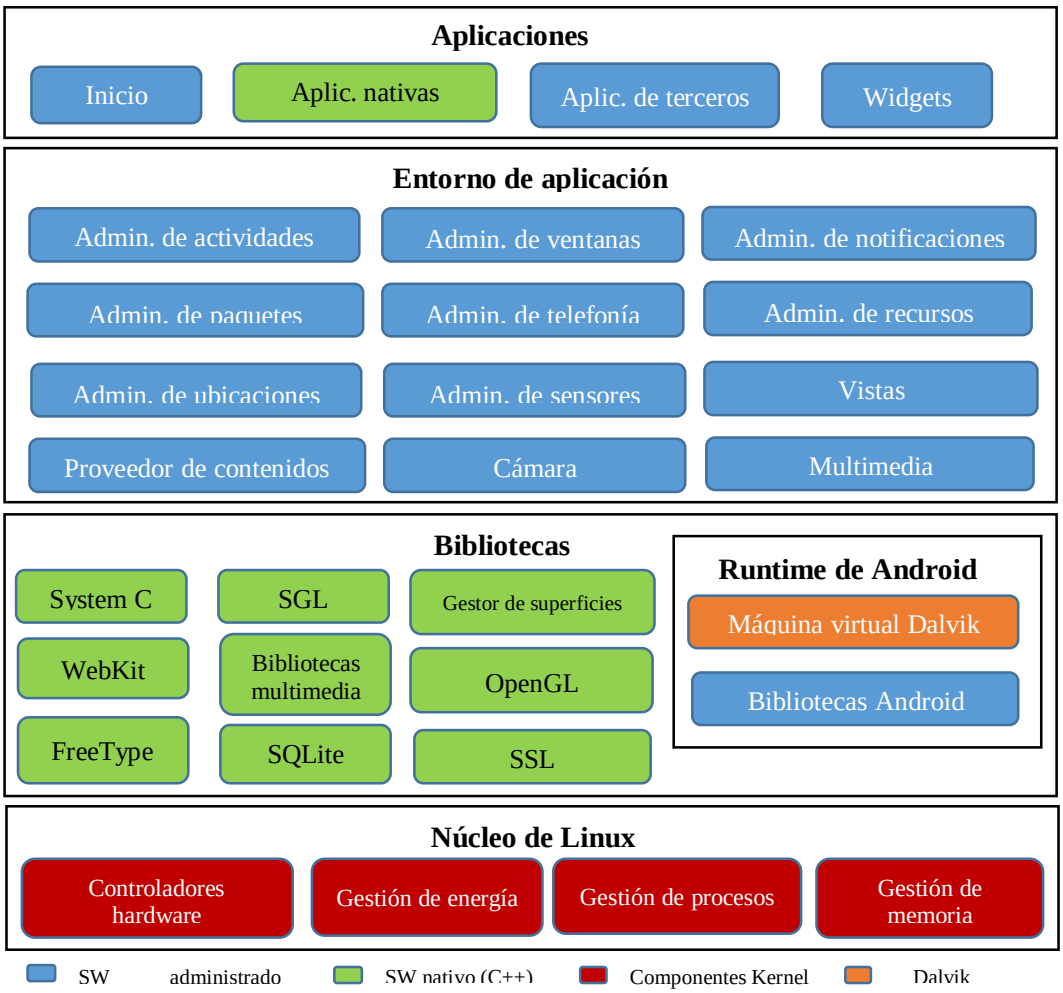



**Figura 2-1: Arquitectura de Android**

#### *2.1.1.3 Versiones*

Existen varias versiones de Android, cada una ha ido corrigiendo errores de las anteriores y añadiendo nuevas funcionalidades. Se muestra una tabla en la que aparecen todas las versiones.

| Versión | Nivel API | Nombre de versión |
|---|---|---|
| Android 4.4 | 19 | KITKAT |
| Android 4.3 | 18 | JELLY_BEAN_MR2 |
| Android 4.2, 4.2.2 | 17 | JELLY_BEAN_MR1 |
| Android 4.1, 4.1.1 | 16 | JELLY_BEAN |
| Android 4.0.3, 4.0.4 | 15 | ICE_CREAM_SANDWICH_MR1 |
| Android 4.0, 4.0.1, 4.0.2 | 14 | ICE_CREAM_SANDWICH |
| Android 3.2 | 13 | HONEYCOMB_MR2 |
| Android 3.1.x | 12 | HONEYCOMB_MR1 |
| Android 3.0.x | 11 | HONEYCOMB |
| Android 2.3.3. 2.3.4 | 10 | GINGERBREAD_MR1 |
| Android 2.3, 2.3.1, 2.3.2 | 9 | GINGERBREAD |
| Android 2.2.x | 8 | FROYO |
| Android 2.1.x | 7 | ÉCLAIR_MR1 |
| Android 2.0.1 | 6 | ÉCLAIR_0_1 |
| Android 2.0 | 5 | ECLAIR |
| Android 1.6 | 4 | DONUT |
| Android 1.5 | 3 | CUPCAKE |
| Android 1.1 | 2 | BASE_1_1 |
| Android 1.0 | 1 | BASE |

**Tabla 2-1: Versiones de Android**

Cuando se ha lanzado una nueva versión, siempre ha sido compatible con las versiones más antiguas. En el caso de modificar alguna funcionalidad, no se eliminan, se etiquetan como obsoletas pero se pueden seguir utilizando.

La versión Android 1.0 no se utilizó comercialmente. Fue a partir de ésta cuando se empezaron a comercializar.

### 2.1.2 iOS

Es un sistema operativo desarrollado por Apple Inc. y destinado solamente a los productos fabricados por Apple, tales como iPhone o iPad. Apple no permite su instalación en hardware de terceros.

Está basado en el Kernel del sistema operativo Mac OS, también creado por Apple.

El lenguaje de programación de las aplicaciones es Objetive-C, un lenguaje creado como superconjunto de C y orientado a objetos.

No es un sistema operativo libre de acceso para los desarrolladores, pero se pueden programar aplicaciones previamente registrándose como desarrollador y abonando una cuota anual. Esta es una de las principales diferencias que tiene con Android.

### 2.1.3 Windows Phone

Sistema operativo desarrollado por Microsoft, sucesor de Windows Mobile y enfocado al mercado de consumo. No es un sistema operativo libre de acceso a los desarrolladores.

La primera versión Windows Phone 7 fue lanzada en el año 2010. Se han lanzado varias actualizaciones hasta llegar a la actualidad en 2013 con la versión Windows Phone 8, versión solamente válida para los nuevos dispositivos. Esto es debido a un gran cambio en el kernel, que lo hace incompatible con los dispositivos basados en versiones anteriores, lo que ha provocado una fragmentación en el mercado de las aplicaciones de este sistema operativo.

Para tener una licencia de desarrollador es necesario suscribirse y abonar una cuota anual, al igual que ocurre con iOS.

El lenguaje de programación utilizado es C#, que es un lenguaje de programación orientado a objetos, basado en C/C++ y desarrollado y estandarizado por Microsoft, o Visual Basic .NET que es otro lenguaje también orientado a objetos.

### 2.1.4 Otros sistemas operativos móviles

Existen más sistemas operativos actualmente en el mercado, pero con una menor cuota de mercado que los tres sistemas operativos explicados anteriormente. Se describen de forma simple a continuación:

**BlackBerry OS**

Sistema operativo desarrollado por RIM y destinado a los dispositivos de la marca BlackBerry y al uso profesional. El código es cerrado, pero puedes ser desarrollador y elaborar aplicaciones, las cuales son programadas normalmente en C o C++.

**Firefox OS**

Sistema operativo basado en Linux y de código abierto. Uno de los más actuales, lanzado en 2013 y destinado a cualquier dispositivo móvil.

**Ubuntu Touch**

Sistema operativo prácticamente nuevo, presentado en 2013, basado en Linux y de código abierto. Fue desarrollado por Canonical Ltd.

**Tizen**

Sistema operativo basado en Linux, de código mixto entre abierto y cerrado. Toma como base MeeGo (otro sistema operativo para dispositivos móviles, mezcla de Maemo de Nokia

y Moblin de Intel), y está patrocinado por Samsung e Intel. Está destinado a múltiples dispositivos como smartphones, tabletas, televisores, servicios de información y entretenimiento en vehículos. Las aplicaciones están basadas en el lenguaje HTML5.

Como se puede observar, existen varios sistemas operativos móviles que se han apuntado a la moda de los smartphones. De momento son tres los que copan la cuota de mercado (Android, iOS y Windows Phone). Lo veremos más detalladamente en el apartado 3.3 de este proyecto.

## *2.2 Aplicaciones educativas similares*

Es conveniente antes de crear una aplicación móvil y subirla a algún mercado, realizar una búsqueda en dichos mercados, para ver si existe alguna del mismo estilo que la que estamos diseñando en este proyecto.

En caso de que existan y que la finalidad sea competir con ellas, habría que realizar comparaciones, y si es posible dar algo diferente en la aplicación que no tengan las demás o mejorar lo ya existente.

En este caso, al ser nuestra primera aplicación, la idea no es competir en ningún mercado. La idea es solamente crear una aplicación con utilidad para alumnos de la Universidad.

En cualquier caso, realizaremos una búsqueda en los mercados de aplicaciones para observar si alguien ha hecho ya alguna aplicación con idéntica utilidad.

Como ya hemos dicho anteriormente, hay dos o tres sistemas operativos que copan el mercado de las aplicaciones, por lo que limitaremos nuestra búsqueda a estos mercados, que son:

**Google Play (Android)**

En la tienda de Android, existen aplicaciones educativas (gratuitas y de pago) relacionadas con la electrónica, tanto digital como analógica.

Aparecen aplicaciones en las que se explican conceptos, se muestran tutoriales (puertas lógicas, semiconductores, teoría de circuitos, mapas de Karnaugh…) o las típicas herramientas tales como tabla de colores de resistencias o conversores de unidades. Otras son del estilo del programa de diseño electrónico Orcad, donde puedes crear circuitos y comprobar su funcionamiento.

Después de búsquedas en este mercado, la aplicación que más relación puede tener con la de este proyecto es:

- **Automata | Comp. Sc. Engg:** aplicación de pago que ofrece un amplio tutorial sobre las autómatas. Dentro del temario incluye temas acerca de máquinas de Moore y de Mealy.

### App Store (iOS)

En la tienda de Apple ocurre lo mismo que en la tienda de Android. Existen aplicaciones explicando conceptos, tutoriales, aplicaciones con herramientas básicas para la electrónica, o aplicaciones de diseño de circuitos. Existen tanto para iPhone como para iPad (teléfono y tableta de la marca Apple).

### Marketplace (Windows Phone)

En la tienda de Windows Phone existen menos aplicaciones comparada con las tiendas de iOs y de Android. Las aplicaciones que hay actualmente con respecto a la electrónica siguen la misma línea que Android e iOS (tutoriales, conceptos, herramientas, diseño).

# 3 Diseño

## *3.1 Propósito de la aplicación*

En este proyecto se va a desarrollar una aplicación móvil destinada a los alumnos de la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior (Universidad Autónoma de Madrid).

Es evidente que los teléfonos inteligentes están muy presentes entre los estudiantes, por lo que se va a aprovechar esta plataforma para mejorar el formato de las guías de problemas de esta asignatura, y por tanto la calidad de la asignatura.

La aplicación ofrecerá una guía de problemas similar a la ofrecida en el formato de papel, pero añadiendo las siguientes mejoras:

- Herramienta de corrección de ejercicios.
- Ayudas parciales.
- Tutorial sobre máquinas de estado Moore.

La aplicación ofrecerá diez ejercicios de circuitos secuenciales y proporcionará un tutorial sobre máquinas de estado de tipo Moore. Se eligen sólo diez pues se busca probar la idea y la tecnología.

El alumno podrá solucionar los ejercicios, y además tendrá la posibilidad de saber si están bien solucionados. Podrá guardar y cargar varias soluciones, además de compartirlas con otros compañeros o con el profesor.

Tras revisar por encima el mercado de las aplicaciones móviles, tal como se describe en el apartado 2 de esta memoria (Estado del arte), no se han encontrado aplicaciones con la misma finalidad que la de este proyecto. Es decir, aplicaciones que ofrezcan ejercicios de circuitos secuenciales, con una herramienta para resolverlos, ayudas parciales y una posterior corrección.

Esto refuerza la idea de desarrollar esta aplicación.

## *3.2 Requisitos de la aplicación*

La aplicación funcionará correctamente si se cumplen los siguientes requisitos:

- Aplicación válida para el sistema operativo Android.
- Soporta versiones de Android desde la 2.2 (Froyo) hasta la 4.4 (KitKat).
- Aplicación válida para todo tipo de tamaños de pantalla en los dispositivos móviles. Esto incluye pantallas desde 2 pulgadas hasta más de 10 pulgadas (teléfonos y tabletas)

- Aplicación válida para densidades de pantalla bajas (ldpi), medias (mdpi), altas (hdpi), extra-altas (xhdpi) y extra-extra-altas (xxhdpi).

## *3.3 Elección del sistema operativo*

Como se ha expuesto en el apartado 2 de la memoria (Estado del arte), existen varias alternativas de sistemas operativos móviles para poder desarrollar aplicaciones.

En este proyecto se ha elegido desarrollar en Android, debido principalmente a dos causas:

1) La primera y más importante, es que Android es un sistema operativo cada vez más presente en los dispositivos móviles.

   Las siguientes figuras muestran la evolución en el tiempo de los sistemas operativos en teléfonos inteligentes y tabletas en España y en el resto del mundo.

   Como se observa, en los últimos 5 años, Android ha ido creciendo y ocupando el primer puesto en los teléfonos móviles en España, y en el resto del mundo.

   Sin embargo, el SO que ha liderado las tabletas ha sido iOS, tanto en España como en el mundo (seguido de Android). Esto es debido a la novedad y éxito de los iPad lanzados en 2010. Por otro lado, se observa una ligera tendencia en Android de alcanzar a iOS.

