UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

MAPAS VEITCH-KARNAUGH EN ANDROID

Rubén Jiménez Benito

NOVIEMBRE 2014

MAPAS VEITCH-KARNAUGH EN ANDROID

AUTOR: Rubén Jiménez Benito
TUTOR: Federico García Salzmann
PONENTE: Eduardo Boemo Scalvinoni

Digital System Laboratory
Dpto. Tecnología Electrónica y de Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Noviembre 2014

Resumen

Este proyecto consiste en la realización de una aplicación para dispositivos Android. Dicha aplicación se centra en la simplificación de funciones lógicas desde 3 hasta 7 variables para ayudar a los alumnos de la asignatura Circuitos Electrónicos Digitales de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid (EPS UAM). Con la aplicación se puede desde entender e interiorizar la simplificación con los mapas de Veitch-Karnaugh, hasta comprobar el resultado de la mayoría de los problemas de los primeros temas de la asignatura.

Con funciones lógicas de hasta 5 variables permite la opción de simplificar con los mapas, ya que perfectamente se pueden manejar a la vez hasta 32 casillas en una pantalla de tamaño normal de un teléfono móvil. Para las funciones de más variables se puede introducir de otras maneras alternativas: Tabla de la verdad, sumatorio de minitérminos, producto de maxitérminos y escribiendo la expresión lógica por teclado. Una vez introducida se traduce al resto de modos y además se simplifica y muestra su esquemático.

La última herramienta incluida es un conversor binario, octal, decimal y hexadecimal que será muy útil a los alumnos en la práctica 4 del laboratorio de la ya citada asignatura.

La aplicación se ha desarrollado soportando los idiomas inglés y español, de manera que se ejecutará en el idioma en que se encuentre configurado el dispositivo. Aunque se ha elegido como predeterminado el inglés para usuarios cuyo terminal no se encuentre en uno de los dos idiomas implementados.

La aplicación se encuentra publicada en Play Store.

Palabras clave

Mapas Karnaugh, mapas Veitch-Karnaugh, Android, tableta, móvil, puertas lógicas, simplificación lógica, conversor, binario, octal, decimal, hexadecimal, esquemático, tabla de la verdad, sumatorio minitérminos, producto maxitérminos, tutorial.

Abstract

This project deal with an application for Android devices, focus on the simplification of logical functions for 3-7 variables using Karnaugh maps. It intent to help students of Digital Electronic Circuits subject of EPS UAM. With this program, the students can understand the mechanic of the Veitch- Karnaugh method as well as to check the result of several problems of combinational circuits.

The app can graphically solve K-maps of up to 5 variables. For functions of more variables, it is only possible to enter the function as truth table, sum of minterms, product of maxterms and by writing the algebraic expression. Once the function has entered, it is translated into other modes and also simplified. The schematic is also synthesized.

Additionally, the tool includes a binary, octal, decimal and hexadecimal converter, that is utilized by students in the Lab Session 4 of the aforementioned subject.

The application support English and Spanish languages, running into the language that is configured on the device. English language is the default.

The application have been posted to the Play Store in October 2014.

Key words

Karnaugh Maps, Veitch-Karnaugh Maps, Android, tablet, mobile phone, logical functions simplification, converter, binary, octal, decimal, hexadecimal, schematic, truth table, sum of minterms, product of maxterms, tutorial.

Glosario

DSLab:	Digital System Laboratory
EPS:	Escuela Politécnica Superior
UAM:	Universidad Autónoma de Madrid
PFC:	Proyecto Final de Carrera
GSM:	Global System for Mobile Communications
EDGE:	Enhanced Data for GSM Evolution
GPS:	Global Positioning System
OS:	Operating System
SMS:	Short Message Service
API:	Application Programming Interface
CPU:	Central Processing Unit
VK:	Veitch-Karnaugh
SDK:	Software Development Kit
AVD:	Android Virtual Device
XML:	eXtensible Markup Language
LIFO:	Last Input First Output

Agradecimientos

En primer lugar quiero dar las gracias a todas las personas que han formado parte de mi vida mientras hacía el proyecto y que han ido viendo todos los avances del mismo poco a poco.

Pero el agradecimiento más especial es para mi padre, con conocimientos de los fundamentos de la programación, aunque fuese en alguna "lengua muerta", ha aportado muchos consejos e ideas útiles a lo largo de toda la carrera, no sólo en este proyecto.

A Eduardo, por brindarme la oportunidad de realizar este proyecto y por su ayuda durante todo este periodo.

A todo el grupo de amigos que he hecho aquí en la escuela y que espero que dure muchos años.

ÍNDICE DE CONTENIDOS

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Cuando se trabaja con funciones digitales y se quieren implementar en la realidad mediante puertas lógicas, es conveniente obtener la función de la forma más sencilla mediante simplificación. Uno de los métodos más sencillos para llevar a cabo está simplificación son los mapas de Veitch-Karnaugh.

Estos mapas simplifican mediante una representación gráfica en forma de tabla de la función, que es muy intuitiva. Pero cuando va creciendo el número de variables empieza a ser cada vez más difícil simplificar con dichos mapas y se vuelve cada vez menos sencillo.

Este problema provoca la necesidad de dar a los alumnos un método para simplificar funciones con un alto número de variables. Además a lo largo del curso, tendrán que resolver centenares de problemas de este tipo, por lo que una vez que lo tengan mecanizado ya no será necesario que sigan teniendo que dibujar y perder tiempo simplificando funciones lógicas. Por lo que esta herramienta les proporcionará una manera más rápida de solucionar estos problemas.

Aprovechando el auge que en los últimos años han sufrido los dispositivos móviles y las tabletas y su uso generalizado entre los estudiantes, se ha decidido que serán estas plataformas para las que se desarrollará la herramienta que ayudará a los estudiantes.

Algunas de las ventajas que ofrecen estos dispositivos:

- Fácil portabilidad. Un terminal móvil ocupa mucho menos espacio que un ordenador y se lleva todo el día en el bolsillo.
- Sustitución del papel y bolígrafo por pantalla y dedo.
- Ahorro de tiempo.
- Facilidad de introducir datos con pantalla táctil.
- Actualización automática proporcionada por Play Store.
- Posibilidad de uso sin acceso a Internet.
- Posibilidad de ser una herramienta de ayuda para estudiantes de ingeniería de todo el mundo.

1.2 Objetivos

El objetivo de este proyecto es desarrollar una aplicación Android que provea de ciertas herramientas al usuario para agilizar la simplificación y traducción de funciones lógicas. Esta aplicación principalmente se orientará a las necesidades de los alumnos de la asignatura de Circuitos Electrónicos Digitales de la EPS, aunque también será de utilidad para cualquier estudiante de otra universidad que tenga una asignatura parecida.

Esta aplicación, combinada con el resto de las aplicaciones desarrolladas en el DSLab, constituirá una manera rápida e interactiva de aprender y ejercitar los fundamentos de Circuitos Digitales. Los estudiantes las tendrán disponibles en Play Store para su descarga. Adicionalmente, en las guías de problemas de la asignatura, se indicarán los ejercicios en los que alguna de las aplicaciones será de utilidad o tienen ese problema disponible.

Desde el punto de vista del estudiante de PFC el objetivo es aprender a programar aplicaciones para dispositivos Android y sobre todo, el lenguaje Java. Ninguno de los dos temas está cubierto por las asignaturas obligatorias de la carrera, por lo tanto constituyen una formación adicional y muy enriquecedora para el futuro. Además, supone una revisión profunda de Circuitos Digitales.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción, motivación, objetivos del proyecto y organización de la memoria.

- Capítulo 2: Estado del arte, sistemas operativos móviles y estudio de aplicaciones en el mercado.

- Capítulo 3: Diseño de la aplicación, finalidad, elección de requisitos y herramientas de la aplicación.

- Capítulo 4: Desarrollo de la aplicación, pasos previos, instalación del entorno de desarrollo, esquema general , conceptos clave, detalles de la implementación, algoritmo Quine-McCluskey y diagrama de clases.

- Capítulo 5: Integración, pruebas, resultados, primera publicación, estadísticas y cuestionario.

- Capítulo 6: Conclusiones y trabajo futuro.

2 Estado del arte

2.1 Sistemas operativos móviles

En este primer apartado se va a realizar una breve introducción a los principales sistemas operativos móviles actuales. Prestando especial atención al sistema Android, en el cual se desarrollará la aplicación en cuestión.

2.1.1 Android

2.1.1.1 Historia

Android es un Sistema Operativo que en la actualidad se ha convertido en el más utilizado en dispositivos móviles y tabletas. Es un sistema que fue desarrollado por Android Inc. basado en el kernel de Linux. Google respaldo económicamente a la empresa desarrolladora hasta que finalmente la compró en 2005.

En el 2007 se creó la Handset Alliance, que es un conjunto de compañías de Hardware, Software y telecomunicaciones que promocionaban los estándares de códigos abiertos para dispositivos móviles. Android es un sistema operativo de código abierto, lo que significa que no hay que pagar nada ni para programar en él, ni para incluirlo en un terminal. Esto es lo que ha provocado su gran popularidad ya que reduce mucho los costes a la hora de fabricar un teléfono o aplicación. Cualquiera puede descargarse el código fuente y modificarlo o instalarlo a su antojo.

2.1.1.2 Características

Algunas de las características más novedosas que incluye el sistema Android y que le han hecho tan popular son:

- **La máquina virtual Dalvik**: Que se encarga de arrancar un ejecutable Dalvik compilado desde el código Java. Es una máquina virtual especializada para Android y optimizada para consumir pocos recursos en dispositivos como los móviles.
- **Multimedia**: Dispone de soporte para multitud de formatos de audio, video e imágenes.
- **Conectividad**: El sistema está completamente equipado y soporta todo tipo de conexiones: Bluetooth, 3G, Wi-Fi, GSM/EDGE y muchas más.
- Soporta también una gran variedad de elementos y sensores para la interactividad con el exterior como: Cámaras, GPS, brújula, acelerómetro, pantalla táctil, sensores de proximidad y de presión.
- **Gráficos optimizados**: biblioteca de gráficos 2D, biblioteca de gráficos 3D.
- **Navegador integrado**: Basado en el motor de código abierto WebKit.

- **SQlite**: Es un sistema de gestión de bases de datos relacional contenida en una librería relativamente pequeña.

2.1.1.3 Arquitectura

La arquitectura de Android está formada básicamente por 4 capas como se puede ver en la siguiente figura:

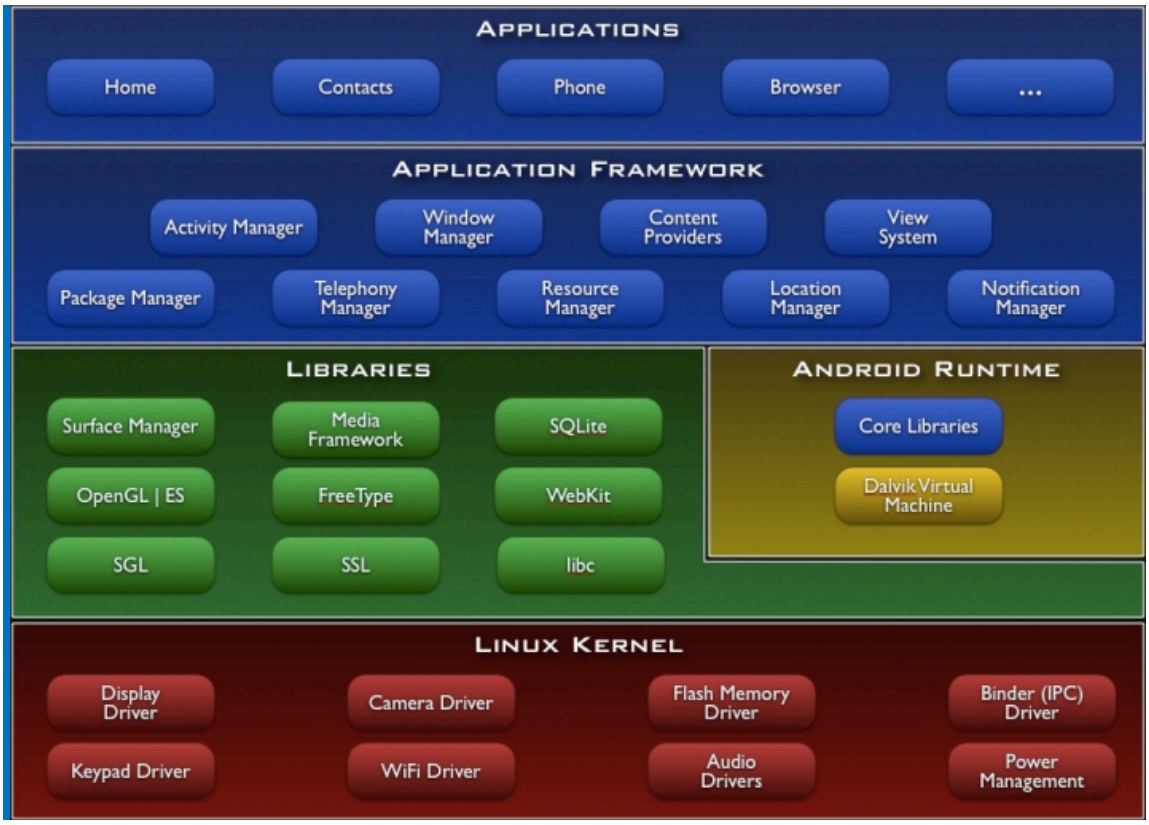

Figura 2-1: Arquitectura de Android

* Imagen obtenida de: http://developer.android.com/

- **Aplications**: Todas las aplicaciones son escritas en lenguaje Java. Las aplicaciones base incluyen un cliente de email, programa de SMS, calendario, mapas, navegador, contactos y mucho más.
- **Aplication Framework**: Debido a la característica de código abierto, cualquiera tiene acceso al código fuente usado en las aplicaciones base. Así se simplifica la reutilización de componentes.
- **Libraries**: Android incluye un conjunto de librerías en C/C++ que están ya compiladas e instaladas en el dispositivo para tareas que se vayan a realizar con frecuencia en las aplicaciones. Algunas de ellas son: Librerías de gráficos, 3D, SQlite y WebKit.
- **Android Runtime**: Con un set de librerías base para proporcionar la mayor parte de las funciones del lenguaje java. Cada aplicación corre su

propio proceso en una instancia de la Máquina Virtual Dalvik de la que ya se habló en las características.

- **Linux Kernel**: Android depende de Linux para los servicios básicos del sistema, como seguridad, gestión de memoria y de procesos, etc.

2.1.1.4 Versiones

Desde la primera versión comercial (de prueba) de Android en 2007 ha habido una gran cantidad de nuevas actualizaciones del sistema. Típicamente corregían errores de las anteriores y añadían nuevas funcionalidades. Como curiosidad, las versiones de Android reciben el nombre de diferentes postres en inglés y cada una empieza por una letra distinta siguiendo el orden alfabético.

Versión	Identificador	Nivel de API
Android 4.4	Kit Kat	19
Android 4.3	Jelly Bean MR2	18
Android 4.2	Jelly Bean MR1	17
Android 4.1	Jelly Bean	16
Android 4.0.3	Ice Cream Sandwich MR1	15
Android 4.0	Ice Cream Sandwich	14
Android 3.2	Honeycomb MR2	13
Android 3.1	Honeycomb MR1	12
Android 3.0	Honeycomb	11
Android 2.3.3	Gingerbread MR1	10
Android 2.3	Gingerbread	9
Android 2.2.2	Froyo	8
Android 2.1	Eclair MR1	7
Android 2.0.1	Eclair 0.1	6
Android 2.0	Eclair	5
Android 1.6	Donut	4
Android 1.5	Cupcake	3
Android 1.1	Banana Bread	2
Android 1.0	Apple Pie	1

Tabla 2-1 Versiones de Android

2.1.1.5 Android en números

En la siguiente gráfica/tabla se puede ver cómo están repartidos los usuarios entre las actuales versiones de Android. Se puede observar que casi un 90 % de usuarios se concentran en las versiones de Ice Cream Sandwich en adelante equivalentes a la versión Android 4.0 y posteriores.

Identificador	API	Porcentaje
Froyo	8	0.7 %
Gingerbread	10	11.4 %
Ice Cream Sandwich	15	9.6 %
Jelly Bean	16	25.1 %
	17	20.7 %
	18	8.0 %
Kit Kat	19	24.5 %

Tabla 2-2: Distribución de versiones Android

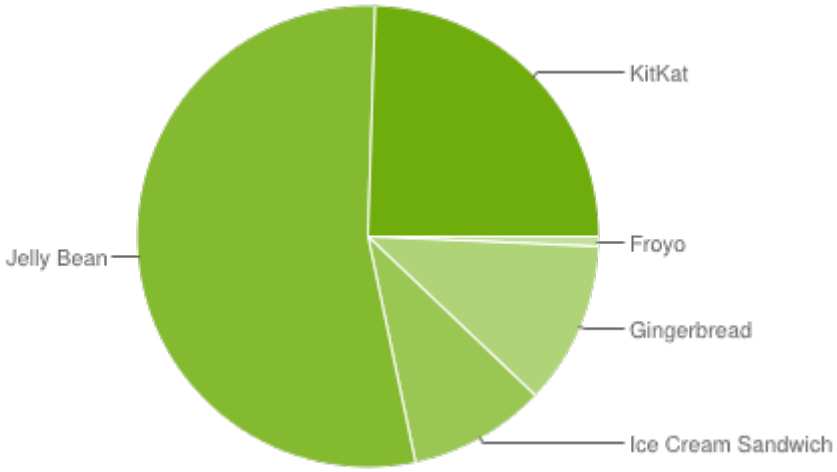

Figura 2-2: Distribución de versiones Android

* Imagen obtenida de: http://developer.android.com/

2.1.2 Otros sistemas operativos

2.1.2.1 iOS

El sistema operativo iOS es el principal rival de Android en la actualidad. Desarrollado por la empresa Apple Inc. y de código cerrado solo se encuentra en dispositivos de dicha marca. Esta exclusividad, su apuesta por la calidad y la experiencia de usuario que propone supusieron una revolución sin precedentes.

Aunque en comparación con Android tiene una cuota de mercado mucho menor, en realidad tiene una gran cantidad de seguidores fieles que parece que nunca se van a perder, además hay estadísticas que dicen que sus usuarios gastan aproximadamente 4 veces más dinero en aplicaciones que los de Android.

2.1.2.2 Blackberry OS

Sistema operativo móvil de código cerrado desarrollado por RIM para los dispositivos de BlackBerry, compañía que en 2008 lideraba el mercado de los Smartphones y ahora lucha por evitar su hundimiento.

2.1.2.3 Symbian OS

Propiedad de Nokia es un sistema operativo que tenía mucha cuota de mercado en el pasado hasta la llegada de los Smartphones.

2.1.2.4 Windows Phone

Es el sistema operativo desarrollado por Microsoft sucesor de Windows Mobile. Aunque en comparación con sus competidores tiene poca cuota de mercado, parece que sigue creciendo poco a poco.

2.1.3 Sistemas operativos en números

Por último se adjunta el siguiente gráfico para entender el reparto actual de la cuota de mercado en el mundo de los Smartphones.

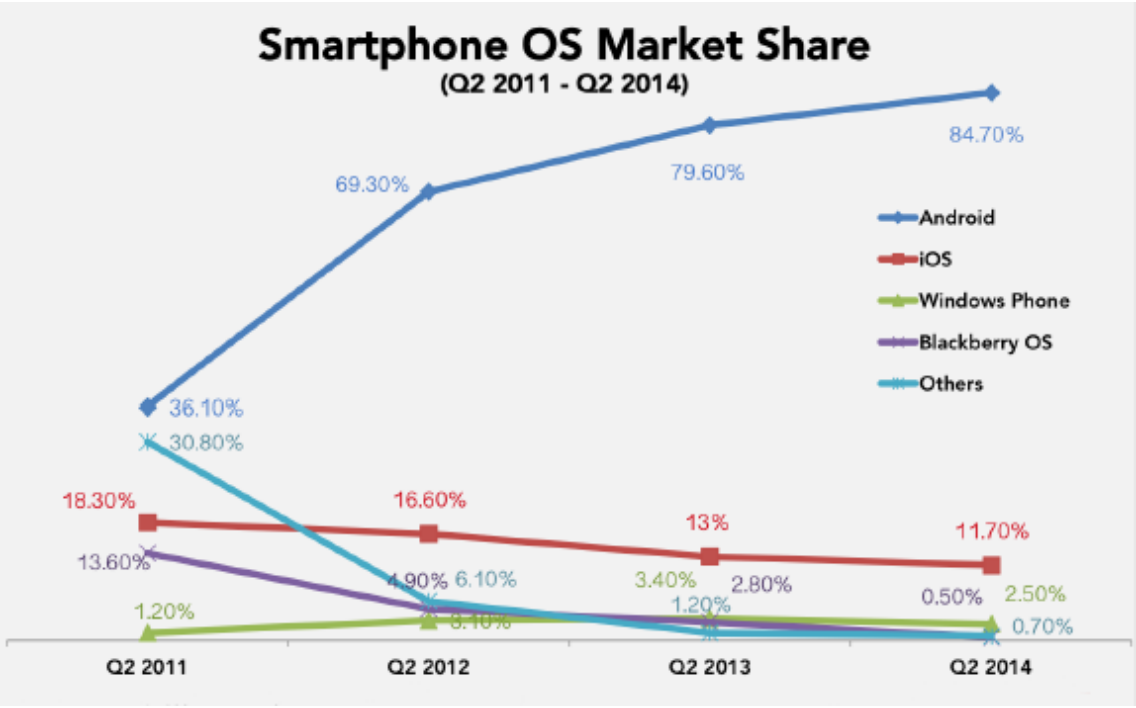

Figura 2-3: Distribución del mercado entre sistemas operativos

* Imagen obtenida de: IDC's Worldwide Mobile Phone Traker

2.2 Aplicaciones similares ya en el mercado

En este apartado se va a realizar un estudio de las aplicaciones más importantes existentes en el mercado. Se buscarán las que tengan funcionalidades similares a las que se desarrollarán en el proyecto, aplicaciones con las que se competiría.

Dado que el proyecto se va a desarrollar para dispositivos Android sólo es necesario comparar con las aplicaciones existentes en esta tecnología. Además como la aplicación resultante se subirá, en un principio, de forma gratuita sólo se estudiarán las aplicaciones con precio cero.

Este paso podría no ser necesario ya que tampoco se busca hacer negocio, de momento, con esta aplicación. Simplemente el objetivo de estas aplicaciones del DSLab es ayudar a los alumnos de la escuela a desarrollar antes su aprendizaje en el ámbito de los Circuitos Digitales.

Las aplicaciones del mercado más parecidas que se han encontrado son:

2.2.1 KMap

La primera aplicación encontrada es KMap. La aplicación es gratuita, pero como casi todas las aplicaciones gratuitas del Play Store, tiene publicidad. Nada más entrar te da a elegir el número de variables que quieres entre 2 y 5, se puede ver en la primera captura.

La segunda captura es de una simplificación de un mapa K de 4 variables, la estética es sencilla formada por botones que se iluminan si está elegido ese cuadro. Gráficamente no diferencia los grupos en los que se han seleccionado los unos para simplificar, pero muestra la lista debajo en texto. La función simplificada la escribe arriba en texto correctamente.

La tercera captura es del mapa de 5 variables, la estética pierde mucho debido al reducido espacio de la pantalla. No se ve donde se encuentra la variable más significativa en el mapa y la palabra expresión está mal escrita y dividida en dos líneas. Aún así, también parece funcionar correctamente.

Por último, también permite la introducción de funciones en tabla de verdad para su posterior simplificación, como se puede ver en la captura 4.

Como comentario adicional, la aplicación no permite cambiar el nombre de las variables y por defecto es ABCDE, lo cual entorpece su aplicación para otros conjuntos de variables.

Se ha puesto a prueba en casos extremos y parece seleccionar siempre el mínimo número de grupos que abarque todos los unos.

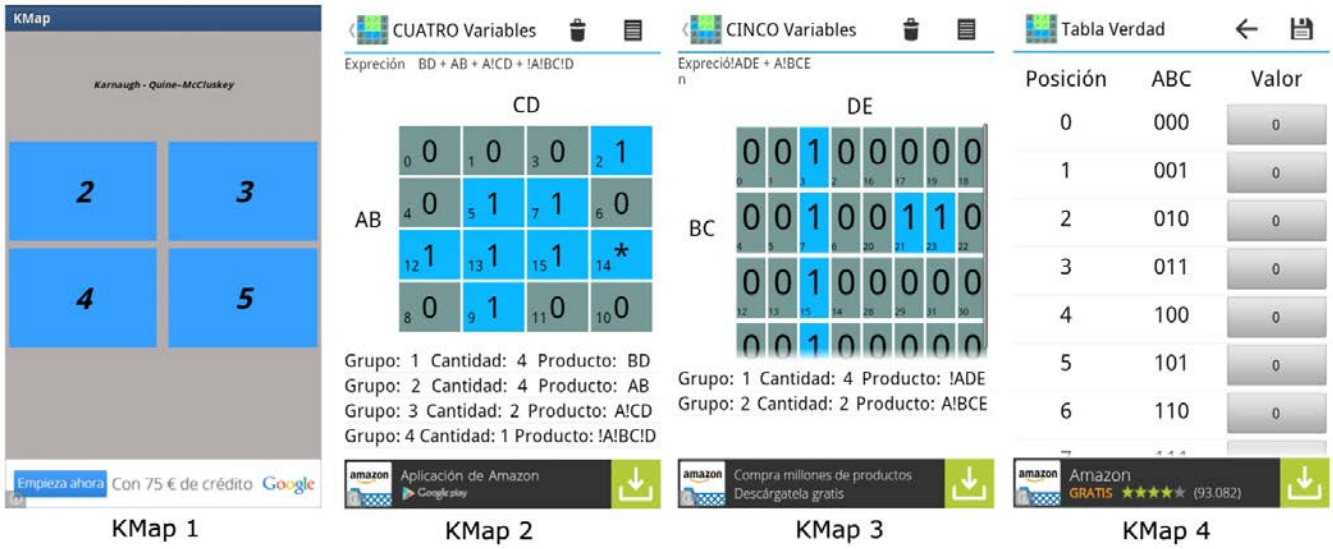

Figura 2-4: Capturas de pantalla de KMap

2.2.2 KMap Solver

El siguiente en ser analizado es el KMap Solver. Esta aplicación consiste en una pantalla a lo largo en la que se encuentran todas las opciones. La estética es buena ya que todo tiene un tamaño constante independiente del número de variables. Permite la selección de mapas de Karnaugh de entre 2 y 4 variables, de las cuales tampoco se puede cambiar el nombre. Al contrario que la anterior no incluye publicidad

En la segunda imagen se puede ver la manera de seleccionar el número de variables y la posibilidad de introducir la función lógica por tabla de verdad.

En la tercera se aprecia el mapa K de 4 variables y que tampoco muestra gráficamente las agrupaciones seleccionadas para la simplificación.

