Tetris Lite

Julio Fernández Lasheras
Master Universitario en Ingeniería Informática
Sistemas Encastados

Jordi Bécares Ferrés
Pere Tuset Peiró

Enero 2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Tetris Lite*
Nombre del autor:	*Julio Fernández Lasheras*
Nombre del consultor/a:	*Jordi Bécares Ferrés*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega (mm/aaaa):	01/2018
Titulación:	*Master Universitario en Ingeniería Informática*
Área del Trabajo Final:	*Sistemas Encastados*
Idioma del trabajo:	*Castellano*
Palabras clave	*Empotrado, Juego, Tetris*

Resumen del Trabajo:

El siguiente proyecto, está basado en el ámbito de los sistemas embebidos, se pretende buscar la importancia de estos sistemas, que tenemos presentes en tantos ámbitos de nuestra vida, que no somos conscientes.

El sistema embebido que se quiere construir es un clásico juego que en la década de los 80 y los 90 causo una revolución teniendo una grandísima aceptación y una fama que aun hoy en día conserva, se trata del juego del Tetris.

Para llevar a cabo la tarea de crear un sistema embebido que corra el juego del Tetris, tendremos que buscar el hardware más adecuado, y desarrollar el software que se adapte a ese hardware y tenga toda la lógica del juego.

El hardware seleccionado, será crucial en el desarrollo del juego, pues debe proporcionar una experiencia de juego fluida, así como contar con los periféricos necesarios para dotar al juego de todas las entradas y salidas.

El Software que se tiene que diseñar, tiene que estar pensado en los elementos hardware seleccionados, y deben orientar el desarrollo del juego a los diferentes dispositivos de entrada y salida con los que se cuenta.

Abstract:

The following project, is based on the scope of embedded systems, I will try to seek the importance of these systems, that we have present in many areas of our lives, despite we are not aware.

The embedded system that is wanted to build is the classic game that in the 80s and 90s caused a revolution with a very great acceptance and fame that even today conserves, it is the game Tetris.

To carry out the task of creating an embedded system that runs the game of Tetris, we will have to find the most suitable hardware, and develop the software that adapts to that hardware and has all the logic of the game.

The selected hardware will be crucial in the development of the game, as it should provide a smooth gaming experience, as well as having the necessary peripherals to provide the game with all the inputs and outputs.

The software must be developed having in mind all of the selected hardware elements, and must guide the development of the game towards the different input and output devices that are available.

A mi familia, por haberme apoyado no solo en esta etapa, sino en todas a lo largo de mi vida.

A Lourdes, por ayudarme cuando más lo necesitaba, por su gran paciencia conmigo, y por apoyarme hasta el final.

A la UOC y al Jordi, por haberme transmitido el modelo educativo y conocimientos

Gracias

Índice

Lista de ilustraciones

Lista de tablas

1. Introducción

1.1. Contexto y justificación del trabajo

En el presente proyecto se desarrolla el clásico juego del Tetris. El Tetris es un videojuego creado en 1984 por Alekséi Pázhitnov, un ingeniero informático ruso, mientras trabajaba en el centro de computación de la academia de las ciencias de la URSS, por lo que no pudo recibir los derechos de autor del juego, sino que se los quedo el gobierno de la URSS, para quien trabajaba.

Alekséi, hizo los primeros prototipos para MS-DOS, sin embargo, el juego empieza a ganar popularidad a medida que se porta a nuevas plataformas, comenzó con el IBM PC, y paso al Apple II, Commodore 64 y a varias consolas después, no nos olvidemos que las consolas son dispositivos embebidos, como la Atari. No es hasta su salida en la Game Boy, en 1989 que este juego gana mucha más popularidad y se convierte en uno de los juegos más populares de la época. Desde entonces ha sido portado a máquinas recreativas, consolas portátiles (PSP, Nintendo DS…) y consolas de sobremesa (Xbox 360, PlayStation 3…), ordenadores, versiones web… en definitiva, ha sido uno de los videojuegos más importantes de la historia con más de 170 millones de copias vendidas en 2016,

Actualmente el nivel tecnológico de los sistemas embebidos, permiten reducir el tamaño de aquellas maquinas arcade, pudiendo tener una gran cantidad de juegos en la palma de la mano y con unos gráficos y una conectividad que no se podía plantear en el origen de las máquinas recreativas.

En el siguiente proyecto, se va a portar a un sistema embebido, que veremos en siguientes apartados de forma detenida, formado por un Launchpad de Texas Instruments, y un kit que lo dote de pantalla joystick y botones para poder jugar.

La mecánica del Tetris es relativamente simple, dispondremos de un tablero con diferentes medidas, en función de la pantalla que dispongamos, resolución o de gustos, pues hay versiones que nos permiten personalizarlo, sobre la que irán apareciendo diferentes piezas. Estas piezas tienen la característica de ser tetrominos, una forma geométrica compuesta por cuatro cuadrados iguales conectados entre si, como podemos ver en la Ilustración 1.

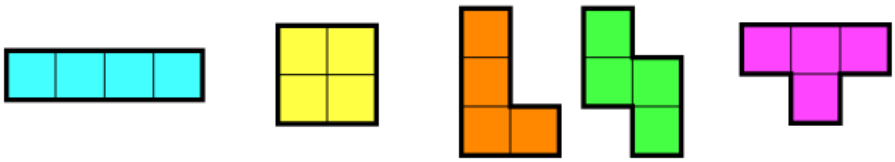


Ilustración 1 - Figuras del Tetris

Estas piezas aparecen por la zona superior de la pantalla, y se van desplazando hacia el inferior del tablero, en el transcurso en el que la pieza está bajando, el usuario puede rotar las piezas para así ajustarlas, de modo que pueda llegar a completar filas completas. Una vez se consigue completar una fila, esa fila desaparece, y si hubiera piezas por encima de ella, se desplazan tantas filas hacia abajo como filas se hayan limpiado. Con el paso del tiempo, aparece una dificultad, la velocidad con la que bajan las fichas es mayor, por lo que tenemos el reto de colocar las fichas en la posición y orientación que más nos convenga en un menor tiempo. El juego termina cuando al aparecer una ficha en la parte superior del tablero, esta se encuentra ya ocupada.

1.2. Descripción del trabajo

El proyecto consiste en ser capaz de seleccionar una placa de desarrollo, unos periféricos, y recopilar todas las reglas y excepciones que pueda tener el Tetris, para así adaptarlo al sistema embebido seleccionado, y crear un prototipo de videoconsola en el que pueda correr el juego.

Para lograr tener el sistema, tendremos que buscar los elementos hardware, seleccionar el IDE que más se adapte para cubrir nuestras necesidades, buscar, modificar o crear las librerías necesarias para controlar todos los elementos, y desarrollar el juego adaptándolo al entorno escogido, resolución de pantalla, controladores…

Esquemáticamente, la solución final sería algo como lo que tenemos en la Ilustración 2:

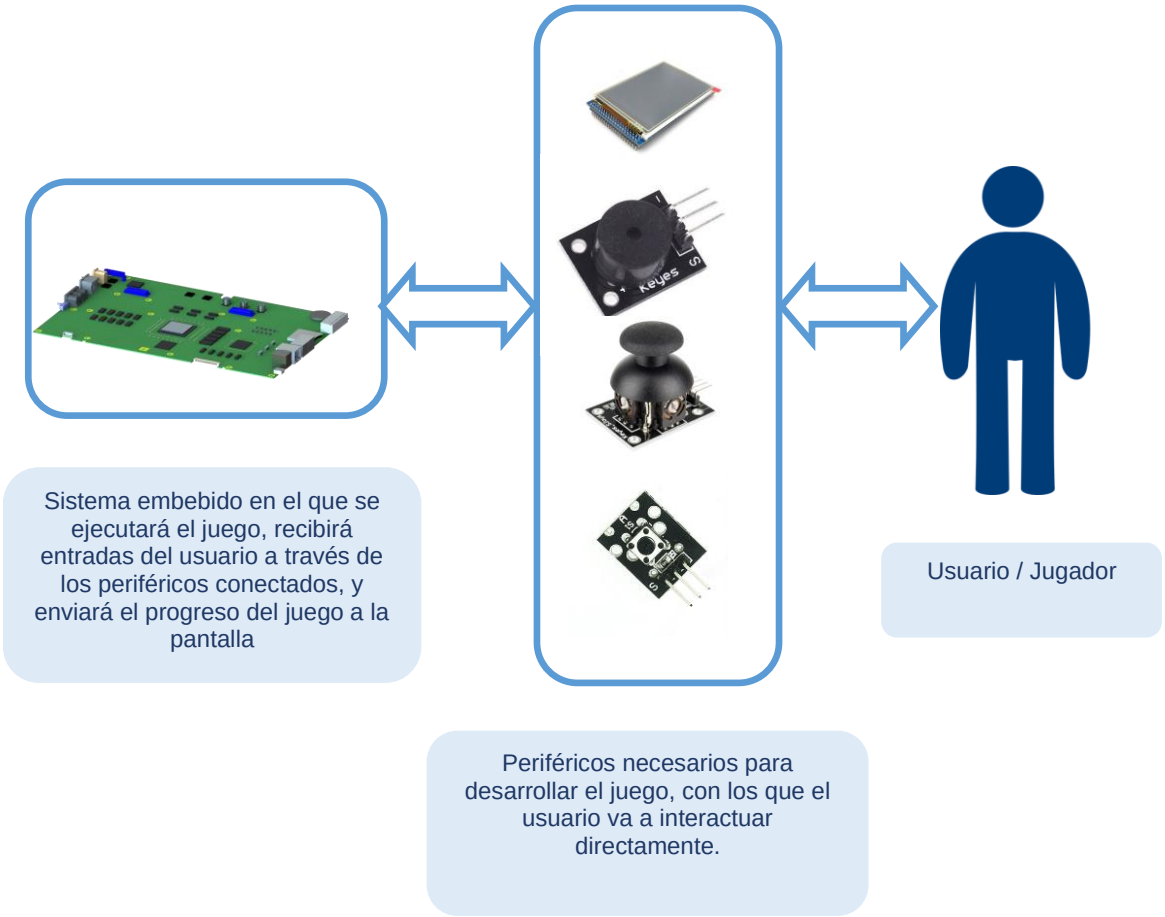


Ilustración 2 - Esquema Solución

1.3. Objetivos del TFC.

El principal objetivo marcado para este proyecto es disponer de manera funcional el juego del Tetris corriendo sobre la placa de desarrollo. Pero para lograr eso, se han dividido los objetivos entre principales, los cuales son necesarios para una operativa básica del juego, y secundarios o mejoras, los cuales aportaran más valor añadido a la solución.

- Objetivos principales:
    - Desarrollar interfaz gráfica basada en el LCD
    - Dotar al sistema de interfaz de usuario vía joystick
    - Implementar funcionalidad de girar la ficha con el botón A (S1)
    - Implementar lógica del juego

- Objetivos secundarios o mejoras:
    - Implementar pantalla inicial del juego
    - Implementar histórico de mayor puntuación
    - Integrar librería de sonido
    - Integrar la funcionalidad de pausar partida
    - Integrar la funcionalidad de guardar partida
    - Implementar diferentes niveles de dificultad en el juego (este objetivo, no estaba entre los propuestos inicialmente, pero se ha desarrollado como una mejora del sistema).

Los objetivos están ordenados según se ha creído que podía ser la importancia en la solución final.

1.4. Enfoque y método seguido.

Como sabemos, existen varias estrategias para desarrollar un proyecto:

- Desarrollar un producto nuevo. En este caso, no tenemos ninguna base, y estamos definiendo nosotros los objetivos en base a los requisitos que nos marquemos
- Adaptar un producto existente. De entre los elementos que nos podamos encontrar en el mercado, buscamos los que más se adapten y puedan cubrir las necesidades marcadas.
- Desarrollar un producto nuevo en base a elementos existentes. Analizamos el mercado en busca de elementos genéricos que nos puedan servir, y los modificamos acorde a los objetivos que tenemos marcados, y desarrollamos las partes que necesitemos para alcanzar el objetivo final.

Por lo que una vez estudiados los enfoques que le podemos dar al proyecto, vemos como el que más se adapta a nuestro caso, es el de “Desarrollar un producto nuevo en base a elementos existentes”.

Por ello, se buscan los elementos hardware necesarios para desarrollar el Tetris, y se analizan para ver que pueden cumplir con los objetivos marcados, y a nivel de funcionalidades, el Tetris al ser un juego creado hace más de 30 años, hay que obtener las normas y adaptarlo al sistema que se está creando.

Para abordar el proyecto, se definen tres etapas fundamentales, las cuales son:

- Documentación y análisis

  En esta primera etapa, la cual en el proyecto ha sido la que más tiempo ha requerido, lo que se hace es analizar los distintos elementos hardware con los que se contaba, y una vez vistas las características que tenían, documentarse de cómo funcionaban, cuál era el entorno de desarrollo, que RTOS se adaptaba mejor y podía cubrir las necesidades…

  Al igual que se analizan las características hardware y la forma de interactuar con el sistema embebido, también hay que analizar el juego del Tetris, ver las normas que tiene, las casuísticas y movimientos prohibidos, y analizar las funcionalidades que se van a querer implementar y si son posibles.

- Desarrollo y pruebas

En este punto, ya están seleccionados los elementos fundamentales, que componentes hardware usar, que entorno de desarrollo, y cuáles eran los objetivos principales a desarrollar y cuales podrían ser las mejoras en caso de disponer de tiempo.

En cada objetivo o funcionalidad cumplida, se buscaba mantener un sistema libre de errores y warnings, tanto de compilación como de usabilidad, por lo que se probaban todos los posibles casos y se depuraba en caso de ser necesario, con esto se ha conseguido que los errores que hubiera se fuesen corrigiendo a la vez que el desarrollo, y así no llegar a una situación final donde tuviese que corregir gran cantidad de errores, con el aumento de tiempo que llevaría.

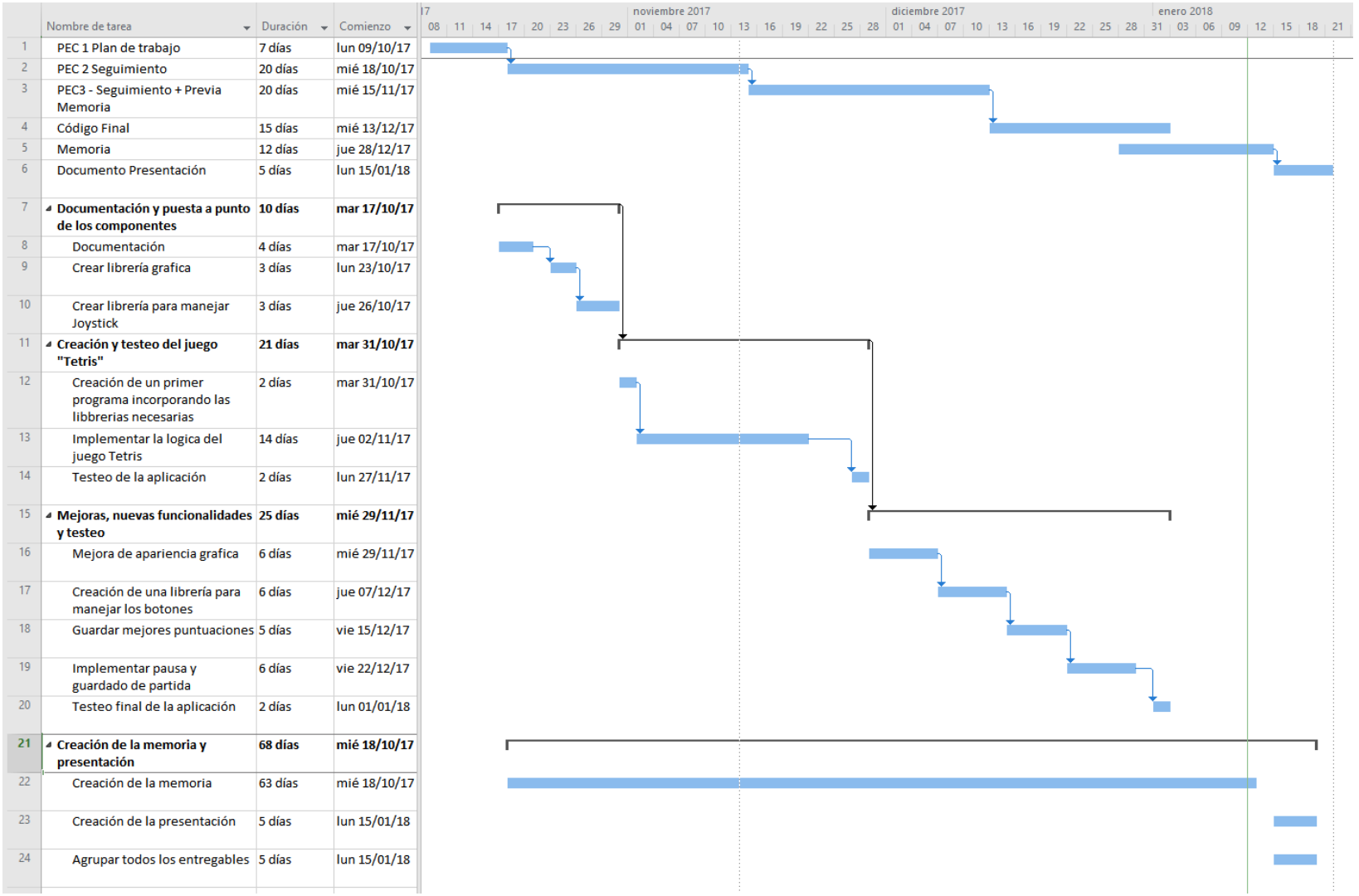
- Memoria

Como todo proyecto que se desarrolle, una vez completado, es necesaria una documentación en la que se expliquen las diferentes fases, como se ha desarrollado, un manual de usuario, o posibles bugs que pudiera tener el desarrollo.