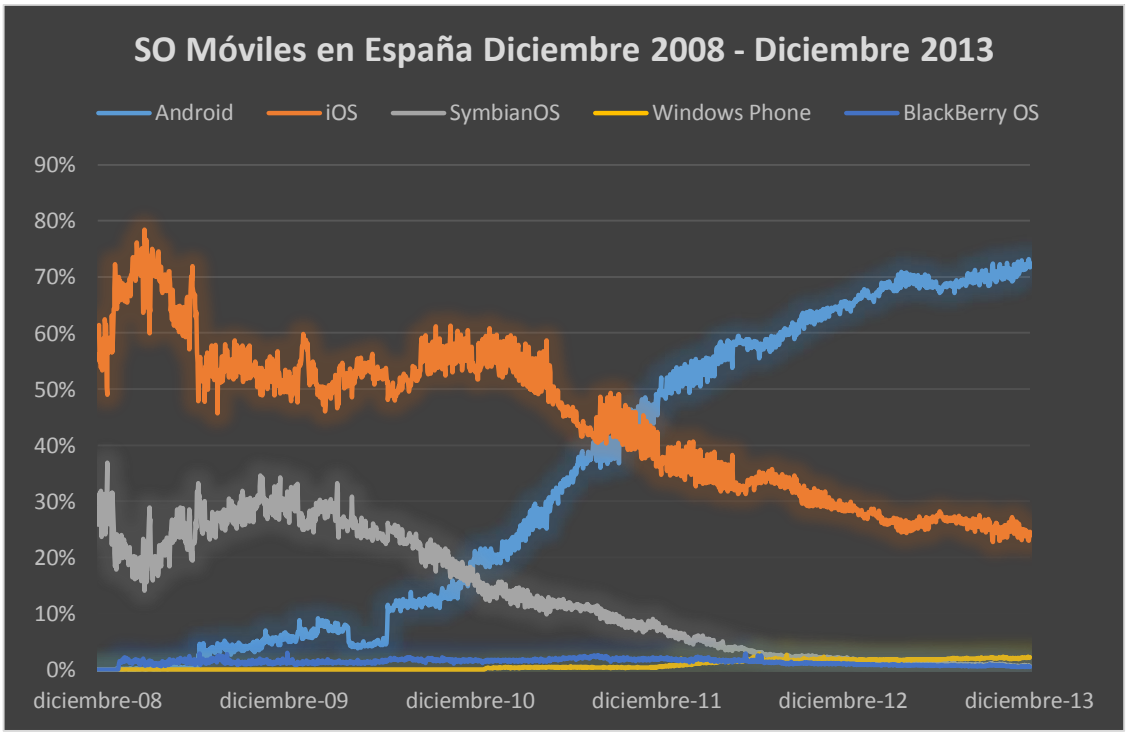



**Figura 3-1: SO Móviles en España Diciembre 2008 – Diciembre 2013**

*Datos sacados de la web http://www.gs.statcounter.com.*

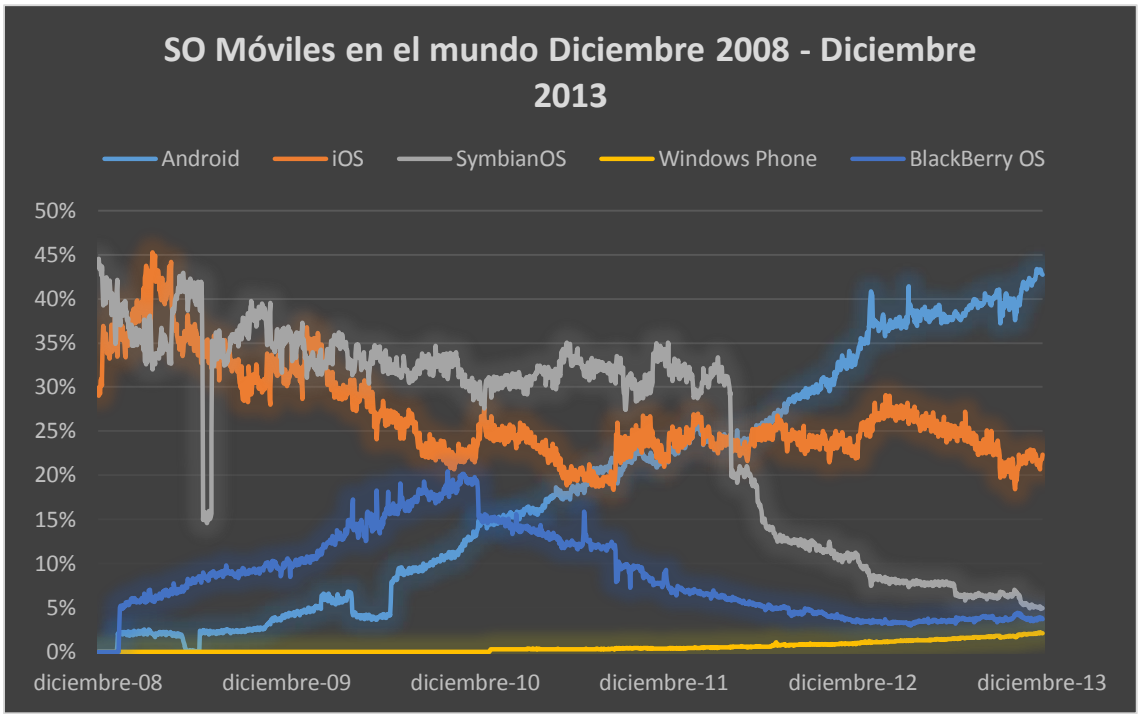



**Figura 3-2: SO Móviles en el mundo Diciembre 2008 – Diciembre 2013**

*Datos sacados de la web http://www.gs.statcounter.com*

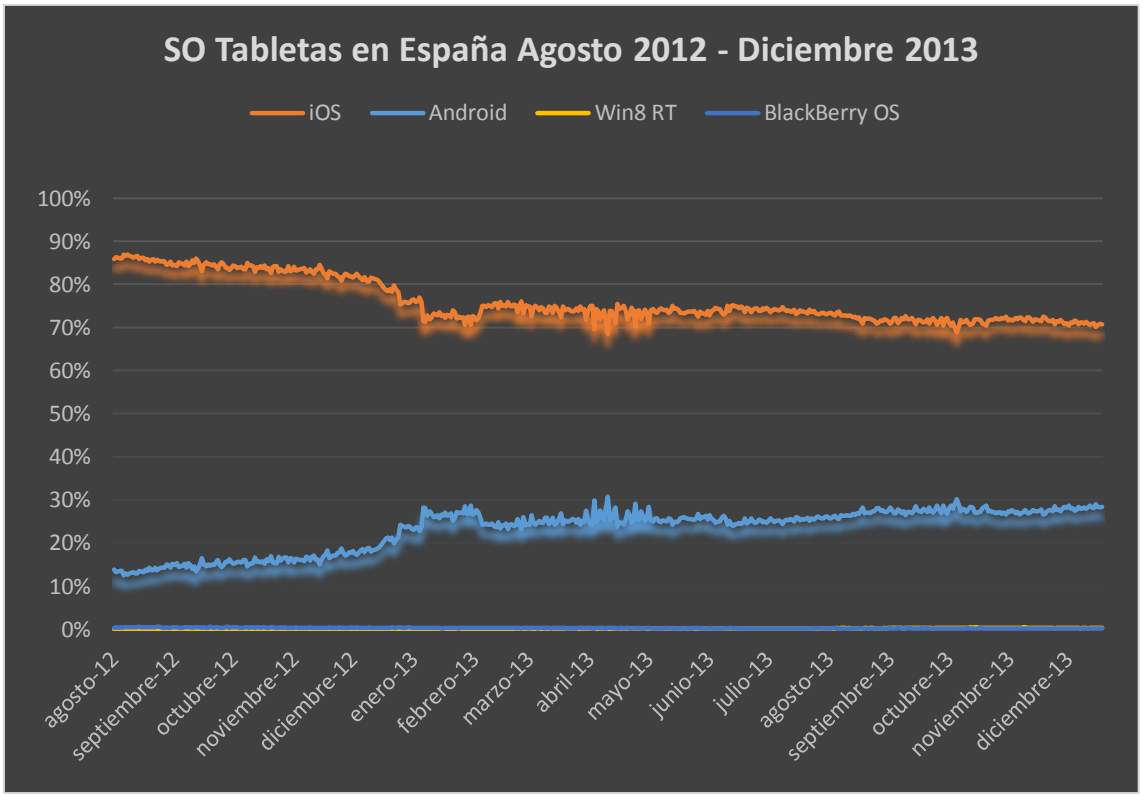



**Figura 3-3: SO Tabletas en España Agosto 2012 – Diciembre 2013**

*Datos sacados de la web http://www.gs.statcounter.com*


**Figura 3-4: SO Tabletas en el mundo Agosto 2012 – Diciembre 2013**

*Datos sacados de la web http://www.gs.statcounter.com.*

2) Es una plataforma libre y de código abierto. Es más sencillo y barato acceder a ser desarrollador en Android. Cualquier desarrollador tiene acceso al código, por lo que existen continuas mejoras y un gran número de desarrolladores que muestran y proporcionan soluciones a cualquier problema que te pueda surgir. Esto facilita en gran manera el desarrollo de aplicaciones.

### 3.3.1 Versiones válidas

La plataforma de desarrollo de Android permite elegir de una manera muy sencilla para qué versiones de Android quieres que funcione tu aplicación. Basta con indicar en un fichero de configuración de la aplicación, llamado "Manifest.xml", la mínima y la máxima versión soportable.

Cualquier función que creen en una versión nueva, no será soportada en versiones antiguas. A su vez, cualquier versión nueva soporta cualquier función existente en versiones antiguas.

Atendiendo al siguiente gráfico proporcionado por Android, se observan los porcentajes de las versiones de Android activas actualmente a 2 de Diciembre de 2013.

| Versión | Nombre | API | Distribución |
|---|---|---|---|
| 2.2 | Froyo | 8 | 1,6 % |
| 2.3.3 – 2.3.7 | Gingerbread | 10 | 24,1 % |
| 3.2 | Honeycomb | 13 | 0,1 % |
| 4.0.3 – 4.0.4 | Ice Cream Sandwich | 15 | 18,6 % |
| 4.1.x | Jelly Bean | 16 | 37,4 % |
| 4.2.x | | 17 | 12,9 % |
| 4.3 | | 18 | 4,2 % |
| 4.4 | KitKat | 19 | 1,1 % |

**Tabla 3-1: Versiones de Android en dispositivos a Diciembre 2013**

*Tabla proporcionada por http://developer.android.com/about/dashboards/index.html*

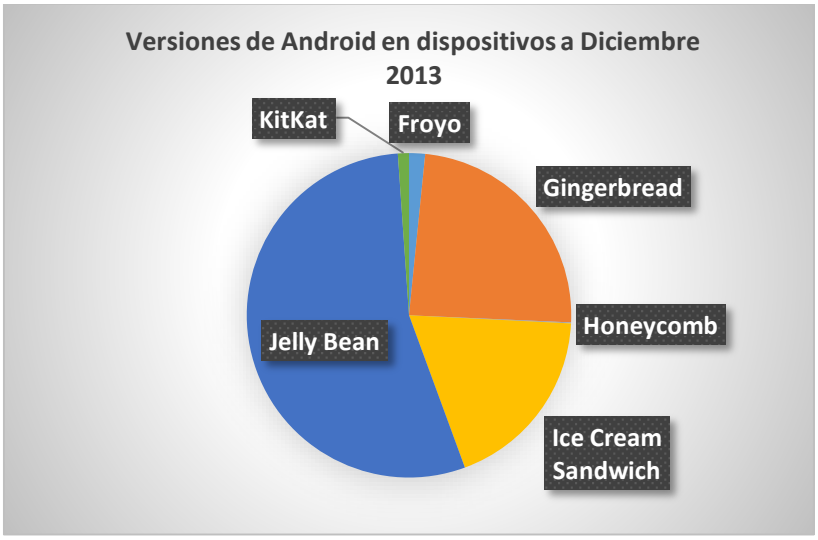



**Figura 3-5: Versiones de Android en dispositivos a Diciembre 2013**

Prácticamente, todos los dispositivos actuales (a Diciembre de 2013) tienen instalada desde la 2.2 en adelante. Por lo tanto, se descarta que la aplicación soporte versiones anteriores a la 2.2.

Es decir, la aplicación soportará desde la versión 2.2 hasta la última versión publicada (a Diciembre de 2013, la versión 4.4). Según salgan nuevas versiones de Android, se irá actualizando la aplicación para que las soporte.

## *3.4 Elección de los tamaños de pantalla válidos*

Actualmente existen multitud de dispositivos Android con diferentes configuraciones de pantalla, como el tamaño y la densidad. Android separa en rangos los diferentes tamaños y densidades de pantalla de la siguiente manera:

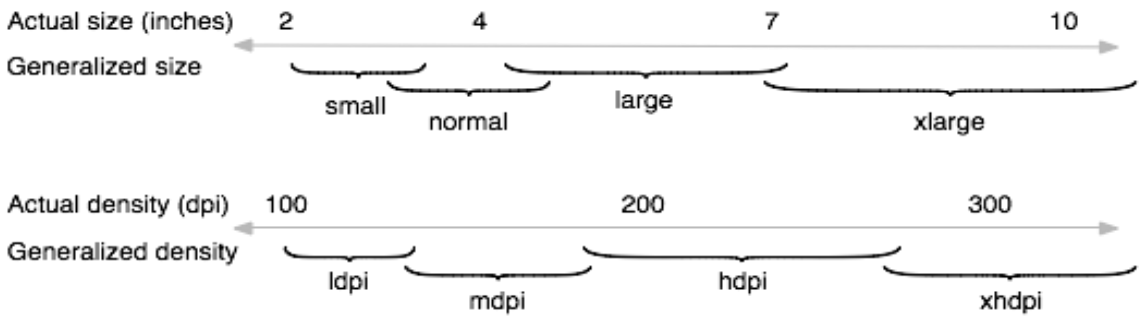



**Figura 3-6: Rangos de tamaño y densidad de pantalla para Android**

*Imagen de http://developer.android.com/guide/practices/screens_support.html*

El tamaño de la pantalla está medido en pulgadas, y la densidad en unidades dpi (puntos por pulgada).

Las siguientes figuras, con datos proporcionados por Android, muestran la distribución de las diferentes configuraciones de pantalla en los dispositivos a Diciembre de 2013.

| | ldpi | mdpi | tvdpi | hdpi | xhdpi | xxhdpi | Total |
|---|---|---|---|---|---|---|---|
| Small | 8,8 % | | | | | | **8,8 %** |
| Normal | 0,1 % | 14,6 % | | 33,4 % | 21,6 % | 9,7 % | **79,4 %** |
| Large | 0,6 % | 3,8 % | 1,4 % | 0,5 % | 0,6 % | | **6,9 %** |
| Xlarge | | 4,5 % | | 0,3 % | 0,1 % | | **4,9 %** |
| Total | **9,5 %** | **22,9 %** | **1,4 %** | **34,2 %** | **22,3 %** | **9,7 %** | |

**Tabla 3-2: Distribución de configuraciones de pantalla en dispositivos a Diciembre 2013**

*Tabla proporcionada por http://developer.android.com/about/dashboards/index.html*


**Figura 3-7: Distribución de tamaños de pantalla en dispositivos a Diciembre 2013**


**Figura 3-8: Distribución de densidades de pantalla en dispositivos a Diciembre 2013**

Con respecto al tamaño de pantalla, predomina la categoría “normal”, a la cual pertenecen los teléfonos móviles convencionales. El resto son teléfonos de pantallas más grandes (cada vez más presentes), de pantallas más pequeñas y tabletas.

En cuanto a la densidad de pantalla hay un reparto algo más equitativo con respecto al tamaño. Hay una distribución muy parecida entre “mdpi”, “hdpi” y “xhdpi”.

La densidad “tvdpi” no es considerada una densidad “primaria”. Está comprendida entre la densidad “mdpi” y la “hdpi” y destinada principalmente a televisiones.

Se va a diseñar la aplicación tanto para móviles como para tabletas. Dado que los móviles están incluidos en tamaños desde “small” hasta “large”, y que las tabletas pertenecen a los grupos “large” y “xlarge”, la aplicación se diseñará para todos los tamaños posibles de pantalla.

La aplicación también mostrará cada imagen adaptada a las diferentes densidades de pantalla “ldpi”, “mdpi”, “hdpi”, “xhdpi” y “xxhdpi”.

## 3.5 Limitaciones de los ejercicios

Los diez ejercicios consistirán en diseñar una máquina de estados a partir de un enunciado. La máquina de estados deberá ser de tipo Moore o Mealy según lo indique el enunciado.

Por cuestiones de simplicidad, se escogerán ejercicios con las siguientes limitaciones:

- Máquinas con tres bits o menos de entrada.
- Máquinas con dos bits o menos de salida.
- Máquinas con cinco estados o menos.
- Solamente se pueden añadir tres valores en cada flecha.

Para enseñar los conceptos, se cree que no es necesario introducir ejercicios más tediosos. Se entiende que un ejercicio es más tedioso si tiene un mayor número de estados, o entradas y salidas con más bits.

Sin embargo no se descarta que en posibles futuras mejoras de la aplicación, se modifiquen estas limitaciones.

## 3.6 Elección del idioma

El idioma elegido para el contenido de la aplicación será el inglés de Estados Unidos.

La aplicación será ofrecida a los alumnos a través de la plataforma de Google Play. Esta plataforma es accesible desde muchos lugares del mundo, por lo que aunque la aplicación esté principalmente diseñada para la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior (Universidad Autónoma de Madrid, España), se ha decidido diseñarla en el idioma más internacional, que es el inglés.

De este modo, la aplicación es útil en cualquier lugar del mundo que tenga acceso a Google Play y con conocimiento de inglés.

Tampoco se descarta que en posibles futuras mejoras se añadan más idiomas a la aplicación.

## 3.7 Módulos de la aplicación

En este apartado se van a exponer de forma resumida todas las funcionalidades de la aplicación.

### 3.7.1 Tutorial de máquinas de Moore

La aplicación ofrecerá un tutorial acerca de las máquinas de estado tipo Moore, con texto desarrollado por Prof. Boemo a partir de sus apuntes de clase.

Será un tutorial en el que se podrá pasar de página hacia delante y hacia detrás, similar a un libro electrónico.

El tutorial podrá ser ampliable en una mejora futura.

### 3.7.2 Ejercicios: Máquina del usuario

Se ofrecerán diez ejercicios que se resolverán diseñando máquinas de estado de tipo Moore y de tipo Mealy. Cada ejercicio tendrá su propio enunciado y la posibilidad de acceder a la máquina del usuario mediante un solo botón.

La máquina del usuario será una pantalla que imitará una máquina de estados. Presentará una máquina Moore o una máquina Mealy dependiendo del diseño que pida el enunciado.

Dispondrá de cinco estados, de los cuales el usuario podrá utilizar los que necesite. El usuario tendrá toda la libertad (dentro de las limitaciones descritas en el apartado 3.5) para resolver el ejercicio. Es decir, podrá utilizar cualquier combinación de estados.

Desde esta pantalla se tendrá acceso con un simple botón a:

- Guardar el ejercicio en memoria.

- Manual de ayuda de la aplicación.

- Borrar las flechas que se necesite: podrá elegir mediante un menú desplegable cuál flecha de las dibujadas quiere borrar.

- Resetear el ejercicio: vuelve al estado inicial la máquina del usuario.

- Corregir el ejercicio: con un simple botón, la aplicación le indicará si la solución es correcta o no.

- Consejos: podrá acceder a la máquina del profesor, la cual proporciona ayudas para la realización del ejercicio.