En la última se ve el mapa K de 3 variables con la simplificación escrita en la parte inferior. También parece que la simplificación funciona correctamente.

Aunque se ha comprobado como en algunos casos escoge más grupos de los estrictamente necesarios para conseguir la función mínima.

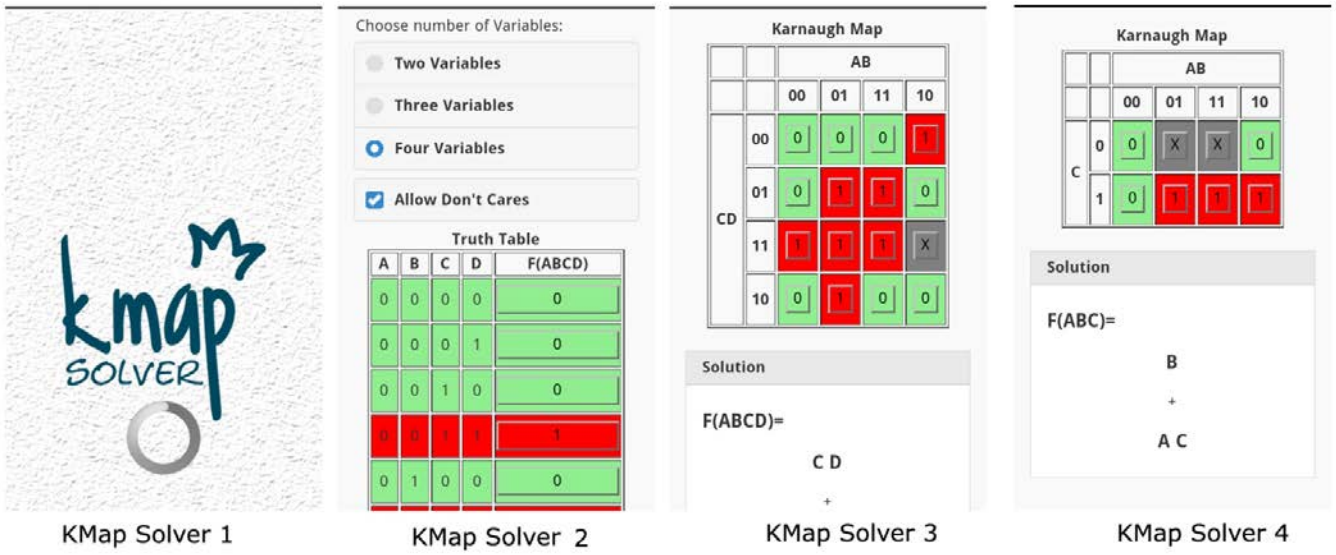

Figura 2-5: Capturas de pantalla de KMap Solver

2.2.3 KVD - Karnaugh-Veitch-Diagram

La aplicación KVD a simple vista es la más sencilla de todas en cuanto a modos de funcionamiento, parece que sólo tiene un modo que es el de 4 variables. Pero si investigas y pulsas la tecla Hardware de menú aparecen otras opciones, tal vez es un error tenerlo tan escondido ya que muchos usuarios no lo encontrarán. En ella puedes cambiar el número de variables entre 2 y 5 pero, como en las anteriores, no puedes cambiar el nombre de dichas variables. KVD tampoco tiene publicidad.

En la captura 1 se puede ver el menú que aparece si pulsas la citada tecla menú Hardware.

En la segunda imagen se ve una captura del correcto funcionamiento del mapa de Karnaugh de 4 variables. En él, se ha seleccionado que aparezcan todas las opciones de vistas que se pueden añadir. Una de esas opciones es la tabla con los pasos del algoritmo Quine-McCluskey. Encima del mapa te enseña todos los términos seleccionados de la tabla y debajo del mapa todas las posibles selecciones de los conjuntos, aunque en la segunda captura no se aprecia bien esto porque solo hay una posible manera de simplificarla con el mínimo número de términos.

En la tercera captura se ve un ejemplo de mapa K de 4 variables en el que en la parte de arriba se tienen 3 términos como todos los posibles, pero en la de abajo, solo selecciona los dos necesarios para minimizar. Con lo cual se ve el correcto funcionamiento de la simplificación.

En la cuarta captura se ve un mapa K de 5 variables en el que en la parte de abajo te propone dos simplificaciones mínimas para el conjunto seleccionado. También

se ve como simplifica correctamente. Esta posibilidad de dar dos distintas simplificaciones es una mejora importante con respecto al resto de aplicaciones.

De momento, de todas las aplicaciones estudiadas hasta ahora, esta es la mejor estéticamente hablando y la que funciona de la manera más fluida con los cambios. Además de que es la única en la que se muestran gráficamente los grupos seleccionados, lo cual no es sencillo de conseguir y hace que la aplicación sea mucho más vistosa.

Como único inconveniente, en general, cuanto más larga se hace la función simplificada, menor es el tamaño de letra en el que se representa, con lo que para cadenas grandes no se lee nada. Aún así, la aplicación ofrece una aplicación Premium de 1 euro en la que dice que este problema está solucionado.

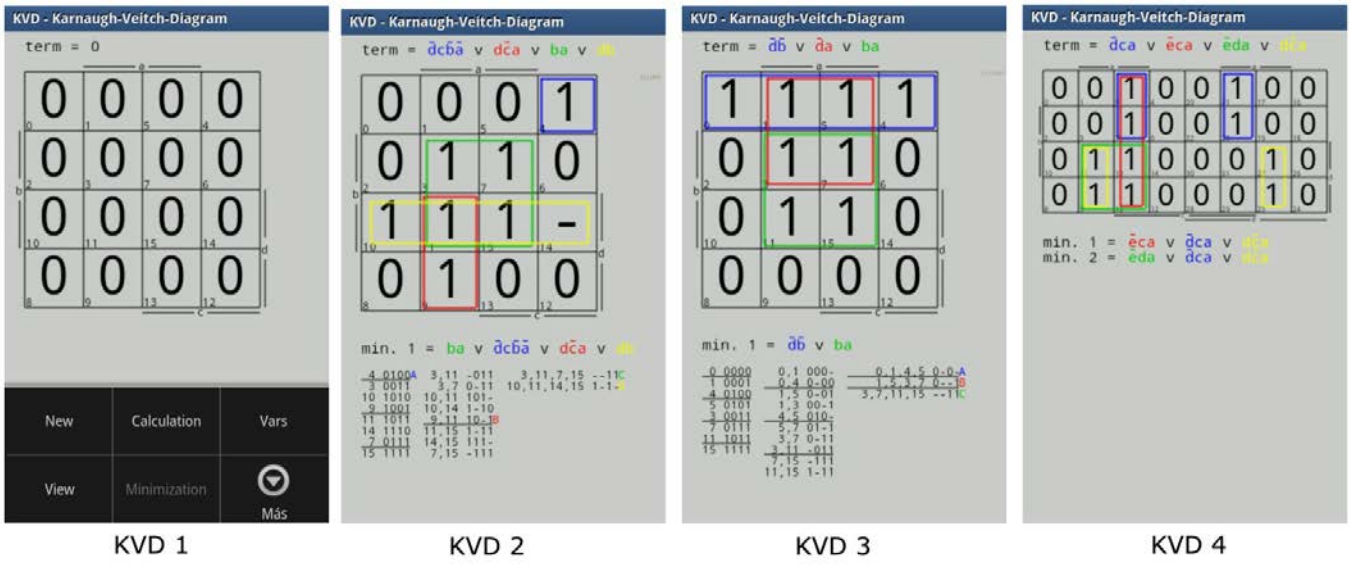

Figura 2-6: Capturas de pantalla de KVD - Karnaugh-Veitch-Diagram

2.2.4 MiniKarnaugh

MiniKarnaugh es la siguiente aplicación que se estudiará. Es una aplicación en la que nada más entrar muestra un anuncio, pero una vez que lo cierras ya no sale ninguno más. Solo dispone de una pantalla, en la que están incluidas todas las funcionalidades del programa, con lo que a primera vista queda muy cargada. Aún así, en cuestión de funcionalidades está muy bien:

- Permite introducir desde 1 a 8 variables, lo que la convierte en la mejor en este aspecto hasta ahora. Aunque si metes alguna más el programa no avisa y se cierra con error, lo cual es un problema.
- Se pueden introducir las funciones de tres maneras, por el mapa de Karnaugh de las variables que sean, por tabla de la verdad o por expresión lógica. Lo de la expresión lógica es una idea que no había en ninguna de las anteriores.

- Permite también don´t cares.

Pero también tiene algunos aspectos negativos como:

- La estética, una mejor organización de los elementos en varias pantallas hubieran mejorado la experiencia del usuario.
- La no necesidad de mostrar mapas de Karnaugh de 8 variables, que se vuelven gigantes tanto en vertical como en horizontal, tal vez sería mejor idea dar sólo la tabla de verdad o la introducción por expresión lógica.
- El problema del no control del número de variables que hace que el programa deje de funcionar.

En las siguientes capturas se ve el funcionamiento general de la aplicación. En la primera, la publicidad del inicio. La segunda un ejemplo de entrada y simplificación del mapa de Karnaugh de 4 variables. En la tercera se prueba la introducción por expresión lógica y simplificación con 3 variables. La última es un ejemplo de cómo se ve la tabla de verdad en el caso de 8 variables.

Figura 2-7: Capturas de pantalla de MiniKarnaugh

2.2.5 Boole

Esta última aplicación es muy distinta a las demás ya que sólo permite introducir funciones lógicas como expresión lógica, para su posterior traducción a: tabla de verdad, mapa de Karnaugh, función simplificada, sumatorio de minitérminos, producto de maxitérminos y diagrama esquemático del circuito. Aunque está última opción tiene trampa ya que si la eliges te dice que aún no está implementada.

En principio no tiene límite de variables y te deja elegir sus nombres, pero la realidad es que en el dispositivo en el que se ha probado a partir de las 9

variables no era capaz de calcularlo y se acababa bloqueando el dispositivo consumiendo muchos recursos de CPU. Con un mejor dispositivo seguro que se pueden simplificar funciones de alguna variable más.

Estéticamente es buena, sencilla de entender y fácil de utilizar. Aunque tiene también sus puntos negativos, el principal de ellos, que no es cómodo introducir funciones largas con el teclado QWERTY. Tampoco te muestra los grupos que utiliza para la simplificación y no tiene esa traducción automática eligiendo las casillas del mapa de Karnaugh que es la principal herramienta que se busca en este proyecto.

En la siguiente captura se pueden apreciar algunas de las características de esta aplicación. En la primera se ve un mapa K de 3 variables y su simplificación. En la segunda la traducción a minitérminos y maxitérminos de una expresión de 3 variables. Y en las dos últimas se ven las tablas de la verdad de una expresión de 9 variables y la de una tabla de dos variables que no son sólo de una letra, ya que esta aplicación permite introducir variables con palabras.

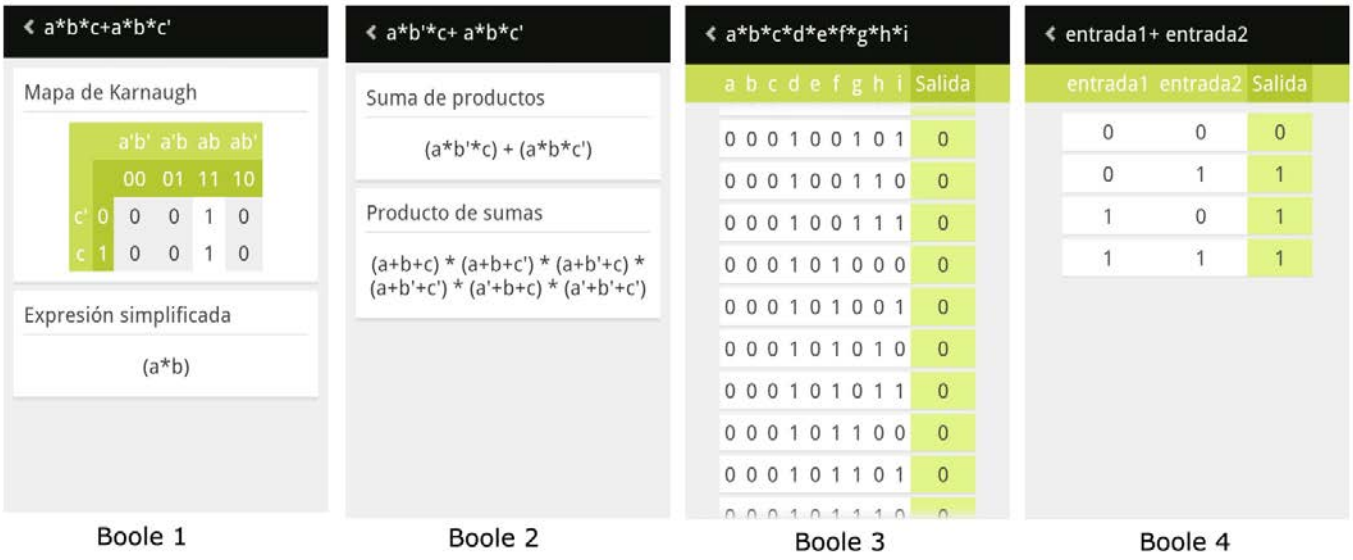

Figura 2-8: Capturas de pantalla de Boole

2.2.6 Tabla comparativa

Para conseguir un análisis más completo del mercado se ha realizado una tabla resumen en la que se pueden ver las características que tienen cada una de las aplicaciones existentes en el mercado, incluyendo la que se ha creado en este proyecto, en la última columna.

Nombre	KMap	Kmap Converter	KVD	Mini Karnaugh	Boole	Karn Map
Logo						
Número de variables	2-5	2-4	2-5	1-8	1-9	3-7
Modificar nombres variables	No	No	No	Si	Si	Si
Tiene publicidad	Siempre	No	No	Sólo un anuncio	No	No
Estética	Simplista y sencilla	Simplista y sencilla	Menú difícil de encontrar	Muy cargada y difícil	Sencilla y bonita	Intuitiva y bonita
Métodos de introducir función (núm. de variables de esa forma)						
Mapa Karnaugh	Si (2-5)	Si (2-4)	Si (2-5)	Si(1-8)	No	Si (3-5)
Tabla de la verdad	Si (2-5)	Si (2-4)	No	Si (1-8)	No	Si (3-7)
Expresión lógica	No	No	No	Si (1-8)	Si (1-9)	Si (3-7)
Métodos de salida						
Tabla de verdad	Si	Si	No	Si	Si	Si
Expresión simplificada	Si	Si	Si	Si	Si	Si
Sumatorio minitérminos	No	No	No	No	Si	Si
Producto maxitérminos	No	No	No	No	Si	Si
Esquemático	No	No	No	No	Aún no	Si
Opciones de simplificación						
Admite "don't cares"	Si	Si	Si	Si	No	Si
Simplifica correctamente	Si	No	Si	Si	Si	Si
Muestra gráficamente las agrupaciones	No (texto)	No	Si	No	No	Si
Ofrece más de una simpl. correcta	No	No	Si	No	No	No
Enseña los pasos de Quine-McCluskey	No	No	Si	No	No	No

Tabla 2-3: Comparativa de aplicaciones

3 Diseño

En el apartado de diseño básicamente se hablará de la finalidad del proyecto y de las decisiones que se han tomado con respecto a los requisitos del mismo y los motivos que incentivaron a tomar todas las decisiones.

También se hará una introducción inicial, no demasiado técnica, de las herramientas que estarán presentes en la aplicación, hablando de las limitaciones que presentan e introduciendo algunas de las clases que se utilizarán.

3.1 Finalidad del proyecto

En este proyecto el objetivo es realizar una aplicación en Android que principalmente simplifique funciones lógicas con el mapa de Veitch-Karnaugh. De manera que pueda ser una aplicación útil para los alumnos de la nueva asignatura de grado de Circuitos Electrónicos Digitales de la Escuela Politécnica Superior.

Esta aplicación ayudará y servirá de herramienta de aprendizaje a los estudiantes de dicha asignatura ya que entre otras cosas les proporcionará:

- Una manera de ver como se agrupan los términos del mapa de una manera que minimice el número de agrupaciones.
- Una manera de comprobar que las soluciones obtenidas por ellos mismos en papel son correctas.
- Herramientas traductoras y conversoras que agilizarán a veces su resolución de problemas, o en el caso del conversor binario, ayudará en alguna de las prácticas de laboratorio.

La aplicación ofrecerá básicamente 4 herramientas, más el manual de ayuda y un tutorial sobre la simplificación de funciones lógicas. Estas herramientas son:

- **Herramienta de selección de variables**: Para elegir el número de variables y sus nombres.
- **Herramienta Karn Auto**: Que simplifica en tiempo real funciones lógicas en mapa de Karnaugh. Es un herramienta en la que se ve cómo actúan dichos mapas gráficamente y de manera automática.
- **Herramienta Traductor**: Traduce y simplifica funciones lógicas desde varios métodos de entrada hasta varios de salida. Esta función admite más variables aún que la anterior, lo que la hace más completa aunque no tan gráfica.
- **Herramienta Conversor**: Para convertir entre los sistemas numéricos binario, octal, decimal y hexadecimal.

Ésta ha sido una breve introducción de las 4 herramientas, ya se explicarán con mejor detalle más adelante.

Después de analizar las aplicaciones que había en el mercado, en el estado de arte de esta misma memoria, se busca implementar una que reúna las características más interesantes de todas ellas. Además se buscará que estén implementadas de la manera más sencilla e intuitiva posible y se dispondrá un completo manual de ayuda para el usuario, manual que no incluían muchas de las aplicaciones existentes.

3.2 Elección de requisitos

En este apartado se van a argumentar todas las decisiones tomadas en el proyecto con los motivos empleados para llegar a ellas. Se explicarán desde la selección del sistema operativo hasta la de idioma.

3.2.1 Sistema operativo

En el apartado de estado del arte se hizo una introducción de las características e historia de los sistemas operativos para móviles más utilizados en la realidad. Pero en ningún momento se llegó a dar ninguna explicación de por qué se eligió el sistema operativo Android. Los motivos principales de dicha elección son:

- La mayor extensión del sistema operativo Android en móviles, muy por encima de los otros sistemas. Este hecho se veía perfectamente en la representación gráfica de la figura 2-3 que no merece la pena volver a adjuntar.

  En resumen, en el último estudio de dicha gráfica, en 2014, se tenía un 84,7% de cuota de mercado para Android, la siguiente era iOS con un 11,70% y la tercera en discordia era Windows Phone con sólo un 2,5% de la cuota.

- Otro de los motivos principales es el tema económico. Conseguir una licencia de desarrollador para el Play Store de Android cuesta un pago único de 25$, mientras que la misma licencia para App Store cuesta 99$ de cuota anual.

  Aun así, si se buscase una aplicación con la que ganar dinero podría compensar pagar la licencia de la App Store ya que la situación entre ambos sistemas operativos se resume con una frase: "Google tiene una cuota de mercado 4 veces superior a Apple, pero Apple tiene unos ingresos 4 veces superior a los de Google". Esta frase no es exactamente real pero los datos son bastante claros con que los usuarios de Apple

gastan más en descargar aplicaciones. El resultado se puede observar en la siguiente gráfica de principios de 2013:

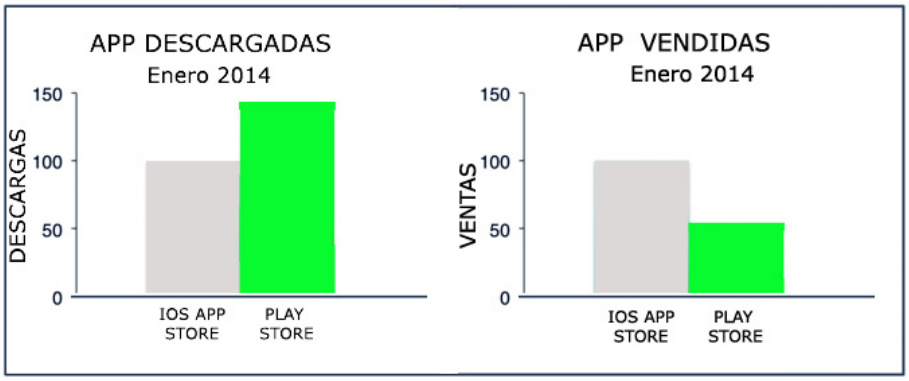

Figura 3-1: Comparación aplicaciones descargadas y vendidas.

* Datos obtenidos de: http://thenextweb.com/

A pesar de que se tienen más posibilidades de ganar dinero programando aplicaciones para Apple, este factor no es tan importante debido a que el principal objetivo de este proyecto no es económico,. Recordemos que el objetivo principal y único de estas aplicaciones es educar y motivar a los alumnos en su aprendizaje de Circuitos Digitales.

- El siguiente motivo es el lenguaje en el que se desarrolla el software de ambos sistemas. Mientras que el lenguaje Java utilizado en Android sirve además para multitud de ámbitos, como programar aplicaciones web y de escritorio. El lenguaje utilizado en Apple, el Objective C, sirve únicamente para desarrollar software de iOS.

- En Android todas las herramientas utilizadas para desarrollar la aplicación se pueden descargar gratuitamente de internet, tanto la herramienta eclipse, como el Android SDK Manager y el Android Virtual Device Manager. Se pueden encontrar en: http://developer.android.com/

  Además Android permite desarrollar sus aplicaciones y testearlas desde cualquier tipo de dispositivo, mientras que para desarrollar Apple solo se puede hacer desde un dispositivo de dicha marca.

En general Android ofrece muchas más facilidades que Apple para desarrollar aplicaciones. Se han dado una cantidad suficiente de motivos para elegir Android como sistema operativo en el que se desarrollará el proyecto.

3.2.2 Versión de Android

Decidir la versión de Android para la que se desarrollará el proyecto es extremadamente importante. Se debe de desarrollar una aplicación que sea

compatible con el mayor número de terminales posibles. Pero si eliges que sea compatible con versiones de Android muy tempranas, hay muchas herramientas que no podrás utilizar porque fueron desarrolladas para versiones posteriores. Se vuelve a adjuntar la tabla 2-2 porque será de ayuda para tomar esta decisión:

Identificador	API	Porcentaje
Froyo	8	0.7 %
Gingerbread	10	11.4 %
Ice Cream Sandwich	15	9.6 %
Jelly Bean	16	25.1 %
	17	20.7 %
	18	8.0 %
Kit Kat	19	24.5 %

Tabla 3-1: Distribución de versiones Android

* Datos obtenidos de: http://developer.android.com/

Las versiones más nuevas son las que acaparan un mayor número de usuarios, así que está claro que cualquier aplicación tiene que ser compatible con ellas, para llegar al mayor número de dispositivos distintos posible. Por ello, la decisión más importante es ver cuál será la mínima versión de API a partir de la que será compatible.

Aunque la API 8 solo contiene un porcentaje de usuarios de un 0,7% se ha decidido elegirla como mínima API posible. El proyecto que se quiere desarrollar no precisa de herramientas de versiones de Android muy avanzadas para su resolución, con lo que no limitaría demasiado elegir esa API.

Además también fue un factor relevante que las anteriores aplicaciones desarrolladas por el DSLab eligieron la API 8 también como mínima y se consideró una buena idea mantener ese requisito.

Por lo tanto la aplicación funcionará en todos los dispositivos que vayan desde la versión 2.2 en adelante.

3.2.3 Dispositivos válidos

La aplicación se va a desarrollar para que sea válida en cualquier dispositivo Android, desde una tableta de 10 pulgadas a un terminal móvil de 2,7 pulgadas. Se va a intentar en todo momento que se ajuste al tamaño de pantalla de cualquier dispositivo en el que la aplicación sea instalada.

En el caso de pantallas en las que hubiese alguna posibilidad de que no entrarán a lo largo o a lo ancho por ser demasiado pequeñas se añadirán elementos ScrollView para que el usuario de dicho dispositivo se pueda desplazar en la dirección que sea necesaria. Se ha añadido un elemento ScrollView en la gran mayoría de las ventanas de la aplicación para que funcione siempre correctamente.

También habrá pantallas en las que todo deba presentarse en un mismo pantallazo, como es el caso de la herramienta Karn Auto que más adelante se describirá. En este tipo de pantallas la aplicación consultará el tamaño del terminal tanto en ancho como en largo y verá cuál de las dos es la que va a delimitar las dimensiones. Y a partir de esa medida delimitadora construirá todo lo demás, de manera que siempre entrará en cualquier pantalla y será más grande o más pequeño en función del dispositivo.

Luego hay algunas pantallas, como es el caso del lienzo de la clase Canvas en que se pinta el esquemático, en el que si se hiciese que todo entrará a la vez en la pantalla, cuanto más grande fuera el circuito, más pequeñas serían todas las puertas y en algunos casos no se distinguiría nada. Por ello se ha decidido que en este tipo de pantallas tan variables, las puertas y los elementos tengan siempre las mismas dimensiones en función del ancho y del largo de la pantalla e incluyan unos elementos ScrollView para desplazarse a lo largo de todo el lienzo.

Para facilitar las cosas y aprovechar al máximo los dispositivos se ha decidido forzar a la aplicación su modo de funcionamiento a "portrait". Esto quiere decir que no se girará la pantalla aunque se desee. Esta consideración se ha realizado para asegurar una mayor calidad en el visionado general de la aplicación. Hay una excepción de esto y en el caso de Karn Auto de 5 variables, se creyó oportuno forzarlo a "landscape", pantalla horizontal, para aprovechar mejor el espacio disponible.

3.2.4 Idioma

Aunque el tutor propuso que el lenguaje de toda la aplicación fuese únicamente el inglés para alcanzar un mayor público potencial. Se pensó que dado a las características que ofrece Android esto era innecesario.

Android permite crear un fichero strings.xml en el que almacenar todas las cadenas de texto de la aplicación. A cada cadena se le asocia una etiqueta que nos facilite el acceso a ella como recurso. Y en cada sitio en el que se quiera introducir una palabra o conjunto de palabras, en lugar de escribirla a mano se accede al índice de la etiqueta. Esta herramienta de Android nos permite varias ventajas:

- Supongamos que después de haber terminado la aplicación, de decenas de archivos XML y java y miles de líneas de código, se quiere cambiar el

texto de todos los botones en los que se escribió, por ejemplo, "continuar" por "ok". Se tendría que emprender la labor enorme de revisar en todos los archivos XML de los layouts y en todas las clases Java la línea que utilizamos para cambiar ese texto estático. Si se hubiera utilizado la manera explicada anteriormente, sólo sería necesario acceder al fichero de srings.xml y cambiar el valor de la etiqueta correspondiente a los botones por el nuevo valor.

- Otra funcionalidad que ofrece este sistema es la del cambio de idioma. Si se dispone de ficheros de cadenas de texto y se quiere añadir un nuevo idioma a la aplicación, sólo hay que crear un nuevo fichero de textos y definir todas las etiquetas de la aplicación con su cadena en el nuevo idioma. Será el propio dispositivo, el que indicará a la aplicación en qué idioma se encuentra, y si lo hay, la aplicación indexará directamente las etiquetas del fichero de ese idioma. En caso de que no se disponga se indexarán a las cadenas del idioma por defecto.