1.5. Planificación del trabajo.

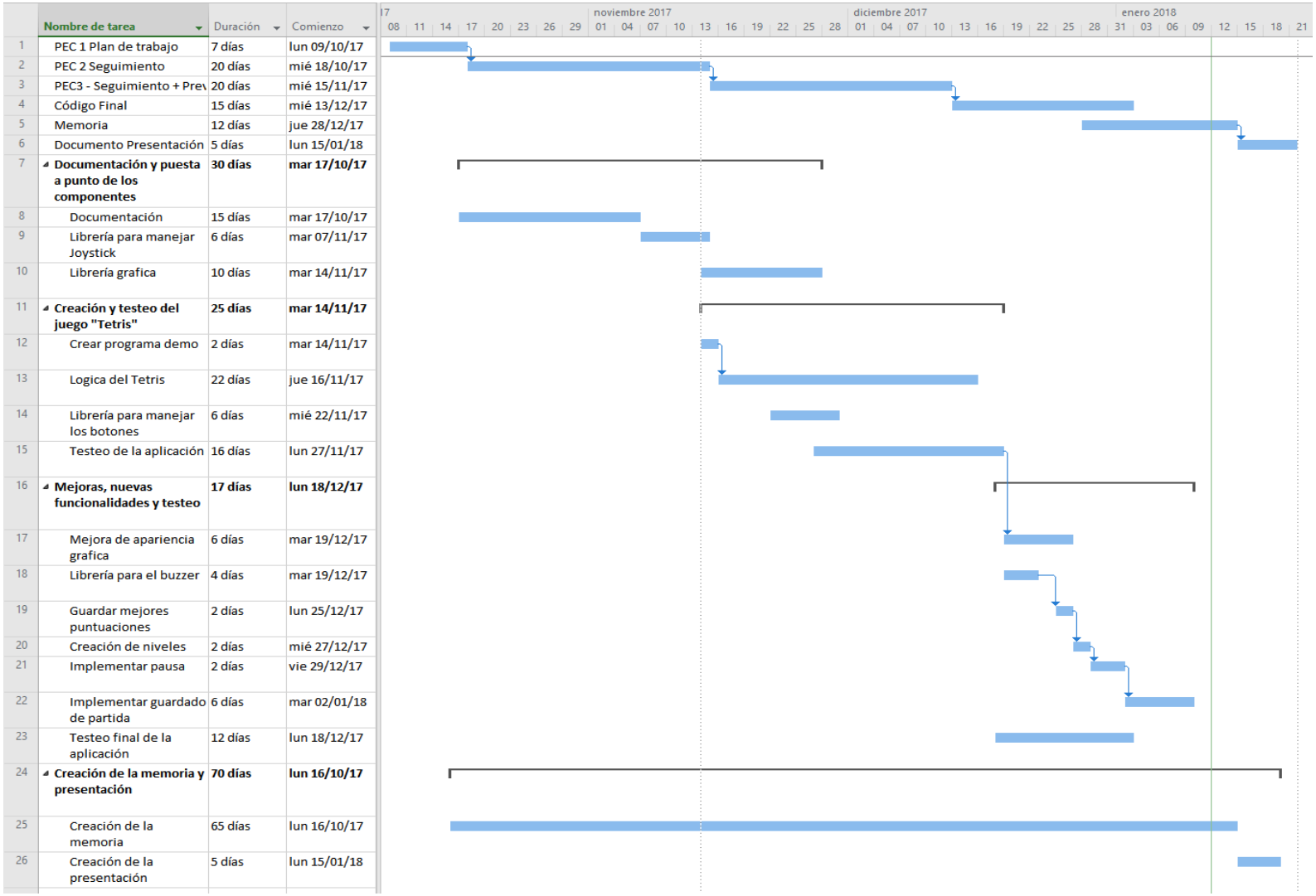
Como ya se ha comentado anteriormente, la planificación de este proyecto se ha hecho en tres fases

- Documentación y análisis. Donde se incluyen tareas como:
    - Documentación del entorno, y estudio de herramientas
    - Librería gráfica, se pretende buscar o crear una librería que nos permita manejar el LCD y de este modo poder representar el juego sobre él.
    - Librería para manejar el joystick, será el principal elemento de control del usuario, por lo que es necesario encontrar la forma de acceder a los datos que nos da de su estado.
- Desarrollo y pruebas. Estarían todas las tareas que tienen que ver con la codificación y el testeo de la misma. Con el fin de ver más claro objetivos principales y secundarios, aparecen separados en el diagrama.
    - Objetivos principales:
        - Creación de un primer programa, para así validar que entendemos el manejo del LCD y el joystick
        - Implementar la lógica del juego
        - Testeo de la aplicación.
    - Objetivos secundarios
        - Mejora de la apariencia gráfica, buscando que sea amigable y atractiva
        - Librería para el manejo de botones, ya que disponemos de dos botones, usarlos en vez de utilizar el botón del joystick
        - Guardar mejores puntuaciones
        - Implementar la pausa y guardado de partida
        - Testeo de la aplicación
- Memoria. Donde se van a agrupar las tareas que tienen que ver con el presente informe, así como la creación de la presentación
    - Creación de la memoria
    - Creación de la presentación
    - Agrupar todos los entregables.



	Nombre de tarea	Duración	Comienzo
1	PEC 1 Plan de trabajo	7 días	lun 09/10/17
2	PEC 2 Seguimiento	20 días	mié 18/10/17
3	PEC3 - Seguimiento + Previa Memoria	20 días	mié 15/11/17
4	Código Final	15 días	mié 13/12/17
5	Memoria	12 días	jue 28/12/17
6	Documento Presentación	5 días	lun 15/01/18
7	**Documentación y puesta a punto de los componentes**	**10 días**	**mar 17/10/17**
8	Documentación	4 días	mar 17/10/17
9	Crear librería grafica	3 días	lun 23/10/17
10	Crear librería para manejar Joystick	3 días	jue 26/10/17
11	**Creación y testeo del juego "Tetris"**	**21 días**	**mar 31/10/17**
12	Creación de un primer programa incorporando las libbrerias necesarias	2 días	mar 31/10/17
13	Implementar la logica del juego Tetris	14 días	jue 02/11/17
14	Testeo de la aplicación	2 días	lun 27/11/17
15	**Mejoras, nuevas funcionalidades y testeo**	**25 días**	**mié 29/11/17**
16	Mejora de apariencia grafica	6 días	mié 29/11/17
17	Creación de una librería para manejar los botones	6 días	jue 07/12/17
18	Guardar mejores puntuaciones	5 días	vie 15/12/17
19	Implementar pausa y guardado de partida	6 días	vie 22/12/17
20	Testeo final de la aplicación	2 días	lun 01/01/18
21	**Creación de la memoria y presentación**	**68 días**	**mié 18/10/17**
22	Creación de la memoria	63 días	mié 18/10/17
23	Creación de la presentación	5 días	lun 15/01/18
24	Agrupar todos los entregables	5 días	lun 15/01/18

Ilustración 3 - Planificación inicial



	Nombre de tarea	Duración	Comienzo
1	PEC 1 Plan de trabajo	7 días	lun 09/10/17
2	PEC 2 Seguimiento	20 días	mié 18/10/17
3	PEC3 - Seguimiento + Prev	20 días	mié 15/11/17
4	Código Final	15 días	mié 13/12/17
5	Memoria	12 días	jue 28/12/17
6	Documento Presentación	5 días	lun 15/01/18
7	**Documentación y puesta a punto de los componentes**	**30 días**	**mar 17/10/17**
8	Documentación	15 días	mar 17/10/17
9	Librería para manejar Joystick	6 días	mar 07/11/17
10	Librería grafica	10 días	mar 14/11/17
11	**Creación y testeo del juego "Tetris"**	**25 días**	**mar 14/11/17**
12	Crear programa demo	2 días	mar 14/11/17
13	Logica del Tetris	22 días	jue 16/11/17
14	Librería para manejar los botones	6 días	mié 22/11/17
15	Testeo de la aplicación	16 días	lun 27/11/17
16	**Mejoras, nuevas funcionalidades y testeo**	**17 días**	**lun 18/12/17**
17	Mejora de apariencia grafica	6 días	mar 19/12/17
18	Librería para el buzzer	4 días	mar 19/12/17
19	Guardar mejores puntuaciones	2 días	lun 25/12/17
20	Creación de niveles	2 días	mié 27/12/17
21	Implementar pausa	2 días	vie 29/12/17
22	Implementar guardado de partida	6 días	mar 02/01/18
23	Testeo final de la aplicación	12 días	lun 18/12/17
24	**Creación de la memoria y presentación**	**70 días**	**lun 16/10/17**
25	Creación de la memoria	65 días	lun 16/10/17
26	Creación de la presentación	5 días	lun 15/01/18

Ilustración 4 - Planificación final

Para poder plasmar y llevar un seguimiento del progreso se ha empleado la herramienta Microsoft Project, con la cual hemos creado los diagramas de Gantt que vemos en la Ilustración 3 y en la Ilustración 4

Como en la mayoría de los proyectos, suele haber diferencias entre la planificación inicial y la final, demoras por problemas con el material, tareas que se habían sobreestimado, confusión en la toma de requisitos…y así ha sido este caso también, al hacer una comparativa entre los dos diagramas, vemos las siguientes desviaciones, dadas principalmente por el desconocimiento del entorno y, por ende, no poder medir correctamente el tiempo que podrían suponer:

- Documentación. En un principio se pensó que iba a ser una tarea rápida, en la que no habría que invertir mucho tiempo, sin embargo, se ha visto como no ha sido así, sino que se ha cuadriplicado la carga, y ha sido una tarea que, aunque no aparezca reflejado, ha estado presente durante casi todo el desarrollo del proyecto.
- Manejo de joystick, LCD, botones y buzzer. Estos componentes, y en especial el joystick y el LCD que fueron los primeros a los que me enfrente en el proyecto, han sido los más subestimados, inicialmente se pensó que iba a ser una tarea fácil, pero esta se retrasó en más del triple.
- Lógica del Tetris. En este caso, la desviación fue de 8 días con respecto a lo señalado inicialmente, trabajar con interfaces gráficas y la lógica compleja del Tetris ha hecho que se produzca esta demora.
- Testeo de la aplicación. Tanto en la primera fase de objetivos principales, como en la segunda fase, se ha prolongado el periodo de prueba, poniéndolo en paralelo con el desarrollo, con el fin de tener un desarrollo limpio al terminar cada funcionalidad o tarea.
- Objetivos secundarios. A excepción de la funcionalidad de guardado de partida, de la cual hablaremos a continuación, todos los objetivos secundarios han visto reducido su tiempo de ejecución, y esto es debido a que al haber empleado más tiempo en las fases iniciales, el conocimiento adquirido me ha permitido avanzar de una forma mucho más rápida.
- Guardado de partida. Esta funcionalidad, que está declarada como objetivo secundario, se ha quedado fuera del desarrollo, por las demoras iniciales en el desarrollo y la complejidad que suponía, crear nuevas vistas con las que indicar como guardarla, la selección de todas las partidas, volver la partida al estado en que se guardó…

Como hemos visto hay variaciones entre la planificación inicial y final, especialmente en las primeras semanas, esto ha hecho que se retrasasen los plazos del resto de tareas, si bien al dedicar más horas, se han podido completar todos los objetivos a excepción del guardado y reanudado de partida.

1.6. Recursos empleados

Para el desarrollo de este proyecto, han sido necesarios los siguientes elementos Hardware:

- Kit de sistemas embebidos de la UOC, compuesto por:
    - Launchpad MSP432P401R

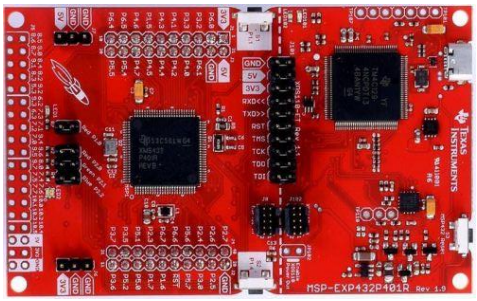


Ilustración 5 - Launchpad MSP432P401R

    - BOOSTXL-EDUMKII - Educational BoosterPack MKII

Ilustración 6 - Educational BoosterPack MKII

- PC Sobremesa (Intel i7, 8Gb de RAM y 2TB de disco duro)
- PC Portátil (Intel i7, 8 GB de RAM y 340 GB de Disco Duro)

Así mismo, también han sido necesarios los siguientes recursos Software instalados en ambos ordenadores:

- Windows 10 64 bits
- Paquete ofimático Microsoft Office 365
- StarUML
- IDE Energia
- Drivers para el manejo del launchpad MSP432P401R

Como recursos documentales, los principales han sido:

https://e2e.ti.com/group/helpcentral/b/weblog

http://energia.nu/guide/

http://energia.nu/reference/

https://www.hackster.io/

https://www.freertos.org/

http://www.43oh.com/

1.7. Productos obtenidos

Una vez vistos los recursos necesarios, y tras un desarrollo, lo que obtenemos al unir todas las piezas es la implementación sobre el launchpad MSP432P401R del clásico juego del Tetris, y que gracias al BOOSTXL-EDUMKII - Educational BoosterPack MKII es capaz de dotar al sistema de una interfaz gráfica gracias a la pantalla LCD que incorpora, así como dotarlo de diferentes entradas, joystick para mover derecha e izquierda la ficha en el juego y dos botones, uno con el que poder rotar las fichas y otro con el que poder pausar el juego. Este módulo que conectamos al launchpad, cuenta también con un buzzer, el cual nos dará la capacidad de añadir la clásica melodía del Tetris antes de empezar una partida, y cada vez que completemos una fila, nos dará un aviso sonoro.

1.8. Breve descripción de los otros capítulos de la memoria.

En el siguiente apartado, vamos a hacer un breve resumen de lo que nos vamos a encontrar en los capítulos de la memoria:

Capítulo 2, Antecedentes:

En este capítulo, vamos ver el estado del arte de los sistemas embebidos, que son, donde están presentes, y veremos los principales sistemas operativos embebidos que hay, así como las razones de porque elegir uno frente a otro, al igual que la elección del entorno de desarrollo. Se vera también cual es el mercado actual tanto de alternativas al dispositivo seleccionado, como el estado del mercado del juego del Tetris.

Capítulo 3, Descripción funcional:

En este capítulo, veremos la descripción del sistema creado, veremos las características del hardware empleado, como se relacionan, y veremos en detalle el desarrollo software, casos de uso, diagramas de secuencia, interfaz de la aplicación…

En este capítulo vamos a hacer una descripción funcional del sistema. Explicando los diferentes módulos, librerías y drivers que lo componen. Sus interrelaciones y modelo de funcionamiento.

Capítulo 4, Descripción detallada:

Se ha separado para este apartado los detalles hardware que no se veía conveniente contar en el apartado anterior, se verán los detalles del Launchpad y del BoosterPack.

Capítulo 5, Viabilidad técnica:

Se analizará si el proyecto técnicamente es posible con los recursos seleccionados y si cumplirán todos los objetivos marcados.

Capítulo 6, Valoración económica:

En este capítulo, se desgranarán los componentes que son necesarios, así como recursos softwares y costes de desarrollo. Una vez vistos estos costes, se verán los costes que son necesarios para industrializar el proyecto, haciendo una simulación con datos orientativos.

Capítulo 7, Conclusiones:

En el capítulo siete, se verán las conclusiones del proyecto realizado, revisando los objetivos cumplidos, y lo que han quedado pendientes, haciendo una autoevaluación del trabajo realizado, y se expondrán una serie de líneas de trabajo futuras con las que mejorar el proyecto.

Capítulo 8, Glosario:

En este capítulo, veremos el glosario con los principales términos usados en el proyecto.

Capítulo 9, Bibliografía

En el capítulo 9, veremos la bibliografía usada para el desarrollo del proyecto y la memoria.

Capítulo 10, Anexos:

En los anexos, será donde veremos el manual de instalación y compilación con Energia, los esquemas técnicos del Launchpad, y la licencia empleada.

2. Antecedentes

2.1. Estado del arte

En este apartado, vamos a estudiar el diferente estado del arte de los elementos que entran en juego en el desarrollo del proyecto:

2.1.1. Sistemas embebidos

Los sistemas o dispositivos embebidos son sistemas electrónicos de propósito específico, es decir, están diseñados para llevar a cabo tareas específicas. Esto es lo que les diferencia de las máquinas de propósito general, como los ordenadores personales, que pueden desempeñar gran variedad de funciones, dependiendo de las características de hardware y software que los definan.

Este tipo de sistemas pueden realizar desde tareas sencillas, como por ejemplo la medición de la temperatura con un termómetro digital, hasta tareas de alta complejidad, como puede ser la decodificación de señales de audio y vídeo con un sintonizador de TV o ejecutar un juego.

Por lo general, son dispositivos de una única placa o tarjeta, conocidos por sus siglas en inglés como SBC (Single Board Computer) o PCB (Printed Circuit Board). Estos dispositivos funcionan con un sistema operativo o con un determinado programa almacenado en memoria y en muchas ocasiones este software está desarrollado por los mismos fabricantes de la placa.

En esta tarjeta se integran todos los elementos o subsistemas de hardware necesarios para efectuar la tarea específica para la que se diseña el sistema: se incluirá una unidad de cálculo o CPU, que en este tipo de dispositivos habitualmente es un microprocesador o microcontrolador. Habrá además un sistema de memoria, un sistema de comunicación (buses) y los dispositivos de E/S o periféricos necesarios. Contendrá, asimismo, un sistema de alimentación: en el caso de dispositivos independientes, el sistema contará con la posibilidad de alojar una batería; si está incrustado en otro sistema mayor, hará uso del sistema de alimentación del mismo.

2.1.2. Sistemas operativos embebidos

Si bien no es necesario, generalmente sobre los sistemas embebidos hay corriendo un sistema operativo, el cual se encarga de la gestión de todas las tareas, control de interrupciones, llamadas a sistema, accesos a memoria…

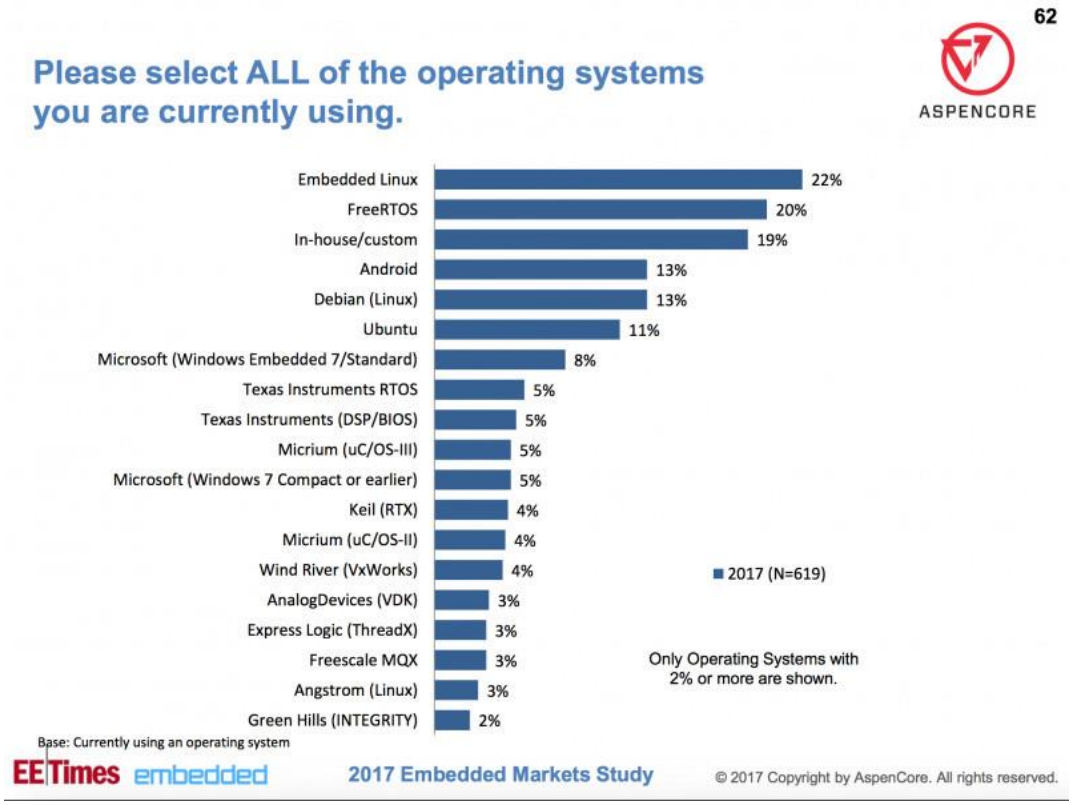


Ilustración 7 - Uso de los sistemas operativos embebidos

Como podemos ver en la Ilustración 7, tenemos tres claros dominantes del mercado de los sistemas operativos embebidos, teniendo el Linux embebido, FreeRTOS y sistemas operativos desarrollados internamente.

En el caso de Texas Instruments – RTOS, tiene solo una cuota de mercado del 5%, y esto es principalmente a que su uso, está pensado más en educación que en servicios productivos.

A la hora de elegir el sistema operativo para desarrollar el proyecto, también se ha tenido en cuenta el entorno de programación y la cantidad de referencias y soporte que se podía recibir.