### 3.7.3 Consejos: Máquina del profesor

La máquina del profesor será una pantalla que simulará también una máquina de estados. Esta máquina de estados no servirá para solucionar el ejercicio, sino que solamente mostrará ayudas y consejos para la resolución del ejercicio.

Estas ayudas y consejos serán accesibles a través de simples botones. Se mostrarán tanto trozos de la resolución total del ejercicio, como textos explicativos para una mejor comprensión del ejercicio.

Intencionadamente, no se pondrán soluciones completas para evitar que se salte el paso de pensar una solución.

Solamente se podrá acceder a la máquina del profesor desde la máquina del usuario.

### 3.7.4 Corregir ejercicios

Existirá la posibilidad de corregir cada uno de los diez ejercicios.

Consistirá en un solo botón perteneciente a la máquina de usuario, que pulsando en cualquier momento indicará “fallo” o “acierto”.

La corrección detectará como “acierto” cualquier combinación de estados de la solución correcta. Esto proporcionará libertad al usuario y no condicionará al enunciado a restringir o fijar los estados a utilizar.

### 3.7.5 Guardar ejercicios

Cada ejercicio tendrá la posibilidad de guardar cualquier cambio realizado. El usuario solamente tendrá que escribir el nombre con el que quiera guardar el archivo. El archivo será guardado en la memoria externa del dispositivo, dentro de una carpeta también creada por la aplicación llamada “SequentialCircuits”.

El archivo guardado contendrá el estado del ejercicio correspondiente en el momento en el que se guardó.

En caso de que el dispositivo no contenga memoria externa (por ejemplo, no disponga de una tarjeta de memoria SD), el ejercicio será guardado en la memoria interna, en el espacio reservado para los datos privados de la aplicación.

La memoria externa también puede ser considerada una partición de la memoria interna, en la cual se pueden guardar datos del usuario. Esto es más usual en los dispositivos más actuales que no disponen de lector de tarjetas de memoria SD.

Esta opción está pensada para que el usuario tenga la opción de poder guardar todas las posibles soluciones que se le ocurran.

### 3.7.6 Cargar y borrar ejercicios

Las soluciones guardadas podrán ser cargadas en la máquina del usuario en el momento que necesite.

La opción de cargar las soluciones del ejercicio será proporcionada dentro del enunciado de cada ejercicio.

Cuando el usuario pulse la opción en cualquier ejercicio, se le proporcionará una lista de las soluciones guardadas de ese ejercicio, tanto en la memoria interna como en la externa. Solamente podrá elegir una solución para cargarla.

En el momento de cargar la solución de un ejercicio, el estado actual de la máquina del usuario será sobrescrito por la solución cargada.

Adicionalmente, se ofrecerá la opción de borrar soluciones guardadas. Esto será posible cuando se muestre la lista de las soluciones guardadas. Solamente bastará con marcar los ejercicios a borrar y pulsar el botón correspondiente. Los archivos serán borrados de la memoria externa o de la memoria interna, según donde se hayan guardado.

Esta opción está pensada para que el usuario tenga la posibilidad de acceder a cualquier solución guardada en cualquier momento de una forma rápida y sencilla.

### 3.7.7 Enviar ejercicios

La aplicación ofrecerá la posibilidad de enviar cualquier solución guardada por correo.

Con un solo botón en el menú principal de la aplicación, se le mostrará una lista con todos los ejercicios guardados, tanto en la memoria interna como en la externa.

El usuario podrá elegir las soluciones que quiera y pulsar el botón de enviar. Entonces aparecerá una alerta en la que el usuario elegirá con qué gestor de correo instalado en el dispositivo quiere enviar el mensaje.

Una vez elegido el gestor de correo, se creará automáticamente el mensaje nuevo, con el "asunto" escrito, las soluciones elegidas adjuntas y un texto explicativo en el cuerpo del mensaje donde se explican las instrucciones para que el receptor pueda cargar esas soluciones. El usuario solamente tendrá que introducir la dirección de correo del receptor.

Esta opción está pensada para facilitar la compartición de las soluciones entre alumnos y profesor.

Las instrucciones para que el usuario receptor cargue los archivos adjuntos en el correo son las siguientes:

- Descargue los archivos adjuntos en el correo. Normalmente serán descargados en la carpeta /sdcard/download del dispositivo.

- Acceda a dicha carpeta con un gestor de archivos. Existen aplicaciones Android que actúan como gestor de archivos.

- Copie o mueva las soluciones descargadas a la carpeta "SequentialCircuits" creada en la memoria externa del dispositivo. Previamente ha debido de instalar y ejecutar la aplicación "Sequential Circuits". Este paso también es posible realizarlo con un gestor de archivos.

- Cuando ejecute la aplicación y vaya a cargar las soluciones, ya estarán disponibles.

### 3.7.8 Manual de ayuda

Se ofrecerá un manual de ayuda de la aplicación. En él se explicará cómo utilizar la máquina del usuario y cómo guardar, cargar o enviar ejercicios.

La ayuda es accesible desde cualquier pantalla de la aplicación. Solamente basta con pulsar el botón de "menú" del dispositivo y ahí aparecerá la opción de ayuda.

### 3.7.9 Menú siempre disponible

Se ofrecerá en cualquier pantalla de la aplicación, a través del botón MENU siempre disponible en cualquier dispositivo, un acceso al manual de ayuda y a la pantalla "Acerca de" de la aplicación.

# 4 Desarrollo

## 4.1 Introducción

En este apartado se va a describir cómo están programadas las partes principales de la aplicación, los problemas surgidos durante el desarrollo de la misma y las soluciones proporcionadas. No se entrará con detalle al código, sino que se explicará de una manera sencilla y lógica la forma en que trabajan las partes principales de la aplicación. Es posible que haya alguna notación referida al código, pero siempre explicada para su buen entendimiento.

## 4.2 Herramientas a utilizar

Para la realización de este proyecto es necesario:

- Ordenador personal
- SDK de Android (viene con plataforma Eclipse y su plug-in ADT Android)
- Java
- Dispositivo Android (teléfono y tableta)

## 4.3 Funcionamiento básico de la aplicación

La aplicación utiliza principalmente "activities" y "views".

Una "activity" es una clase en java que representa cada una de las pantallas de la aplicación. La mayoría de las pantallas de la aplicación están programadas en un fichero XML, también llamado "layout", que contiene los elementos visuales que aparecen en la pantalla (botones, imágenes… llamados "views" ya que heredan de la clase "view").

Las "activities" tienen un ciclo de vida gestionado por defecto por Android. Según se actúe con el dispositivo móvil, la "activity" puede quedarse en pausa, en stop, finalizar, crearse, reiniciarse…Un desarrollador puede definir el comportamiento de la "activity" en cada estado anteriormente indicado, aunque por defecto está gestionado por Android.

Cuando se navega entre las "activities" (pantallas) de una nueva aplicación, al pasar de una a otra, por defecto las "activities" se quedan en estado de pausa o parada en una pila. Si el usuario utiliza el botón BACK del dispositivo, la "activity" más reciente en estado de pausa o parada es la que aparece en pantalla. Es decir, siempre por defecto vuelve a la pantalla anterior.

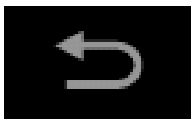


**Figura 4-1: Botón BACK**

Para navegar en nuestra aplicación se utilizan los botones propios de la aplicación (botón de ejercicios, botón de tutorial…). Cada vez que se pulsa uno de estos botones, se inicia una nueva "activity". Para volver hacia atrás, se maneja el botón BACK siempre presente en los dispositivos móviles.

En esta aplicación, el funcionamiento de tener "activities" en pila no interesa. Si el usuario hiciera muchas transiciones seguidas entre "activities" sin finalizarlas, se quedarían todas en pila y debería de pulsar muchas veces BACK para cambiar a otra diferente.

Por lo tanto, cada vez que una primera "activity" llama a otra segunda, se finaliza la primera, para que siempre exista solamente una activa, la segunda.

El botón BACK finaliza la "activity" desde la que se pulsa e inicia la "activity" anterior a ésta.

Resumiendo, solamente hay una "activity" activa en cada momento, y tanto las transiciones "hacia delante" (pulsando cada botón de la aplicación) como las transiciones "hacia detrás" (pulsando el botón BACK) son gestionadas y definidas por nosotros para que exista siempre solamente una "activity" activa.

De esta manera, el usuario sabe que estando en una pantalla, al pulsar el botón BACK siempre va a ir a la anterior, y no va a aparecer ninguna pantalla inesperada.

Existen dos excepciones a esto:

- Cuando se llama desde Ayuda1 a Ayuda2 (ver diagrama de clases en sección 4.15), al no tener más salidas desde Ayuda2, no finalizamos la "activiry" Ayuda1. Por lo tanto, Ayuda1 se queda en pila y al pulsar el botón BACK (en este caso no manejado por nosotros) se vuelve a Ayuda1.

- Cuando se llama desde cualquier "activity" a About (ver diagrama de clases en sección 4.15), al no tener más salidas desde About, no finalizamos la "activity" que ha llamado a About. Por lo tanto, al pulsar el botón BACK (en este caso no manejado por nosotros) se vuelve a la actividad anterior.

## *4.4 Creación de la máquina del usuario (Máquinas Moore y Mealy)*

La máquina del usuario es una de las partes principales de la aplicación. Es la pantalla en la cual el usuario diseña la solución de cada ejercicio.

Se quiere trasladar el diseño en papel de una máquina de estados a un dispositivo móvil. Se entiende como diseño de una máquina de estados a la unión de éstos mediante flechas, indicando entradas y salidas del circuito.

### 4.4.1 Diseños descartados

Antes del diseño definitivo, se pensaron e intentaron dos maneras de realizarlo:

1) Replicar directamente la forma en la que se hace en papel. Se presentaban cinco estados dibujados en una pantalla simulando la máquina de estados. Los estados se unían mediante un trazo libre con el dedo entre un estado y otro. Se creaba una línea con la misma trayectoria indicada con el dedo.

   Esta libertad podía provocar problemas, debido a posibles trazos largos y complejos creados por el usuario.

   Por lo tanto, la opción de dejar el dibujo libre al usuario se descartó.

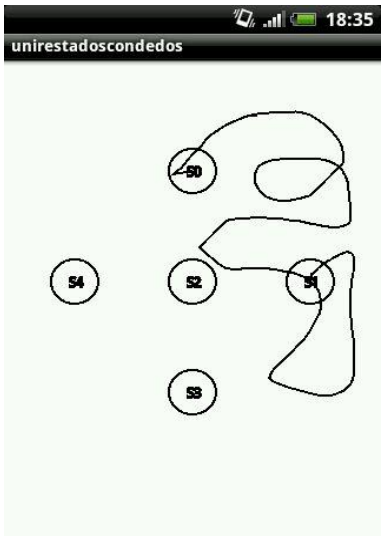



**Figura 4-2: Máquina del usuario trazando flecha con dedo**

2) Para evitar toda esa libertad de dibujo, se decidió dar al usuario la opción de unir los estados pero proporcionándole las flechas. De esta manera el dibujo estaba más controlado.

   La manera de proporcionarle las flechas era mediante imágenes de flechas. Inicialmente, la pantalla mostraba una imagen con cinco estados. Para el dibujo de las flechas, se daba la opción mediante botones desplegables de elegir estado origen, estado destino, entrada y salida. Al pulsar un botón de "Dibujar", una imagen idéntica a la inicial pero con una flecha entre los estados seleccionados se superponía a la anterior. Así continuamente hasta la resolución del ejercicio.

   El inconveniente de esta opción era la cantidad de imágenes a utilizar y almacenar en la aplicación. Esto aumentaba el peso de la aplicación en gran cantidad.

   Por lo tanto se rechazó también esta idea.

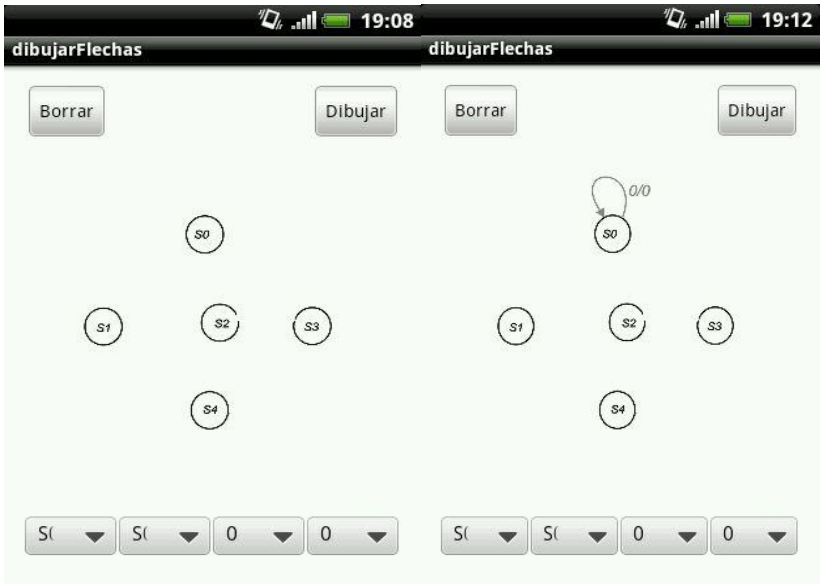



**Figura 4-3: Máquina del usuario mediante imágenes**

### 4.4.2 Diseño definitivo

Finalmente, la máquina del usuario se diseñó de la siguiente manera:

Se crea un lienzo, de la clase "canvas" proporcionado por Android. En él se pueden dibujar, a través de funciones proporcionadas por la clase "canvas", líneas rectas, círculos, figuras geométricas, dibujo libre con el dedo, cualquier tipo de dibujo simulando un papel en blanco y un pincel.

En él se dibujan cinco círculos, numerados de S0 a S4, simulando una máquina de estados.

Para dibujar una flecha entre estados hay que seleccionar un estado origen. Para ello, se pulsa sobre uno de los círculos dibujados obteniendo las coordenadas de éste, previamente diseñadas y conocidas. El estado origen queda guardado a la espera de pulsar en un estado destino.

Una vez que se ha pulsado un estado destino, aparece una alerta con dos editores de texto en los que se introduce la entrada y la salida de esa flecha (en el caso de Moore solamente aparece un editor de texto para la entrada) y un botón para confirmar. La entrada y la salida introducidas quedan guardadas en el programa. Pueden contener 1, 0 o "x" minúscula.

Inmediatamente, con coordenadas origen-destino, entrada y salida guardadas, se dibuja una flecha entre ambos estados con la entrada/salida escrita sobre la flecha, tal y como se hace en papel.

Las flechas han sido previamente diseñadas para ocupar el espacio de la pantalla de una forma óptima y así tener una buena visualización.

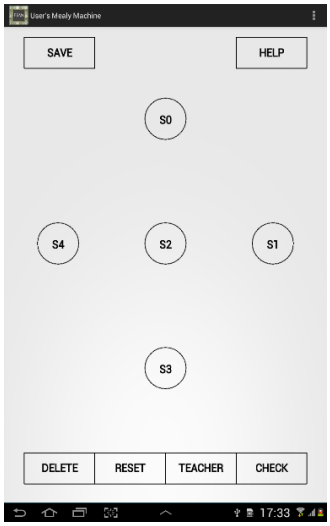



**Figura 4-4: Máquina del usuario definitiva**

### *4.4.2.1 Dibujo de las flechas*

Las flechas, antes de ser dibujadas son definidas mediante "paths" (caminos). En estos "paths", se define a través de coordenadas la trayectoria que debe seguir la flecha. Una vez definido el "path", se procede a "colorear" ese "path".

La trayectoria puede ser recta o curva, dependiendo de cuál sea el camino más óptimo para la buena visualización de la máquina.

Las trayectorias rectas son definidas a través de rectas, con coordenadas (x,y) origen y destino.

Las trayectorias curvas son algo más complejas. Primero se define un "path" en forma de óvalo, el cual se diseña a través de cuatro puntos que son el lado izquierdo, derecho, superior e inferior más salientes. Por lo tanto, con estos óvalos se puede definir la curvatura de la flecha.

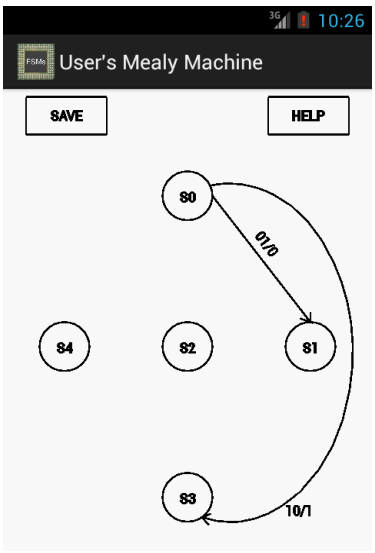



**Figura 4-5: Flecha recta y curva en máquina del usuario**

No se necesita todo el "path" del óvalo para la flecha curva, por lo que se dibuja solamente un tramo del óvalo. Este tramo es definido a través de un ángulo inicial y un ángulo de barrido.