Gracias a esta herramienta de Android se ha decidido que es mucho más útil escribir la aplicación en inglés por defecto, pero añadir un fichero de cadenas en español. De momento sólo se añadirá el fichero en español ya que viendo las descargas de las anteriores aplicaciones del DSLab la mayoría de sus descargas se produjeron en países de habla hispana. Para el resto de países que no sean de habla inglesa, ni española, su programa se ejecutará por defecto en inglés, como las anteriores aplicaciones de DSLab.

3.2.5 Resumen de especificaciones

En resumen, esta aplicación funcionará tanto para dispositivos móviles como para tabletas con sistema operativo Android.

Podrá ser instalado en dispositivos con la versión 2.2 de Android (Froyo) y todas las posteriores.

Se adaptará a todo tipo de pantallas y tamaños.

El idioma será por defecto el inglés, peso si es lanzado desde un dispositivo que esté en español cambiará de idioma automáticamente y sin preguntar.

3.3 Herramientas de la aplicación

En este apartado se van a describir todas las principales herramientas que se implementarán en la aplicación. Se añadirá mucho más detalle que en las anteriores descripciones que se han realizado hasta el momento.

3.3.1 Tutorial

El tutorial de la aplicación consiste en una sucesión de imágenes estáticas que ofrecen una breve explicación del uso de los mapas de Veitch-Karnaugh para la simplificación de funciones lógicas. Estas explicaciones de cómo se debe simplificar y cómo no, serán muy útiles para entender la base de las herramientas ofrecidas en la aplicación, sobre todo Karn Auto.

Mencionar que esto no proporciona un manual para aprender a utilizar la aplicación, ya que éste se encuentra en las herramientas de ayuda.

Las imágenes se mostrarán en pantalla completa y para pasar entre las distintas páginas se deslizará el dedo por pantalla.

Las imágenes del tutorial han sido diseñadas por el profesor Eduardo Boemo.

3.3.2 Herramienta de selección de variables

3.3.2.1 Descripción

La herramienta de selección de variables es una funcionalidad de gran importancia en este proyecto. Es muy útil dar a elegir las variables con las que se trabajará en el resto de herramientas ya que fijar parámetros nunca es bueno.

Por ejemplo en la clase de Circuitos Digitales de la escuela se suele utilizar el conjunto de variables DCBA, por lo que se escogió como conjunto por defecto. Pero esto no quiere decir que siempre se utilice ese conjunto en concreto. Si se deja fijado DCBA como en la mayoría de aplicaciones encontradas en el Play Store se entorpecería mucho al usuario a la hora de simplificar funciones con otras variables.

Una vez dentro de la selección de variables, aparece una lista con todos los conjuntos guardados previamente en ese dispositivo. Se guarda utilizando un mecanismo que ofrece Android llamado SharedPreferences, de las que se hablará en el siguiente apartado.

En esa lista hay uno de los elementos que aparece marcado en amarillo, ese es el elemento seleccionado en la actualidad. Si abriéramos una de las otras herramientas ese sería el conjunto de variables que se utilizaría.

Debajo de la lista hay un botón para crear un nuevo conjunto que si se elige, al igual que si eliges uno de los elementos de la lista, te salta una ventana de diálogo para que edites ese o el nuevo elemento de la lista.

En la ventana de edición puedes elegir la letra que quieres que vaya en cada una de las posiciones. También dispones de unos botones para cambiar el número de variables de ese conjunto. En la parte de abajo hay 3 botones, uno para elegir y guardar el conjunto editado, otro para borrar el conjunto y el último para volver.

3.3.2.2 La clase SharedPreferences

La clase SharedPreferences es un mecanismo que proporciona Android para almacenar datos. Se suele utilizar para datos limitados, como las preferencias de una aplicación, el nivel actual de un juego, los colores de pantalla, etc.

Cada preferencia se almacena con forma de etiqueta (identificador único) y valor asociado y los datos no se guardan en un fichero binario de base de datos, sino en ficheros XML.

El manejo de estas preferencias es muy sencillo. La clase SharedPreferences es la que se encarga de toda la gestión.

Se ha elegido este método para guardar la mayoría de los datos del programa, como los conjuntos de variables, los estados de los mapas, los estados de traducción, etc. Los motivos de esta elección son:

- En esta aplicación no se necesita demasiada memoria para guardar datos con lo que las SharedPreferences serían suficiente.

- Si se hubiese elegido otro método de almacenamiento como memoria interna o utilizar la tarjeta SD, la aplicación hubiera necesitado pedir permisos al usuario. Cuando te descargas una aplicación de Play Store antes de bajarla se pide al usuario que acepte los permisos que requiere la aplicación. Un buen programador debe de minimizar el número de permisos que requerirá su aplicación ya que los usuarios tienden a fiarse menos de las aplicaciones que requieren demasiados permisos. Por lo que se decidió utilizar SharedPreferences para intentar reducir el uso de estos.

3.3.2.3 Limitaciones

En esta herramienta se ha decidido que sólo se puedan crear conjuntos de 3 a 7 variables, ya que menos de 3 son pocas variables como para que las herramientas hicieran un trabajo útil y con 7 variables es suficiente para el objetivo que se tiene. Se podría dejar la posibilidad de aumentar el número de variables para un futuro o se podría hacer una versión Premium para usuarios que quisieran un número de variables aún mayor.

La otra limitación es que por razones obvias no se pueden elegir dos variables con la misma letra. En caso de que se elijan, el programa avisará por pantalla del error y no se podrá continuar hasta que el usuario lo cambie.

3.3.4 Herramienta Karn Auto

3.3.4.1 Descripción

La herramienta Karn Auto es la más importante de la aplicación, era el objetivo principal del proyecto y la que acaba dando el nombre a la misma. Se trata de un simplificador de funciones lógicas en tiempo real por mapa de Karnaugh.

Nada más entrar a la herramienta te da a elegir entre los distintos números de variables permitidas aplicadas al conjunto de variables seleccionado. En esta herramienta no importa el número de variables que indiques en el selector de conjuntos. Ya que los posibles números de variables con los que se puede utilizar Karn Auto están ya definidos.

Además, por si acaso se desea cambiar el conjunto de variables que se eligió te muestra siempre un acceso directo a la herramienta selectora.

Una vez seleccionado el número de variables deseado, la aplicación pasa a una nueva pantalla, que se trata de un lienzo de la clase Canvas. La clase Canvas se explicará en el siguiente punto.

En el lienzo se representa un mapa de Karnaugh con tantas posiciones como sean necesarias. Cada una de las posiciones del mapa puede ser pulsada, cambiando su estado entre las 3 posibilidades que ofrece: 0, 1 o X. Según se van introduciendo, se va traduciendo a tiempo real en la parte inferior del mapa.

A cada agrupación de la simplificación se le asocia un color y se marcan en el mapa con ese color los elementos que pertenecen a ella. A la vez la simplificación de cada grupo se escribe en la parte inferior de ese mismo color.

Por último mencionar que el estado actual del mapa es guardado en todo momento en las ya explicadas SharedPreferences y se guarda siempre y cuando no borres los datos de la aplicación o cambies las variables seleccionadas en el selector, en dicho caso no merece la pena seguir guardando el estado de las antiguas variables.

3.3.4.2 La clase Canvas.

La clase Canvas es usada para crear interfaces de bajo nivel. Esta clase contiene los mecanismos necesarios para usar la pantalla del dispositivo como si fuese un

lienzo para pintar. Se tiene un control total sobre la pantalla a nivel de píxel. Esta clase es muy usada por los desarrolladores de juegos por ello.

Canvas ha sido la clase elegida para este cometido porque permite todo lo necesario para dibujar el mapa K y su estado en cada momento. Cuando se pulsa una casilla del mapa se cambia el estado y se repinta otra vez el mapa entero. La actualización se realiza muy rápido por lo que este método queda muy vistoso y produce una muy buena experiencia de usuario.

3.3.4.3 Limitaciones

A pesar de que el algoritmo de simplificación funciona correctamente para números grandes de variables, la herramienta Karn Auto solo dejará las opciones de 3, 4 y 5 variables. El motivo de limitar 5 variables como el máximo para esta herramienta, es que un mapa de Karnaugh de un número alto de variables deja de tener sentido. Es importante entender que el algoritmo simplificador es el algoritmo Quine-McCluskey y que los mapas K son una herramienta gráfica que ayuda a realizar estas reducciones. Con 6 variables y en adelante la herramienta gráfica deja de ser del todo útil.

Por los motivos previamente descritos, si se quiere simplificar una función lógica con un número de 6 o 7 variables habrá que recurrir a la herramienta Traductor.

En los mapas de Karnaugh de 3 y 4 variables se ha creído conveniente introducir colores para mostrar las agrupaciones y la simplificación de cada uno, mientras que en el de 5 variables se creyó inoportuno porque los cuadros son demasiado pequeños como para que se vea correctamente.

3.3.5 Herramienta Traductor

La herramienta Traductor se crea por la necesidad de un medio alternativo a los mapas de Karnaugh para introducir funciones lógicas para su posterior simplificación. Cuando la función es demasiado grande se ofrecerá la posibilidad de introducirla por varios métodos distintos, en los siguientes puntos de la memoria se expondrán los más importantes. Al contrario que en la herramienta Karn Auto aquí la traducción no será en tiempo real, sino que se realizará a continuación de pulsar el botón correspondiente.

3.3.5.1 Introducir funciones por tabla

La primera alternativa que se implementó para introducir funciones lógicas fue por medio de tablas. Con este modo de introducción se añadieron 3 formas: Tabla de verdad, sumatorio de minitérminos y producto de maxitérminos.

La primera de ellas, la tabla de verdad, se trata de una tabla con dos columnas. En la primera de ellas están todas las combinaciones con ese número de

variables, y la segunda se encuentra por defecto llena de ceros. Si se pulsa alguno de los ceros cambia de estado. Se pueden introducir unos o equis a lo largo de toda la tabla y cuando se haya construido la tabla de verdad completa se puede pulsar la tecla traducir, que se encuentra en la parte inferior de la tabla.

Las otras dos, tienen en la primera columna todas las posibilidades de minitérminos o maxitérminos respectivamente. Si se pulsa un "1" en ellas, quiere decir que ese término estará presente en la función final y si se pulsa una "X" quiere decir que no importa el valor que tome.

En todas las tablas existe también una tecla de reset que lleva a la tabla a su estado inicial. El estado se guardará siempre y cuando no se cambie de conjunto de variables o siempre y cuando no se pulse traducir en ninguno de los otros distintos métodos de traducción.

3.3.5.2 Introducir funciones por expresión lógica

El último método de introducción es por expresión lógica. En él también se utiliza la clase Canvas previamente comentada para pintar en el lienzo una serie de teclas a modo de calculadora. El resto de aplicaciones estudiadas utilizaban el teclado del dispositivo para introducir funciones lógicas, lo que llevaba a dos problemas importantes:

- Había que estar cambiando continuamente del teclado de letras al teclado de símbolos para intercalar operaciones y variables, lo cual era muy molesto.
- No quedaba muy claro cuáles eran los símbolos adecuados para indicar las distintas operaciones lógicas, lo que causaba muchos errores.

Por estos motivos se decidió que crear un teclado propio era la mejor decisión posible para introducir las funciones. El teclado que se ofrece al usuario dispone de los siguientes elementos:

- Las letras del conjunto de variables que el usuario tenga seleccionado en ese momento.
- Los valores 0 y 1, por si el usuario necesita introducirlos en alguna situación.
- Paréntesis de apertura y de cierre para cambiar el orden de las prioridades de las operaciones.
- Las distintas operaciones lógicas que se pueden ver en la siguiente tabla:

Símbolo	Función	Prioridad
¬	NOT	1
*	AND	2
/	NAND	2
+	OR	3
-	NOR	3
#	XOR	3

Tabla 3-2: Símbolos y prioridades de las operaciones lógicas

Además del botón de traducir, existe un botón de ayuda para facilitar al usuario a entender las normas que se siguen para introducir una expresión correcta para el sistema. También existe un botón "borrar" para eliminar el último término introducido en la expresión.

3.3.5.3 Traducciones

Una vez pulsado el botón traducir en cualquiera de las funciones anteriores se salta a una nueva actividad en la que se podrá observar la función introducida vista desde los demás puntos de vista.

Esta pantalla se ha construido utilizando la vista ViewPager de Android de la que se hablará con más detalle en uno de los siguientes puntos. Se trata de una serie de ventanas consecutivas entre las que se puede cambiar de una a otra deslizando el dedo. Las distintas ventanas que se muestran son las siguientes:

- En la primera ventana se vuelve a mostrar la función introducida, para poder saber en todo momento y sin tener que volver atrás, la función de la que se están mostrando las traducciones.
- La segunda muestra la tabla de verdad.
- La tercera muestra el sumatorio de minitérminos.
- La cuarta muestra el producto de maxitérminos.
- La quinta muestra la función simplificada y un botón para acceder a su representación esquemática.

3.3.5.4 Esquemático

La representación del esquemático de la función simplificada es una de las funcionalidades que no ofrecía ninguna de las aplicaciones ya existentes en internet. Se trata de un lienzo Canvas en el que se dibujarán todas las puertas lógicas y conexiones que representan dicha función simplificada. La manera de dibujar las funciones ya se desarrollará más adelante en el apartado de desarrollo.

3.3.5.5 ViewPager

La vista ViewPager es uno de los numerosos patrones de diseño que ha añadido Android en sus versiones más recientes. Este en concreto ha sido incluida desde la versión Ice Cream Sandwich, aunque se puede utilizar en versiones anteriores gracias a algunas librerías de compatibilidad.

Esta vista consigue un aspecto moderno, atractivo y más limpio para la aplicación ya que permite desplazarse entre las diferentes páginas de la aplicación con un simple gesto del dedo. Además evita el uso de botones o enlaces en los diferentes menús.

Uno de los ejemplos más famosos de esta vista es el que se puede ver en el Play Store de Google, aunque cada vez se encuentra en muchas más aplicaciones.

3.3.5.6 Limitaciones

Con respecto a la introducción de funciones mediante expresión lógica se tuvo que introducir un límite de tamaño para dicha expresión. Dicho límite queda marcado por la longitud de la pantalla dibujada en el Canvas y se trata de un total de 80 caracteres. Se podría fácilmente permitir que aceptara más aún, pero se deja como posible mejora futura o para una versión Premium de la aplicación.

A la hora de traducir funciones lógicas, cualquier expresión errónea introducida con las teclas, podría producir efectos fatales en la aplicación. Por lo que se requiere de un control de errores exhaustivo que no permita que el sistema no funcione. Además debería informar el tipo de error por pantalla para que el usuario pueda corregir su error y aprender más fácilmente las reglas de introducción de funciones.

En el esquemático para no limitar el tamaño del lienzo al de la pantalla del dispositivo se tendrá que añadir ScrollView en todas las direcciones, es decir, uno para desplazamiento horizontal y otro para vertical.

3.3.6 Herramienta Conversor

3.3.6.1 Descripción

Esta herramienta es la más sencilla de todas ya que solo se trata de un traductor entre algunos de los sistemas numéricos más utilizados. Estos son:

- Binario.
- Octal.
- Decimal.
- Hexadecimal.

El motivo por el que se ha implementado esta herramienta es que en algunas de las prácticas del laboratorio de la asignatura de Circuitos Digitales de la EPS, los alumnos necesitan convertir entre estos sistemas de medida continuamente. Por ello se ha creado esta herramienta con la que realizar conversiones de la manera más rápida y cómoda posible.

Se trata de una alternativa más eficiente que los métodos que utilizan los alumnos actualmente. Por un lado la calculadora de Windows obliga a estar cambiando la ventana continuamente para apuntar los datos en otro sitio. Y por otro lado, en las calculadoras de mano científicas, hay que cambiar de modo repetidas veces para realizar las traducciones.

3.3.6.2 Limitaciones

Existen algunas limitaciones a tener en cuenta en el conversor:

- Hay que tener cuidado con que no se introduzcan valores ajenos a cada sistema numérico. Por ello se ha desactivado el teclado de los dispositivos para que no se pueda utilizar.
- Solo se incluyen teclas para los 16 caracteres que existen en el sistema hexadecimal y para el resto de sistemas sólo se dejan activadas las teclas de ese sistema en concreto.
- Como las variables en Java tienen un tamaño en concreto hay que tener cuidado al realizar la conversión. Cuando se supera dicho tamaño puede haber problemas y el programa se podría quedar bloqueado, así que para evitarlo hay que controlar que no se supere e indicarlo por pantalla.

3.3.7 Herramientas de ayuda

Aunque todas las herramientas del proyecto se han realizado de la manera más intuitiva posible, siempre conviene facilitar las cosas al usuario para que no tenga problemas a la hora de manejar la aplicación.

Por ello se ha creado un manual completo de ayuda que incluye los siguientes apartados:

- Eligiendo conjunto de variables.
- Herramienta Karn Auto.
- Herramienta Traductor.
- Introduciendo expresiones lógicas.
- Introduciendo expresiones por tablas.
- Herramienta Conversor.

Estas pantallas de ayuda son textos estáticos con algunas imágenes para orientar a los usuarios.

3.3.8 Herramientas de menú hardware

Todos los dispositivos Android disponen de una tecla de menú Hardware que en la mayoría de los programas hacen que salga un menú desplegable. Para seguir con la estética del resto de aplicaciones del DSLab se ha incluido este menú para que en cualquier pantalla de la aplicación se pueda acceder a dicho menú desplegable.

Este menú constará siempre de dos opciones:

- Acceso directo al manual de ayudas previamente descrito, por si el usuario tiene alguna duda no tenga que salirse de donde se encuentre.
- Una pantalla de "acerca de" con los datos más importantes del proyecto.

4 Desarrollo

En el apartado de desarrollo se va a profundizar más en los detalles de implementación de la aplicación. Al contrario que el apartado de diseño que era más parecido a un manual, en éste se hablará de las formas en que se han implementado las pantallas más difíciles de la aplicación y de los principales problemas y retos afrontados durante el proyecto.

4.1 Pasos previos

Programar aplicaciones en Android tiene una dificultad relativa, depende mucho de la experiencia previa del programador. En el caso de un desarrollador experimentado con dominio de por lo menos un lenguaje de programación le será relativamente sencillo, pero en el caso de una persona sin ningún conocimiento de programación hay varios pasos que se recomienda hacer antes de empezar a realizar código en Android.

- Como para cualquier lenguaje conviene tener un conocimiento previo y abundante de los fundamentos de programación, no importa a que lenguajes estén orientados.
- La base de Android es principalmente Java que es un lenguaje orientado a objetos. Es necesario entender la filosofía de este tipo de lenguajes y conveniente entender los conceptos que introducen, como la herencia, el polimorfismo, abstracción, encapsulamiento, etc.
- Una vez se tenga la base comentada anteriormente ya se puede instalar el entorno de desarrollo y continuar con los siguientes primeros pasos en ella.
- Antes de ir a por una aplicación gigante conviene intentar realizar multitud de aplicaciones pequeñas que tengan distinto objetivos, como por ejemplo: Practicar la creación de layouts, introducir botones que produzcan cambios, creación de nuevas actividades, comunicación entre actividades y un largo etcétera.
- Cuando ya se esté preparado, se puede comenzar.

En nuestro caso particular ya se tenía un amplio conocimiento de los fundamentos de programación aprendido duramente a lo largo de la carrera por lo que ese paso no se necesitó.

Entre los muchos lenguajes utilizados, el Java se impartió en una asignatura optativa de tercer curso en la que se aprendió muchísimo de dicho lenguaje y de su filosofía. Aun así al hacer más de dos años de esto, se necesitó un amplio repaso en este tema.

Una vez conseguido el entorno de desarrollo se siguieron un montón de tutoriales por Internet, uno de los más completos que se realizó desde el principio hasta

casi el final fue un curso de Android que se encontraba en la red publicado por la Universidad Politécnica de Valencia, el cual recomiendo a cualquiera que desee aprender.

Después de realizar este curso y muchos tutoriales más, se pasó a comprobar si el proyecto y su algoritmo principal, el Quine-McCluskey, eran factibles de conseguir. Una vez comprobado que los objetivos de la aplicación eran posibles de alcanzar se presentó el anteproyecto y se empezó a trabajar.

A pesar de haberse realizado infinidad de tutoriales, es imposible que a la hora de empezar a programar una aplicación difícil, no te aparezca ninguna duda. Según aparecían problemas concretos durante el desarrollo, se utilizaba Google para encontrar respuestas a cada uno. Una herramienta muy utilizada en este paso fue el canal de youtube "slidenerd", el cual supuso una ayuda fundamental para que este proyecto llegase a su fin.

Además de estas páginas habituales hay muchas más páginas consultadas, se les agradecerá en el apartado de referencias de esta misma memoria.

4.2 Instalación del entorno de desarrollo

El entorno de desarrollo consta de varias herramientas necesarias para la programación Android:

- **Eclipse**: Se trata del entorno de desarrollo más recomendable para Android, es libre y soportado por Google. Además fue el utilizado por los desarrolladores para crear Android.
- **Android SDK Manager**: Para actualizar las herramientas e instalar las nuevas versiones que vayan apareciendo. Te aparece una lista con todos los paquetes disponibles para que elijas los que quieras instalar.
- **Android Virtual Device Manager**: A la hora de emular o depurar la aplicación se puede hacer en un dispositivo físico conectándolo por medio del cable. Pero si se desea hacer pruebas en distintos dispositivos el AVD te permite emular la aplicación en tu ordenador en cualquier terminal con Android. Así puedes probar tu aplicación en una gran variedad de teléfonos o tabletas en cualquier versión.
- **Máquina virtual de Java**: Este software va a permitir que se ejecute código Java en tu equipo. Esta máquina virtual también es conocida como entorno de ejecución de Java o Java Runtime Enviroment (JRE).

4.3 Esquema general de la aplicación

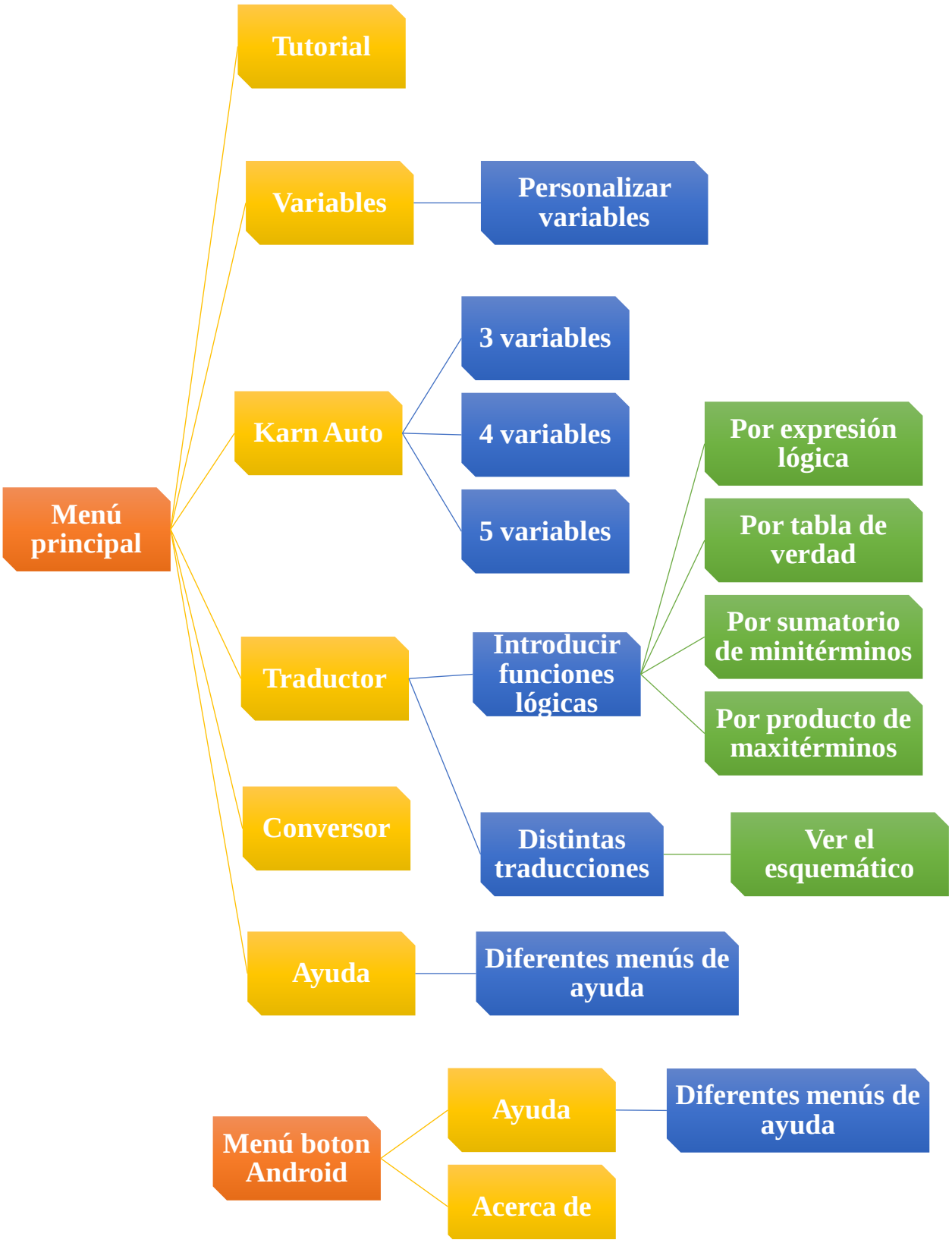

Figura 4-1: Esquema general de la aplicación.

4.4 Conceptos clave

En este apartado se van a explicar algunos conceptos importantes de Android para comprender mejor los siguientes apartados, son conceptos básicos que se utilizarán continuamente a lo largo del proyecto.

4.4.1 Recursos de una aplicación.

Cuando se programa en Android siempre conviene externalizar los recursos del código. Algunos ejemplos son las imágenes, los estilos y los textos. Gracias a la utilización de estos recursos se pueden definir distintos recursos en función del tipo del dispositivo sin necesidad de modificar el código. Aunque también se puede especificar en algunos casos un recurso genérico o por defecto. Todos ellos se guardan en la carpeta /res.

Algunos ejemplos básicos de la aplicación de los recursos son:

- Se crearán actividades con distintos tamaños dependiendo del tamaño de pantalla que tenga el dispositivo, no es lo mismo un móvil que una tableta.
- Se crearán distintos recursos de cadenas de textos para que la aplicación soporte distintos idiomas.
- Se añadirán imágenes de distintas resoluciones para que se representen según la calidad del dispositivo.
- Se definirán estilos para los distintos Widgets de la aplicación para evitar repeticiones de código para Widgets iguales.