Como podemos ver en la Ilustración 8, tanto TI-RTOS como FreeRTOS son dos sistemas operativos muy completos con gran cantidad de servicios y de funcionalidades, sin embargo, es TI-RTOS quien cuenta con un mayor número de drivers, además de contar con un buen soporte en foros y la facilidad que te da que sean los propios fabricantes del Launchpad quienes hayan desarrollado el sistema operativo.

RTOS Features Comparison

	Scheduler Type	CMSIS compliant	Services		Interrupt Latency	Device Specific Support	Device Drivers	Flash Size	Low Power Support
TI-RTOS By TI	• Pre-emptive	X	• Tasks • Swis • Hwis • Software Timers	• Semaphores • Mutex • Mailbox • Events	• Zero-latency interrupts supported (115 cycles for ISRs with TI-RTOS calls)	• Systick • Interrupts • Exception Handling	• SPI • SPI-SD • I2C • UART • Timer • RTC • Timestamp • Watchdog	4K to 10K	• Tick Suppression, • Device specific Power Manager
ARM RTX By Keil	• Round-robin (default) • Pre-emptive • Cooperative	√	• Tasks • SWIs • Software Timers	• Semaphores • Mutex • Mailbox • Events	• Zero-latency interrupts supported	• Systick	X	<4K	• Tick Suppression, • Device specific support coming soon
FreeRTOS By Real Time Engineers Ltd	• Pre-emptive • Cooperative	X	• Tasks • Co-routines (task share stack) • Direct TASK notifications	• Software Timers • Semaphore • Mutex • Events	• Zero-latency interrupts supported	• Systick	X	5K to 10K	• Tick Suppression, • Pre and post sleep macros for customization, • Idle task hook.
Micrium OS By Micrium	• Pre-emptive	X	• Tasks • Software Timers	• Semaphores • Mutex • Mailbox • Events	• Unknown	• Systick	X	5K to 24K	• Idle Task Hook • Tick suppression coming soon

4

Texas Instruments

Ilustración 8 - Comparativas Sistemas Operativos

También cabe destacar las diferencias que existen entre los dos entornos de programación con el que trabaja cada sistema operativo, por un lado, tenemos el Code Composer Studio como vemos en la Ilustración 9, el cual utiliza FreeRTOS, y vemos que es un entorno muy complejo, con gran cantidad de herramientas y funcionalidades para depurar el desarrollo. Está orientado a un público profesional o experto en el área.

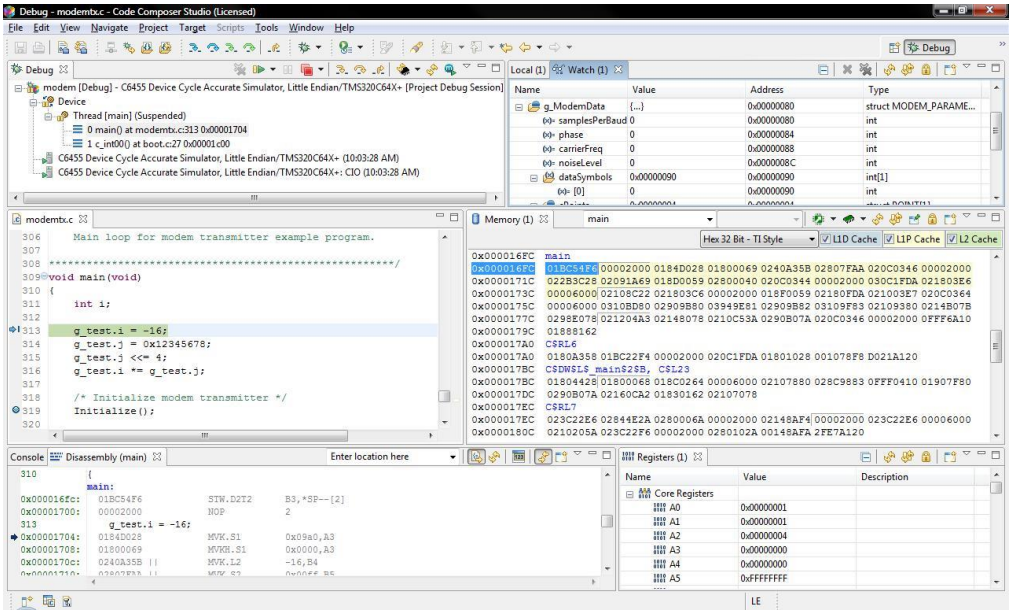


Ilustración 9 - Code Composer Studio

Y como vemos en la Ilustración 10, tenemos la interfaz de Energia, un entorno mucho más sencillo, con una filosofía de programación simplificada, teniendo dos partes principales una para inicializar todo, y otra que este en bucle. Cuenta con gran cantidad de librerías, y soporte para las placas de Texas Instruments, y gran cantidad de dispositivos de Arduino.

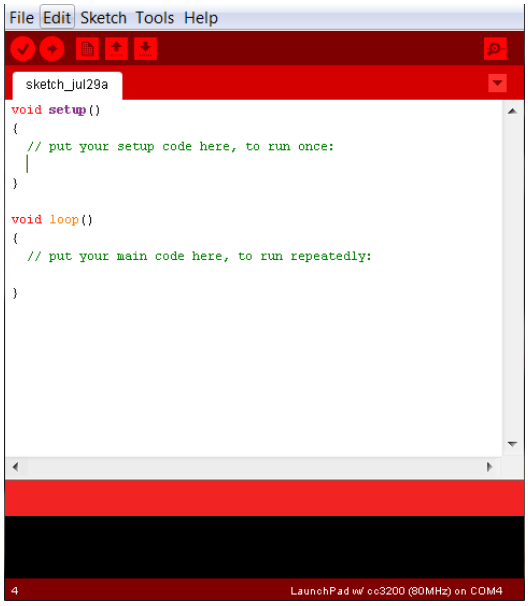


Ilustración 10 - Energia

2.2. Estudio de mercado

Antes de comenzar el proyecto, conviene hacer un estudio de mercado, por si existiese alguna solución ya creada, o que pudiera ayudarnos a hacer más eficiente, o barato el proyecto.

Ilustración 11 - Arduino

Por ello a nivel hardware, la placa más similar en características, sería el Arduino que vemos en la Ilustración 11, se trata de otra placa de desarrollo pensada en educación, aunque su uso en industria está extendido, que cuenta con gran cantidad de periféricos que nos ayudarían a crear el sistema que tenemos actualmente. El principal inconveniente que tiene, es que no dispone de un módulo como el BoosterPack que agrupa todos los componentes necesarios en un dispositivo, haciéndolo más portátil, y con un aspecto de consola.

En cuanto al software del Tetris, vemos como hoy en día no es uno de los juegos más populares, y que debido a las nuevas tendencias, han hecho que este tipo de juego pase a estar instalado en los smartphones, o sea una aplicación web, atrás quedaron los tiempos en que teníamos dispositivos dedicados a este juego como el de la Ilustración 12, y se ha pasado al actual más famoso juego de Tetris, desarrollado por EA tanto para Android como para iOS que vemos en la Ilustración 13, el cual en Android cuenta con entre diez y cincuenta millones de instalaciones.

Ilustración 12 - Maquina Tetris

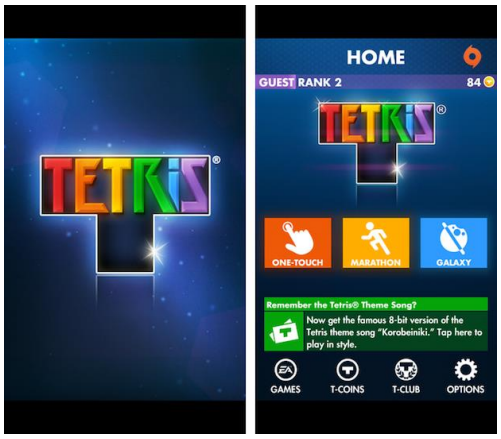


Ilustración 13 - Tetris Android

3. Descripción funcional

A la hora de desarrollar una solución software a medida, es imprescindible analizar convenientemente el proyecto que vamos a desarrollar, analizar la complejidad, requisitos para poder obtener un buen producto.

La fase de análisis comprende la recogida de requisitos, su descripción como requisitos funcionales, requisitos no funcionales y otros requisitos como los requisitos de facilidad de uso. A continuación, se describirán los componentes físicos, así como sus diagramas y como se conectan entre si, se proporcionaran los modelos de caso de uso del sistema y los diagramas de secuencia, de gran importancia para implementar y entender el desarrollo de la aplicación. Por último, se mostrará el interfaz de la aplicación y se describirán las diferentes vistas.

3.1. Sistema Tetris

En este apartado, vamos a explicar cuál es el esquema del sistema desarrollado, así como el diagrama de bloques para ver cuáles son los elementos que tenemos en el sistema, y como interaccionan entre si.

3.1.1. Componentes físicos y diagramas

Los componentes físicos y las características de los elementos que conforman el sistema son:

- Launchpad MSP432P401R: como vemos en la Ilustración 14se trata de una placa de desarrollo que nos permite desarrollar aplicaciones complejas beneficiándonos del bajo consumo que tiene. Esta especialmente pensada para la educación, cuenta con la posibilidad de añadir nuevos módulos, como Bluetooth, Wifi o un BoosterPack entre otros, y todo gracias también a que cuenta con el pineado expuesto, de modo que no es necesario soldar ninguna conexión. Los elementos más relevantes para el proyecto, serían:

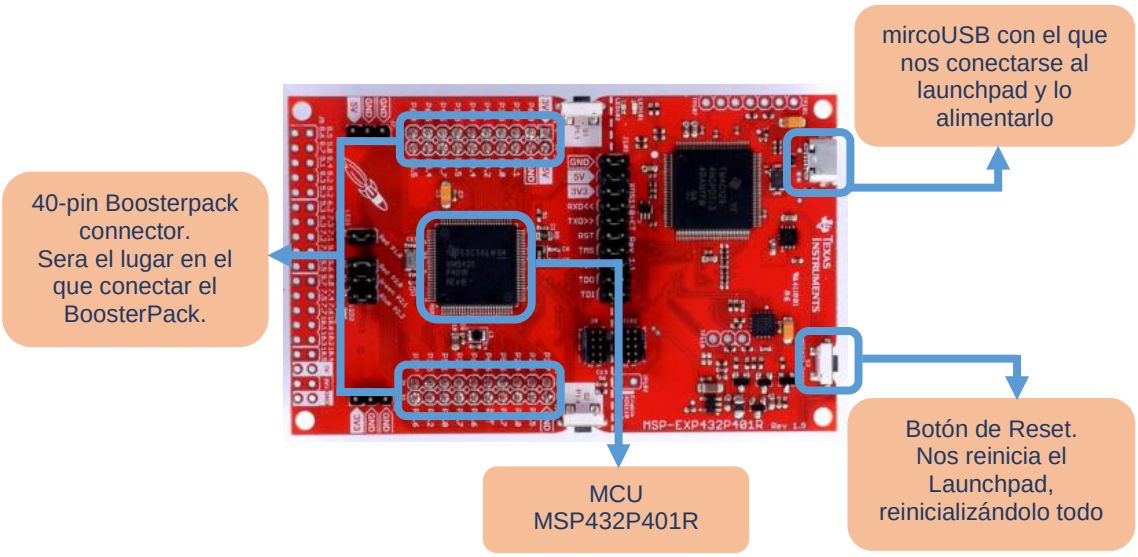


Ilustración 14 - Launchpad MSP432P401R

- Educational BoosterPack MKII. Como vemos en la Ilustración 15, es un elemento muy versátil que proporciona un alto nivel de integración para los desarrolladores y permite un rápido prototipado de una solución completa. Cuenta con gran cantidad de dispositivos de entrada y salida, así como un LCD, joystick, botones, LED… Desde Texas Instruments, cuentan que se diseñó pensando en ser usado con Energia, otro de los motivos por el que se ha empleado en este proyecto. El BoosterPack, cuenta con:

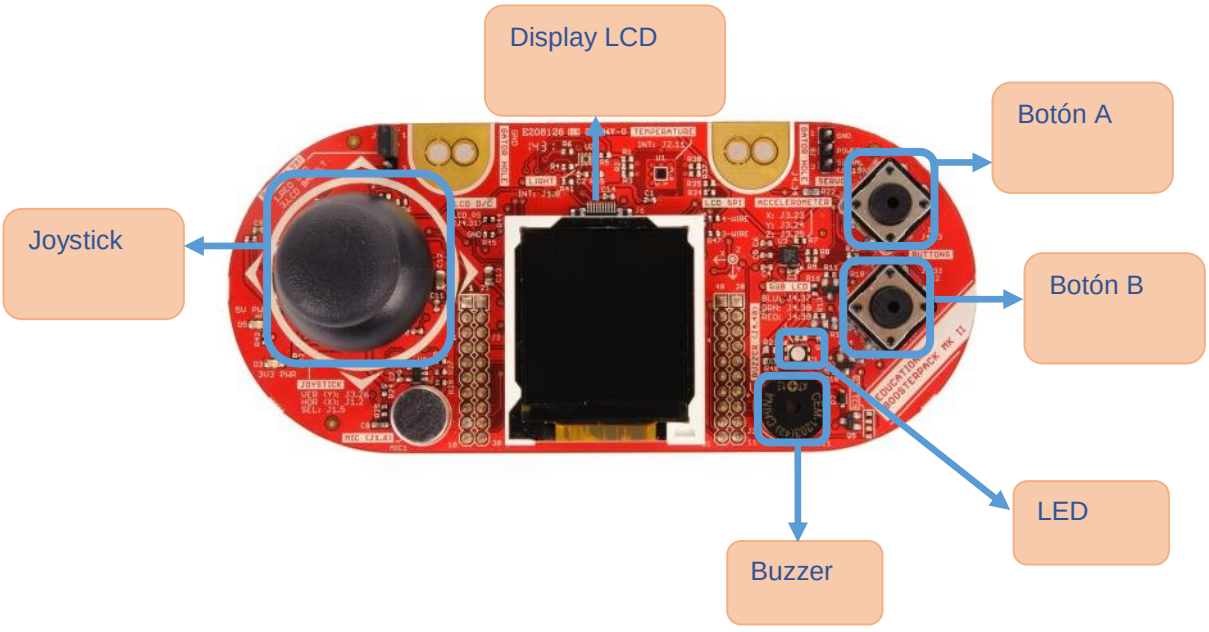


Ilustración 15 - Educational BoosterPack MKII

De los que en este proyecto han sido necesarios:

- 40-pin BoosterPack Connector, serán los pines que tenemos que conectar al Launchpad.
- Color TFT LCD, pantalla en la que mostraremos el Tetris
- 2-axis Joystick, con el que podremos mover las fichas por la pantalla
- User Push Buttons, son dos botones (A y B llamados en la aplicación) con el que podremos rotar las piezas y pausar el juego.
- RGB Multi-Color LED, el cual encenderemos de azul cuando pausemos la partida para dar más información al jugador
- Buzzer, el cual será empleado en la pantalla de inicio para reproducir el tema del Tetris, y durante el juego producirá un sonido cada vez que se complete una fila.

Todos los elementos hardware vistos anteriormente, tienen que comunicarse entre si, como vemos en la siguiente Ilustración 16Ilustración 16.

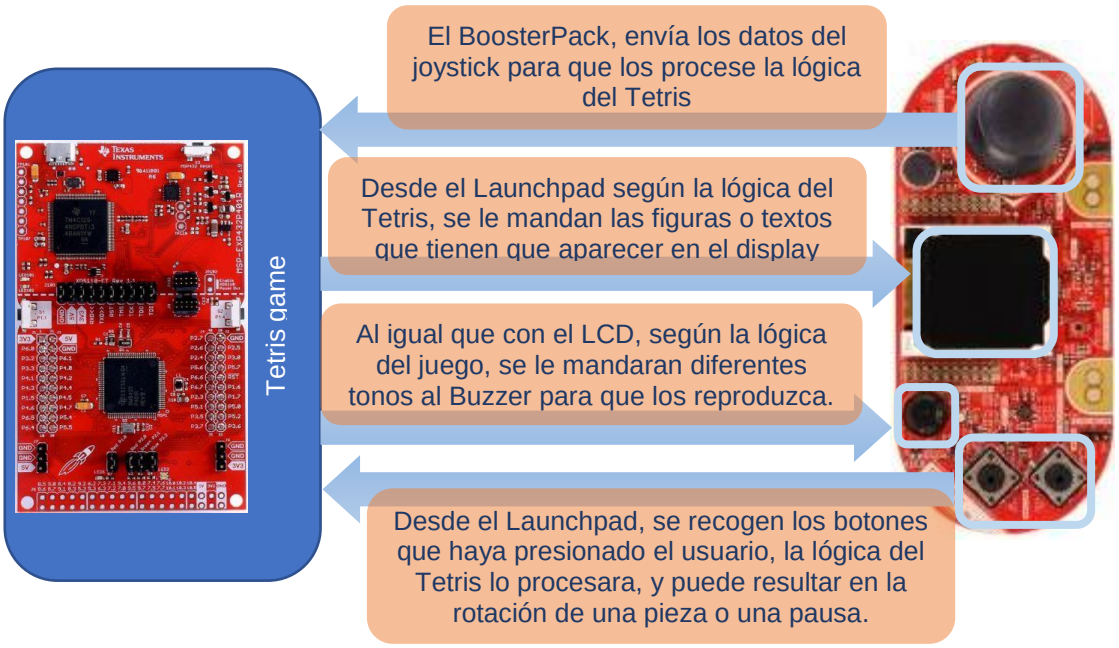


Ilustración 16 - Comunicación elementos

Claro que para poder establecer la comunicación entre componentes y poder hacer toda la lógica del juego, hace falta detrás un Software que lo gestione todo. Podemos distinguir dos tipos de software:

- Software de sistema, el cual da soporte a las aplicaciones, y forman parte de él los drivers, el sistema operativo y el middleware.
- Software de aplicación, es el software de nivel superior que define la función y propósito del sistema embebido. Es el punto de interacción con el usuario.

Cabe resaltar que en función del servicio que vayan a dar los diferentes sistemas embebidos, no todos llevan las mismas capas de software, por ejemplo, los que están basados en una arquitectura basada en microcontrolador no requieren ciertas subcapas del software de sistema como el sistema operativo, sino que es suficiente con un programa o aplicación grabada en la memoria.

En la Ilustración 17 podemos ver las diferentes arquitecturas de los sistemas embebidos.