Después de definir el cuerpo de la flecha, se añade al "path" la punta de la flecha. Previamente habiendo guardado la coordenada del extremo de la flecha donde se va a definir la punta, se definen dos rectas partiendo de ese extremo.

Tras definir el "path" completamente, se "colorea" el camino mediante un "pincel".

Finalmente, para que el texto introducido ("entrada/salida" para Mealy o "entrada" para Moore) aparezca junto a la flecha, se utiliza una función que dibuja un texto sobre un path que se haya definido previamente (en este caso la flecha).

#### *4.4.2.2 Creación de botones*

Para crear botones en el lienzo, se opta por dibujar rectángulos (mediante rectas) simulando la forma de un botón, y leyendo si el usuario pulsa dentro del área que definen dichos rectángulos.

Se añade el texto que define al botón dentro del mismo, a través de una función que dibuja texto sobre el lienzo.

Cuando se pulsa sobre el área que define el rectángulo del botón, se llama a la "activity" correspondiente que lo gestiona.

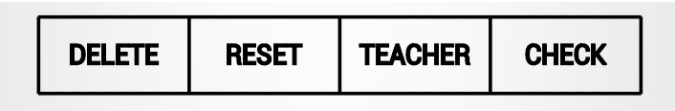



**Figura 4-6: Botones en máquina del usuario**

#### *4.4.2.3 Borrado de flechas*

Cada una de las 25 flechas posibles de dibujar tiene una variable que la habilita o la deshabilita para su dibujo.

Era necesario diseñar una manera para borrar flechas dibujadas por si ocurría alguna equivocación por parte del usuario.

Para ello se habilita el botón de borrado de flechas. Al pulsarlo, aparece una alerta con un listado de las flechas dibujadas hasta el momento. El usuario selecciona las flechas que quiere borrar, lo confirma, y seguidamente se deshabilitan las variables para su dibujo.

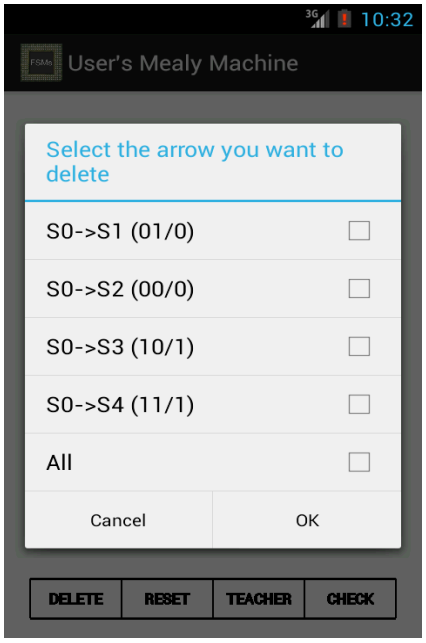



**Figura 4-7: Borrar flechas en máquina del usuario**

#### *4.4.2.4 Salidas en máquina de Moore*

En las máquinas de Moore, la salida está definida en cada estado. Para solucionar esto, se diseñó que el usuario realizara una pulsación larga en cualquier estado, apareciera una alerta y se escribiera la salida.

Para ello, se inicializa un temporizador en el momento en que se pulsa en un estado. Cuando el usuario levanta el dedo y el temporizador supera un valor razonable de tiempo (fijado en 1 segundo), sale la alerta para cambiar la salida.

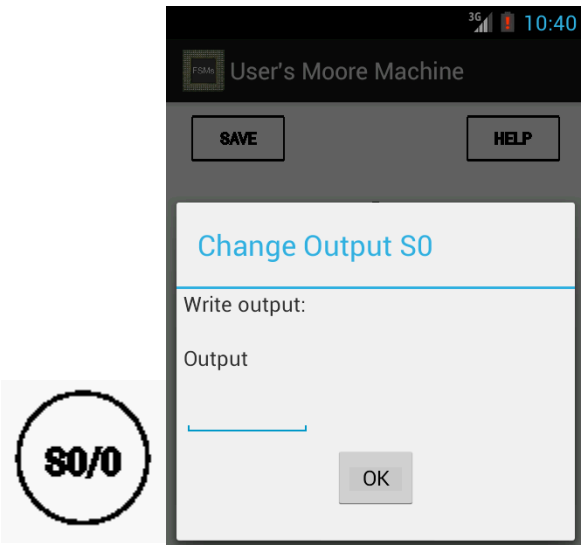



**Figura 4-8: Cambiar salida en estado Moore**

## 4.5 Creación de la máquina del profesor

Se quiere tener una máquina de estados en la cual el usuario diseña su solución, y otra máquina de estados en la que se muestran las ayudas de cada ejercicio.

No era viable mostrar las ayudas directamente en la máquina del usuario ya que podían dar lugar a confusión las flechas de ayuda sobre las flechas del usuario.

Para esto, se habilita un botón, llamado TEACHER, que accede a la máquina del profesor.

Esta máquina del profesor muestra también un lienzo "canvas". En él se dibujan los cinco estados, pero se deshabilita la opción de dibujar flechas. Las flechas que aparezcan en esta máquina serán las correspondientes a cada ayuda.

Se habilitan los botones para las ayudas. Los botones son diseñados de la misma manera que en la máquina del usuario. Al pulsarlos, muestran una alerta con un texto explicativo y se dibujan las flechas correspondientes en la máquina del profesor.

De esta forma, la máquina del profesor solo muestra ayuda, no es posible diseñar nada.

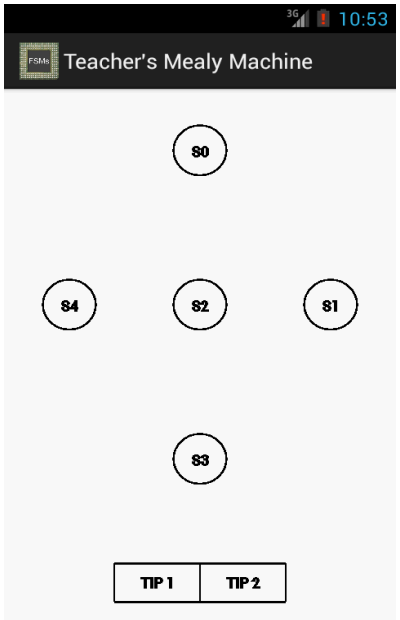



**Figura 4-9: Máquina del profesor**

## 4.6 Guardar y compartir datos entre actividades

La forma de compartir datos entre actividades, o guardar datos permanentes en el dispositivo móvil se realiza mediante las "shared preferences".

Los datos se guardan en un fichero XML ubicado en la parte de datos exclusiva y reservada para la aplicación, en la memoria interna. Los datos se guardan con el dúo etiqueta-valor.

Este fichero es accesible desde cualquier actividad de la aplicación. Aunque se salga de la aplicación o se apague el dispositivo, el fichero no se borra y los datos pueden ser consultados.

Pueden ser borrados desinstalando la aplicación, o manualmente desde el gestor de aplicaciones.

Un ejemplo de utilización de las "shared preferences" en nuestra aplicación es el siguiente:

La máquina del usuario en cada ejercicio se queda guardada con la última versión que el usuario ha dejado. Es decir, si el usuario pulsa por ejemplo el botón BACK desde la máquina del usuario de cualquier ejercicio, el estado en el que la ha dejado sigue intacto cuando vuelve a ese mismo ejercicio.

Cada vez que se dibujan nuevas flechas o modificaciones, se guarda el estado de la máquina en las "shared preferences". Cada vez que se entra en la máquina del usuario, se carga el último estado.

De esta forma, el usuario puede consultar el enunciado las veces que quiera, cambiar de ejercicio, consultar el tutorial, salir de la aplicación…y el ejercicio sigue intacto cuando vuelve a retomarlo.

### 4.6.1 Opciones descartadas

Se barajaron otras opciones como son las bases de datos SQLite o el paso de parámetros entre actividades.

El paso de parámetros entre actividades suponía una carga en el código bastante más compleja, por lo que se descartó.

El uso de una base de datos era una opción muy buena. Pero debido a que no se iban a guardar una gran cantidad de datos, se descartó esta opción. Además, se evitan las consultas a la base de datos.

Por lo tanto, la opción que más se adecuaba a lo que pedíamos era las "shared preferences".

## *4.7 Guardar ejercicios*

Para que el usuario guarde varias soluciones de un mismo ejercicio y después las cargue y las comparta, se decidió hacerlo mediante archivos de texto plano.

Guardar el ejercicio significa guardar todas las flechas dibujadas en la máquina del usuario del ejercicio. Esto conlleva guardar todas las variables que indiquen que esa flecha está dibujada y el texto que va con ellas. En caso de una máquina de Moore, se guarda también la salida configurada en cada estado.

Los ejercicios por defecto se guardan en una carpeta llamada "SequentialCircuits" creada en la raíz de la memoria externa (puede ser una tarjeta SD o una partición de la memoria interna

dedicada a guardar datos). Cada solución guardada lleva una cabecera que indica de qué ejercicio se trata.

En caso de no estar disponible la memoria externa (no hay tarjeta SD, por ejemplo), los archivos se guardan en la memoria interna. Se guardan en la parte reservada para datos de la aplicación SequentialCircuits, sólo accesible por ella.

## *4.8 Cargar ejercicios*

Para cargar los ejercicios guardados, es necesario leer todas las variables guardadas en el fichero, y después guardarlas en las "shared preferences".

Así, nada más entrar a la máquina del usuario, se cargan los datos de las "shared preferences" y el usuario puede ver la solución cargada.

## *4.9 Enviar ejercicios*

Para que el usuario comparta sus soluciones con otros compañeros o con el profesor, se decidió habilitar la opción de enviar por correo cualquier fichero de solución guardado en la memoria externa.

Una vez pulsado el botón de enviar ejercicios en el menú principal de la aplicación, se leen todos los ficheros de solución guardados en memoria interna y externa, y se muestran en una lista.

Cuando el usuario elige los ficheros a enviar y confirma el envío, se prepara un correo nuevo con los ficheros adjuntos. Se crea un "chooser" para que el usuario elija con qué gestor de correo quiere enviarlo.

No es posible adjuntar en el correo los ficheros guardados en la memoria interna. Esto es debido a que están guardados en un lugar de la memoria interna que sólo tiene permiso de acceder la aplicación Sequential Circuits.

La solución a esto sería utilizar los "Content provider", mecanismo para compartir datos entre aplicaciones. Se ha dejado esta opción para explorar en aplicaciones futuras.

## *4.10 Sistema de corrección*

El sistema de corrección de la aplicación es una parte clave.

Dejar total libertad al usuario para que solucione el ejercicio con los estados que quiera, da lugar a múltiples soluciones. Esto hace que el sistema de corrección no sea una simple comparación entre los datos introducidos y una solución correcta.

No era conveniente hacer múltiples comparaciones entre los datos introducidos y todas las posibles soluciones. Quedaría un código muy rústico y largo.

Por lo tanto, tras pulsar el botón CHECK, el sistema de corrección funciona de la siguiente manera:

1) Se obtienen los estados que ha utilizado el usuario para la solución. Un estado ha sido utilizado si ha salido o entrado alguna flecha de él.

2) Se obtienen las flechas (texto entrada/salida en Mealy o entrada en Moore) que entran en cada estado utilizado por el usuario.

3) Se obtienen las flechas (texto entrada/salida en Mealy o entrada en Moore) que salen de cada estado utilizado por el usuario.

4) Se tienen entonces los estados utilizados, cada uno con todas sus flechas salientes y entrantes como identificación. En caso de ser una máquina de Moore se obtienen también las salidas de cada estado.

5) En la aplicación se tiene guardada una solución correcta de cada ejercicio. Es decir, se tienen las flechas entrantes y salientes de cada estado de la solución correcta.

6) Se compara uno de los estados de la solución del usuario con todos los estados de la solución correcta, para ver si coincide con alguno. En caso afirmativo, se apunta ese estado como correcto.

7) Se hace lo mismo con todos los estados de la solución del usuario. En caso de que todos los estados del usuario sean correctos, la solución por tanto es correcta.

De esta manera se abarcan todas las posibles soluciones que puede diseñar el usuario, dejando así total libertad para la realización del ejercicio.

## *4.11 Inserción del tutorial*

El tutorial que ofrece la aplicación consiste en mostrar imágenes en formato ".png" una tras otra. La forma de realización para que simule un libro es la siguiente:

1) Se vuelve a colocar un lienzo en blanco "canvas" sobre la pantalla.

2) Las imágenes ".png" se convierten a mapa de bits mediante una función proporcionada por Android.

3) El mapa de bits de la primera imagen del tutorial se dibuja sobre el lienzo, por lo que al entrar en el tutorial se ve la primera página.

4) Se definen dos zonas de la pantalla como sensibles al toque con el dedo para pasar página hacia delante y hacia atrás. Esas zonas son el 30% del lado derecho de la pantalla (para pasar hacia delante la página) y el 30% del lado izquierdo de la pantalla (para pasar hacia atrás).

5) Cada vez que se toca con el dedo en una de esas zonas, un contador maneja en qué página está, ésta se convierte a mapa de bits y se muestra por pantalla.

## *4.12 Adaptación a todas las pantallas*

Para que la aplicación tenga una visualización óptima en todos los dispositivos Android, es necesario realizar los siguientes diseños:

### 4.12.1 Adaptación de los layouts

Para que cada layout (pantalla) de la aplicación tenga una buena visualización en las diferentes pantallas de los dispositivos, Android proporciona una manera sencilla de diseño.

Solo basta con crear carpetas con el nombre de "layout-tamaño de pantalla", donde "tamaño de pantalla" puede ser "small", "normal", "large" y "xlarge". Dentro de cada carpeta se guardan los layouts, previamente diseñados, para una buena visualización en cada uno de los tamaños de pantalla.

Se realizan cambios como el tamaño de la letra, la posición de los botones, el tamaño de los botones o márgenes. Los diferentes tamaños de pantalla son simulados mediante una herramienta proporcionada por Android, que emula un dispositivo móvil de cualquier tamaño que se elija. Por lo tanto, todos los layouts de la aplicación han sido adaptados para cualquier tamaño de pantalla.

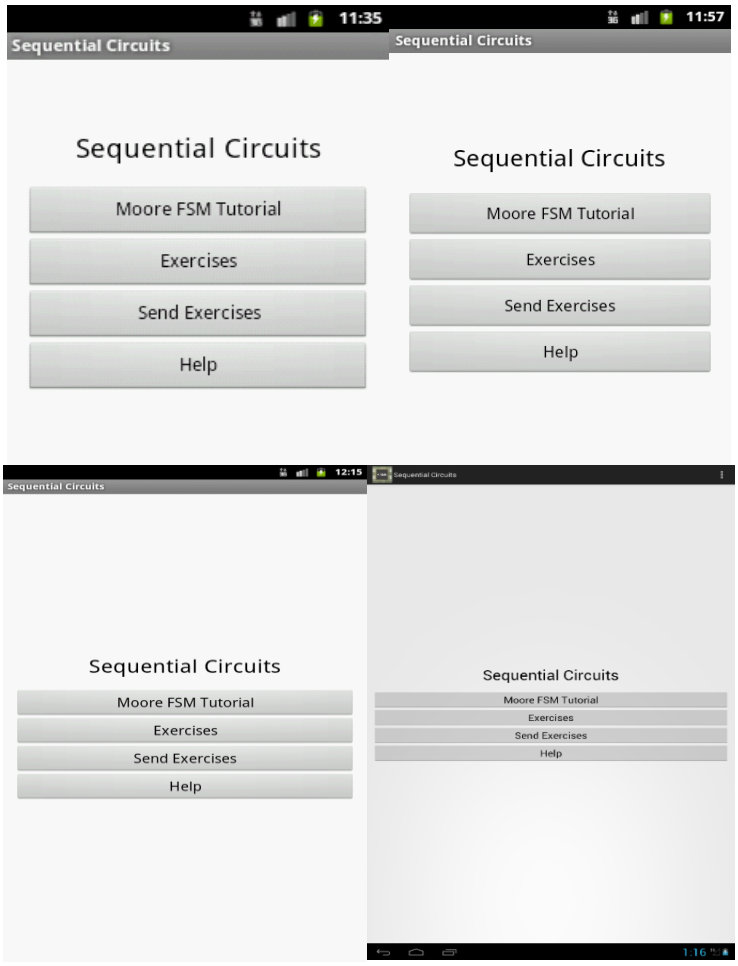



**Figura 4-10: Layouts small y normal (arriba), large y xlarge (abajo)**

Para la adaptación de las imágenes, también se crean carpetas con el formato "drawable-densidad de pantalla", donde "densidad de pantalla" puede ser "mdpi", "hdpi", "xhdpi" o "xxhdpi". En cada una de estas cuatro carpetas se guardan todas las imágenes que aparecen en la aplicación. Según la densidad del dispositivo en el que se ejecuta, obtendrá las imágenes ya adaptadas de su carpeta correspondiente.