4.4.2 La clase Activity.

Una actividad representa una pantalla con la que un usuario puede interactuar. Cada actividad lleva asociada una interfaz de usuario. Para crear una actividad basta con crear una clase que herede de la clase Activity y sobreescribir algunos de los métodos de su ciclo de actividad, que ya se verán en el siguiente punto.

Desde una actividad se pueden lanzar otras actividades, de manera que las anteriores actividades quedan guardadas en memoria en una cola de tipo LIFO (Last Input, First Output) De manera que al pulsar la tecla de atrás se van cerrando las actividades actuales y reapareciendo las actividades anteriores .

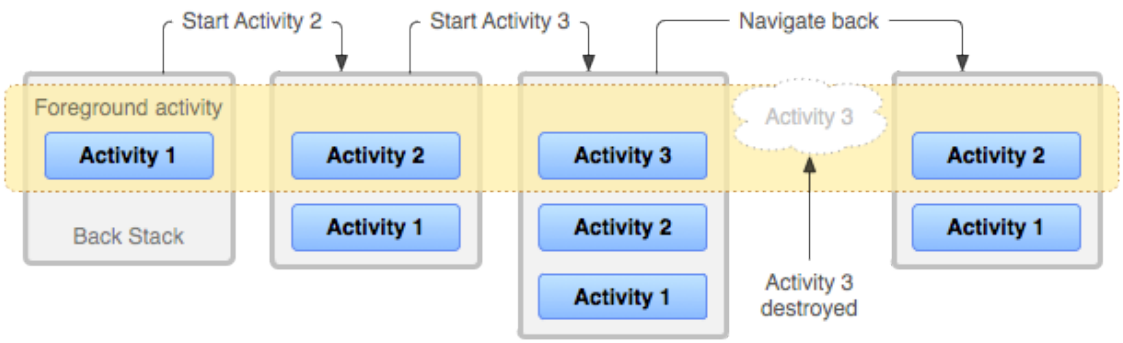

Figura 4-2: Diagrama del botón atrás.

* Imagen obtenida de: https://stuff.mit.edu/afs/sipb/project/android/docs/guide/components/tasks-and-back-stack.html

Por último mencionar que todas las actividades de una aplicación deben de aparecer registradas en su fichero Manifest, porque de lo contrario no funcionarán.

4.4.3 Ciclo de vida de una actividad

El ciclo de vida de una actividad consiste en los diferentes estados por los que puede pasar una actividad. Existe una serie de métodos que se llaman cuando se produce un cambio entre un estado y otro. Los diferentes estados son:

- **Resumed:** Estado en el que se encuentra la aplicación cuando está en primer plano.
- **Paused:** Estado en el que la actividad se encuentra parcialmente visible, porque hay un elemento que se encuentra encima.
- **Stopped:** Estado en el que la actividad está completamente oculta pero aún sigue viva.

El ciclo de vida y los métodos que se ejecutan entre los diferentes estados se pueden ver en el siguiente diagrama:

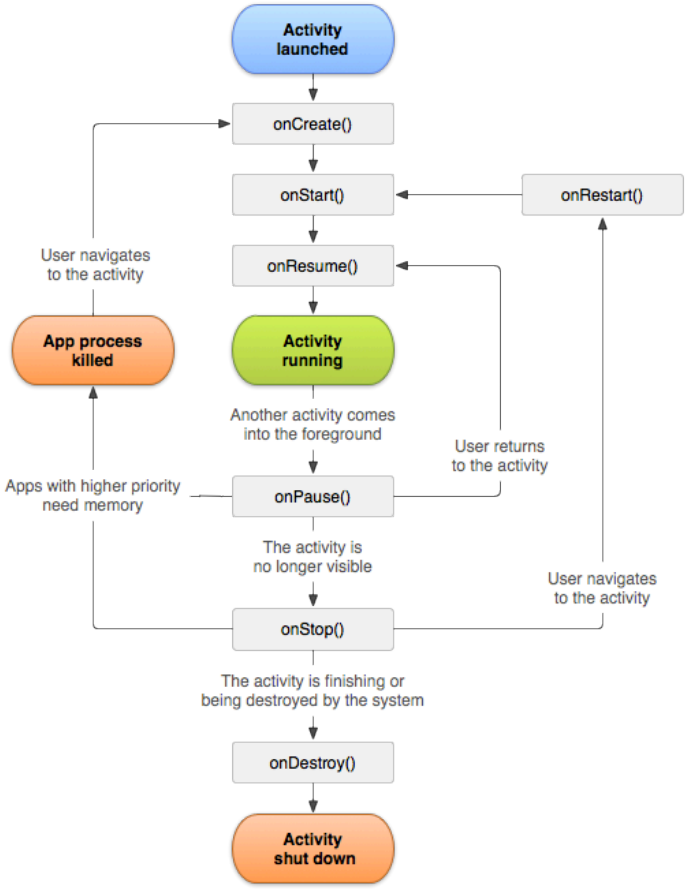

Figura 4-3: Diagrama del ciclo de vida de una actividad.

* Imagen obtenida de: https://stuff.mit.edu/afs/sipb/project/android/docs/guide/components/activities.html

El método más importante es el onCreate(), el cual es llamado cada vez que se inicia una actividad. Este método deberá llamar siempre al método setContentView que será el encargado de cargar la interfaz gráfica. Normalmente ese método llama a un recurso de tipo layout aunque cuando se utiliza la clase Canvas lo que se añade es el lienzo.

4.4.4 Vistas

En este apartado se describirán las dos clases fundamentales a la hora de realizar layouts. Se trata de las clases View y ViewGroup las cuales están presentes en cualquier interfaz Android.

La primera de ellas la clase View representa un bloque básico y elemental a la hora de realizar una interfaz gráfica, también se llaman Widgets. Algunos de los más utilizados en este proyecto son: Button, TextView, EditText, ListView, Spinner, etc.

La segunda clase es el ViewGroup, también llamados layouts que representan un conjunto de Views o Widgets y la manera en la que se representarán en conjunto. Los más utilizados en este proyecto son: el LinearLayout, el RelativeLayout y el TableLayout.

4.4.5 Adaptadores.

La clase Adapter consiste en elementos avanzados que se utilizan para conseguir vistas dinámicas. Ya se explicarán con más detalle cuando se hable de alguna actividad del proyecto en la que se utilice.

4.4.6 Los mensajes Toast.

En Android existen multitud de maneras de notificar mensajes e interactuar con el usuario. Una de las más sencillas son los denominados Toast. Son mensajes que se muestran por pantalla y que pasados ciertos segundos vuelven a desaparecer sin más.

A lo largo del proyecto se utilizarán para avisar al usuario en multitud de actividades. Desde explicar por qué está mal escrita la expresión lógica introducida, hasta para indicar que no se puede aumentar más el número de variables.

4.4.7 Clases ya comentadas en el apartado de diseño

A modo de recordatorio, se debe hacer una especial mención a las clases Canvas, los SharedPreferences y la clase ViewPager que fueron mencionadas en el apartado de diseño.

4.4.8 El fichero Manifiesto

Se trata del fichero más importante de todos los proyectos. Este archivo presenta la información que no puede faltar sobre la aplicación. Algunos de los datos que incluye son:

- El nombre del paquete Java para identificar a la aplicación.
- Descripción de todos los componentes de la aplicación. En este apartado deben de describirse todas las actividades de la aplicación con sus características.
- Las librerías de las que requiere la aplicación para funcionar.
- Especificación de los permisos que tiene la aplicación. En el caso de Karn Map ninguno.
- Declaración de la mínima versión de API en la que funciona la aplicación. En el caso de nuestra aplicación, la API 8 correspondiente a Froyo.

4.5 Detalles de implementación

En este apartado se van a detallar las cuestiones más difíciles de implementar a lo largo del proyecto, así como los problemas derivados de ello y las soluciones ofrecidas. Para mayor claridad se dividirán estas explicaciones por los distintos módulos y actividades de la aplicación.

4.5.1 Tutorial

Como ya se explicó anteriormente el tutorial consiste en una serie de imágenes que se van mostrando en la actividad.

Para desarrollarlo se ha creado un lienzo Canvas que simplemente representa las imágenes ajustadas al tamaño de la pantalla. Las imágenes se seleccionan con sus índices utilizando los recursos de la aplicación.

El lienzo Canvas también debe de detectar los eventos que se producen en la pantalla para cambiar la imagen actual por la siguiente o la anterior.

4.5.2 Herramienta de selección de variables.

El layout que compone el selector de variables, a simple vista es un texto en la parte superior, dos botones en la parte inferior y una lista en medio. Pero como se verá en este apartado está lista no es tan simple como parece.

4.5.2.1 ListView con BaseAdapter

Al no tratarse de una lista estática, sino que en este caso es una lista dinámica en la que se podrán añadir y borrar nuevos campos, se debe implementar con un elemento de la clase BaseAdapter. En resumen los pasos a seguir son:

Lo primero que se debe hacer es preparar la matriz de datos. Estos datos vendrán guardados en la clase SharedPreferences de la siguiente manera:

Etiqueta	Significado
numitems	Número de conjuntos de variables guardados
itemelegido	Número del conjunto que se tiene seleccionado
numvar+i	Número de variables que tiene el conjunto i
vars+i	Nombres de variables que tiene el conjunto i

Tabla 4-1: SharedPreferences para los conjuntos de variables

Se crea una clase Var en la que cada elemento tiene una cadena para guardar el nombre de las variables, un entero para el número de ellas y un booleano para

indicar si es el elegido. Se creará una lista con tantos elementos Var como marque "numitems" y se guardará en ella toda la información.

El siguiente paso es definir en XML como será un elemento simple de la lista. El elemento definido tendrá el siguiente aspecto:

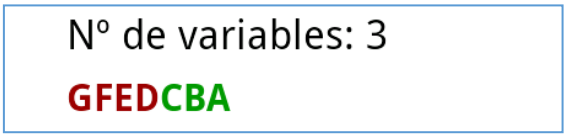

Figura 4-4: Elemento de la lista.

Como se puede apreciar en la foto, un elemento se compone de 3 campos de texto. El superior para indicar el número de variables, uno abajo a la derecha para las variables activadas, que se representarán en verde y otro a la izquierda para las variables desactivadas, en color rojo.

A continuación se creará un adaptador que mapee cada elemento del conjunto de datos con un elemento fila de la lista. Para hacer esto se debe crear una clase que extienda de BaseAdapter y posteriormente añadirla como adaptador al elemento ListView.

La clase que hereda de BaseAdapter debe guardar en una lista todos los elementos que muestra y sobre-escribir alguno de los métodos de esta. El método más importante que se re-implementó es el método getView que es el que llama el adaptador para representar todos los elementos que se puedan de la vista.

Aquí entra en juego, lo denominado en inglés como "layout inflation". Es un término que se utiliza para indicar cuando Android coge un recurso XML y lo convierte en una jerarquía de objetos vista en Java.

Volviendo atrás, el método getView tiene que devolver la vista del elemento i, por lo que "inflatea" un elemento simple de la lista y escribe dentro los datos del elemento i. Es decir, carga el número de variables, las variables activadas y desactivas en sus zonas correspondientes y en el caso de que el booleano de la clase Var sea "true" pinta el fondo de amarillo. Este método se llamará automáticamente y con él se rellenará todo el espacio visible. En la siguiente imagen se puede ver el resultado de la lista.

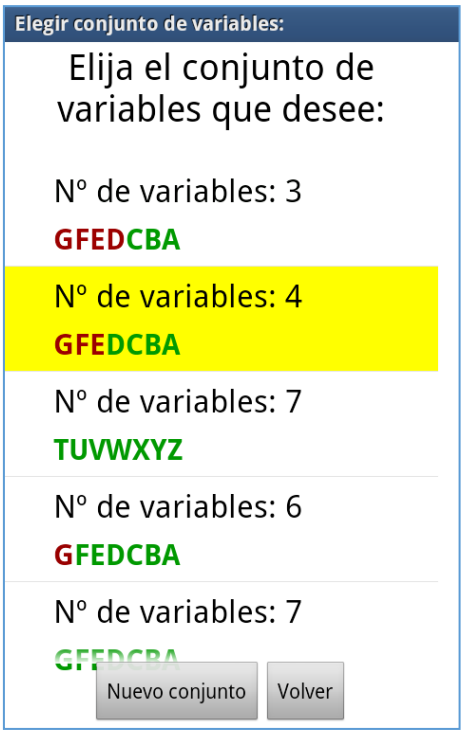

Figura 4-5: Lista de conjuntos de variables.

Por último sólo falta indicar que debe pasar cuando el usuario pulsa uno de los ítems. Se le añadirá al ListView un elemento onClickListener que cuando se pulse, llamará a la actividad de elegir letras pasándole el valor de la posición pulsada.

Con esto podría ser suficiente para que la lista dinámica funcione, pero como se verá en el siguiente apartado se optimizó aún más.

4.5.2.2 Optimización del ListView con BaseAdapter

Esta lista de conjuntos de variables guardados, normalmente no es demasiado larga. Pero si tiende a serlo, irá muy despacio y consumirá mucha memoria. Ya que cada vez que se llama al método getView se realiza una llamada al método inflate además de su inicialización y varias llamadas al método findViewById para coger las etiquetas en las que se debe cargar el elemento de la lista. Resulta que estos métodos son muy ineficientes, por lo que se tiene un problema:

- Cuando se navega por la lista, cada vez que aparecen nuevos elementos se consume más y más memoria porque más objetos nuevos están siendo creados.
- Cuando una de las vistas desaparece ya no se vuelve a usar y se acumula una colección de elementos residuales.
- Tanto la creación como la acumulación de elementos sin uso son muy perjudiciales para el uso de la CPU. Provocan un mayor uso de la batería y un funcionamiento cada vez más lento.

- Las aplicaciones tienen una cantidad de memoria limitada, por lo que si se sobrepasa ese límite, podría dejar de funcionar.

La solución que se va a implementar se basa en el concepto de reciclar vistas. Cuando una vista deja de ser usada, porque al deslizarse por la pantalla ha desaparecido, normalmente significa que otra vista aparece en el lado opuesto. Entonces en vez de eliminar dicho elemento, se desplazará a la parte de abajo y se le sobre-escribirán los datos de la nueva vista.

De esta manera cada vez que se recicla una vista se ahorran las citadas llamadas e inicializaciones y a los métodos inflate y findViewById, que eran los que más CPU consumían. Sin entrar más en los detalles de esta solución así es como se consigue resolver en gran parte todos los problemas explicados.

4.5.2.3 La actividad de elegir letras.

Esta actividad será llamada tanto cuando se seleccione un elemento de la lista de conjuntos, como cuando se elija la opción de crear un nuevo conjunto. Su mayor complejidad es el uso de un nuevo Widget, el Spinner, que tampoco es demasiado difícil. Su apariencia es la siguiente:

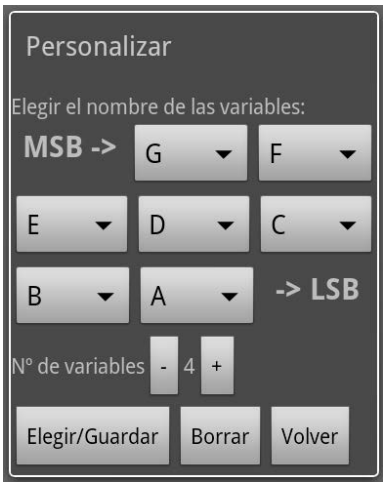

Figura 4-6: Actividad elegir letras.

Debe realizar comprobaciones para que en un conjunto de variables no se repitan letras y para que el número de variables siempre se encuentre entre 3 y 7. En los casos contrarios se comunicará con el usuario utilizando los mensajes Toast que ya se explicaron con anterioridad.

Una vez pulsado el botón de elegir/guardar o el botón de borrar se recoge la información de los Spinners y se actualizan las SharedPreferences para que el programa funcione correctamente. Además de introducir los cambios en las variables, cada vez que cambia o se borra el conjunto seleccionado se ponen a 0 todos los indicadores de estados guardados válidos de la aplicación. El motivo es

que no tiene sentido guardar los estados antiguos para unas variables nuevas. Pero todo esto ya se entenderá mejor más adelante.

4.5.2.4 Principales problemas encontrados

A la hora de reciclar una vista que desaparece de la lista es importante sobre-escribir de manera precisa todos los datos, ya que se tuvieron muchos problemas con este tema. Todos ellos fueron solucionados y se sometió la lista a multitud de pruebas para asegurarse de este correcto funcionamiento.

4.5.3 Herramienta Karn Auto

La herramienta Karn Auto es la siguiente actividad de la que se detallará su implementación. Esta herramienta se divide en 4 actividades.

4.5.3.1 Actividad para elegir variables en Karn Auto

La primera actividad que aparece cuando se elige en el menú principal la opción Karn Auto es una selección de número variables. Debido a que Karn Auto sólo está disponible para 3, 4 y 5 variables, dispone únicamente de esos tres modos de funcionamiento. Por lo que carga desde las SharedPreferences los nombres de variables del conjunto seleccionado actualmente y muestra por pantalla las 3, 4 y 5 variables menos significativas del mismo para dar a elegir al usuario.

Esté menú no tiene gran complejidad de implementación por lo que no se hablará más de él.

4.5.3.2 Actividad Karn Auto de 4 variables

Cada una de las otras 3 actividades de las que dispone Karn Auto corresponde a cada uno de los números de variables permitidos. Todos se implementaron de la manera parecida, así que con ver extendidamente el funcionamiento del de 4 variables, será suficiente para entender el de los demás.

Al igual que los otros dos, el Karn Auto de 4 variables consiste en un lienzo Canvas en el que se dibujará el mapa de Karnaugh con el que se interactuará con el usuario. Al ser la primera clase Canvas que se explica en esta memoria se analizará con mucho detalle.

4.5.3.2.1 Método onCreate

Lo primero que se debe implementar en una clase Canvas es el método onCreate en el que se calcularán todas las medidas a partir de las cuales se dibujará el mapa. Antes de nada y para adaptarse a cualquier dispositivo se ha definido como será el tamaño del mapa que aparecerá en pantalla, en función de una variable paso ("p" en el dibujo). Se puede observar en el siguiente dibujo dicha

apariencia. Lo representado en rojo son los márgenes, en azul la zona para representar el mapa y en la zona verde es donde irán los textos con la función simplificada:

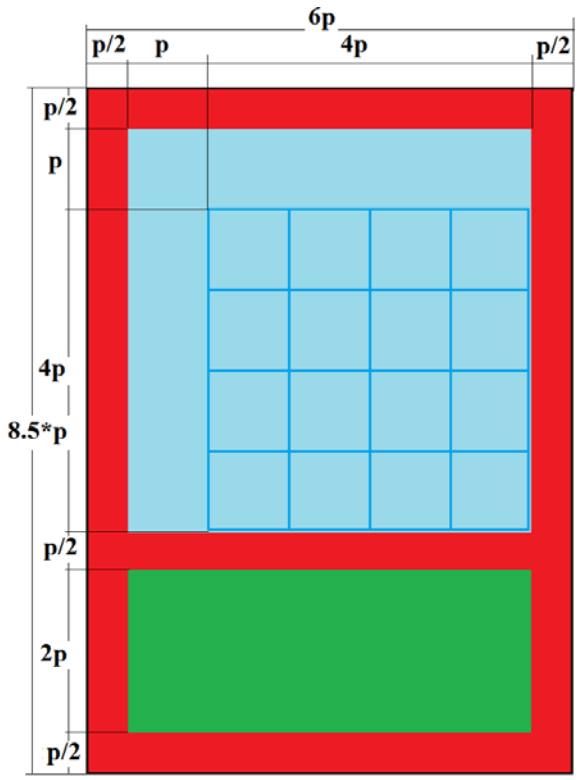

Figura 4-7: Apariencia en función del paso de Karn Auto 4 variables.

Como se puede apreciar en la imagen el ancho máximo será de 6 pasos y la altura máxima será de 8,5 pasos. Con estos datos se utiliza la clase DisplayMetrics para conocer las medidas del dispositivo en el que se está dibujando y con unas simples divisiones se sabrá en ese dispositivo concreto cual es la dimensión delimitadora. Así se consigue que Karn Auto funcione en cualquier dispositivo independientemente de su relación ancho-alto.

Una vez calculada la variable paso en el dispositivo, se procede a indicar cuáles serán las distintas coordenadas con las que se dibujará todo. Las coordenadas que se identificarán son:

- Para pintar la cuadrícula se declararán y calcularán dos matrices de enteros de 6 posiciones, una matriz para las coordenadas x y otra para las coordenadas y. Ambas se calculan muy fácilmente partiendo desde 0 y en función de cuantos pasos "p" se ha dado para llegar a esa coordenada.

- Para escribir en la cuadrícula también se declaran y calculan dos matrices de enteros de 5 posiciones para indicar las coordenadas donde se escribe el texto.

En la siguiente figura se pueden ver varios ejemplos de cómo están distribuidas estas coordenadas. Las indicadas con X e Y son las que sirven para pintar la cuadrícula y las que están indicadas con PT_X y PT_Y son para las posiciones de texto.

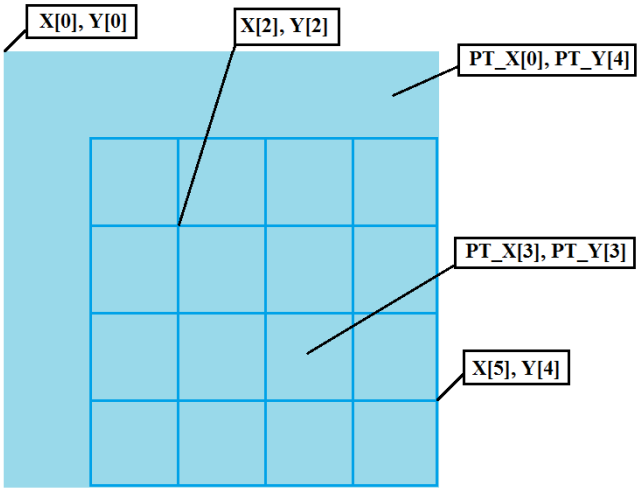

Figura 4-8: Ejemplos de coordenadas en Karn Auto 4 variables.

- Por último las coordenadas para escribir la simplificación consiste en dos matrices con 9 posiciones. La primera para escribir la F con sus variables y el signo igual, y las otras 8 para tantos términos como puedan aparecer en la simplificación. La representación de la función tendrá forma de una matriz de 3x3 en la que se irán añadiendo por orden los términos simplificados.

Otra de las funciones del onCreate es cargar las preferencias, para tener el conjunto de variables que se debe imprimir en el dibujo del mapa y el estado en que se encuentran las casillas del mapa.

Por último se crea una clase de tipo dibujo que será de tipo Canvas y se añade en el setContentView del onCreate asociando la actividad al lienzo Canvas en lugar de asociarlo a un layout.

4.5.3.2.2 Método onDraw

Una vez definida la clase dibujo se llama a su método onDraw que es el encargado de dibujar todo. A continuación y de forma ordenada se explicará todo lo que hace este método:

En primer lugar pinta toda la parte estática del mapa, es decir, la tabla y los textos de fuera del mapa. Para ello utiliza algunas de las coordenadas declaradas previamente.

Posteriormente pinta la parte dinámica, es decir, escribe los nombres de las variables en su sitio e imprime en cada casilla el estado actual correspondiente.

Una vez dibujado pasa a calcular la función simplificada del estado actual. Para ello, crea una lista de términos en la que introduce todos los términos que tengan en el mapa "1" o "X" y le aplica el algoritmo simplificador, llamado algoritmo Quine-McCluskey. Nos devolverá una lista con todos los términos presentes en la función simplificada. Ya se explicarán los detalles de este algoritmo más adelante.

Cuando se tiene la lista de términos, hay que representar la respuesta. Lo cual consta de dos partes para cada término, la visual en la tabla y la textual que se añadirá a la función simplificada final. A cada término se le va a asociar un color de manera que las dos partes equivalentes a cada término se harán del mismo color.

La parte visual de cada término consiste en rodear todas las casillas que componen cada agrupación. Además de un color a cada término se le ha asociado un índice de separación, que marcará el tamaño del rectángulo que rodeará la agrupación de ese término. Los índices de separación no están ordenados, para que círculos cercanos sean lo más diferentes posibles y se distingan mejor. También se alejarán los colores parecidos para que se visualice lo mejor posible.

El método que dibuja los rectángulos se llama drawCircle y recibe como parámetros: el lienzo Canvas, el término que debe dibujar, el pincel paint en el color que toque y el índice de separación. Es un método bastante complicado que básicamente analiza las filas y columnas que se tienen que rodear. También debe observar si la agrupación está dividida, es decir, si se unen columnas o filas opuestas del mapa y actuar en consecuencia. No se entrará más en detalle en lo que hace este método.

En resumen para cada término que haya en la expresión simplificada se llamará a la función drawCircle y, con el mismo color, se imprimirá su expresión en la siguiente posición libre de la función respuesta.

A continuación se pueden ver algunas capturas de la actividad Karn Auto de 4 variables funcionando:

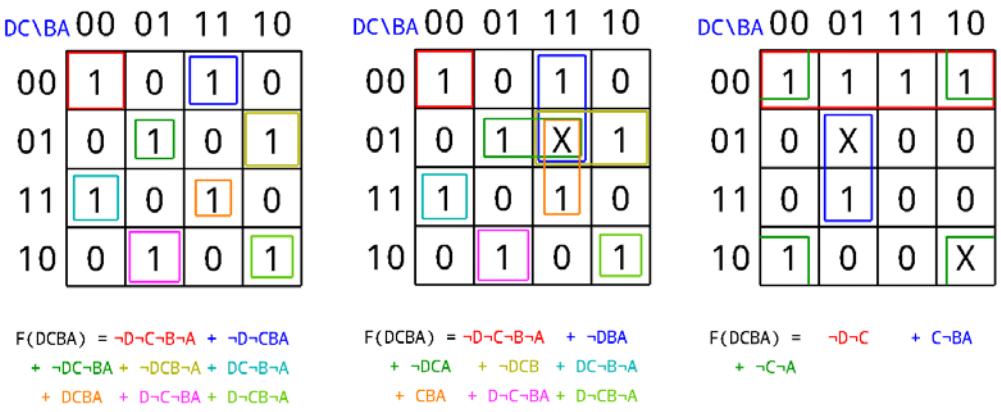

Figura 4-9: Ejemplos de Karn Auto 4 variables funcionando.

4.5.3.2.3 Método onTouchEvent

En la clase dibujo se debe sobre-escribir el método onTouchEvent para interactuar con el usuario. En este caso sólo se comprueba si se ha pulsado alguna de las casillas del mapa y se cambiará su estado. Una vez cambiado el estado se invalida la vista actual para que se tenga que re-pintar todo añadiendo el nuevo cambio.