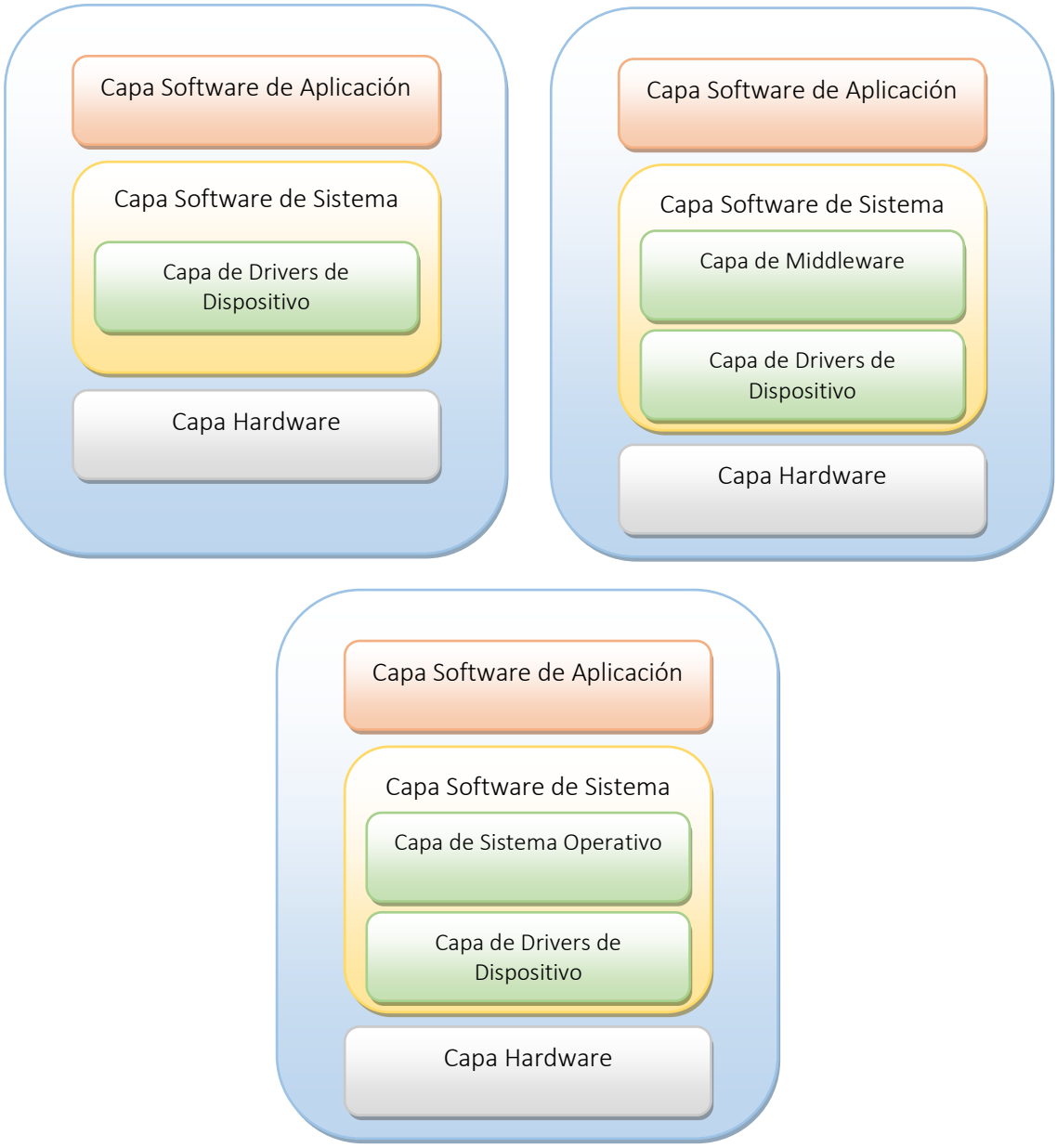


Ilustración 17 - Arquitectura de los sistemas embebidos

De los tres principales modelos que existen a la hora de desarrollar un sistema embebido, para el caso de este proyecto, se ha optado por el último modelo, obteniendo así los beneficios de contar con la capa del sistema operativo y tener las librerías necesarias para controlar los elementos hardware.

Entrando más en detalle, podríamos desgranar la aplicación como vemos.

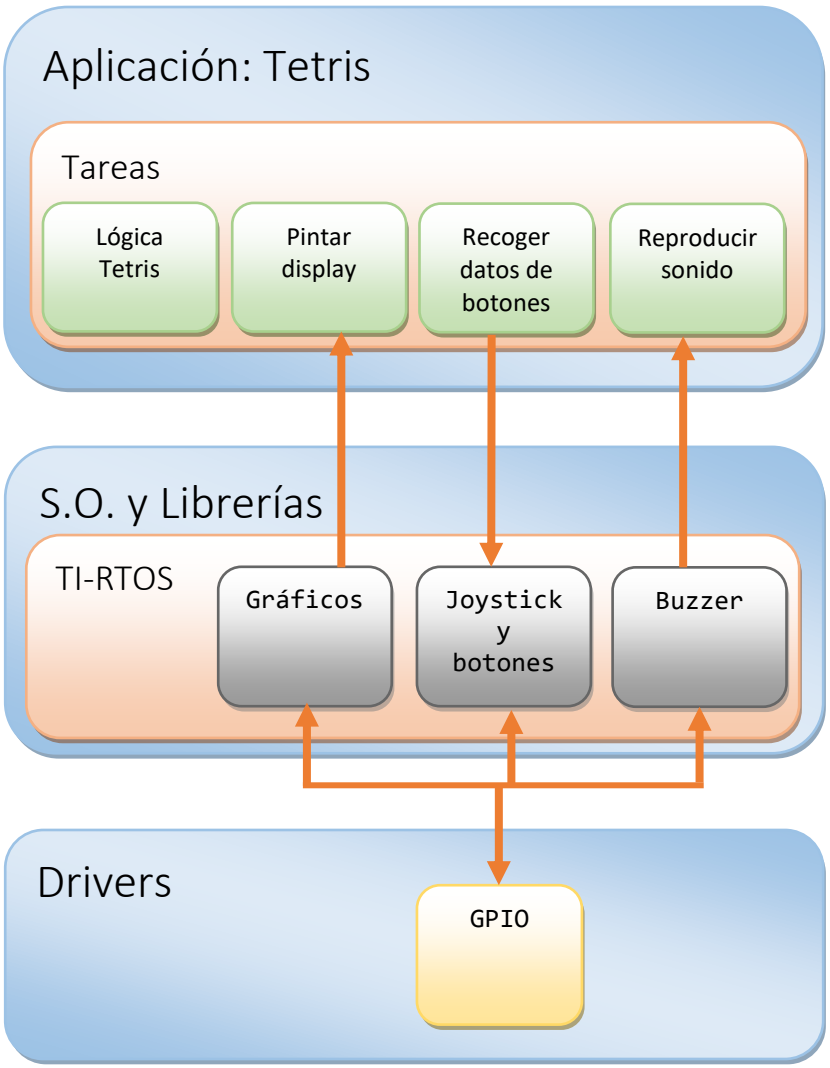


Ilustración 18 - Arquitectura Aplicación

Como podemos ver en la Ilustración 18., se puede separar la arquitectura de la aplicación en tres capas, y cómo interactúan entre si para así llegar a darnos el resultado esperado:

- Drivers: Son el primer nivel software sobre la capa de hardware. La mayor parte del hardware necesita de alguna capa de gestión que lo inicialice y permita acceder a él, ya sea para enviar o recibir datos. Todo sistema embebido como mínimo, va a contar con esta capa.
    - GPIO (General Purpose Input Output) o módulo de Entrada y Salida de Propósito General. Son pines genéricos que pueden ser controlados por el usuario en tiempo de ejecución para la lectura o escritura sobre los elementos hardware que tiene conectados. En nuestro sistema, van a dar alimentación al

BoosterPack, y nos van a permitir controlar todos los elementos de los que está compuesto, display LCD, Joystick, botones, LED y Buzzer.

- Librerías. Se podría decir que un sistema operativo es un conjunto de librerías de software que tienen dos propósitos principales, proporcionar una capa de abstracción para el software que se sitúa en las capas superiores a la del sistema operativo, y minimiza las dependencias del hardware, haciendo el desarrollo de las aplicaciones más sencillo. Por otro lado, es también tarea del sistema operativo la gestión de los numerosos recursos hardware y software del sistema de forma que el sistema completo opere de forma eficiente y fiable. Las principales librerías empleadas en el proyecto son:
    - Gráficos. Nos va a permitir acceder a los recursos del display LCD, gracias a los cuales, vamos a ser capaces de representar toda la interfaz de la aplicación, podremos dibujar la pantalla de inicio, la ventana de instrucciones, y el tablero y todas las fichas
    - Joystick y botones. Con el fin de recoger las acciones u ordenes que el usuario va a querer ejecutar sobre el sistema, dispondrá de dos entradas, el joystick que permita el movimiento de las fichas y los botones que nos permitirán rotar las fichas o pausar la partida.
    - Buzzer. El acceso al mismo, va a posibilitarnos el poder enviar audio al BoosterPack con diferentes frecuencias para de este modo, poder reproducir el tema del Tetris o una señal acústica cuando se completa una fila.
- Aplicación. Va a ser la última capa del sistema, y con la que el usuario va a interactuar, esta debe contener toda la lógica de la tarea que queramos desarrollar, y ser capaz de a través de los componentes mencionados anteriormente, librerías, sistema operativo y drivers acceder al hardware del dispositivo cuando este lo necesite. Las principales tareas que se han mostrado en la ilustración son:
    - Lógica Tetris
    - Pintar display.
    - Recoger datos de botones y joystick
    - Reproducir sonido

El desarrollo de la aplicación, se ha ido haciendo en base al ciclo de vida de los juegos, que vemos en la Ilustración 19

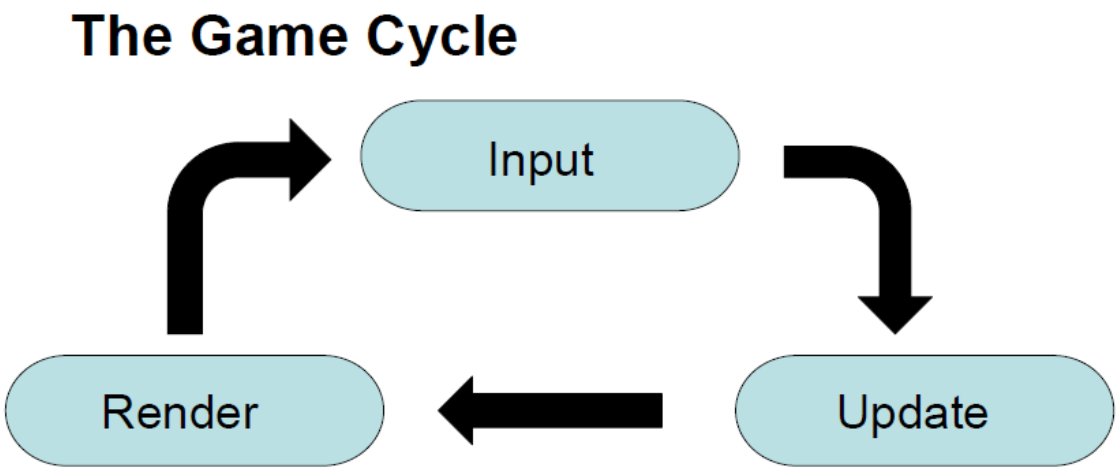


Ilustración 19 - Ciclo de un juego

Como vemos en el esquema, la ejecución de todo juego se realiza de una manera cíclica, teniendo:

- Input. Donde se recoge la entrada que haga el usuario en el sistema, en juegos de PC, esta puede ser la captura del teclado y ratón, en videoconsolas, las pulsaciones sobre los diferentes botones del mando, y en el caso de este proyecto, las pulsaciones que haga el usuario sobre los botones, y la dirección a la que pulse sobre el joystick.
- Update. El juego una vez recogido esa entrada del usuario, que no debemos olvidarnos que en ocasiones la entrada del usuario, será ninguna entrada, el juego debe continuar con su lógica en función de la entrada aplicada por el usuario, así en el Tetris, el update, puede ser girar pieza, mover derecha o derecha, mover más rápido hacia abajo, pausar juego, comprobar que está completa una fila…
- Render. Es el proceso de generar una imagen para que el usuario pueda ver en pantalla el estado del juego. Para poder llegar a la fase render del juego, antes ha debido haber una información por parte de la lógica del juego, ubicada en la parte del update, que diga al render que debe pintar en pantalla.

Y como es de esperar, una vez termina la parte del render, comienza de nuevo la parte del input del usuario, el cual en función del último movimiento suyo y las repercusiones que ha tenido y visto sobre la pantalla, puede hacer que la entrada sea diferente.

3.1.2. Requisitos funcionales

Los requisitos funcionales, a grandes rasgos, son los requisitos generales de los que constara el proyecto.

Este listado de requisitos será de gran utilidad para comprender los casos de uso desarrollados ya que agrupan todas las necesidades de la aplicación de forma abreviada. No se incluye una definición de actores porque no existe tal distinción.

FRQ-001	**Ventana inicial**
Descripción	El sistema debe contar con una vista inicial, que sirva al actor a identificar a lo que va a jugar.

Tabla 1 - Requisito funcional 1, Ventana inicial

FRQ-002	**Ventana HowTo**
Descripción	Como la mayoría de los programas o juegos, son necesarias unas indicaciones básicas acerca de cuáles son los controles

Tabla 2 - Requisito funcional 2, Ventana HowTo

FRQ-003	**Vista Tetris**
Descripción	Esta va a ser la principal vista de la que va a constar el juego, en ella, tendremos: • Tablero de juego • Puntuación • Indicación de la siguiente pieza • Nivel actual

Tabla 3 - Requisito funcional 3, Vista Tetris

FRQ-004	Pausa
Descripción	El sistema debe proporcionarle al usuario la capacidad de pausar la partida en un momento determinado

Tabla 4 - Requisito funcional 4, Pausa

3.1.3. Requisitos no funcionales

Los requisitos no funcionales, son aquellos que no imponen restricciones en el diseño o la implementación del proyecto.

Los requisitos no funcionales describen aspectos del sistema visibles por el usuario que no se relacionan de forma directa con el comportamiento funcional del sistema. Entre los requisitos no funcionales más comunes, tenemos la eficiencia, disponibilidad, precio…

NFR-001	Interfaz intuitiva
Descripción	La interfaz va a ser uno de los elementos más importantes en este proyecto, debe ser atractiva a la par de simple, y buscando que el usuario se focalice en lo que de verdad le importa, el juego.

Tabla 5 - Requisito no funcional 1, Interfaz intuitiva

NFR-002	Sistema embebido de bajo coste
Descripción	Para la realización del proyecto, se tiene que buscar un sistema que tenga un precio contenido, pero que nos ofrezca todas las posibilidades

Tabla 6 - Requisito no funcional 2, Sistema embebido de bajo coste

NFR-001	Libre de licencias de pago
Descripción	Dado que es un proyecto educativo, se busca que, para el desarrollo del sistema, no sean necesarias licencias de pago, o que no permitan su uso para educación.

Tabla 7 - Requisito no funcional 3, Libre de licencias de pago

3.1.4. Desarrollo Software

Una vez que hemos visto los diferentes requisitos funcionales y no funcionales que tiene nuestro sistema, vamos a ver que actores, y cómo interactúan con el sistema

3.1.4.1. Identificación de actores

En este apartado veremos la función del actor en la aplicación. Dado que el proyecto desarrollado es videojuego, pensado para que solo haya un jugador, no tendremos distinción de actores, como podría ocurrir en otro tipo de aplicaciones, como portales webs, herramientas corporativas…

- Actor / jugador. Nuestro actor va a ser el único encargado de interactuar con el sistema. Tanto la primera vez que acceda, como cada vez que pierda una partida, lo primero que se encontrará será la vista del splash de la aplicación, con el nombre del Tetris, y la información de la máxima puntuación, después vera la vista de las instrucciones o HowTo, con los controles necesarios para jugar, y, por último, vera la vista del juego, donde tendrá el tablero, la puntuación, siguiente ficha…

3.1.4.2. Modelo de casos de uso

En el siguiente apartado, vamos a ver los principales casos de uso del sistema, así como cuál es la iteración y una breve descripción de los mismos.

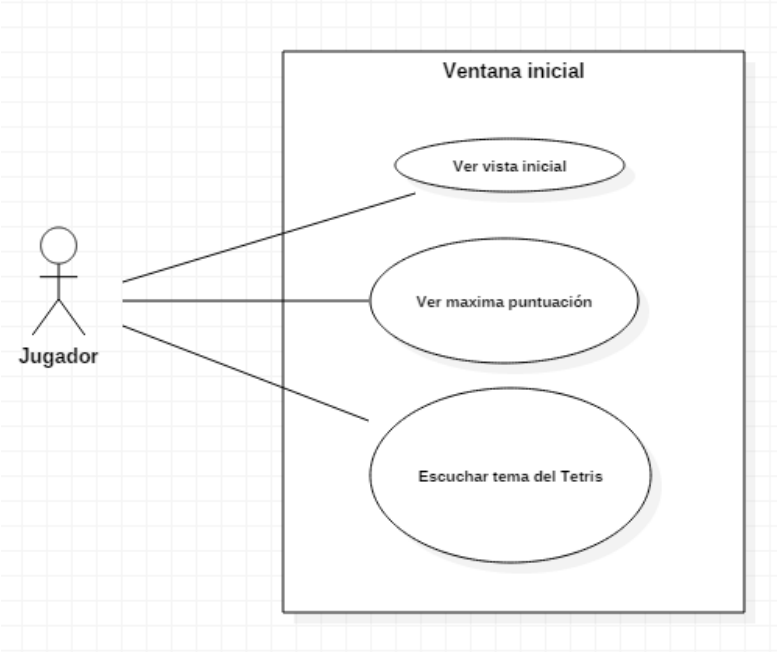


Ilustración 20 - Modelo de caso de uso, Ventana inicial

Como podemos ver en la Ilustración 20, el jugador en la primera vista solo tendrá disponibles las opciones:

- Ver vista inicial, donde veremos el nombre del juego, y un mensaje indicándonos que pulsemos el botón A para continuar
- Ver la máxima puntuación. Antes de empezar a jugar, podremos ver cuál es la máxima puntuación a batir, la primera vez que juguemos, esta será cero.
- Escuchar tema del Tetris. Con el fin de hacer más inmersiva la experiencia del juego, en la ventana de inicio, el usuario estará escuchando la música del Tetris hasta que presione el botón A.

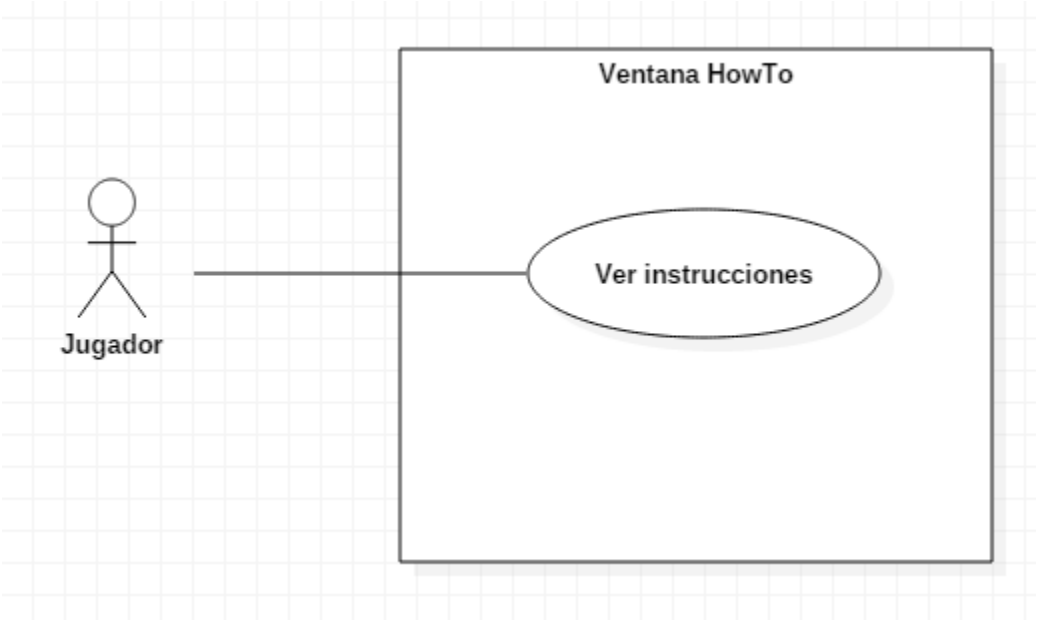


Ilustración 21 - Modelo de caso de uso, Ventana HowTo

En la Ilustración 21, podemos ver la siguiente vista que tenemos en el juego, esta nos sirve para informar al jugador de cuales son todos los posibles controles que va a tener durante el juego.