Los dispositivos que tengan una baja densidad ("ldpi") obtendrán las imágenes de la carpeta "drawable-mdpi", ya que no existe una diferencia grande de calidad entre ambas. Así se evita crear otra carpeta con todas las imágenes que aumenta el tamaño de la aplicación.

### 4.12.2 Adaptación de la máquina del usuario

En cambio, al ser la máquina del usuario un lienzo "canvas" y no un layout, no es posible incluirlo en ninguna carpeta de layouts.

Al tratarse de dibujos mediante coordenadas, no se puede dibujar con coordenadas constantes si se quiere adaptar a cualquier pantalla.

La solución que evitó este problema fue dibujar con respecto a una referencia. Esto es, se dibuja la máquina del usuario en una pantalla de un dispositivo considerado como referencia. Después se diseña un dibujo genérico, válido para todas las pantallas, referenciado al primero.

La forma de hacerlo es la siguiente:

- Se diseña la pantalla de la máquina del usuario en el dispositivo referencia. Se guardan todas las medidas (en píxeles), incluidas el ancho y el alto de la pantalla.

- Se crean constantes de adaptación para todas las medidas, como por ejemplo las horizontales y las verticales:

$$cte1 = \frac{medida\_horizontal_{referencia}}{ancho\_pantalla_{referencia}}$$

$$cte2 = \frac{medida\_vertical_{referencia}}{alto\_pantalla_{referencia}}$$

- Finalmente, se lee el ancho y el alto de la pantalla (en píxeles) del dispositivo en el que se está ejecutando la aplicación, y se calculan las medidas finales referenciadas, como por ejemplo las horizontales y verticales:

$$medida\_horizontal_{dispositivo} = cte1 * ancho\_pantalla_{dispositivo}$$

$$medida\_vertical_{dispositivo} = cte2 * alto\_pantalla_{dispositivo}$$

Con estas medidas, en cualquier dispositivo los dibujos de la máquina del usuario se verán similares pero proporcionados a la pantalla de referencia.

El diseño de la máquina del profesor es el mismo que el de la máquina del usuario.

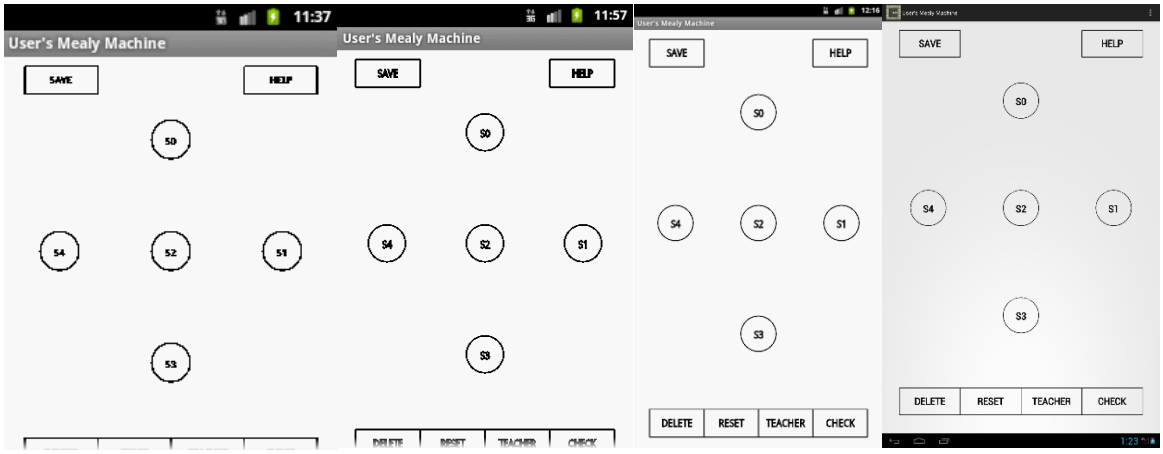



**Figura 4-11: Máquina del usuario small, normal, large y xlarge (de izqda. a dcha.).**

#### *4.12.2.1 Scroll en la máquina del usuario*

En un principio, la máquina del usuario se diseñó para que tuviera una buena visualización en cualquier pantalla sin tener que hacer "scroll" (desplazamiento hacia arriba o hacia abajo de la pantalla).

Tras probar en diferentes dispositivos, se observó que en diferentes versiones de Android, el ancho de la barra de título (barra situada en la parte superior de la pantalla) se hacía más grande. Esto implicaba que la coordenada (0,0) (esquina superior izquierda del lienzo "canvas") bajase con respecto a la pantalla de referencia, por lo que la máquina del usuario se cortaba en la parte inferior de la pantalla.

Para solucionar este problema se decidió habilitar "scroll" en la máquina del usuario, y así evitar cualquier problema de este tipo. El scroll indica la coordenada en el eje "y" hasta donde puede bajar la pantalla.

En nuestro caso, se puso como límite inferior de scroll, la coordenada inferior de la barra de botones, más un margen.

Por lo tanto, en cualquier dispositivo, si la coordenada inferior de la barra de botones no cabe en la pantalla, se podrá utilizar el "scroll" para bajar y ver la máquina completa.

Aprovechando esta situación, se decidió bajar la posición de la barra de botones en la pantalla de referencia, para "descomprimir" por la parte inferior el dibujo de las flechas.

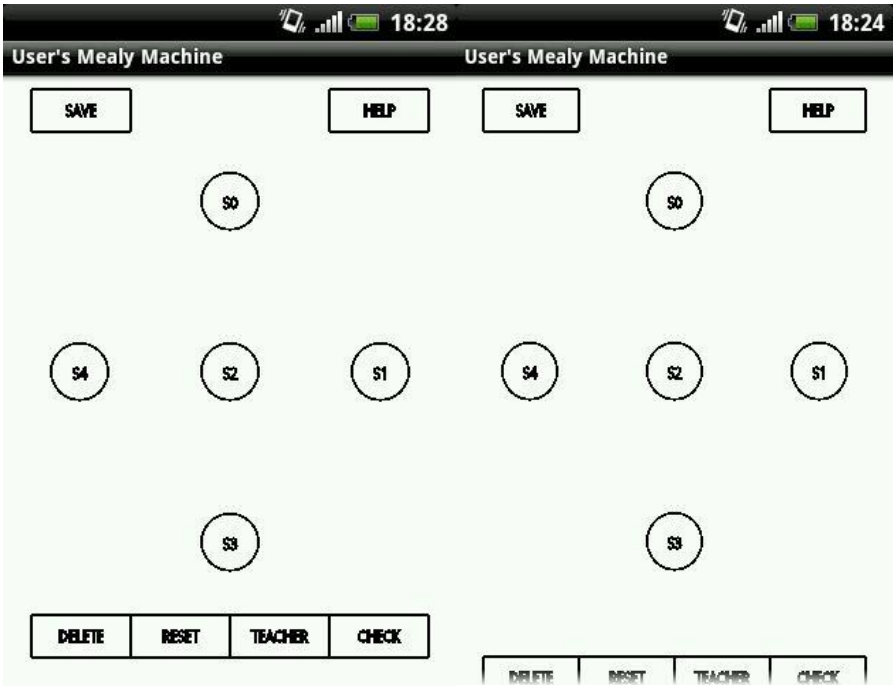



**Figura 4-12: Máquina del usuario en pantalla de referencia sin scroll (izqda.) y con scroll (dcha.)**

### 4.12.3 Adaptación de las imágenes del tutorial

Si mostramos las imágenes tal y como se diseñaron, existirán pantallas en las que se verán mal. Pantallas que tienden a ser cuadradas (relación de imagen muy cercana a 1) hacen que las fotos (todas con una relación de imagen de 1.428) no se vean completamente de altura.

Para solucionar esto, se lee primero la relación de la pantalla en la que se ejecuta la aplicación. Hay que dejar un margen de seguridad entre la relación de la imagen y la relación de pantalla para que no esté muy ajustada la imagen a los bordes de la pantalla.

Por lo tanto, si la relación de la pantalla es mayor que 1.52 (1.428 más un margen de seguridad), la imagen se mostrará tal y como es, ya que no habrá problemas para su visualización.

En cambio, si la relación de la pantalla es menor a 1.52, hay que modificar el tamaño de la imagen para que quepa en la pantalla. Por lo tanto se reduce el tamaño del ancho y del alto en un 25%.

El valor de 1.52 se decidió como límite entre una resolución baja y una resolución media-alta.

El límite del 25% se ha decidido probando con dispositivos con muy baja resolución (320x240 ~ 1.33 de relación de pantalla).

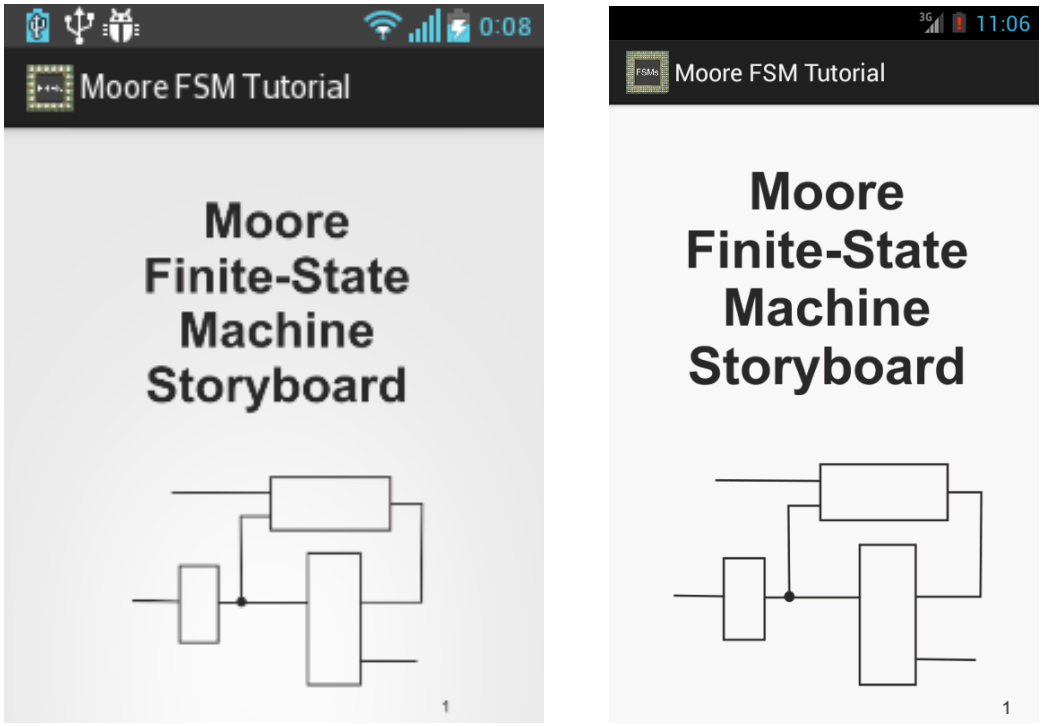



**Figura 4-13: Pantalla con relación menor que 1.52 (izqda.) y mayor que 1.52 (dcha.)**

## *4.13 Instalación en el dispositivo*

El lugar de instalación de la aplicación al descargarse está configurado como "auto".

Esto significa que al obtenerla desde Google Play, la aplicación se instalará, dependiendo de ciertos factores detectados por Android, en la memoria externa o en la interna.

Además, incluye la posibilidad de cambiar el lugar de instalación (entre interna y externa) a través del gestor de aplicaciones de Android.

Cuando la aplicación es instalada en la memoria externa, el fichero ".apk" se guardará en la memoria externa, pero los datos privados de la aplicación serán guardados en la memoria interna.

## 4.14 Menú siempre disponible

En cada pantalla de la aplicación, está siempre disponible el botón menú en el que se ofrece:

- Help: ayuda sobre el funcionamiento de la aplicación
- About: información del desarrollador

Esto es posible gracias a una función que maneja el botón de menú siempre presente en los dispositivos móviles de varias maneras, como las siguientes:

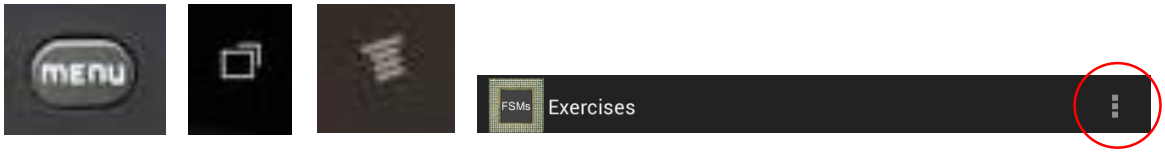



**Figura 4-14: Botón Menú**

Al pulsar este botón, saldrá un menú desplegable con diferentes opciones (en nuestro caso Help y About).

Las opciones que aparecen en el desplegable son editadas en un fichero XML. En la aplicación existe un "menu.xml" con Help y About, y un "menu2.xml" solamente con About (para cuando estemos en la pantalla de Help, no es necesario volver a mostrar Help).

Desde la pantalla de About no existe la opción de un menú.

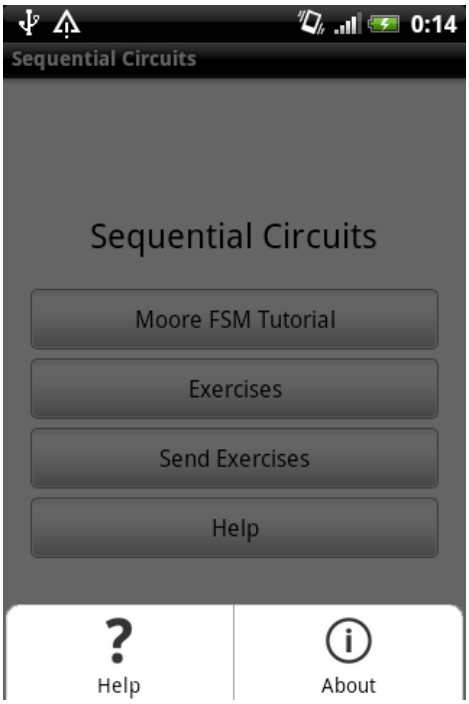
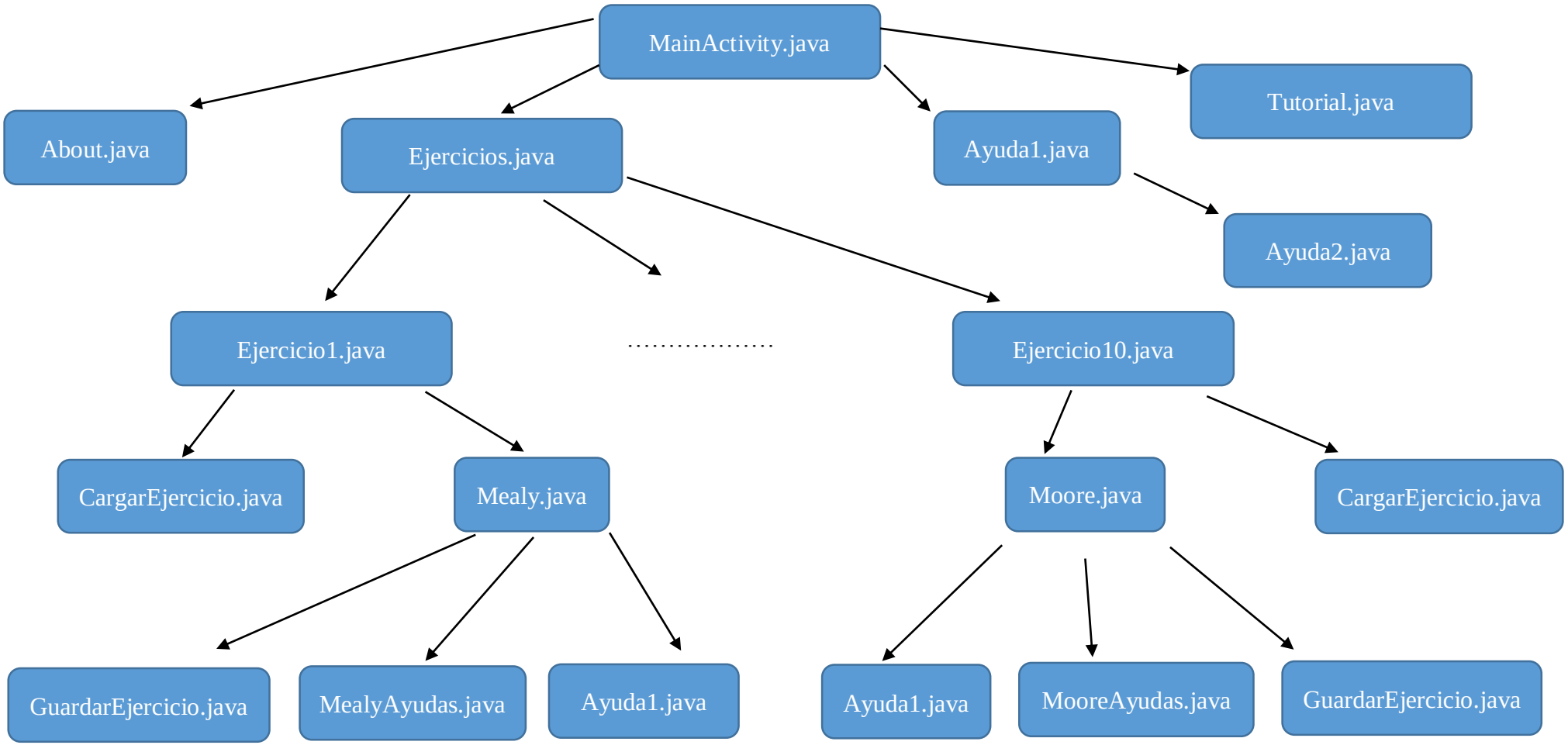



**Figura 4-15: Botón Menú pulsado**

### *4.15 Diagrama de clases del programa*



***MainActivity.java***
Pantalla inicial de la aplicación desde donde que se puede acceder a

- Tutorial
- Ejercicios
- Enviar Ejercicios: función implementada en MainActivity.java
- Ayuda

***About.java***
Pantalla que muestra el "Acerca de" de la aplicación.

***Tutorial.java***
Esta clase gestiona la visualización del tutorial.

***Ayuda1.java***
Muestra el menú de ayuda y gestiona el acceso a cada una de las opciones del menú.

***Ayuda2.java***
Muestra las opciones del menú de ayuda.

***Ejercicios.java***
Pantalla que lista los diez ejercicios disponibles, con botones para acceder a cada uno.

***EjercicioX.java***
Muestra el enunciado de cada ejercicio, la opción de cargar el ejercicio y la opción de resolverlo con la máquina del usuario.