4.5.3.3 SharedPreferences en Karn Auto

Al igual que sucede en el resto de la aplicación el mecanismo que se utiliza en el proyecto para almacenar datos es la clase SharedPreferences. En la siguiente tabla se pueden observar todas las nuevas etiquetas que se utilizan para la información relativa a las herramientas Karn Auto de 3, 4 y 5 variables:

Etiqueta	Significado
estado_ 3var+i	Estado del mapa Karn Auto de 3 variables (0, 1 o X) en la casilla i
estados_validos_3var	Flag que indica si los estados de 3 variables son válidos o no
estado_ 4var+i	Estado del mapa Karn Auto de 4 variables (0, 1 o X) en la casilla i
estados_validos_4var	Flag que indica si los estados de 4 variables son válidos o no
estado_ 5var+i	Estado del mapa Karn Auto de 5 variables (0, 1 o X) la casilla i
estados_validos_5var	Flag que indica si los estados de 5 variables son válidos o no

Tabla 4-2: SharedPreferences para la herramienta Karn Auto

4.5.4 Herramientas para introducir funciones en el Traductor

La herramienta Traductor está compuesta por varias actividades. En este apartado nos centraremos en las actividades de introducción de funciones lógicas, de las que se detallará su implementación en los siguientes apartados.

4.5.4.1 Actividad para elegir variables en Traductor

La primera actividad que aparece en esta herramienta es muy parecida a la que aparecía en Karn Auto. Sólo que ahora el número de variables está fijado y lo que te da a elegir es el modo de introducción que deseas para introducir la expresión lógica. Los distintos modos de introducción se pueden diferenciar en dos grupos. Introducción por tablas e introducción por expresión lógica.

4.5.4.2 Actividad de introducción por expresiones lógicas

Esta actividad es otro ejemplo de lienzo Canvas, como sucedía en el caso de las actividades Karn Auto. En la siguiente imagen se puede ver el esquema que seguirá esta actividad en función de la variable paso. Utilizando las medidas de ancho y alto del dispositivo y las mostradas en el esquema se puede calcular fácilmente el paso máximo con el cual entrará en ambas direcciones en la pantalla del dispositivo. En otras palabras, así se verá la dimensión que delimita, en esa dimensión se verá ocupada entera, mientras que en la otra sobrará un margen mayor.

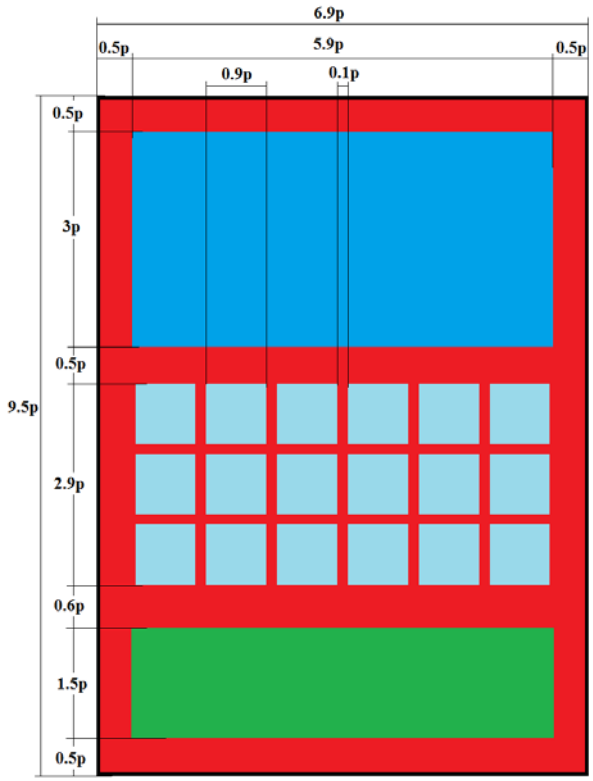

Figura 4-10: Apariencia en función del paso de introducir expresión lógica

En la figura, la parte azul oscura de la pantalla será donde se escriba la función a introducir, las teclas están representadas de azul claro y lo verde serán los botones de traducir, ayuda y borrar.

De manera similar a antes se inicializarán y calcularán coordenadas para posteriormente dibujar y escribir todos los elementos en el lienzo.

Prácticamente todo es igual que lo que se hacía para representar el lienzo en Karn Auto, por lo que no se entrará en más detalle.

4.5.4.2.1 Traducción.

Cuando se pulsa el botón traducir llega uno de los momentos de mayor dificultad de la aplicación. Se tiene una expresión introducida y hay que sacar la tabla de la verdad de dicha expresión con la que fácilmente se podrá traducir al resto de modos que ofrece la herramienta, es decir, su tabla de verdad, sumatorio de minitérminos, producto de maxitérminos, la función simplificada y esquemático.

Pero el paso desde una expresión introducida hasta una tabla de verdad no es nada sencillo. Para realizarlo hay que tener en cuenta los siguientes pasos:

- Primero se debe de averiguar si la función que se ha introducido es correcta y sigue las pautas requeridas para realizar la traducción.
- Se debe de averiguar algún mecanismo para evaluar una expresión.
- Con dicho mecanismo que se haya implementado hay que evaluar todos los valores de la tabla de la verdad para obtenerla.

Estos pasos expuestos para traducir la expresión lógica a una tabla de verdad han sido resumidos de manera muy simple. Pero si se desea profundizar aún más en ese tema se pueden ver los detalles de la implementación completa en el manual del programador, apéndice A.

Una vez se disponga de la tabla de la verdad, el paso a sumatorio de minitérminos es directo ya que es la misma tabla. Por otro lado la traducción a producto de maxitérminos es inversa, simplemente los maxitérminos son los valores de la tabla de verdad que tengan un cero.

Finalmente se guardará todo en las SharedPreferences para pasarlo a la siguiente actividad en la que se mostrarán los resultados. Pero esto ya se explicará en uno de los siguientes apartados.

Se puede ver la apariencia que tiene esta actividad antes de pasar a explicar la siguiente:

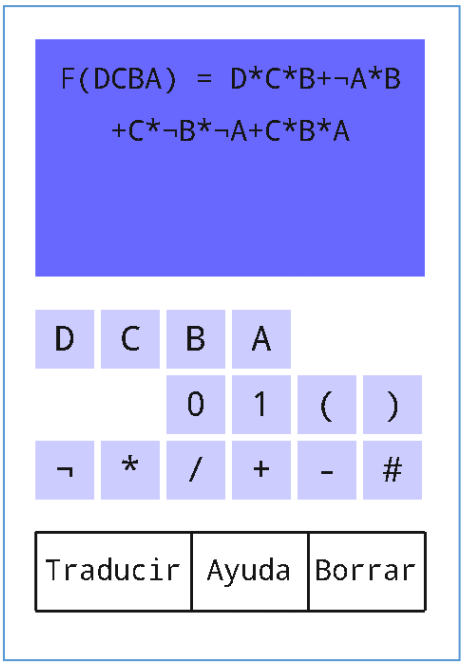

Figura 4-11: Ejemplo de introducción de funciones por expresión lógica

4.5.4.3 Actividades de introducción por tablas

La introducción por tablas está presente en los otros tres métodos de escribir la función de entrada, estos son: Tablas de verdad, sumatorio de minitérminos y producto de maxitérminos. Todos se basan en una clase Canvas como en algunas actividades anteriores tanto en Karn Auto, como en introducir funciones por expresión lógica.

En el método onCreate se realizan el mismo tipo de operaciones que en el de las dos actividades con Canvas anteriores. Solo que en este caso, como las listas suelen ser muy largas (dos elevado al número de variables), se considera que siempre el elemento delimitador será el ancho. Mientras que la altura no se considera. Así se consigue encuadrar esta vista en cualquier dispositivo para que se vea todo. El resto de acciones del método onCreate se harán igual: declarar las coordenadas, cargar las preferencias, etcétera.

El método onDraw es parecido también aunque las tablas se pintarán en vez de con líneas rectas negras, con cuadrados de colores para imitar la apariencia de las tablas de las anteriores aplicaciones del DSLab. Debajo de la tabla se dibujarán unos botones para llamar a las funciones de traducir y de reset.

En el método onTouchEvent se tendrá que analizar sólo la columna del estado de la tabla (segunda columna) y los botones de la parte inferior.

Al ser una actividad con funcionamiento tan parecido al de las anteriores, no merece la pena gastar tiempo con ella. En la siguiente imagen, se pueden ver algunas capturas de las 3 actividades que tienen introducción por tablas.

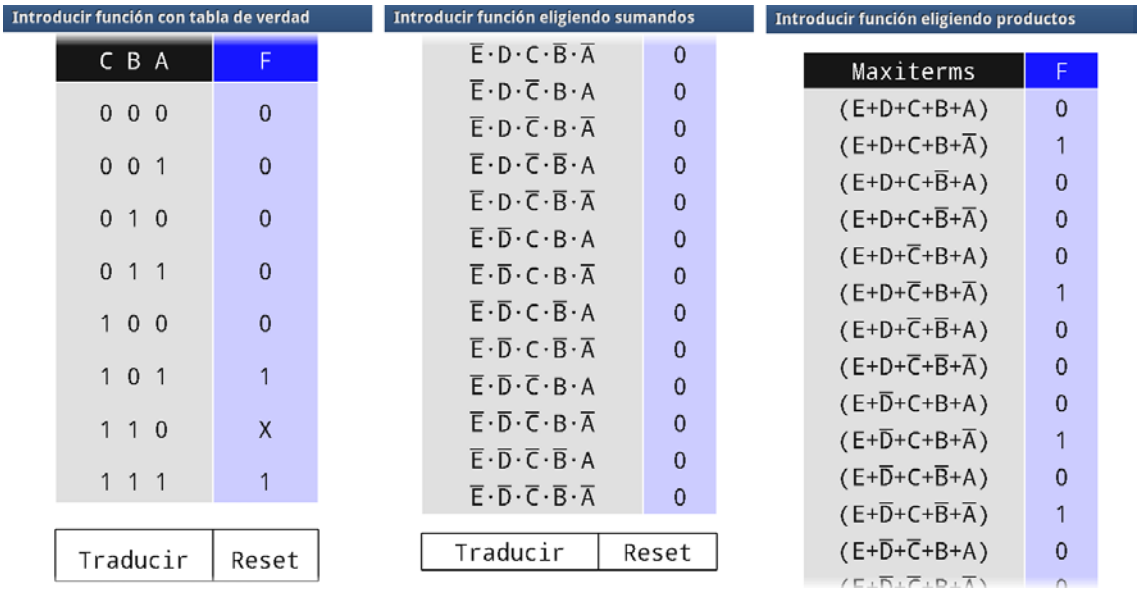

Figura 4-12: Ejemplos de introducción de funciones por tabla

Anteriormente, en la actividad de introducir funciones por expresión lógica, lo que se hacía era evaluar la expresión para crear una tabla de verdad y a partir de ahí traducir fácilmente a minitérminos y maxitérminos. En esta actividad se va a tener ya automáticamente una de las tres tablas, con lo que las otras dos se sacan de manera directa.

Igual que antes se guardará todo utilizando las SharedPreferences con la nomenclatura que se verá en el siguiente apartado.

4.5.4.4 SharedPreferences en Traductor

Como se ha hecho en varios de los apartados anteriores de la memoria, se van a explicar todas las etiquetas que se utilizan y se guardan en la clase SharedPreferences. En este caso se van a explicar las relativas a pasar la información desde los métodos de introducción de funciones hasta la actividad que representa los resultados de la traducción.

Etiqueta	Significado
estados_validos_ tabla_verdad	Flag que indica si los estados de tabla verdad son válidos
tabla_verdad+i	Estado i de tabla de verdad
estados_validos_ sumatorio_miniterminos	Flag que indica si los estados de sumatorio de minitérminos son válidos
sumatorio_miniterminos+i	Estado i de sumatorio de minitérminos
estados_validos_ producto_maxiterminos	Flag que indica si los estados de producto de maxitérminos son válidos
producto_maxiterminos+i	Estado i de producto de maxitérminos
estados_validos_ expresion_logica	Flag que indica si los estados de expresión lógica son válidos
Expresion_logica+i	Estado i de expresión lógica
traduccion_actual	Flag que indica el método de entrada de la última traducción realizada

Tabla 4-3: SharedPreferences para los conjuntos de variables

Para entender mejor el funcionamiento, los flags de estados validos sirven para que se guarde el estado de la tabla o expresión lógica hasta nuevo aviso. Los estados solo se borrarán en los siguientes casos:

- Si se cambia de conjunto de variables elegido.
- Si se borra el actual conjunto de variables al que pertenecen dichos estados.
- Si se realiza una traducción con uno de los otros métodos de entrada, ya que este estado se sobre-escribirá con el valor de la traducción de dicha función.

Si se realiza una traducción sobre un método de entrada, si que se conserva dicho estado_valido para que se puedan realizar cambios sobre él, si es pertinente. Pero sólo se conservarán los de este método, perdiéndose los del resto.

Dependiendo del método de entrada con el que se pulsó el botón de traducir, el flag traduccion_actual cogerá un valor distinto. Así la actividad que muestra los resultados conocerá el método utilizado anteriormente.

4.5.5 Mostrando resultados en la herramienta Traductor

Para mostrar el resultado de las traducciones se barajaron muchas opciones distintas. Al final se pensó que la mejor idea era crear de alguna manera una actividad en la que se representarán todas de forma ordenada, en vez de crear un menú con botones para acceder a las distintas traducciones.

Las opciones finales que se consideraron para realizarlo fueron dos:

- La vista TabHost: que es la que se utiliza en las agendas de la mayoría de dispositivos Android. TabHost ofrece varias pestañas en la parte superior que al pulsarlas te cambian a otra vista distinta.
- El ViewGroup ViewPager: Ya explicado anteriormente, ofrece varias vistas que van cambiando al deslizar el dedo por pantalla. Proporciona una interfaz visual muy moderna y cómoda para representar varias vistas. Es la elegida para esta actividad.

4.5.5.1 Implementación de un ViewPager

ViewPager es una clase que ha sido incluida en una versión reciente de Android y que ha conseguido mucha popularidad. Para poder añadirla al proyecto hay que agregar un paquete de compatibilidad ya que nuestro proyecto está disponible para dispositivos anteriores a la versión en la que aparece dicho elemento.

Una vez importado el paquete de compatibilidad hay que añadir un elemento ViewPager en la vista XML que se desee. La clase ViewPager necesita un adaptador que debe heredar de la clase ViewPagerAdapter. Ya se ha hablado anteriormente de los adaptadores, que sirven para manejar las vistas y la interacción que realice el usuario con cada una de ellas.

Como en usos anteriores de adaptadores hay que sobre-escribir algunos de sus métodos. En este caso hay que hacerlo principalmente con el método: Fragment getItem(int i). Que es el que devuelve el fragmento que se mostrará en pantalla. También se debe sobre-escribir otro método para mostrar los títulos, pero en ese no se entrará con tanto detalle.

Para cada página que se quiera que tenga nuestro ViewPager se debe definir un XML y una clase que herede de Fragment que lo contenga. Esta clase tendrá un método constructor que será al que llame el método getItem del adaptador cuando tenga que devolver el objeto de la clase Fragment en dicha posición.

Por último la actividad principal debe de heredar de FragmentActivity en lugar de la clase Activity de la que heredan casi todas las actividades de la aplicación. En el método onCreate se debe de crear un adaptador de los que se han citado anteriormente y añadirlo al elemento ViewPager para que funcione correctamente.

Por último mencionar que dentro de las clases que heredan de Fragment creadas para cada una de las páginas del ViewPager no se pueden usar las SharedPreferences porque la clase no desciende de Activity. Esto supuso un problema grande porque no se podían sacar las traducciones que se tenían que mostrar. Por ello para que funcionaran correctamente las SharedPreferences se deben de cargar en la actividad principal, que hereda de FragmentActivity, y pasar las cadenas que se deben representar en cada Fragment por los constructores a la hora de crearlos.

4.5.5.2 Realización de las traducciones.

Como ya se explicó en las actividades de introducción de funciones lógicas, la actividad que muestra las traducciones tiene ya calculada un montón de información en las SharedPreferences. Independientemente del modo en el que se introdujese, llegados a este punto se tendrá:

- Una bandera con un indicador del modo de función introducida.
- La tabla de verdad equivalente.
- La tabla del sumatorio de minitérminos.
- La tabla del producto de maxitérminos.
- Y en el caso de que la introducción fuese por expresión lógica, la expresión lógica introducida.

Con todos estos datos ya se tiene información suficiente para representar las cuatro primeras páginas del ViewPager, que son: función de entrada, tabla de verdad, minitérminos y maxitérminos.

Sólo falta la última, la función simplificada. Para simplificar la función únicamente hay que utilizar la tabla de verdad para crear una lista de términos con todos los elementos de la tabla de verdad que tengan "1" o "X". A esta lista de términos se le aplicará el algoritmo Quine-McCluskey para que devuelva la lista de términos de la función simplificada. Dicha lista de términos será lo que hay que imprimir en la página restante.

En la siguiente imagen se muestran capturas de pantallas de las 5 páginas de esta actividad:

Figura 4-13: Ejemplos de las páginas que muestran las 5 traducciones

4.5.6 Mostrar el esquemático

En la actividad que muestra las traducciones, como se puede ver en la captura anterior, hay un botón que conduce a la actividad que muestra el esquemático de la función simplificada. Esta funcionalidad fue la que conllevo uno de los mayores retos de desarrollo de la aplicación. Ya que debe dibujar el esquemático de cualquiera de las funciones simplificadas que pueda devolver el algoritmo.

Otra vez la opción elegida para la resolución del problema va a ser la clase Canvas. En este caso, puede haber circuitos de todos los tamaños, por lo que no se puede hacer que se muestren en un único pantallazo de cualquier dispositivo. Por ello se ha elegido el paso de una manera distinta: Para cualquier dispositivo de entrada, el paso será un onceavo del ancho de pantalla y a partir de este paso calculado se representará todo lo demás.

4.5.6.1 Parámetros de entrada.

Como se ha ido realizando a lo largo de toda la aplicación, en el método onCreate se llamará a la función que carga las preferencias para coger el set de variables actual.

En este caso en concreto se recoge la cadena de caracteres de la función simplificada a dibujar de las SharedPreferences, que fue pasada en un campo llamado "expresion_logica_reducida".

Con esos dos valores ya se tiene toda la información necesaria para realizar el esquemático.

4.5.6.2 Desarrollo.

Las consideraciones para que se pinte un circuito cualquiera tienen que adaptarse a cualquier tipo de función, desde funciones de una sola letra hasta otras con 20 puertas AND. Por ello esta herramienta tuvo un desarrollo muy cuidadoso para el que incluso se tuvieron que definir varias clases para instanciar las puertas.

Al necesitar una explicación bastante extensa se ha decidido sacarlo en un apéndice que se puede ver al final de la memoria y sólo en el caso de que interese ver la implementación de esta actividad. Esta explicación se encuentra en el Manual del programador, apéndice B

4.5.6.3 Ejemplo de correcto funcionamiento.

Para poder hacerse una idea de cómo funciona esta herramienta gráficamente, se ha introducido una función cuya expresión simplificada es bastante larga. Así se puede ver el correcto funcionamiento de dicha herramienta trabajando sobre un circuito grande.

En la imagen se puede ver en la parte izquierda como se representan las variables y sus negadas. Después se ve que hay una etapa de puertas AND en la que sus patas se mapean con las variables y por último una puerta OR que reúne todas las salidas de las puertas AND.

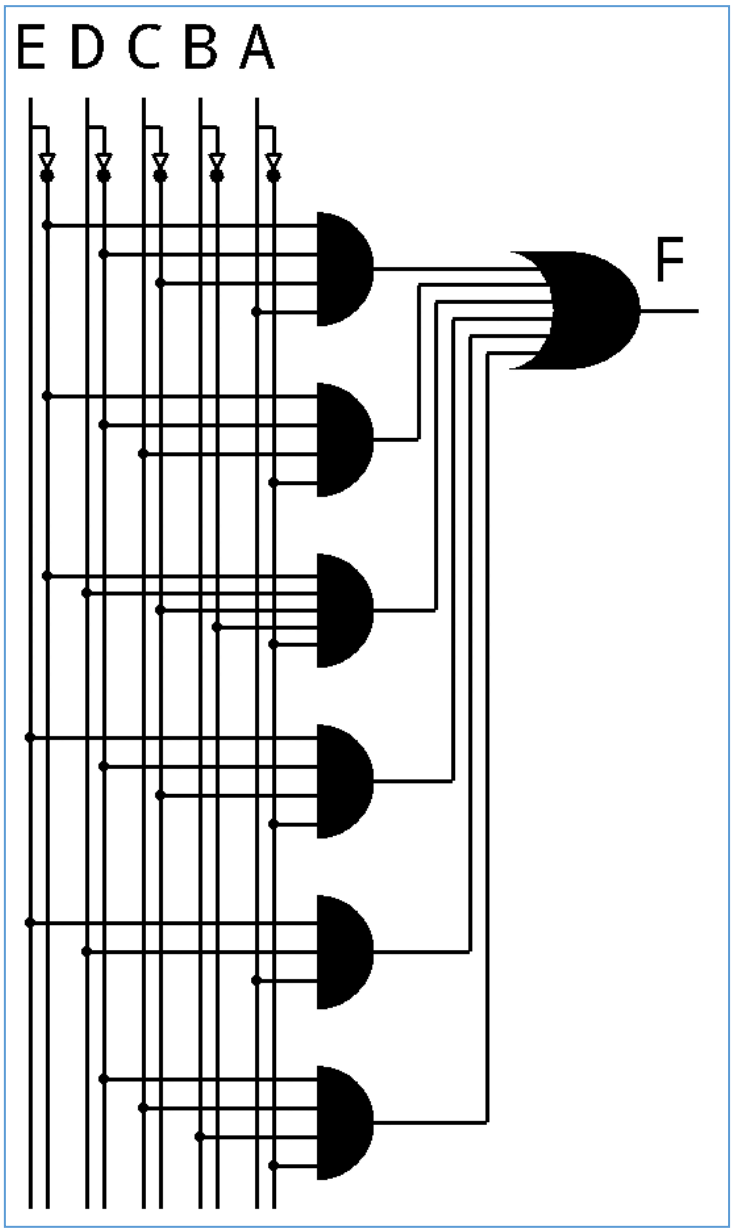

Figura 4-14: Ejemplo de esquemático real

Se recuerda que para entender la implementación completa de esta actividad se puede acceder al apéndice B en el que se indica con todo detalle cómo se ha realizado el esquemático.

4.5.7 Herramienta Conversor

La herramienta Conversor es bastante más sencilla y se basa en un layout con los botones y cuatro campos de texto editables. La única complejidad que conlleva es evitar los errores que puedan surgir de un posible "overflow" de alguna de las variables o por una introducción incorrecta de números ajenos a algún sistema.

Para evitar la introducción de caracteres incorrectos se ha deshabilitado el teclado Software que muestra Android por defecto cada vez que se pulsa dentro de un

campo de texto. Con el teclado desactivado el usuario sólo podría introducir los caracteres de las teclas ofrecidas por el Activity. Pero aun así, esto no soluciona del todo el problema porque aún quedan dos posibles errores:

- Si el dispositivo del usuario tiene un teclado Hardware hay que evitar que las entradas introducidas hagan bloquear el dispositivo al realizar operaciones incorrectas.
- Hay que resolver también los citados problemas de "overflow" relativos al tamaño de las variables.

Para resolver estos problemas se ha implementado el código relativo a la conversión dentro del manejador de excepciones que propone Java. Este manejador consiste en la estructura try-catch. Todo el código introducido dentro del bloque try se intentará ejecutar y si da error, se saltará al bloque Catch que maneja dicho error. En caso de ausencia de error se ejecutará todo el bloque try y nunca se entrará dentro del catch.

Este mecanismo es muy habitual en las aplicaciones Java y en consecuencia de Android. En este caso particular se ha realizado de la siguiente manera:

- En el bloque try lo primero que se hace son las conversiones sin actualizar ninguno de los textos editables.
- Si hay algún error en las conversiones se captura con el método catch y se muestra por pantalla sin realizar ningún cambio en los campos de texto.
- Si no hay error, se actualizan los 4 campos y no se realiza nada más.

Este mecanismo soluciona los dos errores que se tenían y hace que la herramienta Conversor sea muy robusta.

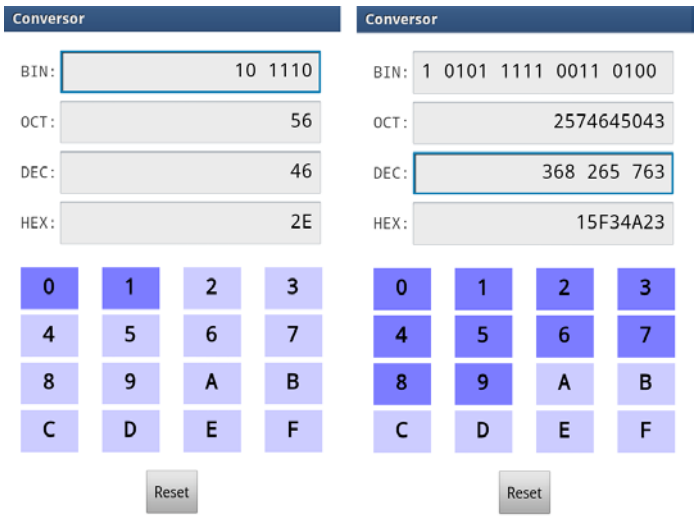

Figura 4-15: Ejemplos del Conversor

4.5.8 Manuales de ayuda

La actividad que incluye los manuales de ayuda fue bastante sencilla de desarrollar, ya que sólo implica una lista estática de varios elementos. En la que al pulsar cada elemento se accede a otra actividad estática consistente en textos e imágenes que siempre tienen la misma apariencia. Al no implicar ninguna dificultad de desarrollo no se expondrán más detalles que estos.

Se pueden ver en la siguiente imagen varias capturas de dicho manual de ayuda.

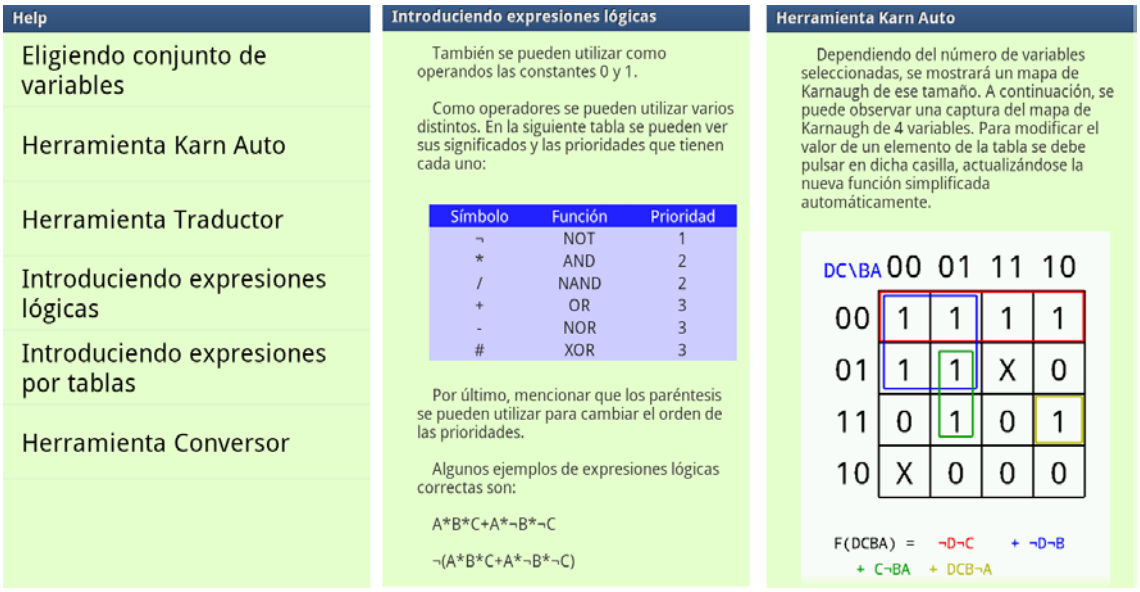

Figura 4-16: Ejemplos del manual de ayuda.