Para poder llegar a esta ventana, antes ha tenido que pulsar el botón A de la ventana inicial. Esta ventana que nos da información, solo dura dos segundos, tiempo suficiente para leer los controles, después pasara a la vista principal del juego.

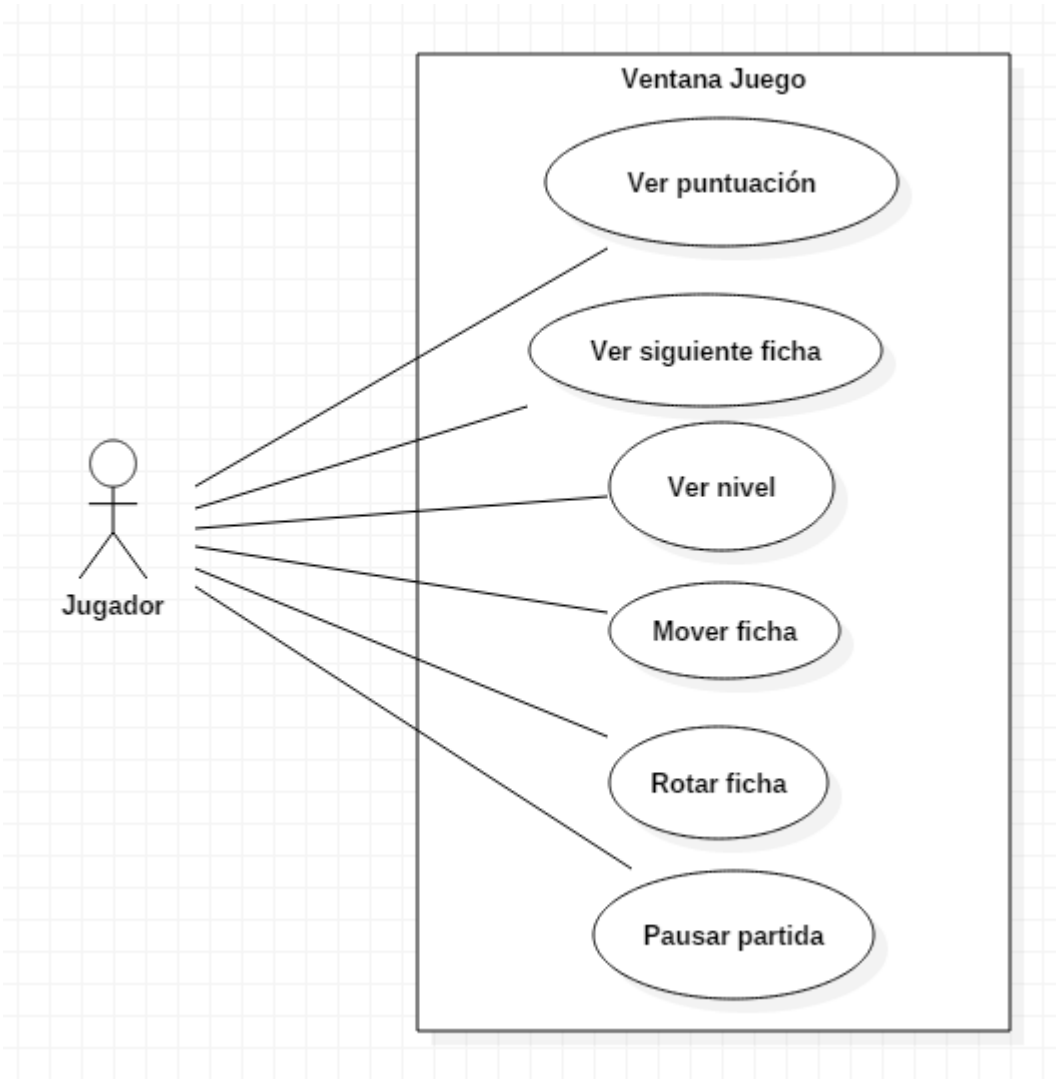


Ilustración 22 - Modelo de caso de uso, Ventana Juego

Como podemos apreciar en la Ilustración 22, este es el principal caso de uso de la aplicación, en el que el jugador va a pasar más tiempo, y donde va a estar toda la lógica del juego. En este caso de uso, el jugador podrá:

- Ver la puntuación de la partida, cada fila que vaya limpiando, será un punto más que subirá al marcador.
- Ver nivel, como es un juego que solo tiene una pantalla, a medida que la puntuación vaya creciendo, así lo hará el nivel. Se han creado 4 niveles, que van aumentando la velocidad con la que las fichas se desplazan verticalmente.

- Ver siguiente ficha, como en el clásico juego de recreativa, el usuario verá en todo momento cual es la siguiente ficha que aparecerá, de este modo podrá trazar estrategias para colocarlas como más le convenga.
- Mover ficha, el jugador podrá mover la ficha que este en juego hacia derecha, izquierda o abajo con el joystick, siempre que las reglas del juego lo permitan.
- Rotar ficha, todas las piezas del Tetris, a excepción del cuadrado, tienen la posibilidad de rotar, por lo que así el usuario podrá rotar las fichas, siempre que las reglas del Tetris lo permitan.
- Pausar partida, pulsando sobre el botón B, el usuario podrá pausar la partida tal y como se encuentre para reanudarlo en otro momento. Durante el periodo de pausa, aparecerá un mensaje en el borde inferior de la pantalla y un LED del BoosterPack se pondrá de color azul.

3.1.4.3. Diagrama de actividad

Trabajar con Energia, junto con TI-RTOS, hace que se simplifique mucho el desarrollo, creando una estructura lógica que debe seguir cualquier sistema embebido, y esta es una primera parte donde se declaren las librerías y variables, y dos funciones principales que van a llevar la ejecución del programa, setup(), y loop(), como podemos ver en la Ilustración 23

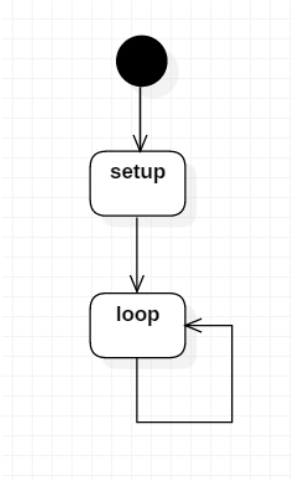


Ilustración 23 - Diagrama de secuencia de Energia

A continuación, veremos cómo son los diagramas de secuencia del Tetris, separados como hemos visto en setup y loop.

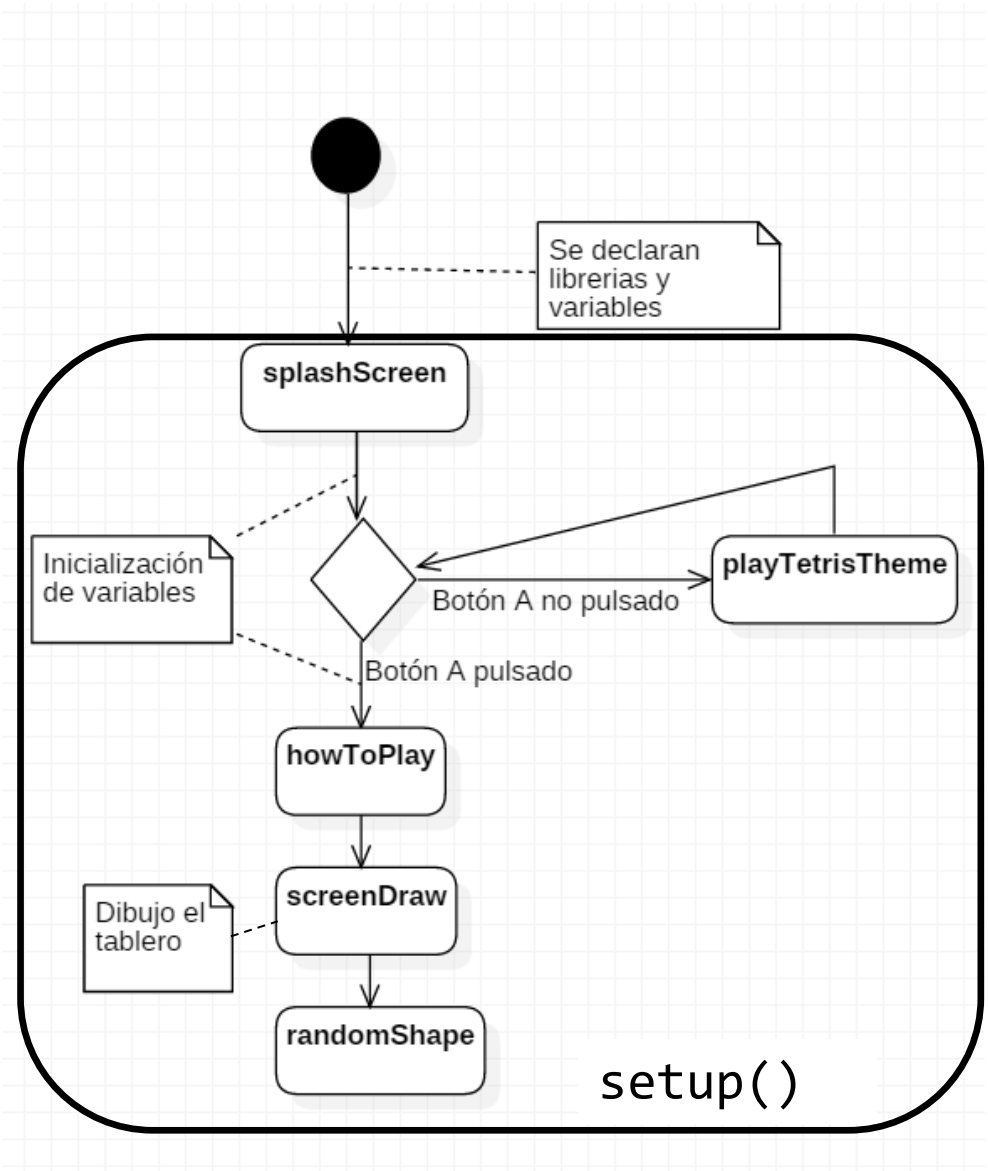


Ilustración 24 - Diagrama de secuencia Tetris, setup

En la Ilustración 24, podemos ver el diagrama de secuencia de la primera parte del programa, comenzamos con la inicialización de variables, y ya en la función *setup*, ejecutamos las siguientes funciones (se ha intentado conservar el nombre que tienen las funciones en el programa, y que su nomenclatura, sea coherente con su función):

- *splashScreen*: Sera la vista inicial de la aplicación, en la que veremos el título del juego, y la puntuación más alta. Se nos indicará pulsar el botón A para continuar, mientras no se haga, estará sonando el tema del Tetris.
- *howToPlay*: una vez pulsado, y durante dos segundos, aparecerá la ventana con la información de los controles.

- *screenDraw*: en esta función, vamos a dibujar todos los elementos de la vista principal del juego, el tablero, los cuadros de puntuación, nivel y la casilla de siguiente ficha.
- randomShape: Las piezas que van a ir apareciendo en el Tetris, son siempre aleatorias, por ello antes de comenzar hay que seleccionar las fichas aleatorias y que no se repitan.

En el siguiente diagrama en la Ilustración 25 veremos lo que ocurre en la función loop, que recordemos, es la que está continuamente ejecutándose en bucle, y la que va a llevar la ejecución del juego.

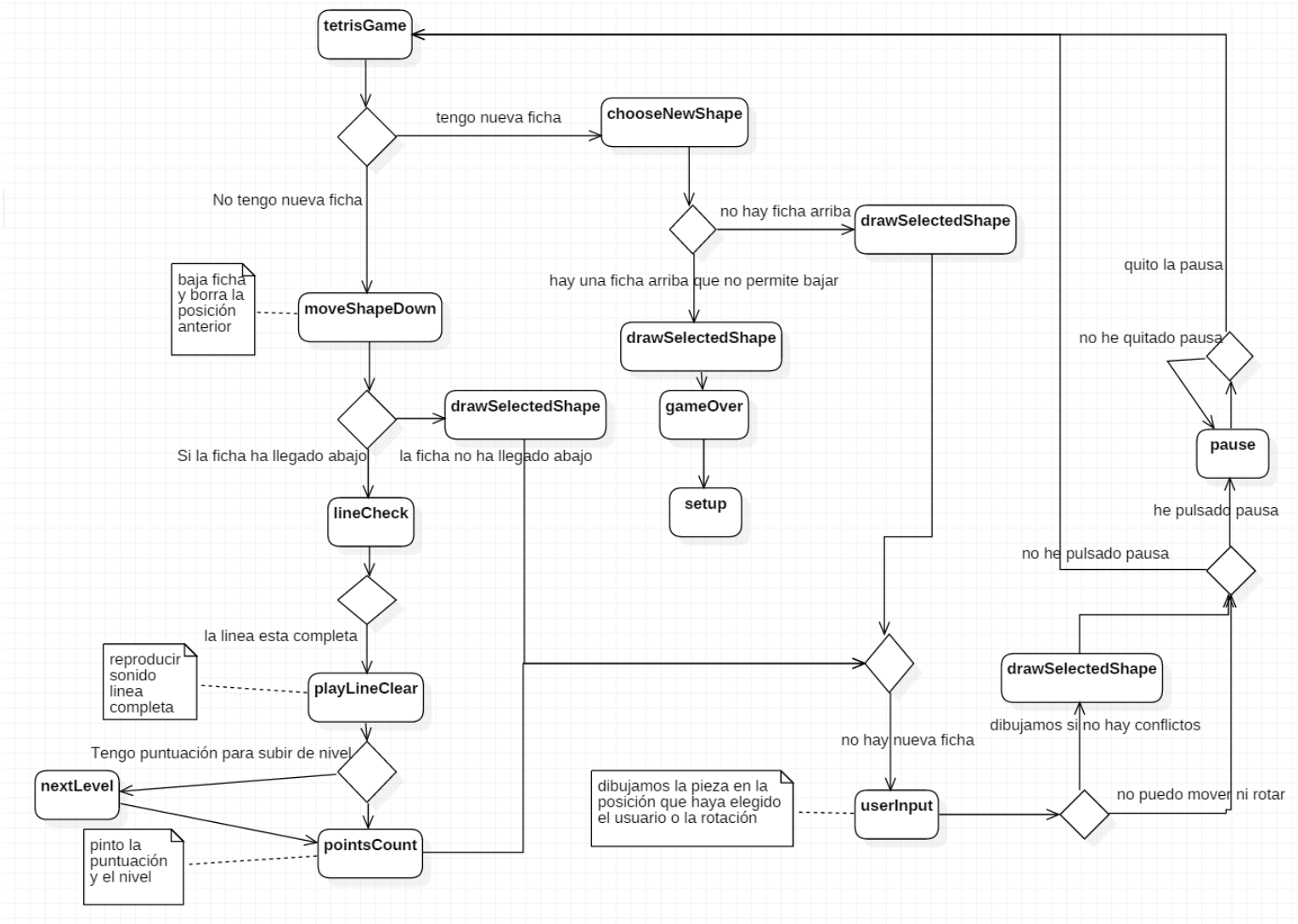


Ilustración 25 - Diagrama de secuencia Tetris, loop

La secuencia del juego, comienza con una decisión, de si tengo o no ficha seleccionada, en caso de tenerla, comprobamos si hay una ficha arriba que no permita bajar, si la tenemos, dibujamos la ficha, pero tenemos un *gameOver,* con lo que el juego se ha terminado y volvemos a la función *setup*, para volver a comenzar todo el proceso. En caso de no haber ficha arriba, dibujamos la pieza seleccionada.

Por el contrario, si nos encontramos con que no hay ficha nueva, lo que hacemos es bajar la ficha y borrando la posición que tenía antes, en caso que esta ficha haya llegado abajo del tablero, o no pueda bajar más porque tenga una ficha, comprobamos si se puede limpiar alguna fila, y de ser así, se aumenta la nueva puntuación, se comprueba si hay que subir de nivel, o si tengo una nueva máxima puntuación. Tras esto, continuaremos la ejecución del juego como si la ficha no hubiera llegado abajo, o como en el caso anterior no se hubiera dado un *gameOver*.

Lo siguiente es volver a comprobar si no hay ficha nueva, y recoger la entrada del usuario en la función *userInput*, aquí las entradas que tiene el usuario son las posiciones del joystick o la opción de rotar la pieza siempre que esta lo permita. Si puedo realizar el movimiento o rotación que el usuario desea, lo hago y dibujo la nueva posición de la ficha, en caso de no poder mover, no hacemos nada, y continuamos la ejecución.

Llegados a este punto, habría dado por concluido la lógica del Tetris, y se estaría repitiendo continuamente, pero dado que hemos implementado la función de pausa, antes de continuar con el bucle del Tetris, vamos a comprobar si hemos pulsado el botón B que activa la pausa, si no lo hemos pulsado, el juego no se pausa y continua al bucle, pero de haberlo pulsado, el juego se quedara en la función *pause*, comprobando continuamente si el usuario ha pulsado de nuevo el botón B para quitar la pausa, cuando lo quite, volveremos a empezar en la función del *tetrisGame* que repite la secuencia anterior.

3.1.4.4. Interface de usuario

Dado que el proyecto se trata de un videojuego, es muy importante el cuidado de la interfaz, esta estará dividida en tres vistas principales, las cuales veremos a continuación.

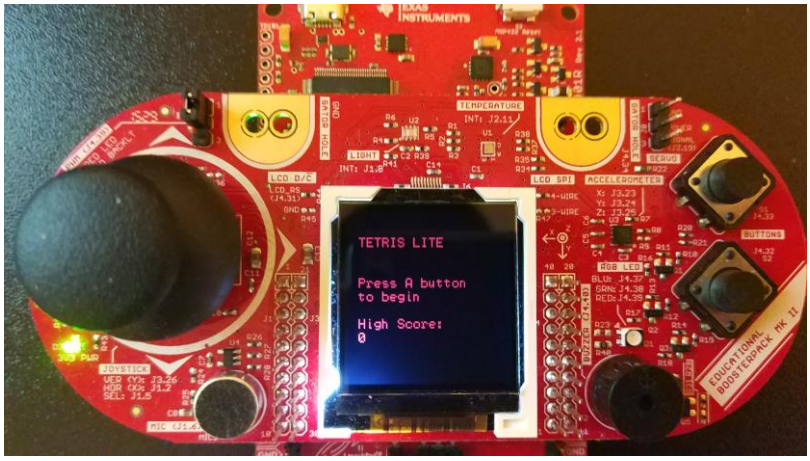


Ilustración 26 - Splash Screen

En la Ilustración 26 vemos la que es la primera vista que va a ver el usuario cuando vaya a utilizar el sistema, en esta vista, podemos ver el título del juego, la máxima puntuación, la cual en el comienzo es cero, pues no hay registros previos guardados, y un mensaje que indique de pulsar el botón A para continuar. Adicionalmente sonara el tema del Tetris mientras se este en esta vista.

Como los mensajes que van a aparecer son muy simples y para recordar el estilo de las máquinas recreativas, todos los mensajes de la interfaz están en inglés.