***Mealy.java***
Máquina del usuario de tipo Mealy. Sirve para resolver el ejercicio. Desde ella se pueden hacer las siguientes tareas:

- Borrar flechas: función implementada en Mealy.java
- Resetear: función implementada en Mealy.java
- Acceso a la máquina del profesor (consejos)
- Corregir el ejercicio: función implementada en Mealy.java
- Guardar el ejercicio
- Ayuda

***Moore.java***
Máquina del usuario de tipo Moore. Sirve para resolver el ejercicio. Desde ella se pueden hacer las siguientes tareas:

- Borrar flechas: función implementada en Moore.java
- Resetear: función implementada en Moore.java
- Acceso a la máquina del profesor (consejos)
- Corregir el ejercicio: función implementada en Moore.java
- Guardar el ejercicio
- Ayuda

***CargarEjercicio.java***
Alerta que muestra los ejercicios disponibles para cargar y los carga.

***GuardarEjercicio.java***
Alerta que permite poner un nombre al archivo para guardar y lo guarda.

***MealyAyuda.java***
Máquina del profesor Mealy. Muestra una máquina de estados Mealy y botones con consejos.

***MooreAyuda.java***
Máquina del profesor Moore. Muestra una máquina de estados Moore y botones con consejos.

# 5 Integración, pruebas y resultados

## *5.1 Publicación en Google Play*

Una vez terminada la aplicación, se han realizado los siguientes pasos para la publicación en el mercado de aplicaciones Google Play.

1) Crear una cuenta como desarrollador en Google, abonando 25 $ e introduciendo una serie de datos. He aquí algunos importantes:

    - ***Nombre de desarrollador:*** DSLab UAM.
    - ***Sitio web:*** http://arantxa.ii.uam.es/~euroform_dslab/android.htm
    - ***Correo contacto:*** dslab.uam@gmail.com

2) Añadir nueva aplicación. Antes de la publicación, hay que seguir una serie de pasos de configuración e introducción de datos de la aplicación.

    - ***Ficha de la aplicación***
      Se introducen una serie de datos que aparecerán en la ficha de la aplicación, publicada en Google Play. La ficha se puede realizar en varios idiomas. En nuestro caso, se ha realizado en español y en inglés. He aquí algunos datos importantes:

        - ***Nombre de aplicación:*** Sequential Circuits.
        - ***Categoría de aplicación:*** Educación.
        - ***Nivel de madurez:*** Alto.
        - ***Descripción de la aplicación:*** breve descripción que resume la funcionalidad de la aplicación. Se introducen palabras clave, referentes a la aplicación, para aparecer en más búsquedas.
        - ***Capturas de pantalla:*** Se incluyen capturas de pantalla de la aplicación de teléfonos móviles, tabletas de 7 y 10 pulgadas. Esto sirve para que al ver la ficha de la aplicación, se compruebe cómo sería la visualización de la aplicación en dichos dispositivos.
        - ***Icono de la aplicación:*** se incluye el icono que aparecerá junto al nombre de la aplicación en la ficha.

    - ***Precio y distribución:***

        - Se configura si la aplicación es gratuita o de pago. En nuestro caso, será gratuita.
        - Se distribuye en todos los países posibles. Estos son 139 países + resto del mundo.

- ***Fichero APK***
  Desde la plataforma de programación Eclipse, se crea el fichero ".apk", que es subido a la plataforma. Antes de subir el fichero, se confirma que funciona instalándolo en un dispositivo de prueba.

3) Una vez subido el ".apk" y confirmando que se desea publicar la aplicación, tarda alrededor de 2 horas en publicarse.

4) Ya publicada la aplicación, se confirma que la descarga y la instalación en un dispositivo de prueba es correcta. La ficha de la aplicación es: https://play.google.com/store/apps/details?id=com.SequentialCircuits&hl=es

## *5.2 Correcciones después de la primera publicación*

Actualizar la aplicación es muy sencillo. Basta con modificar lo necesario en el código, crear otro ".apk" nuevo y volver a subirlo desde la cuenta de desarrollador.

Tardará alrededor de otras 2 horas en publicarse la nueva versión de la aplicación. Los dispositivos avisarán a los usuarios de que hay una nueva versión y según estén configurados se actualizará automáticamente o no.

Después de la primera publicación el día 16/12/2013, se han detectado errores en la aplicación y se ha tenido que actualizar 5 veces en 8 días:

- 1º corrección: algunas soluciones de ejercicios no contempladas (solucionado).
- 2º, 3º, 4º y 5º corrección: mejoras en traducción al inglés, erratas en enunciados y textos.

## *5.3 Estadísticas y datos proporcionados por Google Play*

La consola de desarrollador proporciona datos acerca de las descargas de la aplicación subida. Al momento de escribir esta memoria los resultados son los siguientes:

- **Instalaciones totales:** instalaciones desde la primera publicación hasta la fecha indicada. Sigue una tendencia lineal, con una media de aproximadamente 6 instalaciones y 2 desinstalaciones por día.

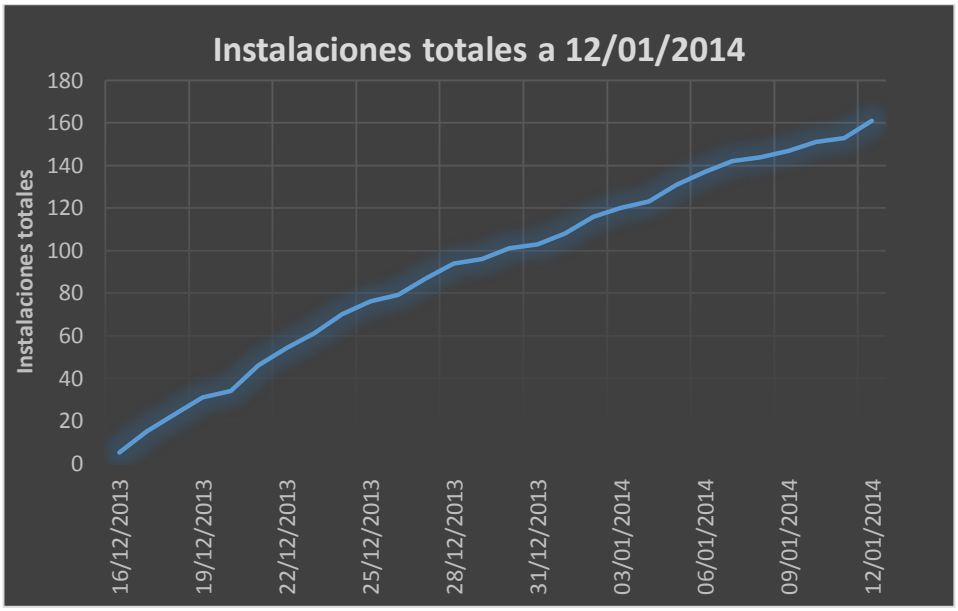



**Figura 5-1: Instalaciones totales a 12/01/2014**

- **Instalaciones actuales:** descargas de la aplicación actualmente activas, sin desinstalar.

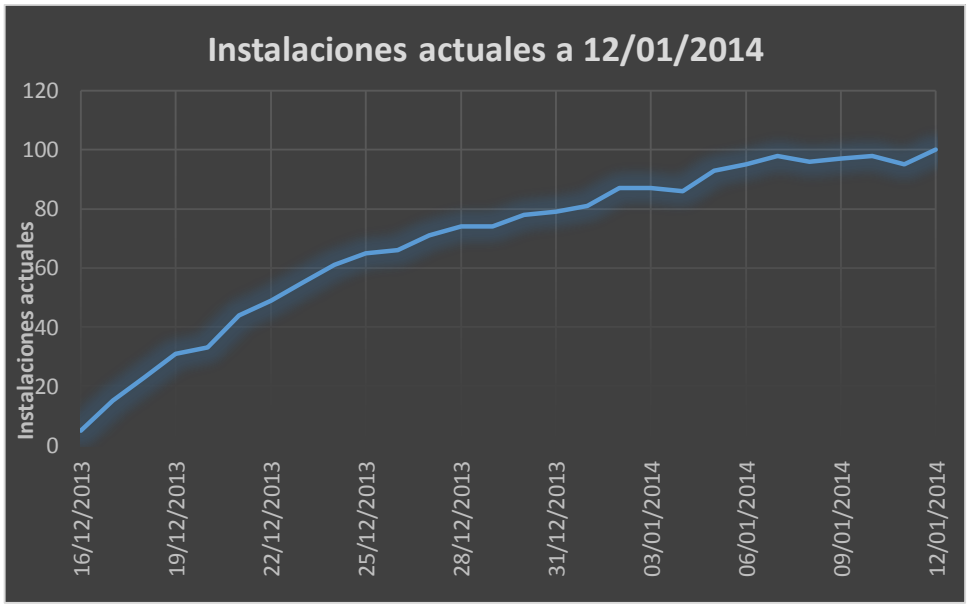



**Figura 5-2: Instalaciones actuales a 12/01/2014**

- **Instalaciones por país:** número de instalaciones por país.

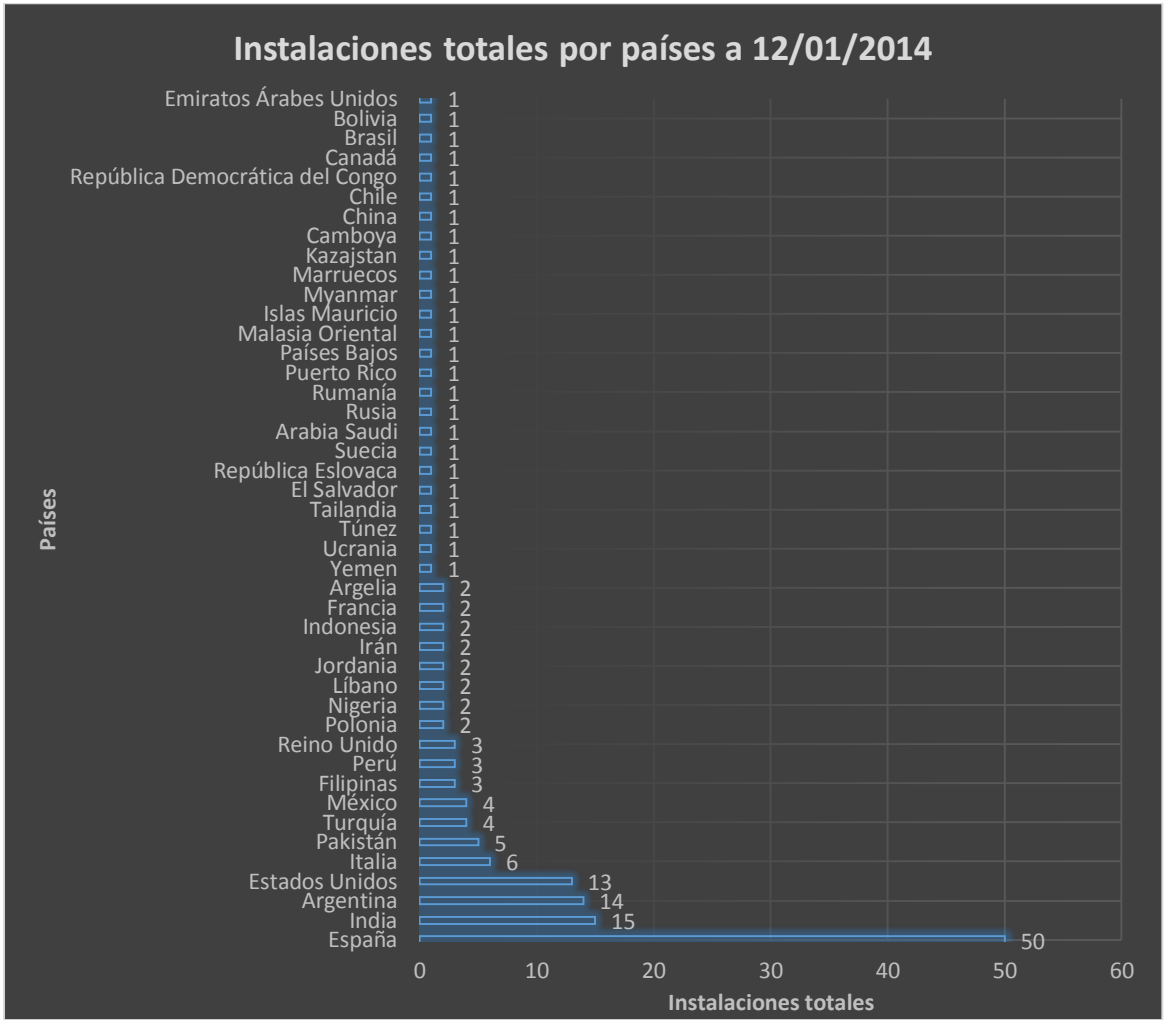



**Figura 5-3: Instalaciones por país a 12/01/2014**

- **Versiones de la aplicación activas:** a fecha de 12/01/2014 la mayoría de las instalaciones están actualizadas a la última versión subida.


**Figura 5-4: Versiones de la aplicación a 12/01/2014**

## *5.4 Pruebas en la asignatura*

Las pruebas en la asignatura Circuitos Electrónicos Digitales se desarrollarán durante el periodo Febrero – Mayo de 2014, periodo en el cual se imparte la asignatura.

Los alumnos podrán además valorar ciertos aspectos de la aplicación con el cuestionario propuesto en la sección 6.2.

La aplicación ha sido publicitada en los tablones de los pasillos de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid a través de un código QR, que enlaza con la página de la aplicación en Google Play.

**Figura 5-5: Código QR de Sequential Circuits**

## 5.5 Ejemplo de utilización de la aplicación

1) Acceder al menú Exercises y elegir el ejercicio que desee. Para acceder a la máquina del usuario y poder resolver el problema, pulsar en Mealy (o Moore cuando corresponda).