4.5.9 Página de acerca de

Para finalizar este apartado, la última actividad que se ha desarrollado en esta aplicación es la actividad de "acerca de" en la que se incluye alguna de la información básica del proyecto. Esta actividad se ha desarrollado siguiendo la estética de las actividades de "acerca de" de las anteriores aplicaciones del DSLab.

Figura 4-17: Ejemplos del manual de ayuda.

4.6 Algoritmo Quine-McCluskey

4.6.1 Introducción

El algoritmo de Quine-McCluskey es un algoritmo alternativo a los mapas de Veitch-Karnaugh para simplificar funciones lógicas.

Por un lado los mapas de Karnaugh son un método gráfico para simplificar las funciones de manera bastante efectiva. Pero presenta algunos problemas importantes:

- A partir de las 4 o 5 variables empieza a ser bastante difícil de aplicar.
- Depende mucho de la habilidad del usuario para encontrar la simplificación mínima.
- Es difícil de implementar computacionalmente.

Por otro lado, el algoritmo Quine-McCluskey presenta las siguientes ventajas:

- Por su modo de aplicación facilita la simplificación para funciones de más de 4 variables.
- Como es un método sistemático, no depende para nada de la habilidad del usuario.
- Su forma tabular lo hace más eficiente a la hora de implementarlo en lenguajes computacionales

Por esto, aunque el algoritmo por mapas de Veitch-Karnaugh es el que da nombre a gran parte de la aplicación, es el algoritmo de Quine-McCluskey el que funciona por dentro para realizar las simplificaciones. Aún así, como Karnaugh es el nombre más comercial y popular en el mundo de la electrónica, se ha

elegido para el nombre a la aplicación. Además la parte gráfica se hace a modo de mapa de Karnaugh, que es lo que al final el usuario observa.

4.6.2 Funcionamiento

El funcionamiento general del algoritmo Quine-McCluskey se puede resumir en dos pasos:

- Primero se deben encontrar todos los implicantes primos de la función a simplificar.
- Segundo, utilizando todos los implicantes primos de la tabla anterior, se deben de buscar los implicantes primos esenciales que sean los mínimos necesarios para representar dicha función.

Este funcionamiento se ha descrito de la manera más reducida posible, en los siguientes apartados se detallarán estos dos pasos para entender el algoritmo completo. Además se ha añadido un ejemplo para entender mejor el funcionamiento de este algoritmo.

4.6.2.1 Paso 1: Encontrar implicantes primos.

En primer lugar se añaden en una tabla todos los minitérminos, es decir, todos los elementos con unos y equis de la tabla de verdad. Ya que lo que se busca es la optimización, las redundancias pueden combinarse también con los minitérminos para conseguir la función más simplificada. Los minitérminos de la tabla deben de estar agrupados según el número de unos que tenga cada uno.

Una vez que se tiene la tabla con los elementos agrupados, el siguiente paso consiste en juntar los términos de grupos contiguos. Se juntarán todos aquellos términos que difieran en sólo un bit dando como resultado un término con un guión (-) en ese bit y el resto de bits igual. Se deben de agrupar todas las combinaciones posibles de términos que difieran un bit. Es importante indicar que minitérminos han dado como resultado cada agrupación.

Se repetirá este paso con los elementos obtenidos hasta que no se puedan obtener elementos nuevos de cada grupo contiguo. Cada elemento final, es decir, que no se haya podido combinar con ninguno debe de ser marcado.

4.6.2.1.1 Ejemplo del paso 1:

La tabla de verdad de la función que se quiere simplificar es la siguiente:

D C B A	F	D C B A	F
0 0 0 0	**0**	1 0 0 0	**0**
0 0 0 1	**1**	1 0 0 1	**0**
0 0 1 0	**0**	1 0 1 0	**1**
0 0 1 1	**1**	1 0 1 1	**1**
0 1 0 0	**0**	1 1 0 0	**0**
0 1 0 1	**0**	1 1 0 1	**1**
0 1 1 0	**X**	1 1 1 0	**1**
0 1 1 1	**1**	1 1 1 1	**X**

Tabla 4-4: Tabla de verdad del ejemplo.

Se agrupan los minitérminos con 1 y X en una tabla según el número de unos que tenga cada término:

Número de unos	minitérmino	D C B A
1	1	0 0 0 1
2	3	0 0 1 1
	6	0 1 1 0
	10	1 0 1 0
3	7	0 1 1 1
	11	1 0 1 1
	13	1 1 0 1
	14	1 1 1 0
4	15	1 1 1 1

Tabla 4-5: Primera tabla de primos implicantes.

En el siguiente paso se agrupan todos los términos que sólo difieren en un bit, entre grupos contiguos.

Minitérminos	D C B A
1, 3 (Elemento 1)	0 0 - 1
3, 7	0 - 1 1
3, 11	- 0 1 1
6, 7	0 1 1 -
6, 14	- 1 1 0
10, 11	1 0 1 -
10, 14	1 - 1 0
7, 15	- 1 1 1
11, 15	1 - 1 1
14, 15	1 1 1 -
13,15 (Elemento 2)	1 1 - 1

Tabla 4-6: Segunda tabla de primos implicantes.

Se vuelve a realizar el mismo paso y los elementos que no se utilizan en ninguna de las agrupaciones se marcan en la tabla superior, en este caso han sido dos elementos.

Minitérminos	D C B A
3, 7, 11, 15 (Elemento 3)	- - 1 1
6, 7, 14, 15 (Elemento 4)	- 1 1 -
10, 11, 14, 15 (Elemento 5)	1 - 1 -

Tabla 4-7: Tercera tabla de primos implicantes.

Ya no se pueden agrupar más elementos por lo que se marcan los 3 de la anterior tabla como elementos que no se pueden agrupar más. La salida de este paso serán esos 5 elementos marcados en las tablas. Se puede ver como sigue en el apartado correspondiente del paso 2.

4.6.2.2 Paso 2: Encontrar implicantes primos esenciales.

Con todos los grupos marcados en el paso anterior, los cuales son todos grupos de manera que no hay uno mayor que los contenga, se debe iniciar este paso. Para alcanzar la simplificación mínima para la función, sólo es necesario seleccionar algunos de esos conjuntos para cubrir todos los minitérminos en los que la salida de la función era 1.

Con este cometido se generará una tabla de conjuntos en la que se marcarán todos los minitérminos que cubre cada conjunto. Para comenzar la selección, primero se elegirán todos aquellos elementos esenciales que son los únicos que

incluyen alguno de los minitérminos. Es obvio que al haber términos que solo están incluidos por un conjunto, ese conjunto tiene que estar siempre en la simplificación.

Por último se eligen el resto de conjuntos mínimos para añadir los términos que queden si no se han añadido todos, al añadir los esenciales. En algunos casos pueden quedar varias simplificaciones mínimas distintas.

4.6.2.2.1 Ejemplo del paso 2:

Ahora hay que crear una tabla en la que se deben incluir todos los elementos de salida del apartado anterior e indicar que minitérminos abarcan cada uno de esos elementos. En este caso no hace falta tener en cuenta los términos que tuviesen redundancias ya que esos sólo se seleccionarán si conviene. La tabla es la siguiente:

Elemento	1	3	7	10	11	13	14
1	X	X					
2						X	
3		X	X		X		
4			X				X
5				X	X		X

Tabla 4-8: Tabla para buscar los primos implicantes esenciales.

En la tabla se han marcado aquellos minitérminos que solo están incluidos por uno de los elementos de la tabla. En este caso son los minitérminos 1, 10 y 13. Por ello los elementos 1, 5 y 2 deberán estar sin ninguna duda en la función simplificada.

Como se puede observar también en la tabla, al añadir a la función simplificada los elementos 1, 2 y 5 se añaden otros elementos además de por los que se escogieron. En este caso todos los minitérminos que ya se tendrán en la función son: 1, 3, 13, 10, 11 y 14.

Ya sólo queda el minitérmino 7 por escoger, para ello hay que coger el elemento 3 o el 4, y como ambos son del mismo tamaño podría valer cualquiera de los dos para obtener la función simplificada mínima. Las dos opciones de conjuntos de elementos que se tienen son:

- Elementos 1, 2, 5 y 3, que equivale a ¬D*¬C*A + D*C*A + D*B + B*A
- Elementos 1, 2, 5 y 4, que equivale a ¬D*¬C*A + D*C*A + D*B + C*B

Para apreciar como se ha realizado una de estas dos simplificaciones en la herramienta Karn Auto de 4 variables, se añade la siguiente captura de pantalla:

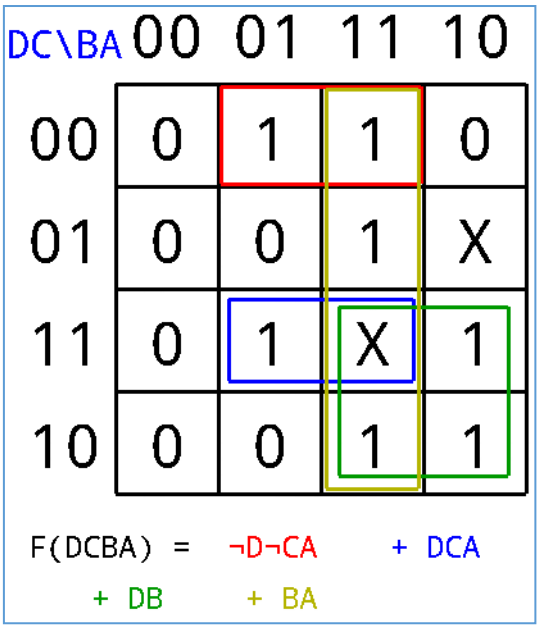

Figura 4-18: Correcto funcionamiento del algoritmo Quine-McCluskey.

4.6.3 Desarrollo

Gracias a la popularidad y la gran utilidad del algoritmo Quine-McCluskey, existen multitud de algoritmos Quine-McCluskey disponibles en la red para descargar y utilizar.

La velocidad a la que se ejecute este algoritmo será de fundamental importancia para que el usuario goce de una buena experiencia al utilizar la aplicación. Por este motivo se ha decidido utilizar una función ya implementada y optimizada de las que existen en Internet. Se podría haber implementado una función que realizará las simplificaciones sin demasiada dificultad pero no se habrían conseguido unos resultados tan buenos en tiempo a la hora de realizar las simplificaciones.

De todas las implementaciones del algoritmo Quine-McCluskey que se encontraron, al final se decidió utilizar la de la página literateprograms.org ya que funcionaba de manera muy eficiente. Aunque esta implementación tenía un problema, no aceptaba la introducción de equis a la hora de crear términos a reducir. Esto era muy importante, porque se quería que la aplicación admitiese redundancias. Así que, utilizando la base del algoritmo original, se realizaron cambios para conseguir que funcionase también con dichas redundancias.

Finalmente se consiguió mantener la rapidez del algoritmo original y se consiguió añadir la posibilidad de poner equis en los términos a la hora de simplificar funciones lógicas, obteniendo así un algoritmo muy completo. Más adelante, en el apartado de pruebas, se hablará de los tiempos de ejecución del algoritmo.

4.7 Diagrama de clases

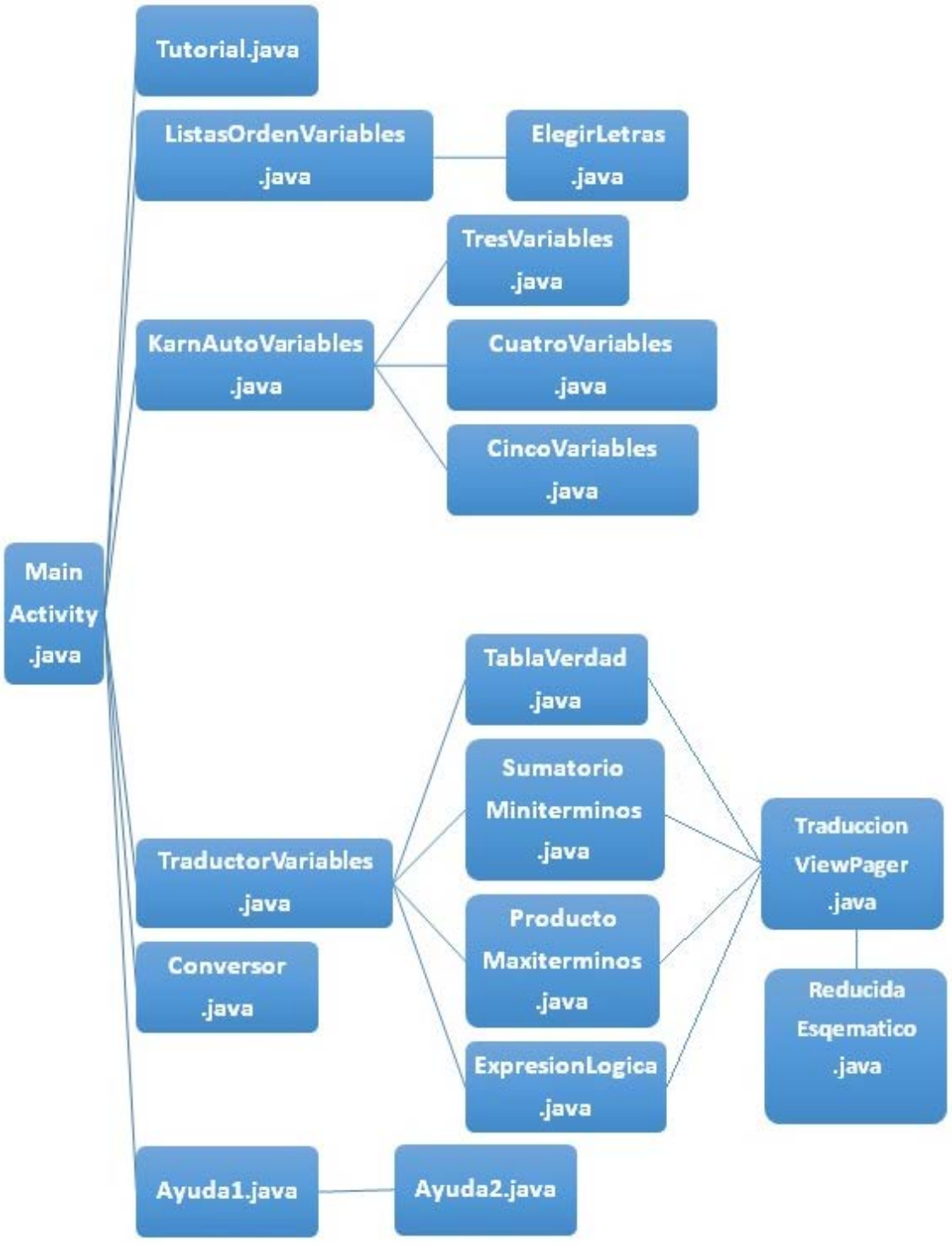

Figura 4-19: Diagrama de clases.

4.7.1 Descripción de las clases principales.

- MainActivity.java: Ofrece el menú principal y desde ahí se comprueba si es la primera vez que se ejecuta la aplicación para crear la lista de SharedPreferences por defecto.

- Tutorial.java: Ofrece unas imágenes a modo de tutorial para orientar al usuario. Gestiona el movimiento entre páginas.

- ListaOrdenVariables.java: Se encarga de mostrar la lista de conjuntos de variables disponibles en la aplicación. Añade un VarCustomAdapter a la lista para que tenga el funcionamiento dinámico que se desea.

- ElegirLetras.java: Esta clase sirve para editar o borrar un set de variables.

- KarnAutoVariables.java: Menú de la herramienta Karn Auto para elegir el número de variables entre 3 y 5. También ofrece la posibilidad de ir a ListaOrdenVariables.

- TresVariables.java: Simplificador de funciones lógicas por mapa de Karnaugh de 3 variables.

- CuatroVariables.java: Simplificador de funciones lógicas por mapa de Karnaugh de 4 variables.

- CincoVariables.java: Simplificador de funciones lógicas por mapa de Karnaugh de 5 variables.

- TraductorVariables.java: Menú de la herramienta Traductor para elegir el método de introducción deseado. También ofrece la posibilidad de ir a ListaOrdenVariables.

- TablaVerdad.java: Ofrece el método de introducción de funciones eligiendo ceros, unos y equis en una tabla de verdad.

- SumatorioMiniterminos.java: Ofrece el método de introducción de funciones eligiendo ceros, unos y equis en una tabla con todos los minitérminos posibles para las variables elegidas.

- ProductoMaxiterminos.java: Ofrece el método de introducción de funciones eligiendo ceros, unos y equis en una tabla con todos los maxitérminos posibles para las variables elegidas.

- ExpesionLogica.java: Ofrece el método de introducción de funciones escribiendo una expresión lógica.

- TraduccionViewPager.java: Es la clase en la que se ofrecen todas las traducciones posibles de la función introducida

- ReducidaEsquematico.java: Ofrece la representación esquemática de la función simplificada.

- Conversor.java: Es la clase que contiene la herramienta Conversor, para realizar conversiones entre los sistemas numéricos: binario, octal, decimal y hexadecimal.

- Ayuda1.java: Menú con una lista de todas las ayudas disponibles.

- Ayuda2.java: Representa la ayuda elegida en el menú ayuda1.

- DialogoTextoCheck.java: Ventana desplegable que será llamada al inicio de varias herramientas para explicar brevemente su funcionamiento. Incluye una casilla para que esta ventana no vuelva a mostrarse.

4.7.2 Descripción de los métodos principales.

En este apartado se definirán algunos de los métodos más utilizados:

- onCreate: Es el método invocado cuando se carga una actividad y debe de definir todo lo necesario para su correcto funcionamiento.

- onCreateOptionMenu: Para crear el menú que se desplegará al pulsar el botón de menú Hardware que ofrecen los dispositivos Android.

- onOptionItemSelected: Para definir qué se debe hacer cada vez que se pulsa una tecla del menú anterior.

- CargarPreferencias: Función que se encarga de cargar las preferencias necesarias para la actividad en la que se encuentra.

- GuardarPreferencias: Función para guardar las preferencias utilizadas o editadas en esa actividad.

- onActivityResult: Método encargado de manejar el resultado de retorno de las actividades llamadas desde la actual.

- onDraw: Método para diseñar la vista en un elemento de la clase Canvas.

- onMeasure: Método para definir las dimensiones del lienzo Canvas.

- onTouchEvent: Método para definir la acción a realizar si la pantalla del dispositivo es pulsada.

5 Integración, pruebas y resultados

5.1 Pruebas

En este apartado se explicarán algunas de las muchas pruebas que se han realizado para comprobar que la aplicación funciona correctamente. Muchas de estas pruebas se han ido realizando a lo largo de la etapa de desarrollo para ir comprobando el funcionamiento de cada actividad que se iba implementando.

Cabe destacar que la prueba definitiva para comprobar el correcto funcionamiento de la aplicación se realizará cuando comience la asignatura para la que se ha desarrollado. Ésta dará comienzo en enero de 2015 ya que pertenece al segundo cuatrimestre. Serán los propios estudiantes de la escuela que cuando utilicen la aplicación para resolver sus problemas hagan la prueba final de la aplicación.

Aun así, la aplicación ya ha sido sometida a múltiples test y se ha puesto a prueba en muchos casos extremos para comprobar su buen funcionamiento.

5.1.1 Pruebas en herramienta Karn Auto.

Para comprobar el correcto funcionamiento de la herramienta Karn Auto se han debido poner a prueba sus dos funcionalidades principales.

La primera de ellas es la parte encargada de la simplificación. Para testear dicha funcionalidad se han cogido libros que explican los mapas de Karnaugh y se han probado sus ejemplos en la aplicación para ver si simplificaban igual. Además de eso también se han comprobado los resultados manualmente. Obteniendo resultados correctos en los Karn Auto de los tres tamaños distintos disponibles.

La otra funcionalidad que se ha sometido a múltiples test es la que pinta los rectángulos indicando las agrupaciones que fueron elegidas. En parte fue comprobada a la vez que la anterior funcionalidad, pero además se provocaron casos extremos para ponerla a prueba.

En las siguientes imágenes se muestran algunos de esos casos extremos, en ellas se puede comprobar el funcionamiento de la simplificación y del dibujo. En la primera, se prueba el funcionamiento si están seleccionadas las 16 casillas y a la vez se testea el uso de las equis. En la segunda se prueba si dibuja bien agrupaciones que se encuentran en lados opuestos horizontales. En la tercera comprueba los verticales y en la cuarta la combinación de ambos.

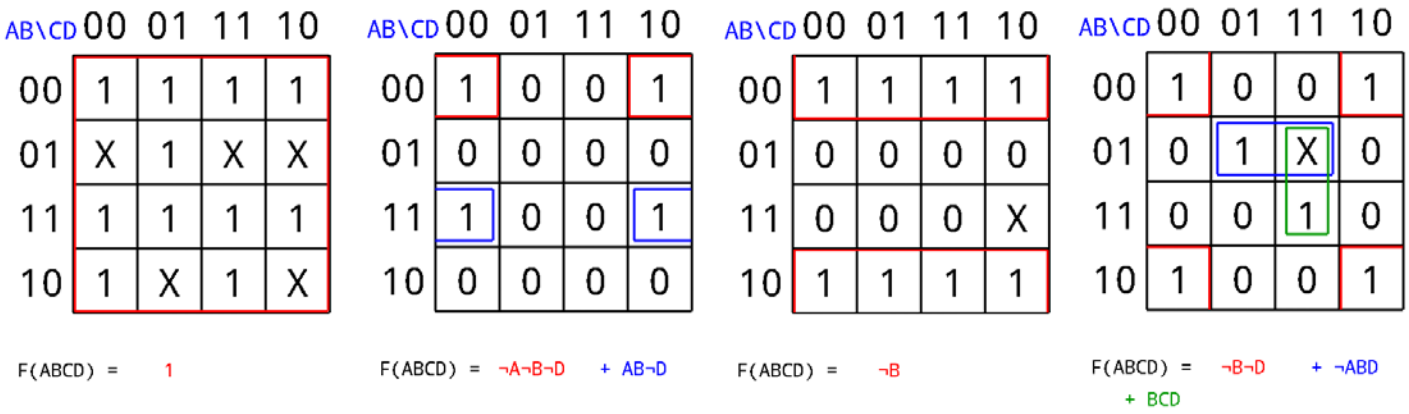

Figura 5-5-1: Casos extremos de Karn Auto.

5.1.2 Pruebas en herramienta Traductor.

La herramienta Traductor tuvo un testeo similar al de la herramienta Karn Auto basándose en ejemplos de libros y en algunos de los ejercicios de las guías docentes de la asignatura de Circuitos Digitales de la EPS. No hay ninguna captura reseñable de dichas pruebas, simplemente, destacar que no se detectó ningún error.

5.1.3 Pruebas en representación de esquemático.

Las pruebas sobre la función que representa el circuito esquemático se basaron principalmente en poner al circuito en los casos más extremos que se ocurrieron. Todos ellos se adjuntarán con sus capturas correspondientes. Algunos de estos casos son:

- Puerta OR pequeña (con 2 entradas):

El primer caso que se mostrará es un circuito cualquiera con una puerta OR pequeña, en este caso en concreto será del valor mínimo que puede tomar, dos entradas.

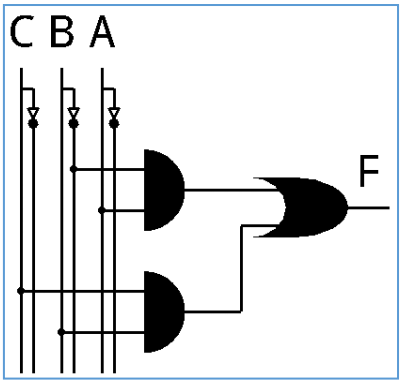

Figura 5-2: Casos extremos de esquemático: Puerta OR pequeña.

- Puerta OR grande (con por ejemplo 24 entradas):

Ahora se pondrá a prueba el extremo contrario, una puerta OR muy grande:

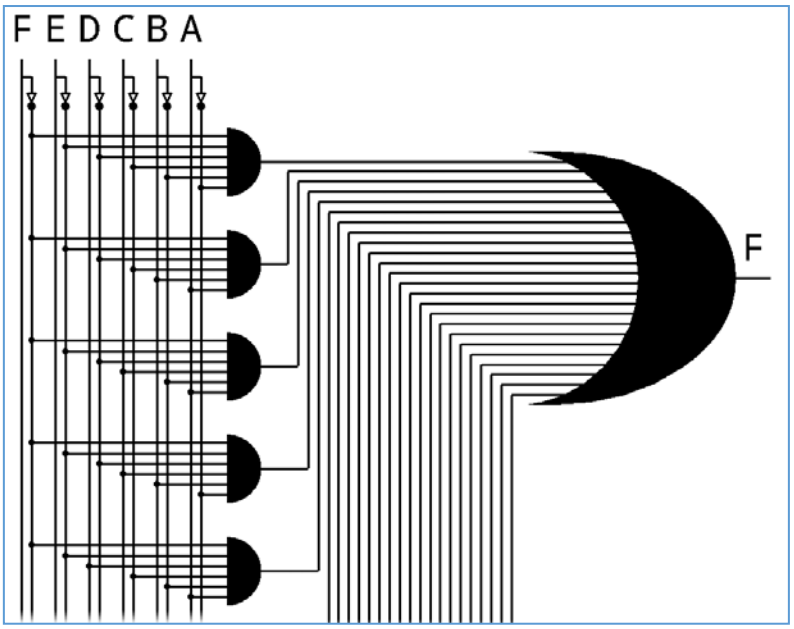

Figura 5-3: Casos extremos de esquemático: Puerta OR grande.

- Sin puerta OR:

Si solo hay una puerta AND en el circuito, entonces la puerta OR solo tendría una pata. Por ello, no se debería pintar la puerta OR y dibujar la salida directamente en la única puerta AND. En la siguiente imagen se puede ver un ejemplo del correcto funcionamiento de este suceso.