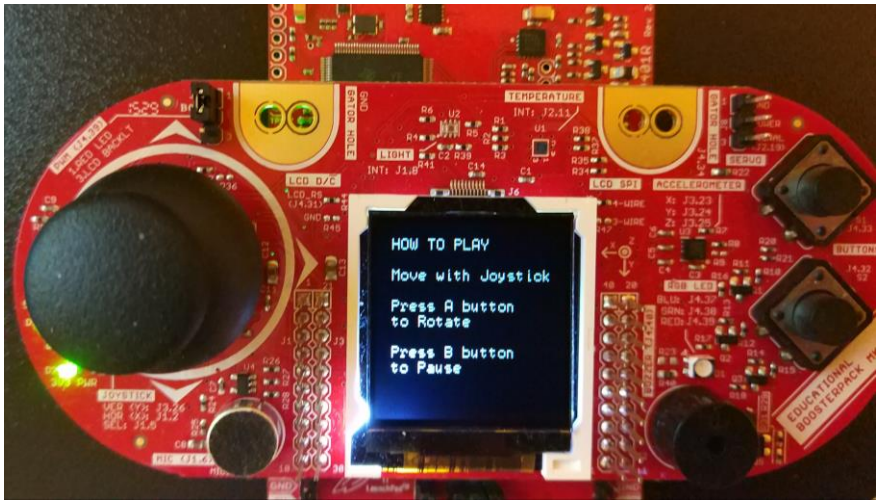


Ilustración 27 - HowTo

En la Ilustración 27, vemos la ventana de instrucciones, donde nos aparecen los controles que tendremos durante el juego. Esta vista, solo dura dos segundos, y el usuario no podrá interactuar con ella.

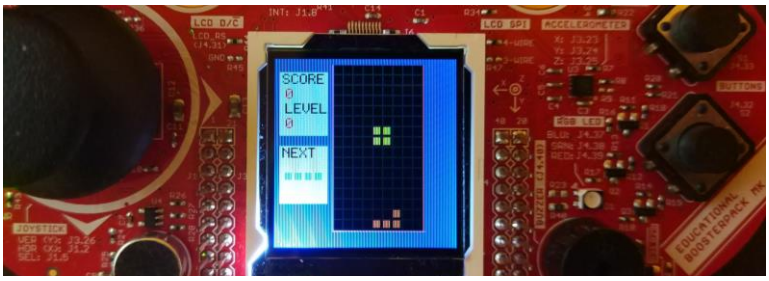


Ilustración 28 - Vista Tetris 1

Una vez pasadas las vistas anteriores, pasamos a la Ilustración 28, donde tenemos la vista principal de la aplicación, en ella podemos ver el tablero de juego con las piezas que hayamos colocado ya, y las que tengamos en movimiento, un cuadro donde ver la puntuación y el nivel, y otro cuadro donde poder ver la siguiente ficha.

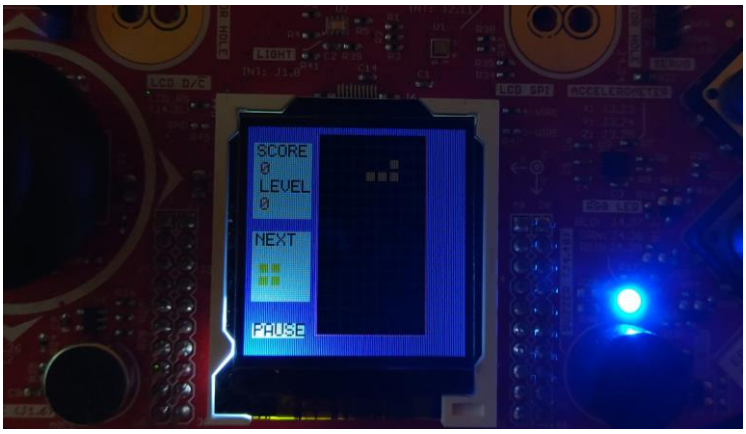


Ilustración 29 - Vista Tetris 2, Pause

Mientras se esté ejecutando la partida, el jugador puede pulsar el botón B, el cual nos pausa la partida, y como vemos en la Ilustración 29, al pulsarlo, se nos enciende un LED azul, nos aparece un mensaje “PAUSE”, y el juego detiene su ejecución.

Ilustración 30 - Vista Tetris 3, Game Over

En caso de haber perder la partida, y que no puedan entrar nuevas fichas, habremos perdido la partida, y como vemos en la Ilustración 30, tendremos la última ficha superpuesta a la que ya había, y un mensaje diciendo “GAME OVER”. Tras esto, volveríamos a la Ilustración 26.

4. Descripción detallada

Como antes hemos visto, las especificaciones técnicas del Launchpad MSP432P401R en la Ilustración 31, serían:

- Las características técnicas son:
  - MCU MSP432P401R de bajo consumo y alto rendimiento, que se compone por:
    - Procesador a 48MHz 32-bit ARM Cortex M4F con unidad de punto flotante y aceleración DSP.
    - Consumo: 80uA/MHz active and 660nA RTC standby operation
    - Digital: Advanced Encryption Standard (AES256) Accelerator, CRC, DMA, HW MPY32
    - Memoria: 256KB Flash, 64KB RAM
    - Timers: 4 x16-bit, and 2 x 32-bit
    - Communication: Up to 4 I2C, 8 SPI, 4 UART
  - 40 pin BoosterPack Connector, and support for 20 pin BoosterPacks
  - Onboard XDS-110ET emulator featuring EnergyTrace+ Technology
  - 2 buttons and 2 LEDs for User Interaction
  - Back-channel UART via USB to PC

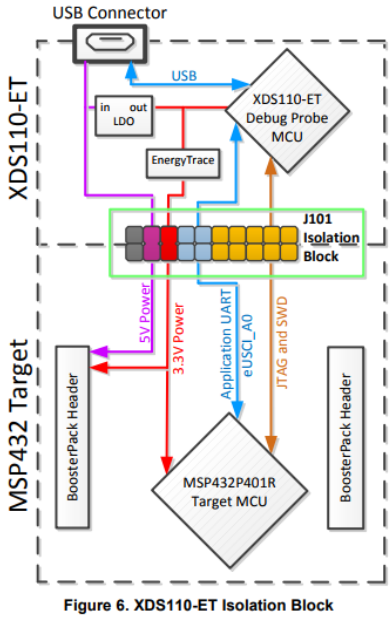


Figure 6. XDS110-ET Isolation Block

Ilustración 31 - Esquema Launchpad

Y las características técnicas del Educational BoosterPack MKII, las vemos en la Ilustración 32.

- TI OPT3001 Light Sensor
- TI TMP006 Temperature Sensor
- Servo Motor Connector
- 3-Axis Accelerometer
- User Push Buttons
- RGB Multi-color LED
- Buzzer
- 40-pin Stackable BoosterPack Connector
- Color TFT LCD Display
- Microphone
- 2-Axis Joystick with Pushbutton

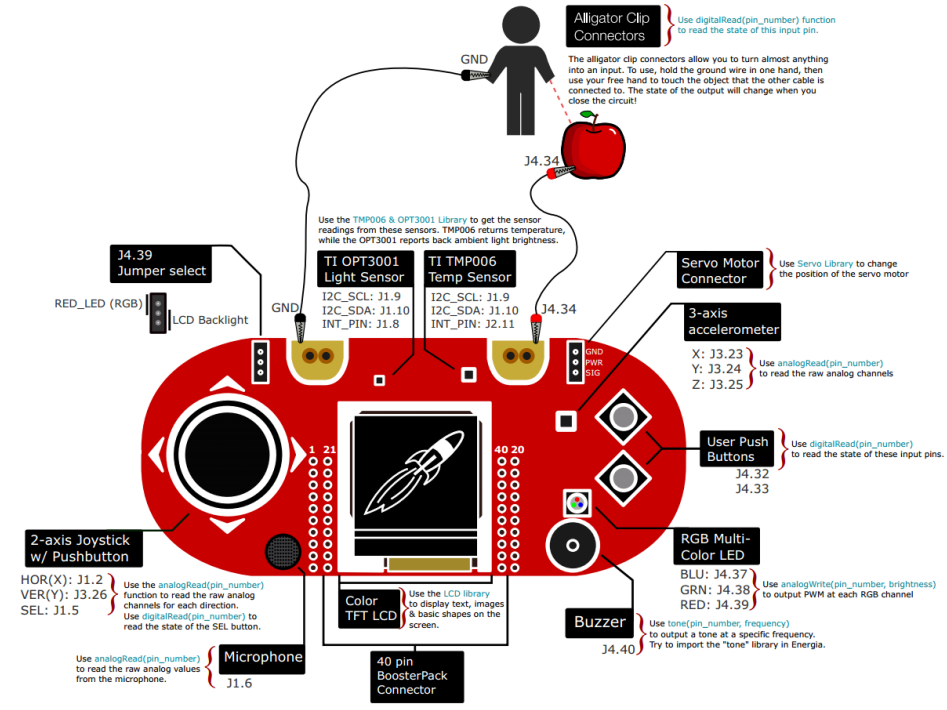


Ilustración 32 - Esquema BoosterPack

5. Viabilidad técnica

Antes de comenzar cualquier proyecto, hay que realizar un análisis técnico, para saber si con los recursos disponibles y los objetivos marcados, y el tiempo fijado, seremos capaces de llegar a completarlo.

En este análisis, comenzaremos por analizar el juego Tetris, y ver que requisitos tiene, y si podemos cumplirlos:

- El sistema debe ejecutarse sobre un sistema embebido, el cual debe procesar toda la lógica del juego
- El sistema requiere de una interfaz visual con la que el usuario pueda tener una visión de la partida.
- El sistema debe estar dotado de pulsadores, botones, y / o joystick para que el usuario pueda interactuar con el sistema

Adicionalmente es deseable:

- El sistema debe contar con una interfaz audio para añadir efectos sonoros al juego.

Y analizar si estos requisitos los cumplen el kit de Launchpad y BoosterPack que incorpora el kit de la UOC, o si, por el contrario, haría falta buscar una nueva solución.

- El Launchpad MSP432P401R, es un dispositivo de bajo consumo, bajo coste y bajo rendimiento, con un procesador ARM® 32-Bit Cortex®, con una frecuencia de reloj de 48 MHz, 256 KB de flash y 64 de RAM, pensado para pequeños proyectos y educación. El juego del Tetris no requiere de grandes cálculos ni necesita tener gran cantidad de memoria, por lo que sobre el papel, este Launchpad nos permitiría correr el juego.
- El Educational BoosterPack MKII, es un módulo de Texas Instruments, el cual dotara a nuestro sistema de un pequeño display LCD, un joystick, dos botones, un buzzer, así como otros sensores que en un principio no necesitamos, como micrófono, sensor de luz y acelerómetros.

  Con los elementos disponibles en el BoosterPack, vemos que tenemos todos los recursos necesarios para poder cubrir las necesidades de crear el juego.

Se ha visto como técnicamente parece posible la creación del juego, sin embargo, también es necesaria una parte de soporte y documentación para la implementación.
Para ello, es necesario:

- Entorno de programación compatible con el Launchpad seleccionado
- Entorno de programación amigable.
- Información y ejemplos sobre el manejo del IDE, al igual que programas de ejemplo para ejecutar en el Launchpad.

Tras analizar estos requisitos y los principales IDE, se escogió Energia, pues la filosofía de trabajo parecía más sencilla, y contaba con gran cantidad de ejemplos y foros con los que resolver posibles dudas.
Por ello se puede concluir, que el proyecto es técnicamente viable, no viendo ningún impedimento en los objetivos principales ni secundarios, a excepción de tener el audio reproduciéndose a la vez que se está ejecutando el juego.

6. Valoración económica

A continuación, presentamos el presupuesto de desarrollo en materiales y mano de obra. En el hemos intentado reflejar todos y cada uno de los componentes utilizados, tanto los que son directamente de adquisición en el mercado como los que han sido elaborados, adaptados o modificados para el proyecto.

6.1. Hardware

En la siguiente tabla, podemos ver todos los elementos hardware necesarios para el desarrollo del proyecto. Si bien estos componentes vinieron con el kit de la UOC, se ha obtenido el precio oficial de la tienda estadounidense y se ha hecho la conversión a euros de los mismos.

Descripción	Unidades	Precio unitario	Importe
Launchpad MSP432P401R	1	10.82 €	10.82 €
Educational BoosterPack MKII	1	24,99 €	24,99 €
Cable microUSB - USB	1	3,00 €	3,00 €
		TOTAL	**38,81 €**

Tabla 8 Valoración económica. Materiales

Para la conversión a euro, se ha aplicado el siguiente cambio a fecha 8/01/2017 1 USD = 0,833

6.2. Software

En este caso, vemos los elementos Software necesarios para desarrollar el proyecto, si bien, mi equipo ya contaba con licencia de Windows 10, así como del paquete ofimático de Microsoft, se ha incluido pensando en un desarrollo desde cero.

Descripción	Unidades	Precio unitario	Importe
Windows 10 64 bits	1	259,00 €	259,00 €
Microsoft Office 365	1	149,00 €	149,00 €
Energia	1	0,00 €	0,00 €
		TOTAL	**408,00 €**

Tabla 9 Valoración económica. Software

6.3. Desarrollo

Con el fin de obtener una estimación de los costes del desarrollo, este se ha dividido en varias fases, y se han tenido en cuenta las siguientes consideraciones:

- Toma de requisitos y documentación: Durante esta etapa, se establecen cuáles son los objetivos principales a desarrollar, y cuáles serían los secundarios o mejoras. También, se realiza una gran labor de documentación para familiarizarse con el entorno y buscar cuales serían las mejores herramientas de desarrollo.
- Desarrollo software: en esta etapa, se plasman los requisitos recogidos en la fase anterior implementándolos como se han diseñado en la fase de documentación.
- Pruebas e informe final: antes de dar por finalizado ningún desarrollo, hay que llevar a cabo una fase de pruebas, las cuales deben ser satisfactorias, si no, se volvería a la fase de desarrollo hasta que todo quede validado, y se realiza un informe con las conclusiones.

Aunque para un proyecto como este entrarían varios actores, para esta primera estimación de costes, se ha puesto un precio/hora fijo independientemente de la tarea, de 20€/hora, teniendo como resultado:

Descripción	Unidades	Precio unitario	Importe
Toma de requisitos y documentación	80	20,00 €	1600,00 €
Desarrollo Software	80	20,00 €	1600,00 €
Pruebas e informe	40	20,00 €	800,00 €
		TOTAL	**4000,00€**

Tabla 10 Valoración económica. Desarrollo

Con todo esto, obtenemos unos costes totales de:

Descripción	Unidades	Precio unitario	Importe
Hardware	1	38,81 €	38,81 €
Software	1	149,00 €	149,00 €
Desarrollo	1	4000,00€	4000,00€
		TOTAL	**4187,81€**

Tabla 11 Valoración económica. Costes totales

6.4. Costes de industrialización

Si bien, este proyecto no está pensado en ser industrializable bien por el bajo rendimiento del equipo, el cual haría difícil añadir nuevas funcionalidades, como por lo pequeña que es su pantalla y por el aspecto y forma que tiene, que haría muy difícil su uso fuera de prototipo o educación, imposibilitándolo como consola portátil.

Como en este tipo de proyectos entran en juego varios actores, se ha buscado el precio hora de un analista, salario medio anual 30.000 €, un desarrollador, salario medio anual 25.000€ y una persona de pruebas, salario medio anual 22.000 €, para dar la estimación de desarrollo más acorde a un proyecto industrializable

Descripción	Unidades	Precio unitario	Importe
Toma de requisitos y documentación	40	20,00€	800,00 €
Desarrollo Software	60	15,00 €	900,00 €
Pruebas e informe	20	12,00 €	240,00 €
		TOTAL	**1940,00€**

Tabla 12 Valoración económica. Costes de industrialización, desarrollo

E incluyendo una carcasa y una batería con el fin de poder hacerlo una consola portátil.

(se han reducido los costes del Launchpad, así como del Educational BoosterPack un 30% y se ha hecho una estimación de los costes de la carcasa, pues no hay ninguna para el Launchpad ni para el BoosterPack y de la batería en función de lo visto para elementos similares en diferentes distribuidores).

Descripción	Unidades	Precio unitario	Importe
Launchpad MSP432P401R	1	7,57 €	7,57 €
Educational BoosterPack MKII	1	17,49 €	17,49 €
Batería	1	0,83 €	0,83 €
Carcasa	1	1,5 €	1,5 €
		TOTAL	**27,39 €**

Tabla 13 Valoración económica. Costes de industrialización, hardware

Viendo la tabla anterior, vemos como en caso de querer industrializarlo, tendríamos un coste unitario de 27,39 €, el cual a mi parecer es un coste muy elevado para correr el juego del Tetris, por lo que habría que buscar una nueva placa de bajo coste que nos permita ejecutar el juego, así como un módulo de pantalla, joystick y botones para poder interactuar con el juego.

7. Conclusiones

7.1. Objetivos

Como todo proyecto, antes de comenzarlo, hay que tomar una serie de requisitos y objetivos, y de ellos analizar la importancia que tienen para así separar los objetivos principales, los cuales son necesarios para satisfacer las necesidades básicas y sin los cuales el sistema no sería capaz de funcionar, y unos objetivos secundarios, los cuales aporten un extra de valor y den más consistencia al proyecto.

A continuación, vamos a ver de todos los objetivos marcados inicialmente, tanto principales como secundarios cuales se han podido cumplir, y cuales por unos u otros motivos se han tenido que quedar fuera del margen del proyecto.

7.1.1. Objetivos conseguidos

Los objetivos principales que se han cumplido son:

- Implementar lógica del juego. Como es de esperar, al tratar de realizar el juego del Tetris, lo primero que debemos es tener implementada la lógica básica del Tetris, tener el control del tablero, la generación de las fichas, control de los bordes y de la parte inferior tanto para saber si una ficha ha llegado al fondo del tablero, como para saber si se tiene que quedar sobre otra.
- Dotar al sistema de interfaz gráfica basada en el LCD. Para poder jugar al Tetris, es necesario tener una interfaz gráfica en la que poder ver el tablero, las fichas, puntuación… por ello se va a emplear el Educational BoosterPack MKII el cual dotará al sistema de un LCD entre otros componentes, y habrá que hacer los desarrollos necesarios para poder mostrar todo el juego por pantalla.
- Dotar al sistema de interfaz de usuario vía joystick. En cualquier juego, es básica la iteración que hace el usuario con el sistema. En este caso, al contar con un joystick, implementar la funcionalidad de mover las fichas, derecha, izquierda y abajo con los movimientos del joystick.
- Implementar funcionalidad de girar la ficha con el botón A (S1). Al igual que es necesario poder mover las fichas en las direcciones del juego, también es necesario poder rotar las mismas, por ello se dota al sistema de la opción de rotar las fichas al pulsar el botón A.