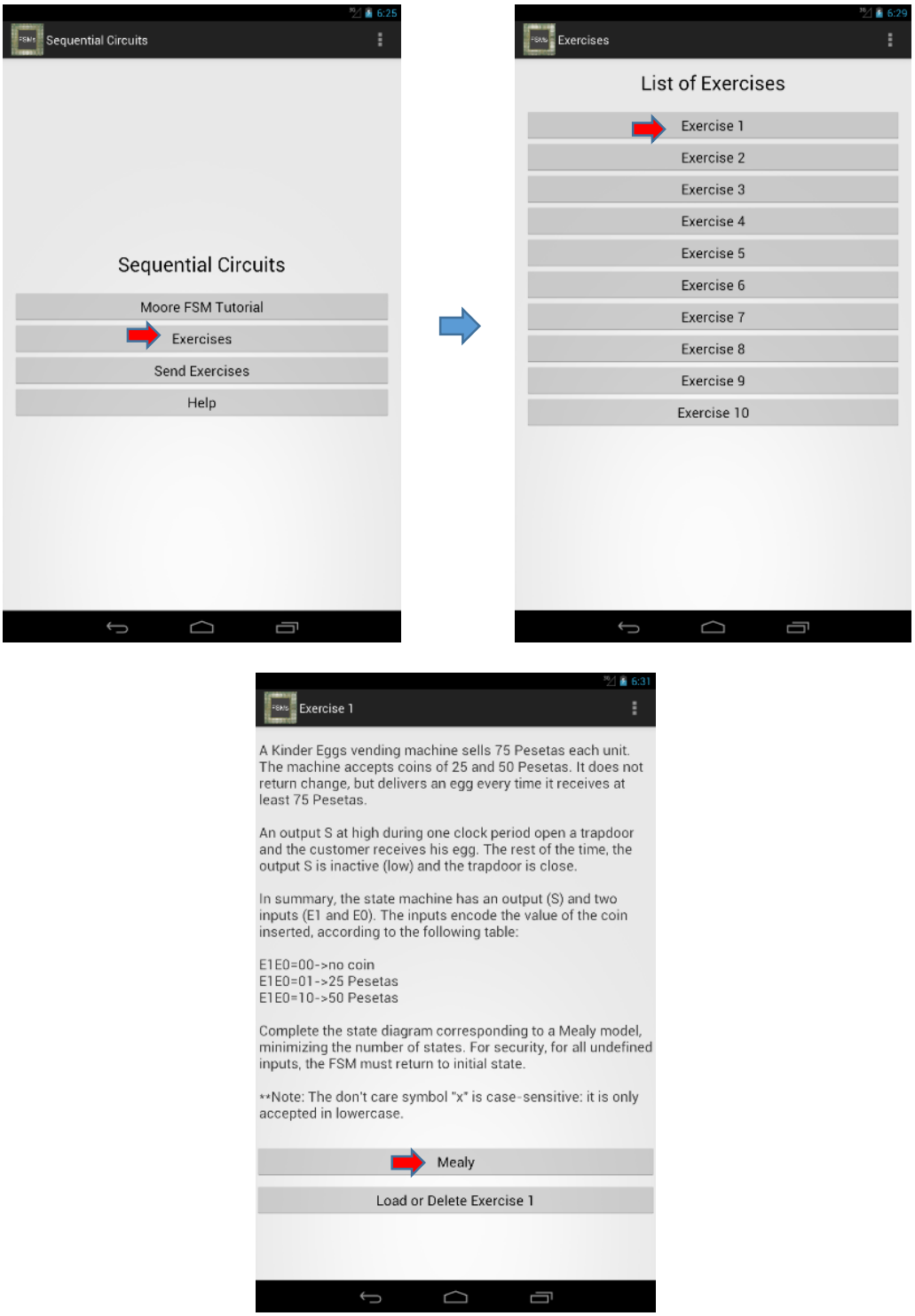



**Figura 5-6: Ejemplo – Acceder al enunciado**

2) Una vez que se accede a la máquina del usuario, se resuelve dibujando flechas. Las flechas se dibujan pulsando primero en un estado (se convertirá en azul) y segundo en otro estado (saldrá alerta para introducir entrada y salida). Una vez introducidas la entrada y la salida, se pulsa el botón OK y la flecha es creada.

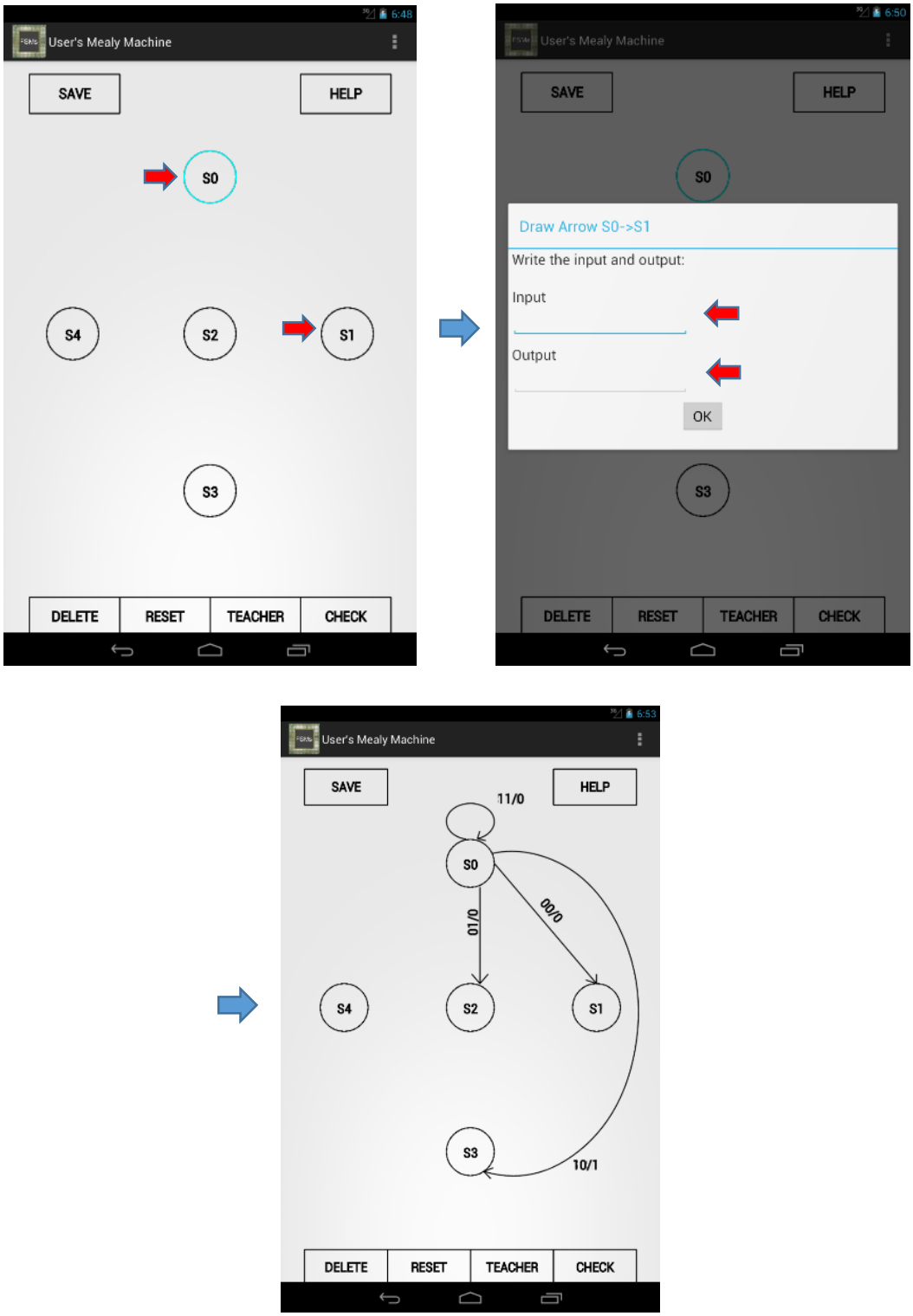



**Figura 5-7: Ejemplo – Resolver el ejercicio**

3) Si se quieren borrar flechas durante la realización del ejercicio, habrá que pulsar en el botón DELETE y aparecerá una alerta con las flechas dibujadas. Se selecciona la que se quiere borrar y se pulsa el botón OK. Si se quiere resetear el ejercicio entero solo basta con pulsar el botón RESET. Si se quiere volver a leer el enunciado, se pulsa el botón BACK y al volver a la máquina del usuario, el ejercicio estará como lo haya dejado anteriormente.

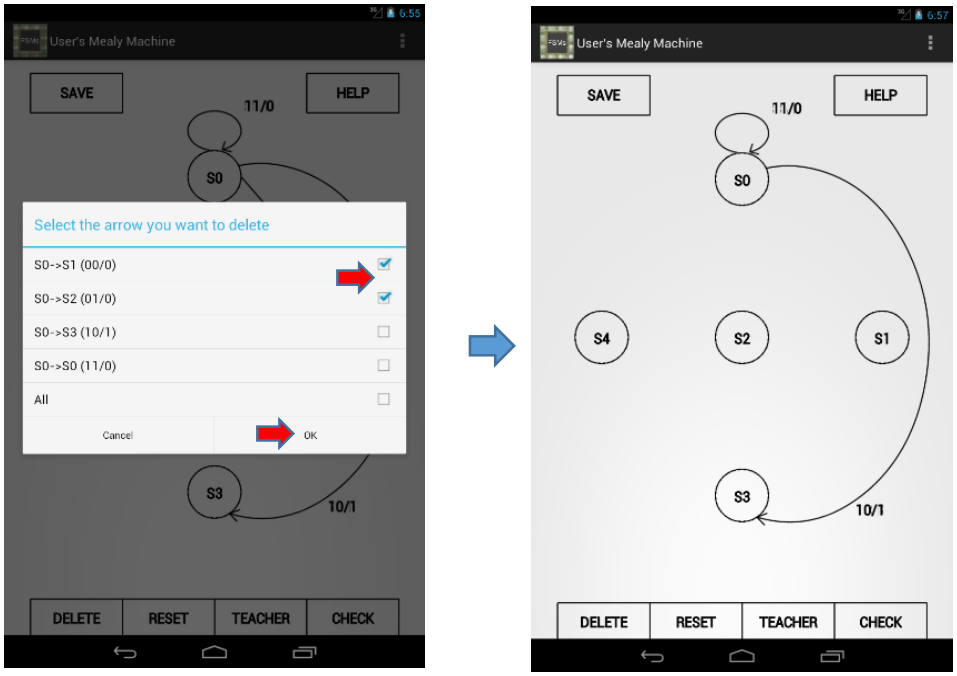



**Figura 5-8: Ejemplo – Borrar flechas**

4) Si se quiere corregir el ejercicio, se debe pulsar el botón CHECK , y éste mostrará un mensaje en la parte inferior de la pantalla indicando fallo o acierto

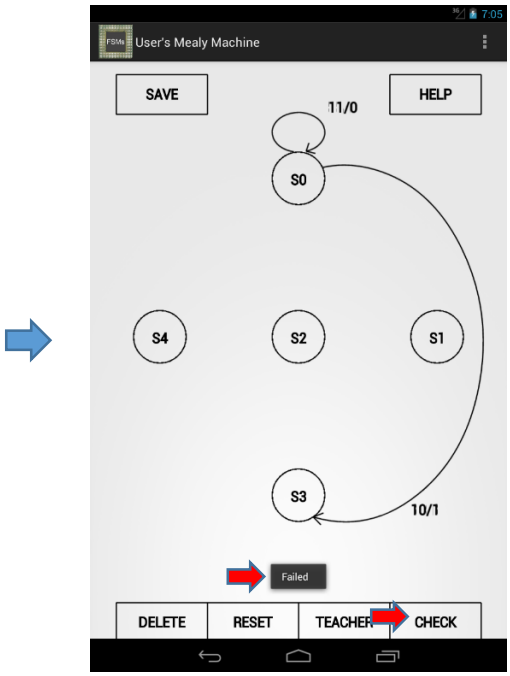



**Figura 5-9: Ejemplo – Corregir el ejercicio**

5) Si se quiere acceder a las ayudas, se pulsa el botón TEACHER. Dependiendo del ejercicio, se ofrecerán diferentes ayudas. Sólo basta con pulsar el botón de la TIP que quieras y la ayuda se mostrará.

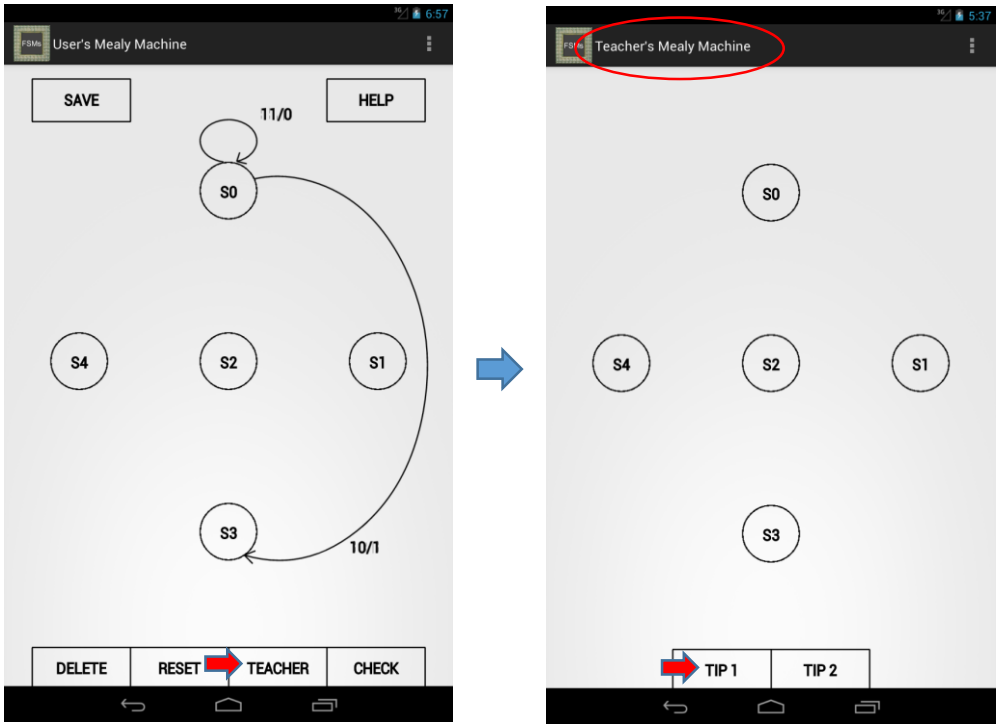



**Figura 5-10: Ejemplo – Acceder a los consejos**

6) Si se quiere guardar el ejercicio, hay que pulsar el botón SAVE. Aparecerá una alerta en la que habrá que introducir el nombre con el que se desea guardar el archivo para después pulsar el botón Save de la alerta.

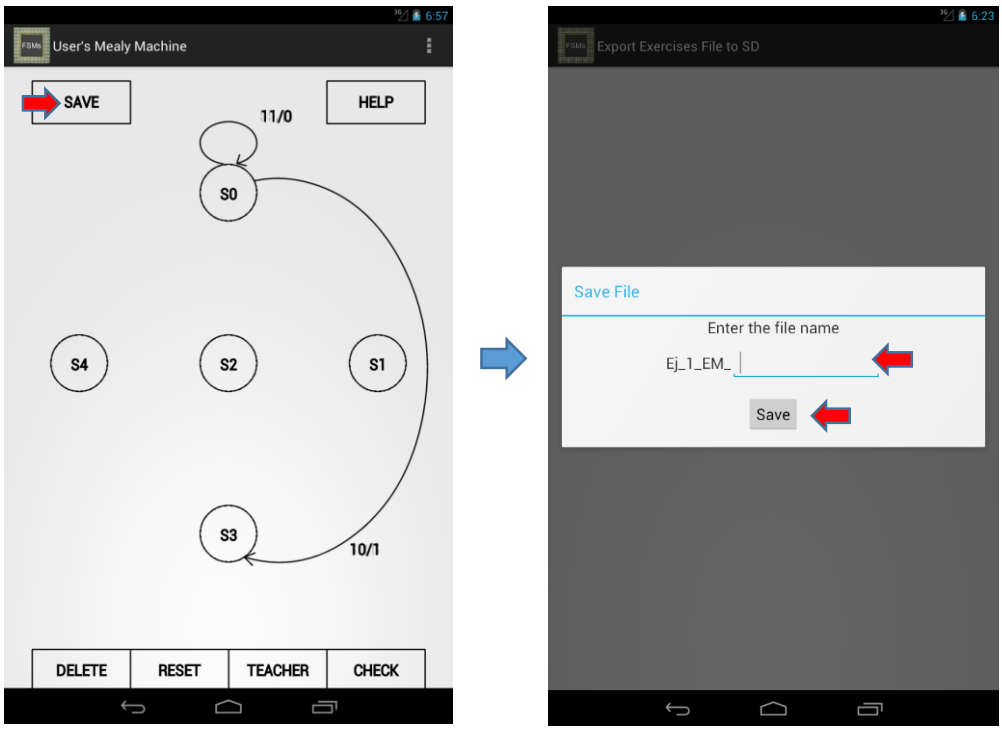



**Figura 5-11: Ejemplo – Guardar ejercicio**

7) Si se quiere cargar un ejercicio, desde el enunciado hay que pulsar en Load or Delete Exercise. Aparecerá una alerta con las soluciones de ese ejercicio guardadas. Hay que elegir una y pulsar el botón Load. Después se volverá al enunciado, y se le indicará a través de un mensaje si se ha cargado el ejercicio o no. Si vuelve a entrar en la máquina del usuario, podrá ver que el ejercicio se ha cargado.