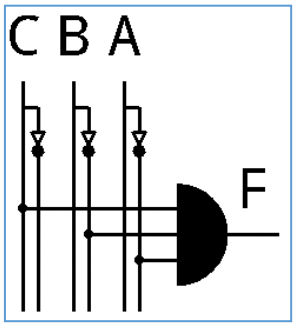

Figura 5-4: Casos extremos de esquemático: Sin puerta OR.

- Sin puerta AND:

El siguiente caso especial es el mismo que antes, pero relacionado con las puertas AND. En vez de dibujarse la puerta, simplemente se debe de dibujar una recta que atraviesa el espacio que ocuparía.

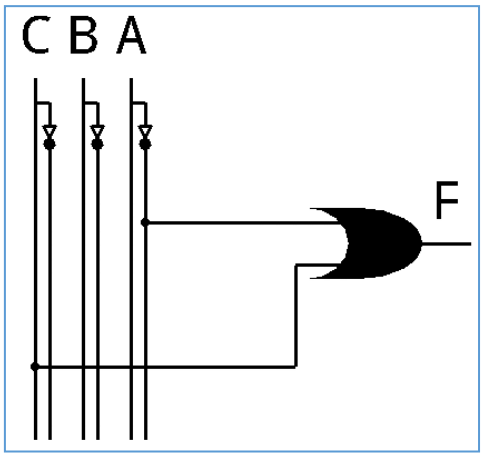

Figura 5-5: Casos extremos de esquemático: Sin puerta AND.

- Sin puertas AND y OR:

Otro caso de funcionamiento extremo que se debe de probar es si funciona correctamente con una función simplificada que resulte ser una sola variable. En este caso no habrá ni puertas AND ni OR. Por ejemplo, la imagen muestra la función F=C:

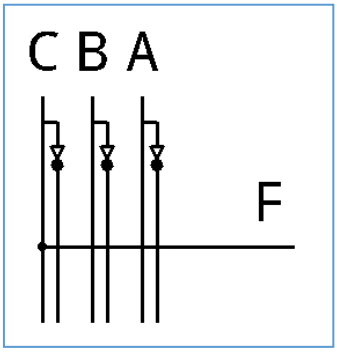

Figura 5-6: Casos extremos de esquemático: Sin puertas AND y OR.

- F=0 y F=1:

Los últimos casos especiales que se tratarán son los casos F=0 y F=1, en los que no haría falta dibujar ni siquiera ninguna variable. En la siguiente captura se puede ver la decisión que se tomó en ese caso:

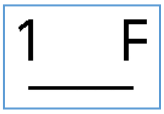

Figura 5-7: Casos extremos de esquemático: F=0 y F=1.

5.1.4 Pruebas en herramienta Conversor.

Para comprobar el funcionamiento del Conversor, se han chequeado varias conversiones realizadas por Karn Map con conversiones realizadas en otras calculadoras. También se ha probado a introducir teclas inválidas en todos los sistemas para ver si son rechazadas. En todos los casos ha funcionado sin problema.

También para intentar provocar errores, se ha probado a instalar la aplicación en un dispositivo con teclado Hardware, para introducir caracteres no válidos. El Conversor no se ha visto afectado por estos caracteres y ha seguido funcionando sin fallos.

5.1.5 Pruebas en otros dispositivos.

Se han realizado las pruebas anteriormente descritas en otros tamaños de dispositivos Android, como son varias tabletas de 7 y 10 pulgadas y dispositivos con pantallas más anchas que altas. En todos los casos probados la aplicación se adaptaba sin problema.

5.2 Resultados

En este apartado se han realizado algunos cálculos del tiempo computacional de ejecución de algunas funcionalidades de la aplicación. Para ello se han introducido varias líneas en el código que se encargan de ver el tiempo que tarda en realizarse.

En la mayoría de los casos, el tiempo de ejecución variaba un poco entre distintas ejecuciones por lo que se realizó la media aritmética entre varios intentos.

Descripción de tarea	Tiempo de ejecución (en milisegundos)
Simplificación y dibujo en Karn Auto 3 variables	12
Simplificación y dibujo en Karn Auto 4 variables	20
Simplificación en Karn Auto 5 variables	25
Traducción de expresión lógica a tabla de verdad de 3 variables	19
Traducción de expresión lógica a tabla de verdad de 4 variables	30
Traducción de expresión lógica a tabla de verdad de 5 variables	36
Traducción de expresión lógica a tabla de verdad de 6 variables	42
Traducción de expresión lógica a tabla de verdad de 7 variables	50
Simplificación en Traductor con 3 variables	2
Simplificación en Traductor con 4 variables	8
Simplificación en Traductor con 5 variables	14
Simplificación en Traductor con 6 variables	45
Simplificación en Traductor con 7 variables	85

Tabla 5-1: Tabla de tiempos de ejecución.

Cabe destacar que las medidas de los tiempos de ejecución fueron tomadas en un dispositivo de la marca Samsung y modelo Galaxy Ace 2, que funciona con la versión 2.3.6 de Android.

Los resultados obtenidos son muy buenos, ya que ninguna de las acciones cronometradas tomó más de 100 milisegundos de media en ejecutarse. Por lo tanto se han conseguido algoritmos muy eficientes que ofrecerán una experiencia muy dinámica al usuario.

5.3 Primera publicación

La publicación de la primera versión de la aplicación se produjo el día 22 de Octubre de 2014. Esta publicación se produjo en la cuenta de desarrollador del laboratorio, llamada DSLab y se llamó Karn Map a la aplicación.

El 29 de Octubre, una semana después, se subió la segunda versión. Esta versión añadía el tutorial que no estaba incluido en la primera y también solucionaba varios problemas que se encontraron en la anterior versión. En el futuro, posteriormente a la entrega de la memoria y de la lectura del PFC se seguirán

añadiendo versiones, si fuese conveniente, para solucionar cualquier problema o añadir alguna funcionalidad nueva.

En los primeros días sólo se difundió la aplicación entre compañeros de clase y del laboratorio, consiguiéndose las primeras descargas. Pero el día 27 de Octubre se intentó dar un gran impulso a la aplicación de varias maneras. Este impulso se podrá apreciar en las gráficas del apartado de estadísticas.

El tutor del proyecto envió la aplicación por e-mail a profesores de asignaturas de Circuitos Electrónicos Digitales o similares de otras universidades. La idea es que esos profesores puedan hablar de la aplicación a sus alumnos aumentando las descargas.

También se intentó promocionar la aplicación por la red social Twitter, pidiendo difusión a distintas cuentas populares entre ingenieros. Este tipo de cuentas eran de todo tipo, desde cuentas con chistes de ingenieros a cuentas que avisan de trabajos disponibles en el mercado. Lo único que se buscaba eran cuentas con gran número de lectores, los cuales fueran en su mayoría gente con posible interés en la aplicación.

Al igual que sucedió con las otras aplicaciones del DSLab se espera un incremento grande de las descargas en la primera semana del segundo cuatrimestre. Ya que es cuando empieza la asignatura principal para la que está diseñada y se avisará a todos los estudiantes para que se la puedan descargar.

5.4 Estadísticas

En este apartado se van a mostrar algunas de las estadísticas que ofrece la consola del desarrollador de Play Store. Estas estadísticas fueron recogidas el día 2 de Noviembre, 11 días después de la primera publicación de la aplicación.

5.4.1 Instalaciones totales de la aplicación.

Para comenzar se mostrará un gráfico de las descargas totales que ha tenido la aplicación en función del tiempo:

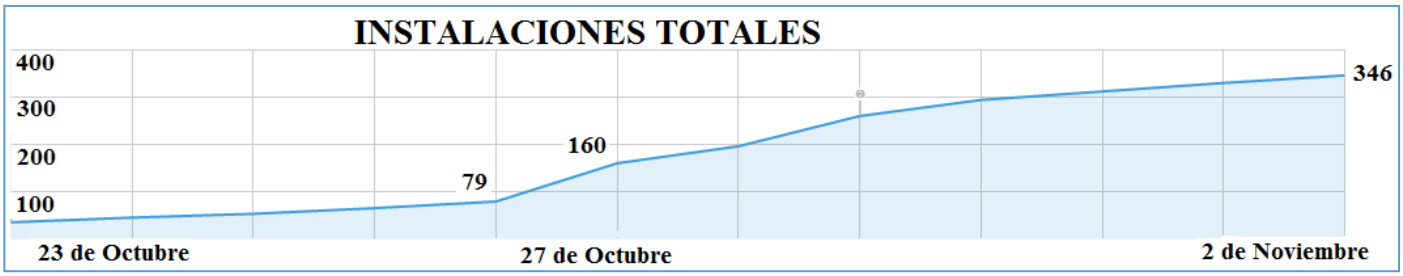

Figura 5-8: Instalaciones totales de la aplicación

En la gráfica se puede observar como el número de instalaciones de la aplicación iba creciendo con una pendiente constante. Pero el día 27 de Octubre y los posteriores esa pendiente creció considerablemente. El motivo de esta gran subida es la propaganda que se hizo de la aplicación durante esa semana de la que se habló en el apartado de la primera publicación. Se publicó el enlace por Twitter y se mandó por correo a profesores y distintos estudiantes a los que podría interesar la aplicación.

5.4.2 Instalaciones actuales de la aplicación.

En este caso ya no se habla de las descargas totales, sino de los usuarios que se descargaron la aplicación y que aún la conservan en sus teléfonos.

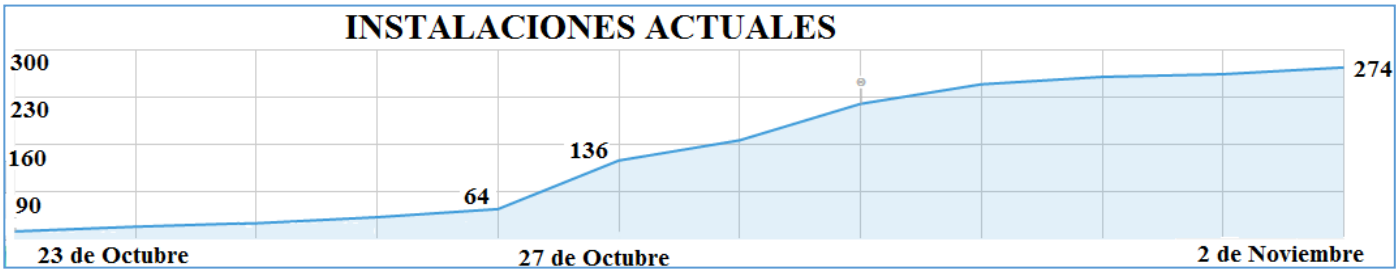

Figura 5-9: Instalaciones actuales de la aplicación

La gráfica presenta aproximadamente la misma pendiente que la anterior, por los mismos motivos, ya que están estrechamente relacionadas. Cabe destacar, que de las 346 personas que han descargado la aplicación, 274 la han conservado en su dispositivo, lo cual supone un 79,2%. Con este dato se puede entender que la aplicación está gustando y resultando útil a los usuarios.

5.4.3 Instalaciones totales por país.

La siguiente estadística que se mostrará es el número de descargas totales por país. Sólo se han incluido los 10 países con más descargas, ya que la lista sería muy larga si se muestran todos los países con al menos una descarga.

La gran mayoría de ellas se concentran en España, esto es debido a que es el país en el que se ha desarrollado y donde se ha publicitado principalmente la aplicación. Por ello sorprende ver el gran número de descargas que se han producido en países como Brasil y México en los que no se le ha dado ninguna publicidad especial a la aplicación.

Aún así, este tema depende mucho del factor suerte. Mientras se publicitaba por la red se llegaron a encontrar entradas en foros en los que la gente hablaba de la aplicación. Gente ajena al DSLab, que se encontraba el enlace, les parecía interesante la aplicación y decidían publicarla en sus blogs personales. Al final, por mucho que intentes publicitar tu aplicación, son los usuarios, si la aplicación es buena, los que mejor la pueden dar a conocer a más público.

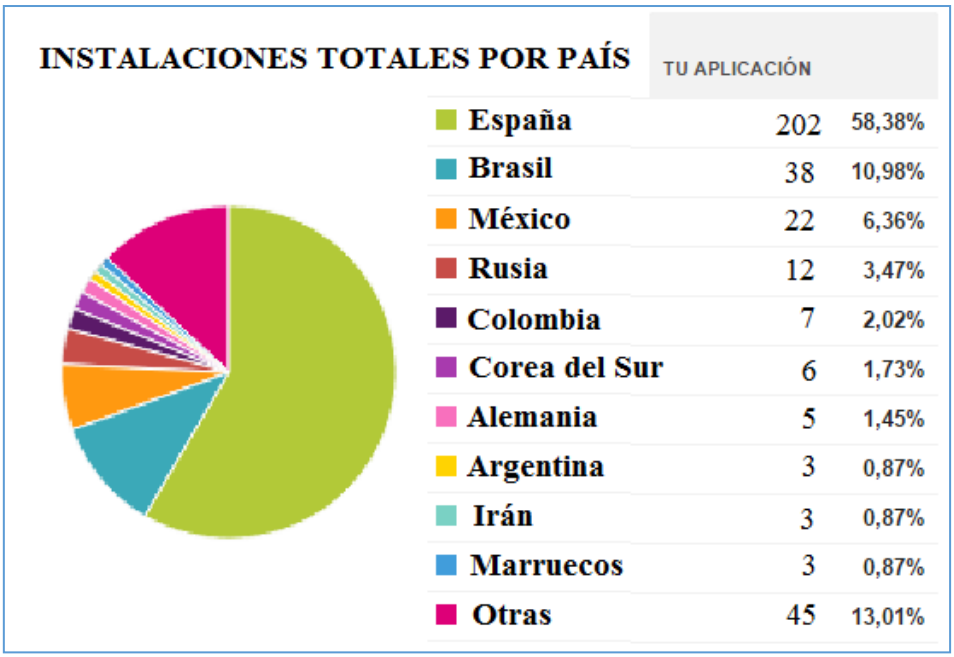

Figura 5-10: Instalaciones de la aplicación por países.

5.4.4 Instalaciones totales de la aplicación por versión.

Por último se muestra el siguiente gráfico en el que se puede apreciar lo bien que funciona Play Store para actualizar la aplicación en todos los dispositivos. En sólo 5 días desde el lanzamiento de la segunda versión ya está instalada en casi el 90% de los dispositivos. Mientras que la primera versión está al borde de desaparecer.

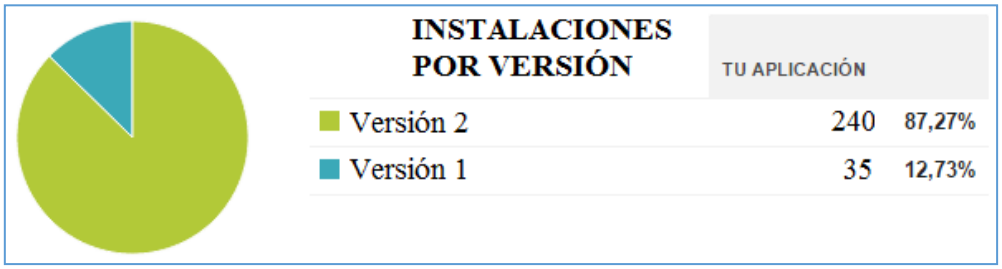

Figura 5-11: Instalaciones de la aplicación por versión

5.4.5 Conclusiones.

Para concluir este apartado de estadísticas, cabe destacar que la aplicación está teniendo una muy buna acogida por los usuarios. En el Play Store tiene una puntuación de 4,89 estrellas sobre 5 con un total de 19 puntuaciones. Además se han recibido bastantes comentarios positivos de los usuarios en general, la mayoría por Twitter y por lo que se ha leído en foros y los blogs anteriormente citados. Por último, alcanzar las 274 instalaciones en tan sólo 11 días se considera una muy buena cifra para el DSLab, además en el futuro se esperan muchas más.

5.5 Cuestionario de la aplicación

	Muy malo (1)	Malo (2)	Normal (3)	Bueno (4)	Muy bueno (5)
Organización general					
Aspecto de los gráficos					
Facilidad de uso					
Fiabilidad y estabilidad					
Consumo de batería					
Velocidad					
Tutorial y manuales de ayudas					
Utilidad del tutorial					
Calidad del tutorial					
Utilidad de los manuales					
Selector de variables					
Utilidad					
Velocidad					
Funcionamiento correcto					
Facilidad de uso					
Herramienta Karn Auto					
Utilidad					
Velocidad					
Funcionamiento correcto					
Facilidad de uso					
Herramienta Traductor					
Utilidad					
Velocidad					
Funcionamiento correcto					
Facilidad de uso					
Herramienta Conversor					
Utilidad					
Velocidad					
Funcionamiento correcto					
Facilidad de uso					
¿La recomendarías?: Comentarios y sugerencias:					

Las estadísticas tomadas a partir de este cuestionario se realizarán durante el segundo cuatrimestre del curso 2014-2015, periodo en el que se imparte la asignatura Circuitos Electrónicos Digitales para la cual está diseñada esta aplicación. Los datos recogidos serán estudiados para buscar posibles modificaciones de cara a una nueva versión que mejore a la que esté disponible en ese momento.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En resumen, los principales requisitos del proyecto se han cumplido:

- Insertar un tutorial teórico sobre la simplificación gráfica de funciones lógicas.

- Proporcionar una herramienta que simplifique gráficamente funciones lógicas por mapa de Veitch-Karnaugh de hasta 5 variables.

- Proporcionar una herramienta alternativa que simplifique funciones lógicas mayores, de hasta 7 variables. Que permite introducir funciones de manera algebraica, minitérminos y maxitérminos.

- Representación esquemática de las funciones simplificadas.

- Conversor de sistemas numéricos: binario, octal, decimal y hexadecimal.

- Funcionamiento correcto en tabletas y dispositivos móviles de cualquier versión de Android.

- Mantener el estilo de las 2 aplicaciones anteriores del DSLab. Incluyendo aspectos como: el tamaño de letra, colores, formato de menú inicial, icono, resolución y relación de aspecto de gráficos, formato de botones, menú de ayuda, página de "acerca de" y funcionamiento del tutorial.

- Estudio del estado del arte y de aplicaciones similares ya en el mercado.

- Desarrollar una aplicación práctica, fiable y rápida en los cálculos.

- Publicación en Play Store y revisión de errores.

A lo largo de las fases del proyecto se han aprendido diferentes herramientas y conceptos de interés:

- Programación en Java y su filosofía.
- Entorno de desarrollo Eclipse.
- Programación en Android y su arquitectura.
- Programación en XML.
- Revisión profunda de la asignatura de Circuitos Electrónicos Digitales.
- Publicar aplicaciones en el Play Store de Google.
- Organización y planificación de un proyecto.

6.2 Trabajo futuro

Esta aplicación ha sido desarrollada desde cero, y la idea principal es que se continúe mejorando y manteniendo en el futuro. Algunas de las mejoras que se podrían hacer a lo que ya hace esta aplicación son:

- Añadir soporte para idiomas de los países en los que la aplicación sea más popular.

- Añadir la posibilidad de funcionamiento con más de 7 variables.

- Habilitar el uso de todas las pantallas de la aplicación tanto en portrait (vertical) como en landscape (horizontal).

- Ofrecer la posibilidad de dar nombres a las variables que sean palabras en vez de letras.

- Admitir la introducción de expresiones lógicas de aún más caracteres. También se podría añadir la posibilidad de pinchar en medio de la cadena e intercalar caracteres entre los ya existentes.

- Añadir información adicional sobre los Circuitos Integrados adecuados más óptimos para implementar el esquemático representado en la realidad.

También se pueden añadir multitud de funcionalidades nuevas a la aplicación en cuestión, como:

- La posibilidad de añadir algún ejercicio para comprobar si el usuario ha entendido los métodos de simplificación de funciones lógicas.

- Añadir al punto anterior, también un modo examen con cronómetro y envío de ejercicios para corregir y puntuar los ejercicios.

- Añadir más métodos de introducción de funciones. Como por ejemplo podría ser dibujando un esquemático, con botones para añadir puertas lógicas y rectas para unir las patas.

- Crear una versión gratuita con publicidad y otra de pago que no tenga publicidad y ofrezca más funcionalidades.

- Adaptar a otros sistemas operativos como Apple y Windows Phone.

MANUAL DEL PROGRAMADOR

Este manual se reservará para explicar los detalles más complicados, relativos a la programación y al desarrollo de la aplicación.

A - Apéndice A

En este apéndice se van a explicar los detalles de implementación de la actividad para introducir funciones lógicas utilizando su expresión.

La expresión será introducida por el usuario utilizando un teclado dibujado en pantalla. Por lo que la entrada que se tendrá es una expresión lógica que puede ser correcta o no, por lo que el primer paso que hay que realizar es una comprobación de dicha expresión.

A.1 Comprobación de la expresión.

Antes de pasar a evaluar la expresión, se debe comprobar que es correcta, porque de no ser así las siguientes funciones podrían dar errores. Esta función es de vital importancia ya que de ella va a depender que la aplicación pueda bloquearse en caso de expresiones incorrectas.

Además es importante avisar al usuario sobre el error que contiene su expresión para que pueda aprender de él y así entender mejor el método de introducción desarrollado. En la ventana de ayuda aparece mucha información e incluso varios ejemplos de lo que hay que hacer y de lo que no hay que hacer, pero aún así, siempre viene bien orientar para que los usuarios disfruten de una buena experiencia al utilizar la aplicación.

En esta función se comprueban todo tipo de errores, en general se comprueba si la expresión está vacía, si el balanceo de paréntesis es incorrecto y si hay dos caracteres consecutivos incorrectos. En la siguiente tabla se pueden ver todos los errores que pueden aparecer al ejecutar esta función. Para mostrarlos por pantalla se utilizarán mensajes de la clase Toast.

Expresión incorrecta por:
Balanceo de paréntesis incorrecto
Hay dos negadores consecutivos
Hay dos variables sin operador entre ellas
Hay dos operadores seguidos
Hay una variable seguida de un elemento incorrecto
Hay un paréntesis de cierre seguido de un elemento incorrecto
El primer caracter es incorrecto
El ultimo caracter es incorrecto
Hay un negador seguido de un elemento incorrecto
Hay un operador seguido de un elemento incorrecto
Hay dos paréntesis sin nada dentro
Hay un paréntesis de apertura seguido de un elemento incorrecto
No se puede hacer la expresión más grande

Tabla 0-1: Posibles errores en comprobar expresión lógica.

Si la expresión ha conseguido pasar está función test con éxito se continua con el siguiente paso.

A.2 Necesidad de conversión infijo-postfijo.

Las expresiones que estamos acostumbrados a ver normalmente son expresiones infijas, con los operadores entre los operandos correspondientes. Pero este tipo de expresiones implican una dificultad muy grande a la hora de evaluarlas, debido a que no se encuentran ordenadas y dependen de las prioridades de los operadores y de los paréntesis.

Para facilitar esta tarea existe la notación postfijo, también llamada notación polaca en honor al matemático que la inventó. La idea es que los operadores se escriban detrás de los operandos y no en medio de las expresiones como se hace en la notación usual. Varios ejemplos de expresiones equivalentes en ambos sistemas se pueden ver en la siguiente tabla.

Notación infija	Notación postfija
¬A	A¬
A*B	AB*
A*¬B	AB¬*
¬A*B	A¬B*
¬(A*B)	AB*¬
A*B+C	AB*C+
A*(B+C)	ABC+*
A*B*C+A*¬B*¬C	ABC**AB¬C¬**+

Tabla 0-2: Ejemplos de expresiones en notaciones infija y postfija.

A.3 Algoritmo de conversión infijo-postfijo.

Una vez explicado en qué consiste la notación postfijo, se va a detallar cómo se realiza esta conversión. Para desarrollar este algoritmo se ha recurrido a las diapositivas de la asignatura de Estructura de Datos y Algoritmos que se impartía en la Escuela Politécnica Superior en segundo curso del plan antiguo.

Hay que tener en cuenta que cada operador tiene una prioridad determinada, se pueden ver estas prioridades en la siguiente tabla:

Función	Prioridad
( )	0
NOT	1
AND	2
NAND	2
OR	3
NOR	3
XOR	3

Tabla 0-3: Prioridades en conversión infijo-postfijo.

En general, la idea con la que se ha implementado el algoritmo es:

- Para empezar, se tiene que declarar una pila para ir metiendo los operadores.
- Se irán leyendo valores de la cadena infija y se analizarán de uno en uno.
- Si el valor leído es un operando, lo imprimirá directamente en la traducción parcial postfija.

- Si se trata de un operador se irán sacando todos los operadores de mayor o igual preferencia que haya en la pila y los concatenará a la expresión traducida. En cuanto se encuentre uno de menor preferencia se para y se introduce en la pila el nuevo operador.
- En caso de que se encuentre un paréntesis de apertura se introduce en la pila y se sigue con el algoritmo. Cuando se encuentre el de cierre se sacan todos los valores hasta el de apertura y se introducen en la expresión de salida.
- Cuando se acabe de leer la expresión infija se sacan e introducen en la expresión todos los operandos que queden en la pila.

En la siguiente tabla se puede ver cómo funciona el algoritmo explicado anteriormente al analizar la expresión A+B*(C+D)*E+F:

Símbolo	Traducción parcial	Pila de operadores
A	A	
+	A	+
B	AB	+
*	AB	+*
(	AB	+*(
C	ABC	+*(
+	ABC	+*(+
D	ABCD	+*(+
)	ABCD+	+*
*	ABCD+*	+*
E	ABCD+*E	+*
+	ABCD+*E*+	+
F	ABCD+*E*+F	+
EOS	ABCD+*E*+F+	

Tabla 0-4: Ejemplo de ejecución del algoritmo de conversión infijo-postfijo.

El resultado final es ABCD+*E*+F+.

En caso de fallar en la conversión también avisará por pantalla con un mensaje de tipo Toast, aunque no será habitual ya que las expresiones fueron sometidas a un duro test.

A.4 Evaluación de expresiones postfijo.

Una vez desarrollada la conversión se debe aplicar un mecanismo de evaluación. Éste también está basado en la utilización de pilas. El algoritmo desarrollado está compuesto por los siguientes pasos:

- Se van leyendo los elementos de la expresión postfijo de uno en uno hasta que se acaben.
- En caso de que se trate de un operando lo introduce en la pila de operandos.
- En caso de que sea un operador, se sacan los dos últimos operandos de la pila, se realiza la operación y se introduce el resultado en la pila de operandos.
- En caso de que el operador sea un negador sólo se saca para operar el último operando de la lista.
- Cuando se acaban los elementos de la expresión el resultado es el que se encuentra en la pila.

En la siguiente tabla se puede ver un ejemplo del funcionamiento del algoritmo de evaluación con la expresión infijo: ¬(1*0*0+1)+(1*¬0), cuyo equivalente en postfijo es: 10*0*1+¬10¬*+.

Símbolo	Op1	Op2	Valor	Pila de operandos
1				1
0				1,0
*	1	0	0	0
0				0,0
*	0	0	0	0
1				0,1
+	0	1	1	1
¬	1		0	0
1				0,1
0				0,1,0
¬	0		1	0,1,1
*	1	1	1	0,1
+	0	1	1	1
EOS				

Tabla 0-5: Ejemplo de ejecución del algoritmo de evaluación postfijo.