Con estos objetivos cumplidos, el usuario final podría disfrutar del juego del Tetris sin ningún problema, sin embargo, con el fin de tener un sistema más completo, se han completado los siguientes objetivos secundarios:

- Implementar pantalla inicial del juego. Se crea una pantalla inicial del juego (splash) con la que el usuario al coger el dispositivo, sepa que juego se trata, así como poder mostrar algo de información. Se creará también una ventana de instrucciones (howto) entre la ventana inicial y el juego con el fin de explicar al usuario como debe interactuar con el sistema, aunque se han buscado los controles más intuitivos.
- Integrar la funcionalidad de pausar partida. Casi todos los juegos tienen la opción de pausar una partida, siempre puede haber algún imprevisto, y la función de pausar y continuar cuando quieras, es muy útil.
- Implementar histórico de mayor puntuación. El objetivo de juegos que no tienen una historia, sino que son repeticiones de lo mismo, el principal aliciente que tienen es tratar de batir tus propias puntuaciones, por ello se implementa la funcionalidad de ver la mayor puntuación en la pantalla de bienvenida para que de este modo antes de empezar una partida sepas cual es la puntuación máxima a batir.
- Integrar librería de sonido. Para poder estimular más al jugador, se implementa el manejo de señales sonoras, de modo que sonara la melodía del Tetris en la pantalla de inicio, y durante el juego, cada vez que se complete una fila, sonara un pitido a modo de confirmación acústica de lo que ha ocurrido.
- Implementar diferentes niveles de dificultad en el juego. Al ser un juego repetitivo, para que suponga un reto al jugador, se han creado cuatro niveles de dificultad los cuales, en función de la puntuación obtenida, irán aumentando la velocidad con la que las fichas se desplazan verticalmente. En este prototipo se ha desarrollado de forma que llegar al nivel máximo de dificultad se dé muy pronto con el objetivo de poder llegar a mostrarlo en la demostración.

7.1.2. Objetivos no conseguidos

Como hemos podido ver, todos los objetivos principales se han cumplido, y los objetivos secundarios casi en su totalidad, sin embargo, tenemos los siguientes que no se han podido cumplir, principalmente por falta de tiempo

- Integrar la funcionalidad de guardar partida. A principio de proyecto, se pensó que implementar la funcionalidad de guardar una partida para después continuarla y tener varias partidas empezadas podría ser una buena funcionalidad.

  Este objetivo no se ha podido desarrollar por la falta de tiempo, al comienzo del proyecto la fase de documentación se prolongó más de lo esperado, llegando al final del desarrollo sin tiempo para poder desarrollar esta compleja funcionalidad. Si que se ha desarrollado la funcionalidad de pausar partida, no es la misma funcionalidad, pero es la más similar.

- Integrar librería de sonido(II). En el sistema, se ha implementado la función de escuchar el tema del Tetris en la vista inicial, sin embargo, intente que esta melodía estuviese sonando durante todo el desarrollo del juego, como en las máquinas recreativas. Se observo que debido a la falta de potencia del Launchpad, cada vez que refrescaba elementos en la pantalla como, cuando limpiaba una fila o cuando actualizaba la siguiente ficha, el audio se veía afectado, produciéndose cortes notables, por ello se decidió quitar el audio durante el juego y solo incluirlo en la ventana de inicio.

7.2. Conclusiones

Como hemos podido ver en el apartado anterior, hemos sido capaces de completar todos los objetivos principales, los cuales dotan al sistema de la función básica del juego del Tetris, así como casi todos los objetivos secundarios, que añaden mejoras tanto a nivel gráfico, como acústico y de manejo, pudiendo pausar el juego, y los objetivos secundarios que no se han podido completar, ninguno de ellos afecte al correcto funcionamiento del juego, sino que le añadirían nuevas funcionalidades.

Con ello, tenemos como resultado una versión lista para ejecutarse sobre el Launchpad MSP432P401R y el Educational BoosterPack MKII, lista para disfrutar del Tetris Lite.

7.3. Autoevaluación

Cuando abordas un nuevo proyecto, este puede resultar muy simple, o tener complicaciones según vas avanzando. En este caso, no ha resultado un proyecto sencillo, principalmente por el reto que suponía enfrentarse a un sistema nuevo que no conocía anteriormente, un nuevo entorno de desarrollo, y un cambio en la forma de programar.

No obstante, la realización de este proyecto me ha ayudado a darme cuenta de cómo con esfuerzo y sacrificio se pueden llegar a, uniendo elementos electrónicos, junto con un software adecuado, nos pueden permitir crear juegos con los que en la infancia pasabas el rato, y motivo por el cual, creo que muchos estudiantes como fue mi caso, optaron por el estudio de una titulación en informática.

El haber terminado este proyecto, me ha dado la capacidad de poner en valor todo lo aprendido, y darme cuenta una vez más de lo importante que son todas las asignaturas estudiadas, sin las cuales no habría conseguido terminar este proyecto, ni saber hacer un análisis de requisitos, un diseño de la solución, la implementación, las pruebas ni la documentación.

Y aunque durante la realización del proyecto, ha habido momentos de frustración e incluso cierto nivel de estrés por las entregas y los plazos, la satisfacción obtenida una vez terminado todo, te hace quedarte con el resultado final, y hacer que esos momentos menos buenos se te olviden por completo.

De hecho, una vez llegado a este punto, y contando con los elementos hardware del proyecto, así como los conocimientos obtenidos, muy posiblemente que busque implementar algún que otro juego o darle un segundo uso al BoosterPack.

7.3.1. Reflexión crítica

Llegados a este punto, tengo un alto grado de satisfacción dado que he sido capaz de cumplir casi todos los objetivos, algo que, a mediados del proyecto, cuando más confuso estuve al haber comenzado toda la documentación y tratar de entender todo, parecía una gran montaña la cual no podría ni saber por dónde escalar.

Sin embargo, esa primera mitad del proyecto, sirvió para obtener una buena base sobre lo que había que hacer, y como hacerlo, y tras ello, pude avanzar a un buen ritmo, completando todos

los objetivos principales, e incluso pensando en cómo mejorar la interfaz de usuario para que fuese algo más atractiva.

Dado que el comienzo del proyecto no fue todo lo rápido que debió ser, tanto por desconocimientos del entorno y plataforma, como por no haber dedicado más horas, no he sido capaz de completar todos los objetivos secundarios marcados al inicio. La funcionalidad de guardar partida, fue un requisito que deje para el final, y como consecuencia, no hubo tiempo material para poder llevarlo a cabo.

En contra partida, si que se añadieron diferentes niveles al juego, un objetivo que no estaba marcado en la propuesta inicial, pero que hace que el juego se convierta en un reto, y sea más entretenido.

7.3.2. Análisis crítico

Debido tanto a la situación laboral, como al ya mencionado desconocimiento del entorno y plataforma, no he sido capaz de seguir el ritmo marcado en la planificación inicial, teniendo que hacer varios ajustes hasta la última entrega, ni he logrado completar las diferentes PECs con el nivel que debiera.

La metodología empleada, creo que ha sido la correcta, separando el proyecto en varias fases y buscando que los desarrollos estuvieran separados por funcionalidades, creando una primera estructura bien depurada sobre la que ir añadiendo el resto de funcionalidades u objetivos básicos, y después los secundarios.

7.4. Líneas de trabajo futuro

Una vez terminado el proyecto, y habiendo recapitulado sobre los objetivos completados y no, es hora de mirar hacia el futuro, y pensar que se podría mejorar, o que funcionalidades añadir.

- Guardar partida. Como hemos visto, el guardado de la partida, ha sido un objetivo secundario que se ha quedado sin desarrollar por falta de tiempo.
- Investigar posibles mejoras para añadir audio mientras se juega. En el tiempo que se dispuso, se investigó, analizó y probo la posibilidad de incluir el tema del Tetris mientras se está jugando, sin embargo, el resultado obtenido no fue el esperado, por lo que, en un futuro, se podría investigar más en esta línea por buscar alguna solución.
- Selección del nivel al comienzo. Al igual que se implementó que con la puntuación se aumentase el nivel, posibilitar el seleccionar el nivel en el inicio de la partida.

- Mejorar la interfaz de usuario. Este es un punto en el que se podría estar trabajando constantemente, buscar una experiencia de usuario lo más satisfactoria posible que impacte e incite a jugar.
- Dar estadísticas del número de piezas que aparecen de cada tipo, o del tiempo promedio que se tarda en completar una fila.
- Utilizar el módulo Wifi contenido en el kit de la UOC, para subir las mejores puntuaciones a algún servidor, o integración con red social.
- Hacer una nueva prospección sobre otras placas sobre las que poder desarrollar el juego con más características para así poder incluir más juegos y funcionalidades, o placas más baratas para optimizar costes.

8. Glosario

Arduino: compañía tecnológica que diseña placas de desarrollo basadas en microcontroladores

ARM: arquitectura de procesador y microcontroladores.

Buzzer: transductor electroacústico que produce un sonido.

Code Composer Studio:

Cortex: Arquitectura de procesadores de ARM

CPU: unidad de procesamiento de datos.

EA: Electronics Arts, compañía de diseño de videojuegos

Energia: herramienta de desarrollo para sistemas embebidos

GPIO: General Purpose Input/Output, Entrada/Salida de Propósito General

Joystick: dispositivo de control de dos o tres ejes.

LED: Diodo emisor de luz

LCD: representación visual por cristal liquido

MCU: microcontrolador

MS-DOS: Sistema operativo de disco de Microsoft

PCB: placa de circuito impreso

RGB Multi-Color LED: diodo emisor de luz con mezcla de colores, rojo, verde y azul.

RTOS: Sistema Operativo de Tiempo Real

SBC: Single Board Computer, ordenador de placa única o reducida.

9. Bibliografía

Todas las referencias citadas a continuación, han sido consultadas, bien durante el desarrollo del proyecto, como en el desarrollo de la presente memoria, por lo que las fechas están comprendidas entre 17 de Noviembre de 2017, y el 14 de Enero de 2018

https://e2e.ti.com/group/helpcentral/b/weblog

http://energia.nu/guide/

http://energia.nu/reference/

https://www.hackster.io/

https://www.freertos.org/

http://www.43oh.com/

http://blogs.solidworks.com/tech/2014/03/circuitworks-and-flow-simulation-working-together.html

https://www.addicore.com/Dual-Axis-XY-Joystick-Module-with-Push-Button-p/139.htm

http://arduinomodules.info/ky-006-passive-buzzer-module/

https://www.jagelectronicsshop.com/product-page/button-module

https://www.itead.cc/itdb02-3-2s-v2.html

http://www.averagemanvsraspberrypi.com/2014/08/how-to-design-printed-circuit-board.html

https://www.microsoft.com/es-es/store/d/windows-10-pro/df77x4d43rkt/48F2

https://products.office.com/es-es/compare-all-microsoft-office-products?tab=1

https://spanish.alibaba.com/product-detail/4400mah-5-volt-battery-18650-rechargeable-battery-for-heating-clothes-shoes-gloves-60572573604.html?spm=a2700.8698675.29.2.6274e91befstQp&s=p

http://www.ti.com/tool/MSP-EXP432P401R

http://www.ti.com/lit/ug/slau597e/slau597e.pdf

10. Anexos

En los anexos, veremos un manual de instalación, compilación y ejecución de un programa con Energia, unos diagramas del Launchpad, y una licencia contenida en el código necesario para reproducir el tema del Tetris.

10.1. Manual Energia

Energīa

Home | Download | Guide | Reference | Blog | Store | Getting Help | IRC | Energia Projects | Even

How to set up Energia on Windows

Need help? The Energia discussion forum is hosted on 43oh.com.

Mac OS X Users & Linux Users

Installing the LaunchPad drivers

To use Energia you will need to have the LaunchPad drivers installed. The drivers allow your PC to "see" the LaunchPad on a serial COM port when it is connected.

+ MSP-EXP432P401R LaunchPad

Energia (MT) uses a different method for building Sketches for the MSP-EXP432P401R. Because of a limitation in this method you *must* install Energia in a location *without spaces*.

If you have already used the MSP-EXP432P401R LaunchPad with TI's Code Composer Studio then you probably already have the driver installed. If not, then follow the steps below to install the driver.

1. Do not connect your MSP-EXP432P401R LaunchPad to your PC. If you already plugged it into your PC then unplug it before proceeding to step 2.
2. Download the LaunchPad drivers for Windows: MSP-EXP432P401R XDS110 Driver Package
3. Unzip and double click DPinst.exe for Windows 32bit or DPinst64.exe for Windows 64 bit.
4. Follow the installer instructions.
5. Connect your MSP-EXP432P401R LaunchPad to your PC. The MSP-EXP432P401R will be automatically recognized.
6. All done, you are now ready to program the MSP-EXP432P401R with Energia.

Ilustración 33 - Energia 1

Como indica, lo primero que tenemos que hacer, es descargarnos el driver para el Launchpad MSP432P401R e instalarlo como se muestra en las siguientes capturas.

Nombre	Fecha de modifica...	Tipo	Tamaño
amd64	06/01/2016 11:50	Carpeta de archivos	
i386	06/01/2016 11:50	Carpeta de archivos	
boot_usb.inf	05/09/2015 2:10	Información sobre...	5 KB
boot_usb_amd64.cat	05/09/2015 2:10	Catálogo de segur...	13 KB
boot_usb_x86.cat	05/09/2015 2:10	Catálogo de segur...	11 KB
DPinst.exe	08/12/2011 15:43	Aplicación	540 KB
DPInst64.exe	02/12/2009 8:03	Aplicación	1.024 KB
stellaris_icdi_com.inf	05/09/2015 2:10	Información sobre...	2 KB
stellaris_icdi_com_amd64.cat	05/09/2015 2:10	Catálogo de segur...	9 KB
stellaris_icdi_com_x86.cat	05/09/2015 2:10	Catálogo de segur...	9 KB
stellaris_icdi_debug.inf	05/09/2015 2:10	Información sobre...	5 KB
stellaris_icdi_debug_amd64.cat	05/09/2015 2:10	Catálogo de segur...	15 KB
stellaris_icdi_debug_x86.cat	05/09/2015 2:10	Catálogo de segur...	12 KB
xds110_debug.cat	05/09/2015 2:09	Catálogo de segur...	11 KB
xds110_debug.inf	05/09/2015 2:09	Información sobre...	5 KB
xds110_ports.cat	05/09/2015 2:09	Catálogo de segur...	9 KB
xds110_ports.inf	05/09/2015 2:09	Información sobre...	3 KB