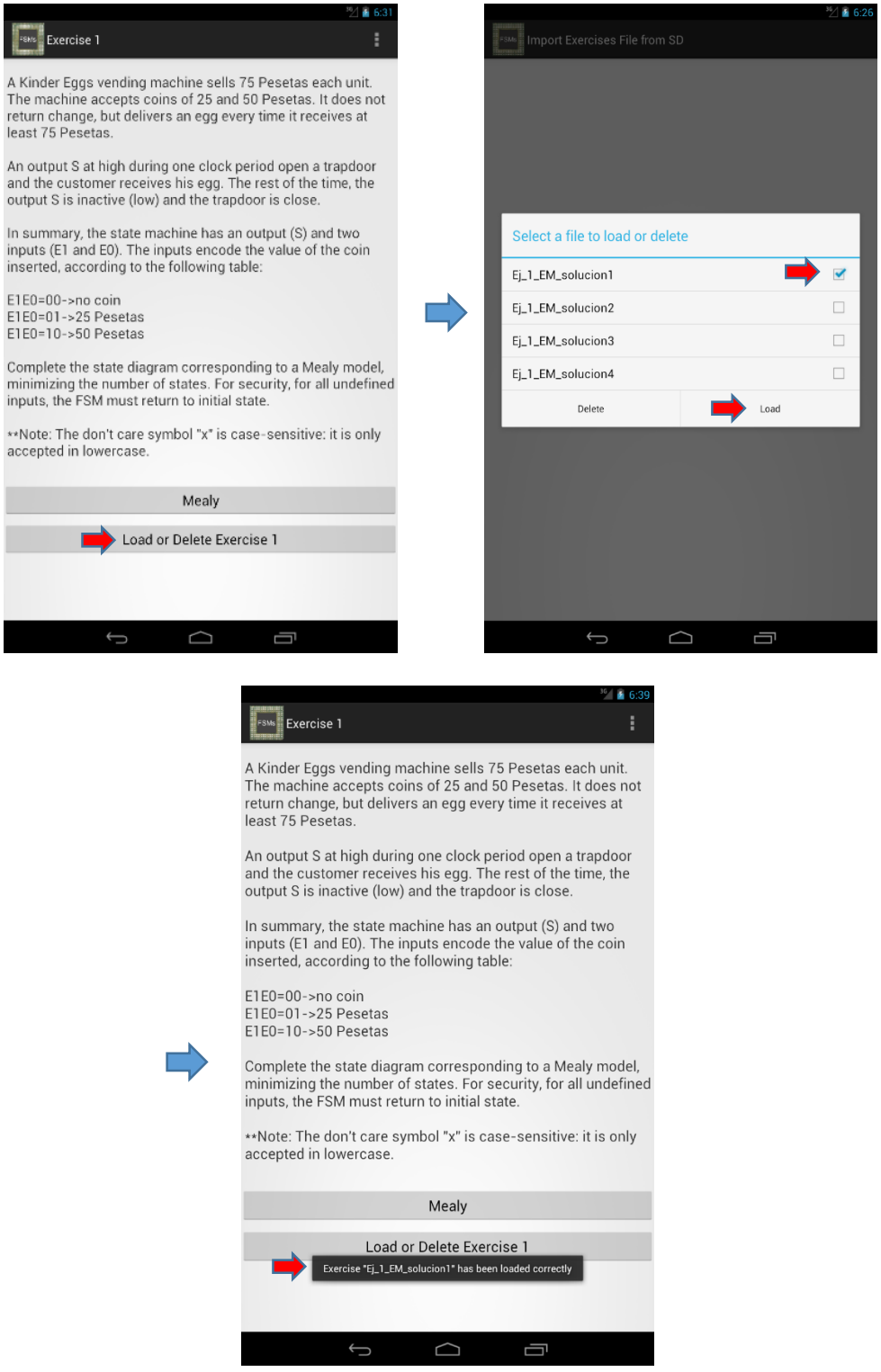



**Figura 5-12: Ejemplo – Cargar ejercicio**

8) Si se quiere enviar un ejercicio, desde el menú principal hay que pulsar en Send Exercises. Aparecerá una alerta mostrando los ejercicios guardados. Hay que seleccionar los que se desea enviar, y pulsar el botón Send. Aparecerá otra alerta mostrando los gestores de correo con los que poder enviar los ejercicios. Se escoge el necesario y se crea un nuevo correo en el que solamente hay que indicar el destinatario (asunto y adjuntos ya están hechos).

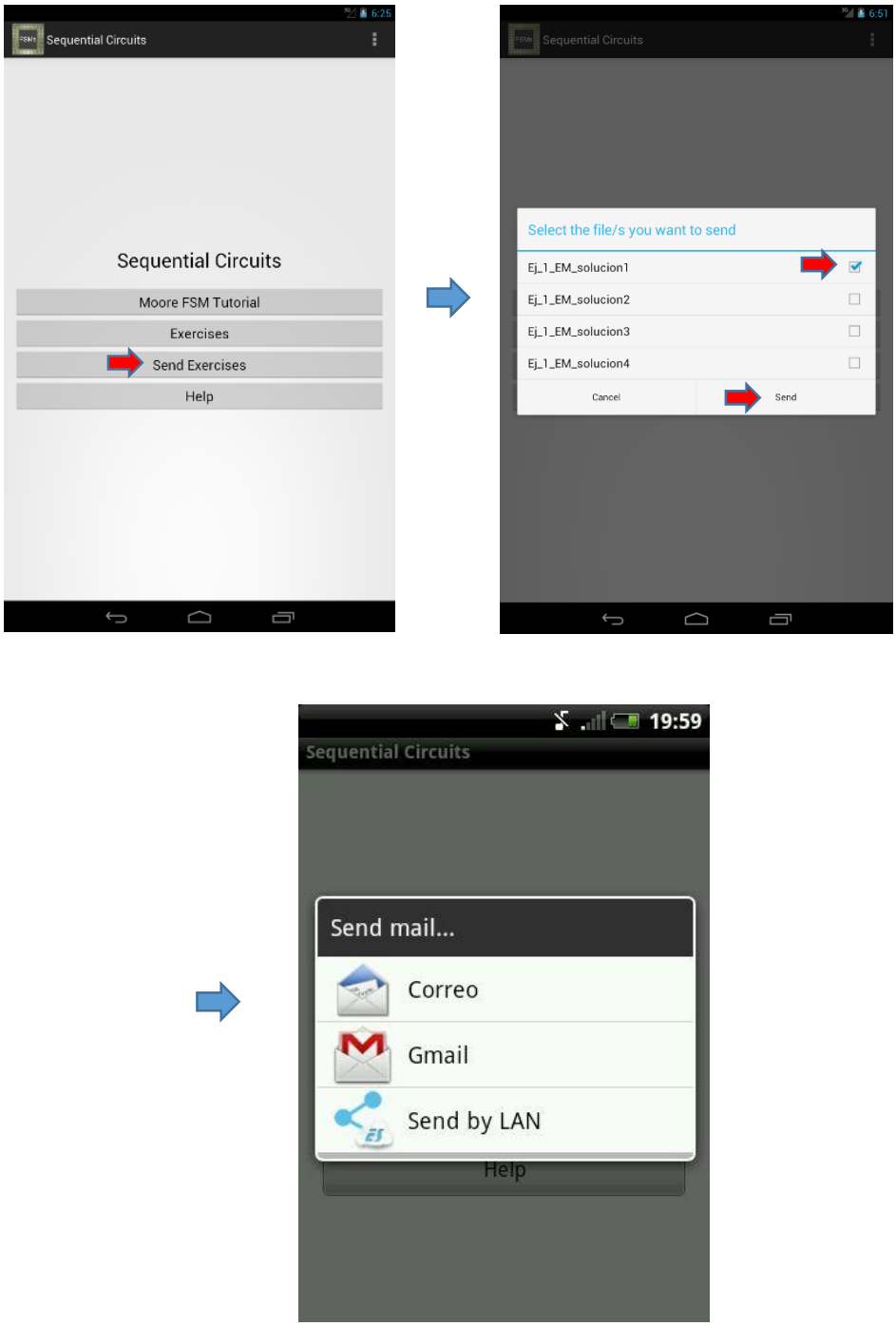



**Figura 5-13: Ejemplo – Enviar ejercicios**

9) Si se quiere leer el tutorial sobre máquinas de Moore, desde el menú principal hay que pulsar el botón Moore FSM Tutorial. Para avanzar de página hay que pulsar en la parte derecha de la pantalla. Para retroceder de página hay que pulsar en la parte izquierda de la pantalla.

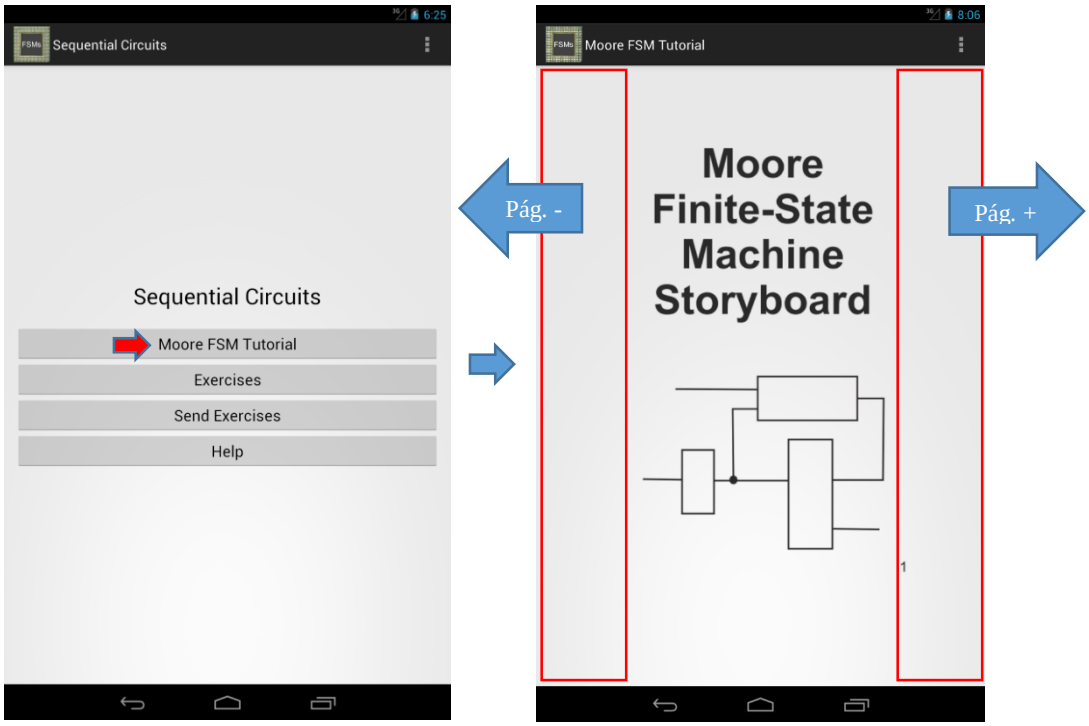



**Figura 5-14: Ejemplo – Tutorial**

# 6 Conclusiones y trabajo futuro

## *6.1 Conclusiones*

Las principales conclusiones del proyecto son:

Se ha realizado una aplicación prototipo que pretende sustituir a la guía de problemas en papel o PDF que se entrega en todas las escuelas de ingeniería.

Se han obtenido algunos modelos de ejercicios aptos para ser resueltos en un teléfono de manera gráfica.

Se han identificado y resuelto las principales técnicas de programación de utilidad en el proyecto, tales como:

- Definición de mecanismo para trasladar el diseño de una máquina de estados en papel a teléfono móvil o tableta.
- Se ha definido una estrategia para corregir ejercicios de FSMs.
- Se ha estandarizado una serie de ayudas y mensajes de error, los cuales serán estándares de futuras aplicaciones.
- Definición de un mecanismo para realizar tutoriales tipo e-book.
- Guardar y cargar soluciones.
- Transmitir soluciones por correo.
- Obtención de los valores óptimos de tipo de letras y tamaño, resolución de los gráficos, márgenes, adaptación a diferentes pantallas, etc. Todas estas variables se han fijado y serán de gran utilidad en los siguientes PFC sobre el tema.

El código contiene unas 9000 líneas de código, y se han dedicado cerca de 1000 horas.

La parte a la que más tiempo se ha dedicado ha sido la adaptación de la aplicación a todas las pantallas.

En cuanto a resultados, la prueba con los alumnos de la asignatura Circuitos Electrónicos Digitales se desarrollará durante el periodo Febrero – Mayo, periodo en el cual se imparte esta asignatura en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.

Los alumnos podrán descargar la aplicación desde la plataforma Google Play y podrán rellenar el cuestionario de la sección 6.2, diseñado para valorar diferentes aspectos de la aplicación.

## *6.2 Cuestionario*

| | Muy malo (1) | Malo (2) | Normal (3) | Bueno (4) | Muy bueno (5) |
|---|---|---|---|---|---|
| Organización general | | | | | |
| Aspecto de los gráficos | | | | | |
| Facilidad de uso | | | | | |
| Utilidad de la opción enviar | | | | | |
| Utilidad de la Ayuda General (F1) | | | | | |
| Fiabilidad y estabilidad | | | | | |
| Consumo de batería | | | | | |
| Velocidad | | | | | |
| Ordenación de los ejercicios | | | | | |
| Dificultad de los ejercicios: | | | | | |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| Utilidad de las “tip” | | | | | |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| Calidad del Tutorial | | | | | |
| Utilidad del Tutorial | | | | | |
| Comentarios y sugerencias: | | | | | |

**Tabla 6-1: Cuestionario sobre la aplicación**

*Dificultad: del 1 al 5 de menos a más

## *6.3 Trabajo futuro*

Entre los principales trabajos futuros se pueden mencionar:

Programación:

- Opción on-line. Imágenes y cualquier recurso necesario alojado en servidor, para reducir el peso de la aplicación.
- Aplicación en idioma español.
- Teclado digital.
- Generador automático de aplicación.
- Adaptar a Apple IOS.
- Análisis y optimización en consumo de batería

Docencia:

- Separación Mealy y Moore.
- Temporización detallada de una FSM (sobre todo, caso Mealy).
- Identificación de nuevas formas de pasar de "resolución en papel" a esquemas equivalentes en teléfonos móviles.
- Aplicación a otros ámbitos educativos como discapacitados, o niños de pre-escolar.

Comercial:

- Incorporación de publicidad.
- Posicionarla en redes sociales.
- Separación de la aplicación en dos: una gratis y otra de pago, con las soluciones.
- Problemas extras de pago.

# Referencias


[1] Fco Javier Ceballos, “Java 2 Curso de Programación”, 4º edición, 2010.
[2] Jesús Tomás Girones, “El gran libro de Android”, 2011.
[3] Ed Burnette, “Android”
[4] EPS-UAM, “Desarrollo de aplicaciones para dispositivos móviles”
[5] http://stackoverflow.com/
[6] http://developer.android.com/training/index.html
[7] http://www.androidcurso.com/index.php/curso-android-basico/tutoriales-android-basico
[8] http://www.inforjmr.es/?p=531
[9] http://www.android-spa.com/
[10] http://androideity.com
[11] http://www.sgoliver.net/blog/?p=1731
[11] http://www.sgoliver.net/blog/?p=2019
[12] http://gs.statcounter.com/
[13]http://lordscales91.blogspot.com.es/2012/10/opinionprogramacion-multiplataforma.html
[14]http://latecladeescape.com/t/Compiladores,+int%C3%A9rpretes+y+m%C3%A1quinas+virtuales
[15]http://barraespaciadora.com/2013/05/10/comparativa-de-sistemas-operativos-para-smartphones/
[16] http://columna80.wordpress.com/2011/02/17/arquitectura-de-android/
[17]http://www.all-things-android.com/es/content/entender-la-jerarquia-de-archivos-android
[18] https://play.google.com/store
[19] https://itunes.apple.com/
[20] http://es.blackberry.com/apps/app-world.html

# Anexos

## A Presupuesto

**1) Ejecución Material**

- Compra de ordenador personal (Software incluido)..... 700 €
- Amortización de impresora láser durante 6 meses ..... 50 €
- Teléfono Móvil..... 300 €
- Tableta..... 300 €
- Material de oficina ..... 50 €
- Total de ejecución material ..... 1.400 €

**2) Gastos generales**

- 16 % sobre Ejecución Material ..... 224 €

**3) Beneficio Industrial**

- 6 % sobre Ejecución Material ..... 84 €

**4) Honorarios Proyecto**

- 960 horas a 15 € / hora ..... 14400 €

**5) Material fungible**

- Gastos de impresión ..... 60 €
- Encuadernación ..... 200 €

**6) Subtotal del presupuesto**

- Subtotal Presupuesto ..... 16368 €

**7) I.V.A. aplicable**

- 21% Subtotal Presupuesto ..... 3437.28 €

**8) Total presupuesto**

- Total Presupuesto ..... 19805,28 €

Madrid, Febrero de 2014

El Ingeniero Jefe de Proyecto

Fdo.: Pablo Molinero Merino
Ingeniero Superior de Telecomunicación

# B Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES SECUENCIALES BAJO ANDROID. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

## Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el

deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

## Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.