Con lo que el resultado de esa operación da 1.

Por último mencionar que a pesar de lo estricto que se es al buscar errores en la función que comprueba si una expresión es correcta, esta función también va a buscar algún error adicional. Se analizan los siguientes posibles errores:

- Si hay un operador y se deben de sacar los dos últimos elementos de la pila de operandos, pero no hay tantos, se mostrará por pantalla el siguiente mensaje: "No hay suficientes operandos para un operador".
- Si el operador es un NOT y no hay ningún operando, el error será: "No hay operando para un NOT".
- Por último, al final del algoritmo solo debe quedar un operando en la pila, si queda alguno más significa que la expresión era incorrecta. En este caso el mensaje Toast será: "Demasiados operandos en la expresión".

A.5 Traduciendo la expresión lógica.

Con todos los algoritmos que se han introducido hasta ahora, el objetivo principal que se busca cuando se presiona la tecla traducir es transformar la expresión lógica introducida en una tabla de verdad.

Para ello se aplican por el orden indicado, los algoritmos para comprobar si la función es correcta y el que convierte la expresión a postfijo. Tras estos pasos nos queda una expresión postfija con valores que no son numéricos. Para calcular la tabla de la verdad lo que se tiene que hacer, es hacer un bucle que recorra todos los posibles valores que pueden tener esas variables. Como se sabe, serán (2 elevado al número de variables) posibles valores.

Para cada conjunto de valores se evaluará la expresión generando la tabla de verdad de esa expresión. Con esa tabla de verdad se podrían obtener fácilmente también el sumatorio de minitérminos y el producto de maxitérminos. Ya que para el sumatorio es la misma tabla y para el producto, justo la opuesta.

B - Apéndice B: Mostrar el esquemático

En este apéndice se va a explicar con todo detalle el funcionamiento de la actividad que muestra los esquemáticos de cualquiera de las funciones simplificadas por el traductor.

Como ya se explicó en la memoria para cualquier dispositivo se calculará un paso en función del que se representará todo lo demás, en este lienzo Canvas dicho paso valdrá un onceavo del ancho de la pantalla del dispositivo.

Para entender el funcionamiento de esta actividad se va a dividir en varias etapas distintas.

B.1 Primera etapa: Dibujar las variables.

El primer paso en que se ha dividido la tarea es el de dibujar las variables y sus negadas. Se realizará en la parte izquierda del lienzo Canvas y se prolongarán en vertical. El esquema que representa el aspecto que tendrán estas variables en función de "p" es el siguiente:

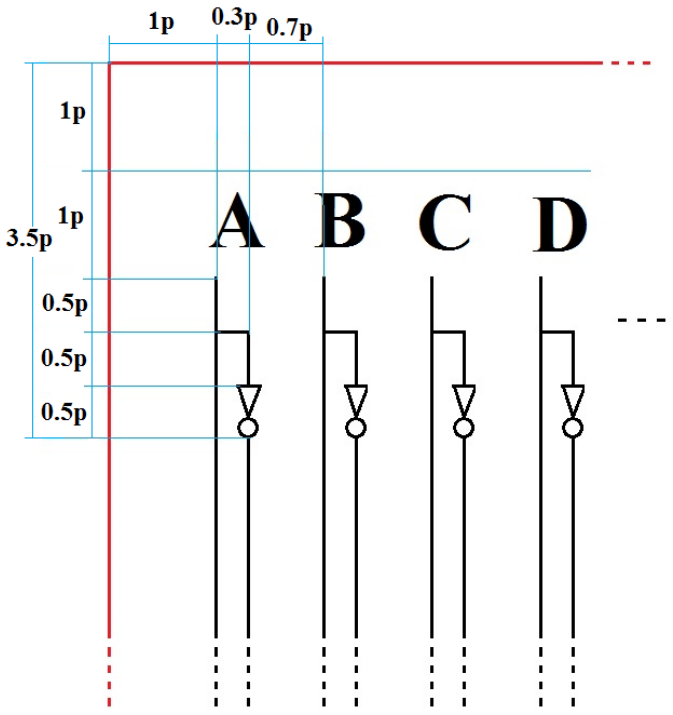

Figura 0-1: Esquema de la colocación de las variables en el esquemático.

La prolongación en vertical se realizará hasta la última puerta AND y habrá tantas variables como las que indique el conjunto de variables actual.

B.2 Segunda etapa: Dibujar las puertas AND.

La segunda etapa es la que se encarga de dibujar tantas puertas AND como términos haya y unir cada una de sus patas con la variable correspondiente.
Para facilitar el funcionamiento de esta etapa se ha creado una clase AND que se definirá a continuación:

B.2.1 La clase AND.

La clase AND se ha definido para representar cada una de las puertas AND del circuito. Esta clase posee los siguientes atributos:

- Una cadena de caracteres con el término al que se refiere.
- Una cadena con el nombre de las variables que contiene y otra para el valor de cada una.
- Las coordenadas del comienzo en x e y
- Las coordenadas de la pata de salida de la AND en x e y.
- El paso "p" que se está utilizando.

La clase contiene un único constructor en el que se pasan algunos de los valores para con ellos calcular el resto.

También se han implementado varios métodos consultores:

- int getNumPatas(): Que devuelve el número de patas que tiene esa puerta AND.
- int getCoordenadaYDePata(int numpata): Que devuelve la coordenada Y de la pata que se indica en el argumento sumada a la coordenada Y de inicio de esa puerta. Tiene en cuenta el número de patas que tiene esa puerta AND para realizar un reparto equitativo y simétrico del espacio para sus entradas, ya que todas las puertas AND del circuito tendrán el mismo ancho y alto. Se recuerda que el número de patas de una puerta AND solo puede ir desde 1 hasta un máximo de 7, que es el número máximo de variables.

Para entender este método se adjuntará la siguiente captura con un dibujo de todas las posibles puertas AND según su número de patas:

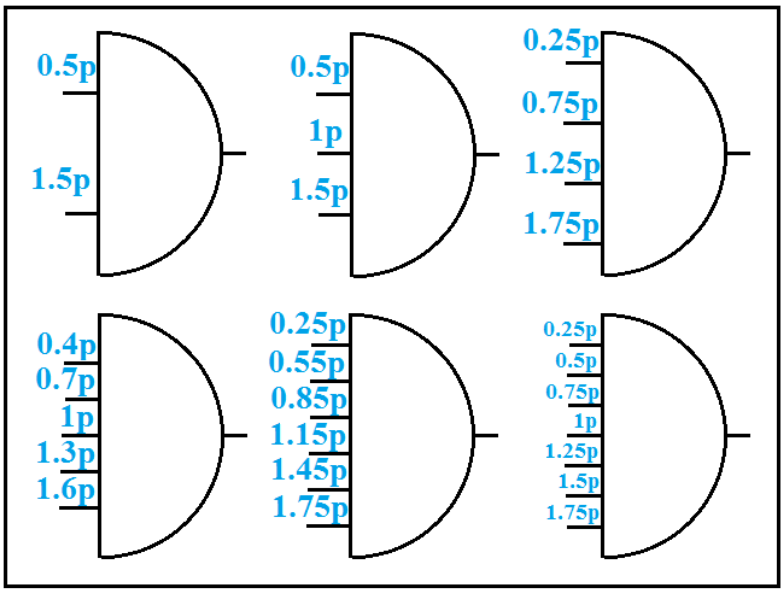

Figura 0-2: Esquema de las distintas puertas AND posibles.

Para completar el esquema, señalar que todas las AND serán de 2*p de alto y una semicircunferencia de radio "p". Estará coloreada de negro y su salida estará justo en la mitad de la altura.

B.2.2 Colocación de las puertas AND.

En el método onCreate se creará una lista de ANDS con tantas puertas AND como términos hubiese en la cadena de la función simplificada.

Al constructor de las puertas AND hay que mandarle la cadena de caracteres del término que le toca y las coordenadas X e Y iniciales y finales. Así que para entender donde estarán colocadas estas puertas se ha realizado el siguiente esquema, a partir de la última variable representada en la parte izquierda del lienzo.

Como se puede ver en la imagen:

- En el eje Y, las puertas estarán medio "p" más abajo del fin de las puertas NOT. Ya que los cruces que se hagan entres las variables y las patas de las puertas deben de realizarse una vez que la puerta NOT esté acabada de dibujar. También destacar que existirá un margen de 1*p entre cada nueva puerta AND.
- En el eje X simplemente destacar que hay una distancia de 1*p entre la última variable y el inicio de la puerta.
- Como ya se comentó anteriormente las puertas miden 2*p de alto y 1*p de ancho.

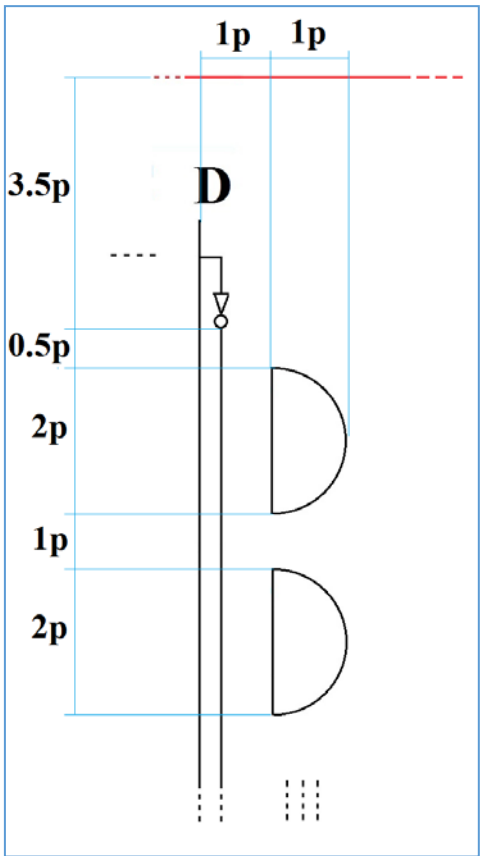

Figura 0-3: Esquema de la colocación de las puertas AND en el esquemático.

B.2.3 Dibujando las puertas AND.

Ya en el método onDraw, cuando todas las puertas AND están definidas y tienen calculadas sus coordenadas, se llama a la función llamada pintar_and con cada una de ellas. Esta función recibe en sus argumentos, la puerta AND que se quiere pintar, el lienzo Canvas y el pincel Paint.

En resumen, lo que esta función realiza es un semicírculo de color negro en las posiciones que indica el objeto de la clase AND. También, para todas las patas que tenga dicha puerta, une con una recta negra, desde la coordenada de la pata hasta la posición de dicha variable en la parte izquierda. Por último dibuja un puntito negro en el punto de corte con dicha variable.

Como consideración adicional, en el caso de que la puerta que se está dibujando solo tuviese una pata, lógicamente no se pintará ninguna puerta. Simplemente se unirá la variable correspondiente a la coordenada de salida de la supuesta puerta AND, de manera que atraviesa la etapa.

B.3 Tercera etapa: Dibujar la puerta OR.

La tercera y última etapa complicada es la que se encarga de dibujar la puerta OR que une todas las salidas de las puertas AND ya explicadas.

Al igual que se realizó antes se creará una clase OR que represente dicha puerta. Esto se ha realizado para posibles expansiones del problema en el futuro, ya que en este caso solo habrá una instancia de esta clase y se podría haber ahorrado.

B.3.1 La clase OR.

La clase OR se ha definido de manera similar a la clase AND que se describió anteriormente. Esta clase tiene los siguientes atributos:

- Coordenadas X e Y de la esquina superior izquierda de la puerta lógica.
- Coordenadas X e Y de la esquina inferior izquierda de la puerta lógica.
- Coordenadas X e Y de la salida.
- Un número entero con el número de puertas AND de la anterior ronda.
- El paso "p" que se está utilizando.
- Una variable inventada que consiste en un segundo paso llamado paso_por_cada_numand. Que se utilizará a menudo multiplicado por el número de puertas AND.

Mencionar que para que todas las patas queden dentro del espacio y no justo en los bordes, se pintará la puerta desde un poco más arriba de la posición de comienzo superior, hasta un poco más abajo de la posición de comienzo inferior.

El método constructor de esta clase recibe solo las coordenadas de inicio el número de ANDs y el paso. Con esos parámetros calcula todos los demás:

- El "paso_por_cada_numand" indica varias cosas distintas. Una de las más importantes es que de él dependerá el alto de la puerta. Dependiendo del número de puertas AND en la fase anterior, este cogerá valores distintos. Por ejemplo en el caso de que solo exista una AND será de 0.9, si hay 3 o 4 valdrá 0.6 y si hay más siempre valdrá 0.3. Esto se hace para evitar que una posible OR de pocas entradas sea muy pequeña.

- Para la esquina inferior izquierda se pone la misma coordenada X que en la superior y para la coordenada Y se le suma a la de la superior paso_por_cada_numand * (número de ands -1) * paso. Esto quiere decir que el alto de la puerta OR dependerá del número de patas, ya que todas estarán separadas por una cantidad igual a paso_por_cada_numand * paso.

- Por último para la coordenada Y de la salida se hace con la diferencia entre las dos de comienzo partido de 2. Para la X, se coge la de comienzo

y se le suma 1.1*paso y también (número de ands) * paso_por_cada_numand * paso * 0.75. No es necesario entender este valor exactamente, simplemente es importante ver que el largo de la puerta OR también depende del número de patas.

- Las medidas expuestas anteriormente, suenan muy aleatorias, pero son fruto de mucha experimentación y dibujos en papel, para intentar que la puerta OR quede lo más natural posible. Recordemos que según estas fórmulas, la puerta OR va a ser más alta y más larga cuando su número de patas aumenta. Se podrán apreciar mejor estos detalles observando distintas representaciones realizadas por esta actividad.

En cuanto a los métodos consultores de la clase OR:

- double getCoordenadaYDePata(int numand): Devuelve la coordenada Y de la pata consultada. Se calcula de manera lineal, es la coordenada Y de comienzo más paso * paso_por_cada_and * (numand-1).
- double getCoordenadaXDeGiro(int numand): Devuelve la coordenada X de giro de la pata que recibe por argumento. Ya se entenderá en el futuro cuando se comente la colocación de la puerta OR.

B.3.2 Colocación de la puerta OR.

En el método onCreate se creará una instancia de la clase OR. Al método constructor de dichas puertas hay que mandarle de parámetros: las coordenadas X e Y de comienzo de la puerta, el número de ANDs y el paso. De esos 4 parámetros se tienen ya los dos últimos. Sólo falta saber cuáles serán las coordenadas del inicio.

Cada puerta AND de la segunda etapa tiene que llegar a la puerta OR que las reúne todas. Pero no podrá ser con una simple línea recta, sino que tendrá que hacer algún giro antes de entrar, para llegar en horizontal. Así que se ha decidido que se coloque la puerta OR de manera que la primera entre recta y a partir de ahí todas tendrán que hacer un pequeño giro, esto se entenderá mejor viendo la siguiente figura en función del paso "p" y el "paso_por_cada_and" que se denominará como "q".

En la imagen se puede ver como la puerta OR estará más lejos de la etapa de las puertas AND cuanto mayor sea el número de estas. Ya que cuantas más sean, se necesitará más espacio para realizar los giros.

Ya teniendo las coordenadas de inicio y el resto de los datos necesarios, se puede instanciar el objeto de la clase OR. Y solo faltaría dibujarlo.

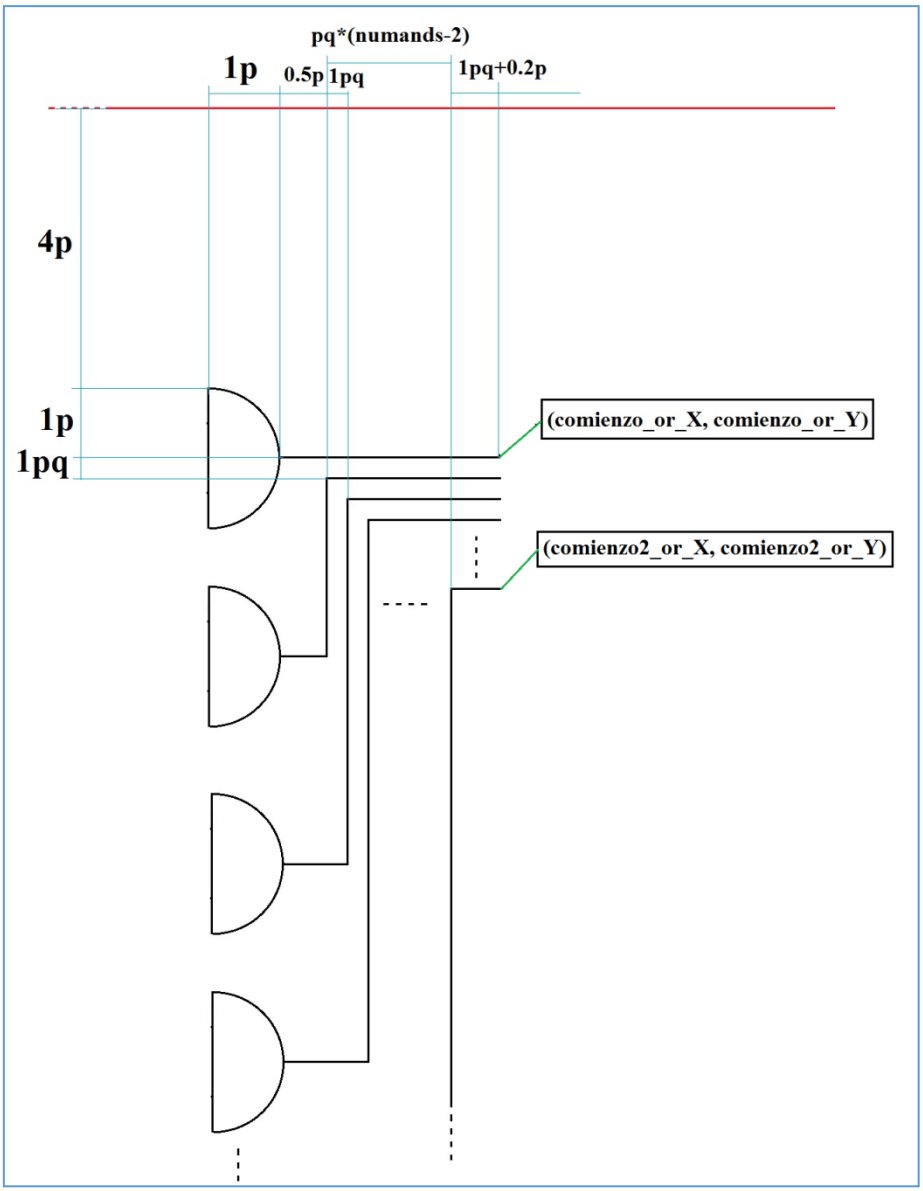

Figura 0-4: Esquema de la colocación de las puertas OR en el esquemático.

B.3.3 Dibujando la puerta OR.

Para dibujar la puerta OR se crea una función que se llama pintar_or que recibe como argumentos la puerta OR a dibujar, el lienzo Canvas y el pincel Paint.

Se va a explicar en resumen como actúa esta función:

- Primero dibuja un rectángulo negro relleno que empieza un poco más arriba del comienzo superior y acaba un poco más abajo del extremo inferior. Esto es debido a que la primera y la última pata entran justas a dichos comienzos y se quiere que la puerta abarque un poco más.
- Segundo se dibuja una especie semicírculo negro detrás de dicho rectángulo que irá hasta el final de la puerta.

- Tercero se dibuja una especie de semicírculo blanco (del mismo color que el lienzo Canvas) al principio del rectángulo que sobre-escribirá parte del rectángulo y dará el aspecto clásico que tienen las puertas OR.
- Por último se pintan todas las rectas. Para ello desde cada puerta AND se avanza en horizontal hasta la coordenada devuelta por getCordenadaXDeGiro. Después en vertical hasta la coordenada que indique getCoordenadaYDePata. Y por último hasta dentro de la puerta. Estas rectas pasarán por encima del semicírculo blanco para dar esa sensación de que se introducen hasta el interior de la puerta.

En el siguiente dibujo se pueden apreciar de manera aproximada los pasos en los que consiste el mecanismo que dibuja las puertas OR:

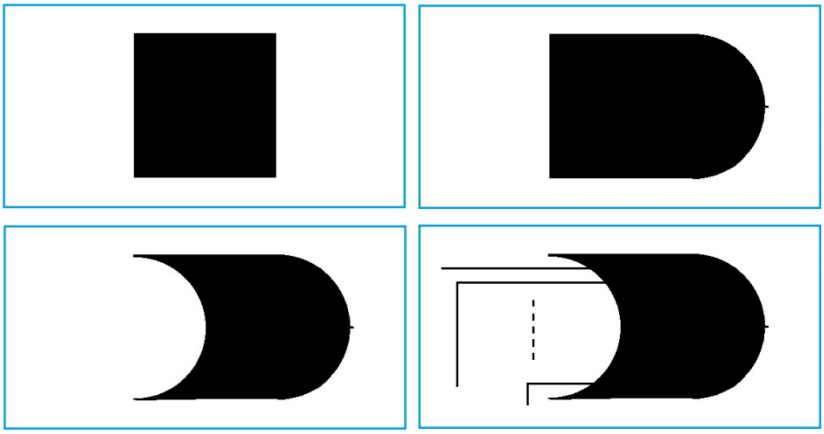

Figura 0-5: Esquema de los pasos para dibujar las puertas OR.

B.4 Última etapa: Dibujar la salida del esquemático.

Para finalizar, el último paso consiste en poner una recta corta que salga de la salida del circuito y que se escriba justo encima la letra F que indica que esa es la función salida. Hay varios casos que debe contemplar esta función:

- El caso más normal, en el que la salida es la puerta OR que reúne los términos.
- Se puede dar el caso de que la función simplificada solo tenga un término, por lo que no hará falta que se dibuje la puerta OR y la salida estará directamente en la única puerta AND.
- Existe la posibilidad de que ni siquiera haga falta que se pinte una puerta AND ya que la función sea directamente equivalente a una de las variables. Por lo que la F saldrá de dicha variable.
- Si la función de salida fuese 0 o 1 habrá que tomar alguna medida para que la actividad también se adapte a ella.

B.5 Ejemplo de funcionamiento.

En este ejemplo se ha introducido la función B*A +D*¬C para ver su esquemático. Se han añadido algunas medidas en función de "p" para que se vea cómo ha sido aplicado todo lo que se ha ido explicando a lo largo del apéndice.

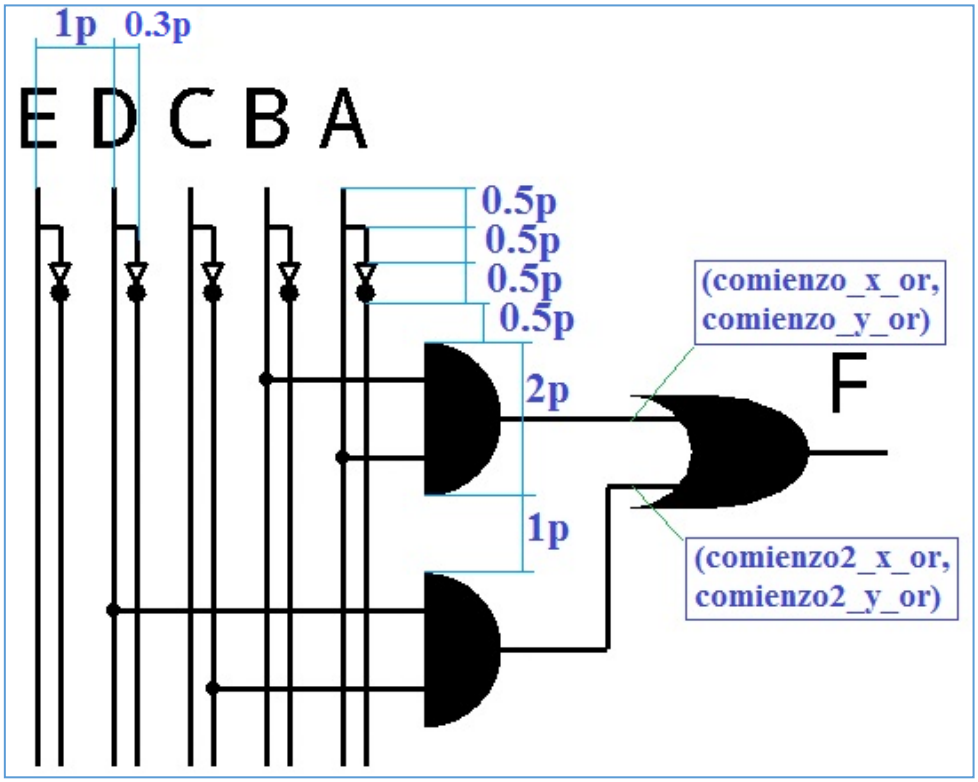

Figura 0-0-6: Ejemplo de un esquemático real.

Referencias

[1] Jerome (J.F.) Di Marzio, “Android, A Programmer’s Guide” Mc Graw Hill

[2] Bruce Eckel, “Piensa en Java”, Prentice Hall

[3] Jesús Tomás Gironés, “El gran libro de Android”, MARCOMBO, S.A.

[4] http://www.androidcurso.com/

[5] Jonathan Simon, "Android Development"

[6] Material “Desarrollo de aplicaciones para entornos móviles”. EPS UAM.

[7] http://developer.android.com/about/dashboards/index.html

[8] http://stackoverflow.com/

[9] https://play.google.com/store/apps

[10] http://en.literateprograms.org/

[11] https://www.youtube.com/user/slidenerd

[12] http://android.alorse.net/

[13] http://www.androidhub4you.com/2013/02/muftitouch-listview-multi-click.html

[14] http://piedralibre.wordpress.com/2010/11/25/android-manifest-una-introduccion/

[15] http://cseweb.ucsd.edu/classes/su10/cse140/Quine.pdf

ANEXOS

A. PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 979 €
- Dispositivo móvil Android..... 350 €
- Tableta Android..... ..... 200 €
- Entorno de desarrollo ..... 0 €
- Material de oficina ..... 20 €
- Total de ejecución material ..... 1.549 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 247,84 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 92,94 €

4) Honorarios Proyecto

- Horas realizando tutoriales: 99 horas
- Horas programando la aplicación: 182 horas
- Horas escribiendo la memoria: 131 horas
- Total: 412 horas a 15 € / hora ..... 6.180 €

5) Material fungible

- Gastos de impresión..... 40 €
- Encuadernación..... 20 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 8.069,78 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto ..... 1.694,65 €

8) Total presupuesto

- Total Presupuesto..... 9.764,43 €

Madrid, Noviembre de 2014
El Ingeniero Jefe de Proyecto

Fdo.: Rubén Jiménez Benito. Ingeniero de Telecomunicación

B. PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de MAPAS VEITCH-KARNAUGH EN 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.