Ilustración 34 - Energia 2

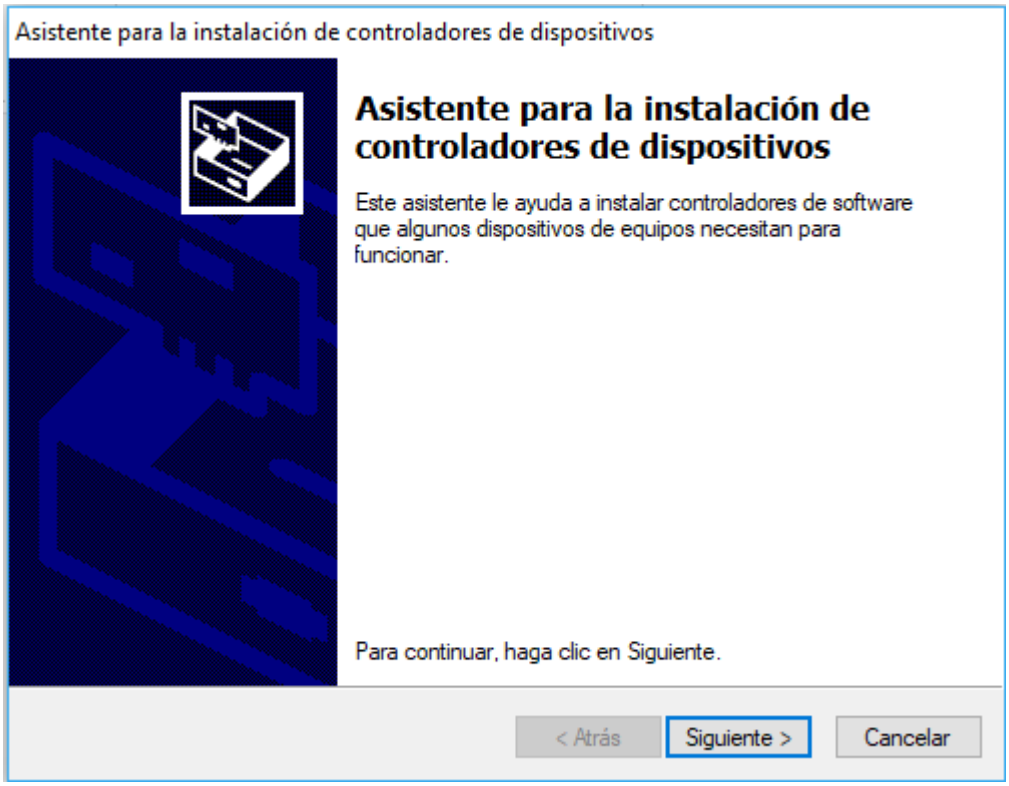


Ilustración 35 - Energia 3

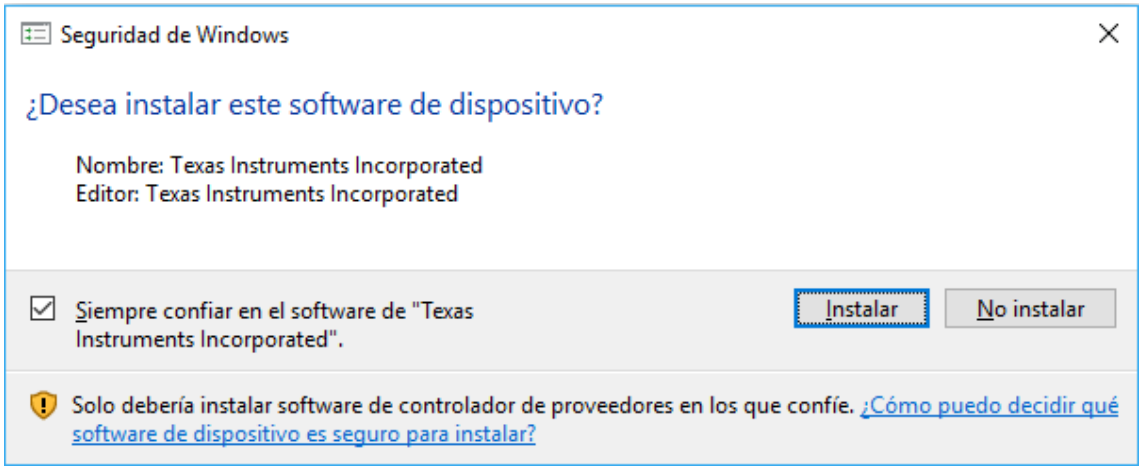


Ilustración 36 - Energia 4

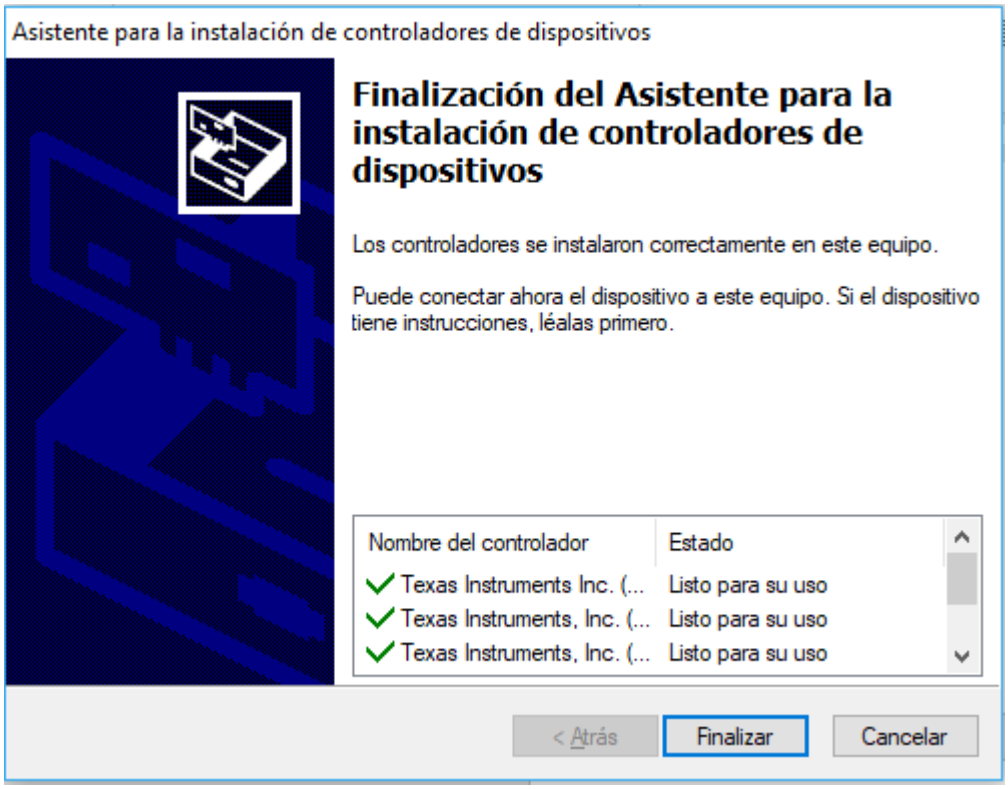
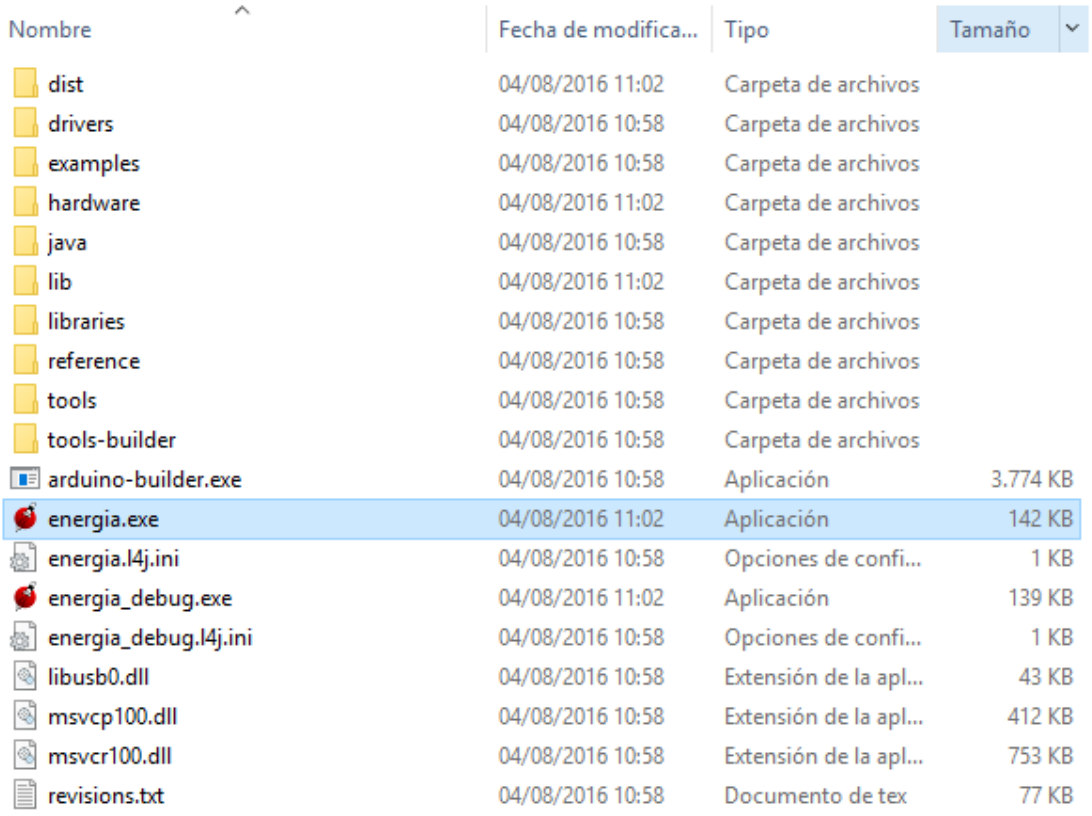


Ilustración 37 - Energia 5

Una vez instalado, nos descargamos Energia, y lo ejecutamos:



Nombre	Fecha de modifica...	Tipo	Tamaño
dist	04/08/2016 11:02	Carpeta de archivos	
drivers	04/08/2016 10:58	Carpeta de archivos	
examples	04/08/2016 10:58	Carpeta de archivos	
hardware	04/08/2016 11:02	Carpeta de archivos	
java	04/08/2016 10:58	Carpeta de archivos	
lib	04/08/2016 11:02	Carpeta de archivos	
libraries	04/08/2016 10:58	Carpeta de archivos	
reference	04/08/2016 10:58	Carpeta de archivos	
tools	04/08/2016 10:58	Carpeta de archivos	
tools-builder	04/08/2016 10:58	Carpeta de archivos	
arduino-builder.exe	04/08/2016 10:58	Aplicación	3.774 KB
energia.exe	04/08/2016 11:02	Aplicación	142 KB
energia.l4j.ini	04/08/2016 10:58	Opciones de confi...	1 KB
energia_debug.exe	04/08/2016 11:02	Aplicación	139 KB
energia_debug.l4j.ini	04/08/2016 10:58	Opciones de confi...	1 KB
libusb0.dll	04/08/2016 10:58	Extensión de la apl...	43 KB
msvcp100.dll	04/08/2016 10:58	Extensión de la apl...	412 KB
msvcr100.dll	04/08/2016 10:58	Extensión de la apl...	753 KB
revisions.txt	04/08/2016 10:58	Documento de tex	77 KB

Ilustración 38 - Energia 6

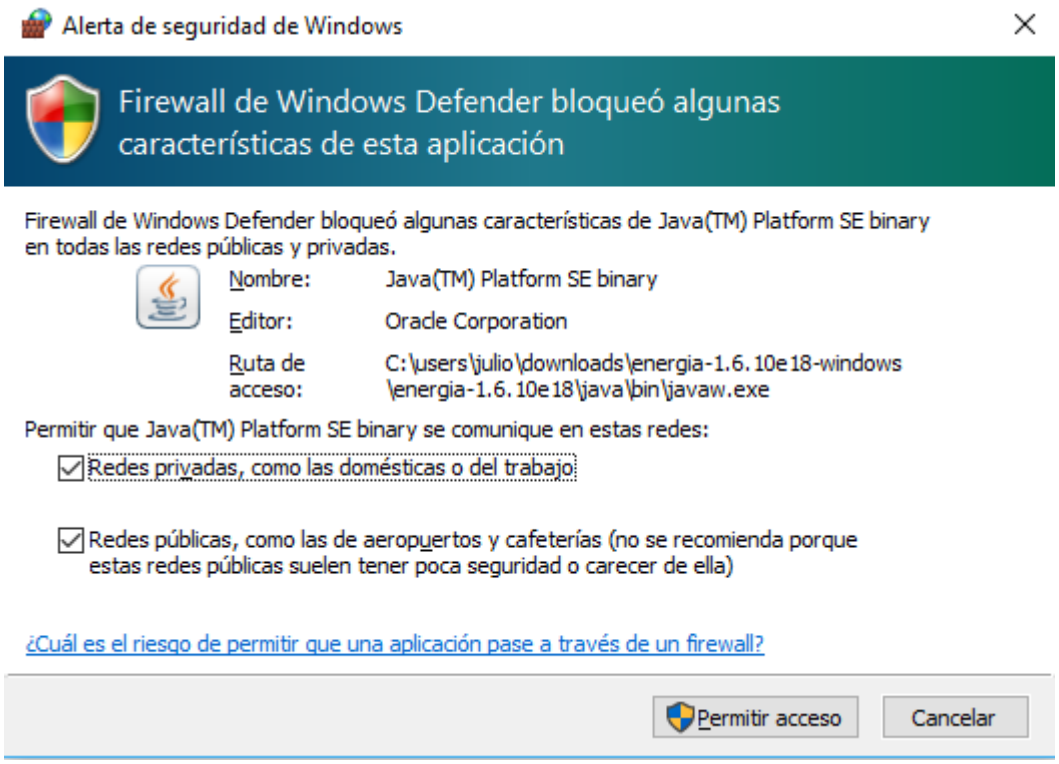


Ilustración 39 - Energia 7

En el gestor de tarjetas buscamos la msp432, y la instalamos

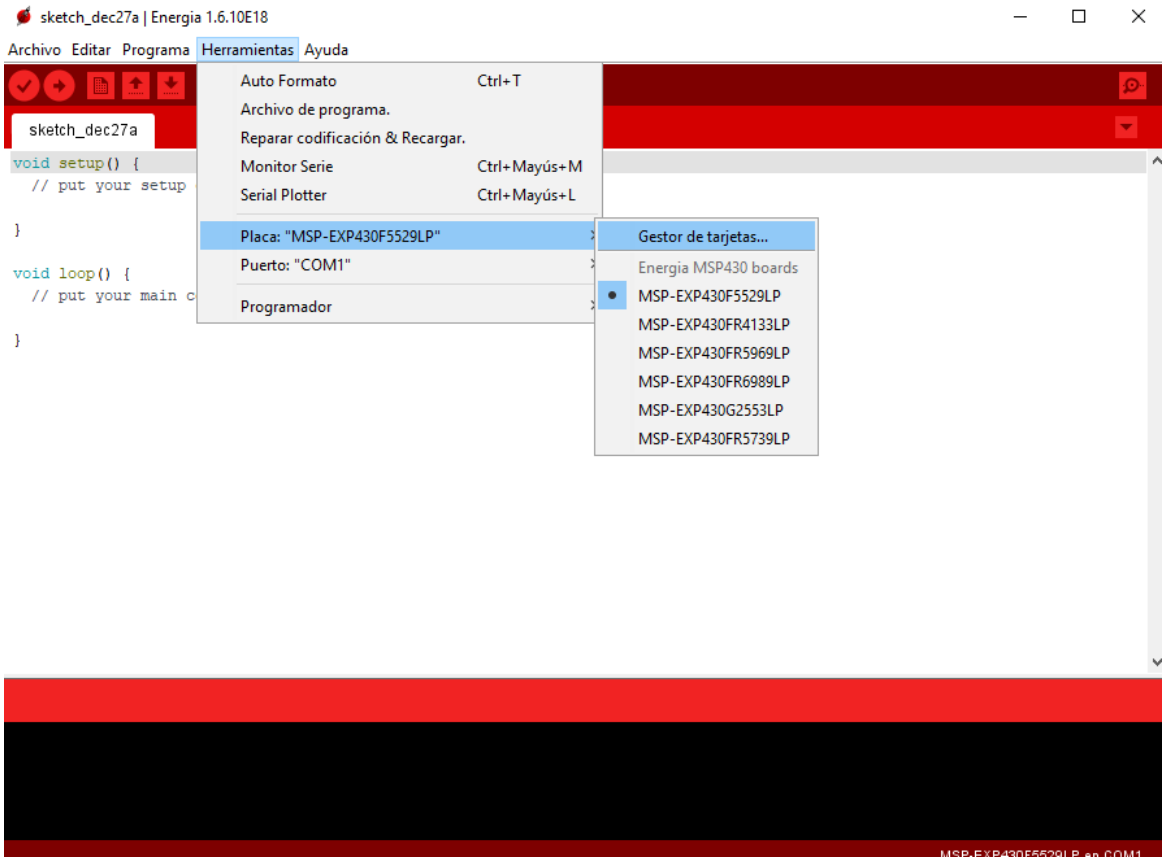


Ilustración 40 - Energia 8

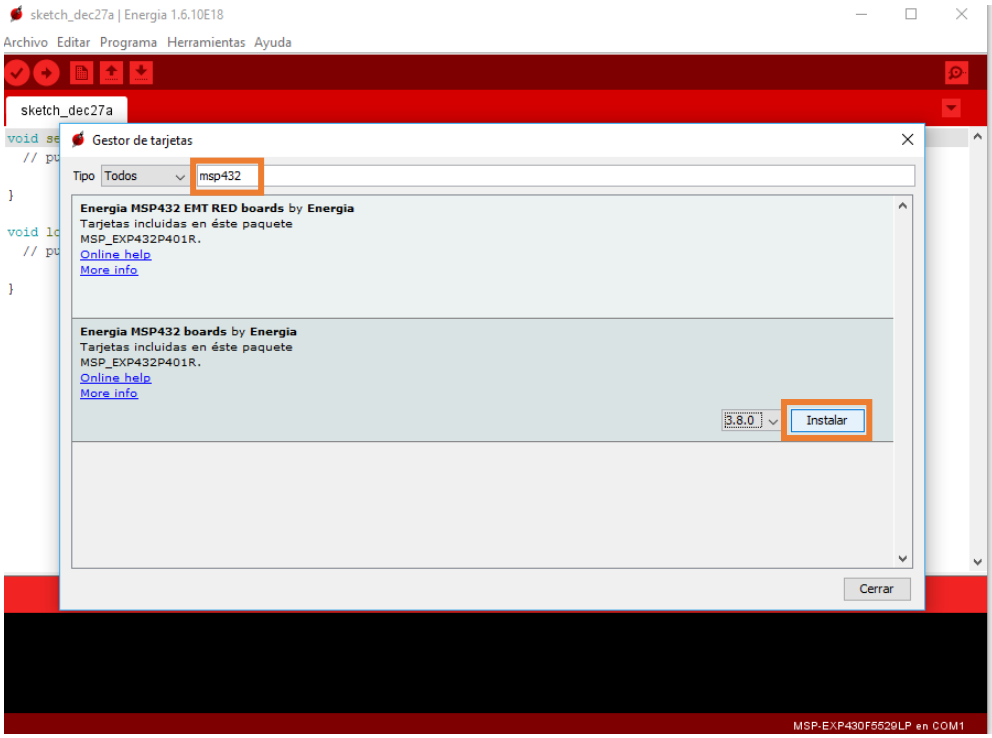


Ilustración 41 - Energia 9

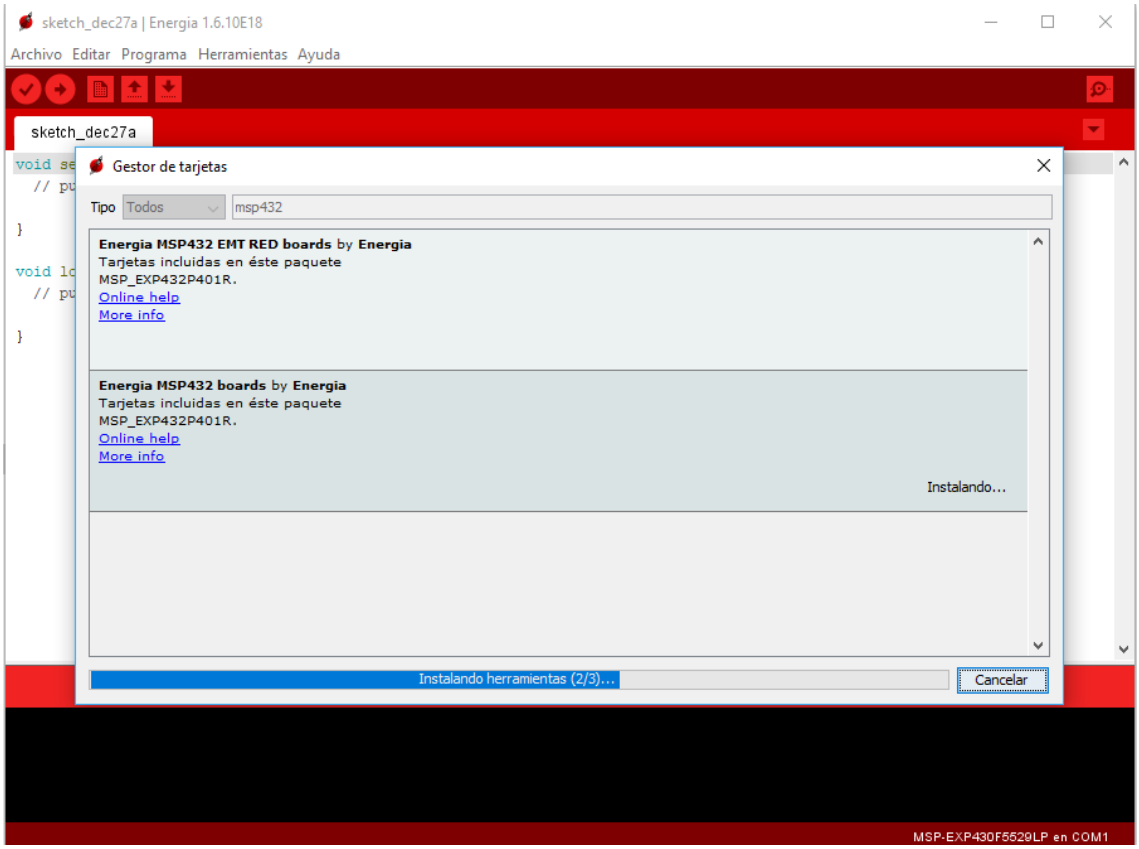


Ilustración 42 - Energia 10

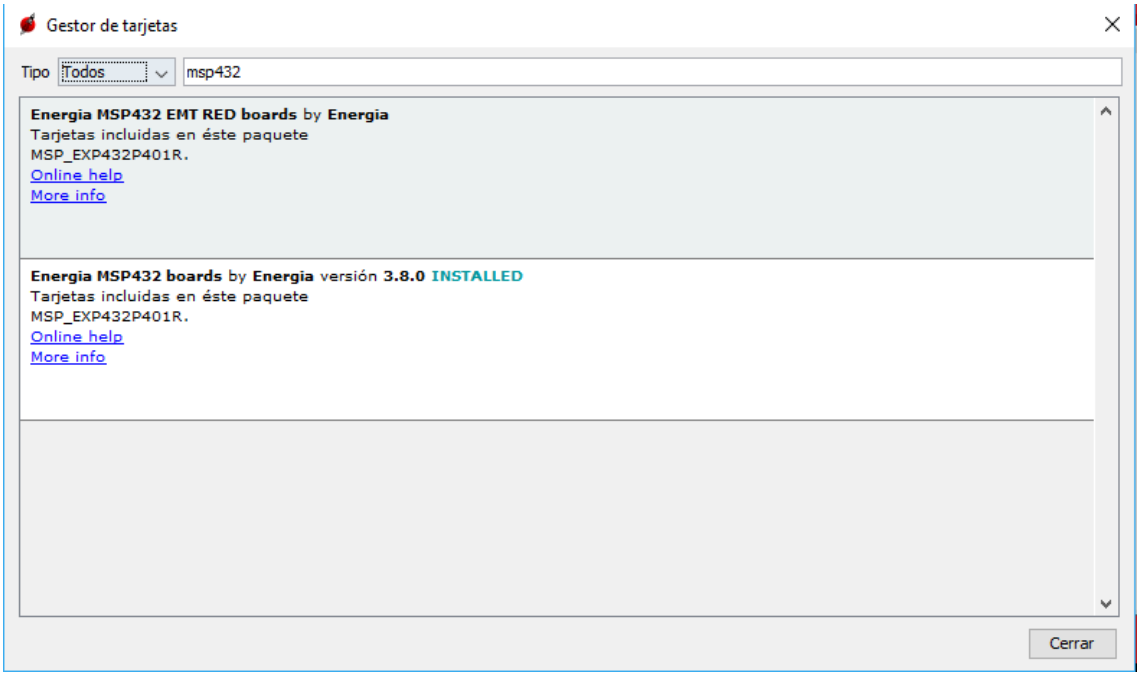


Ilustración 43 - Energia 11

Una vez instalada, la seleccionamos, y elegimos el puerto COM en el que este configurado el Launchpad

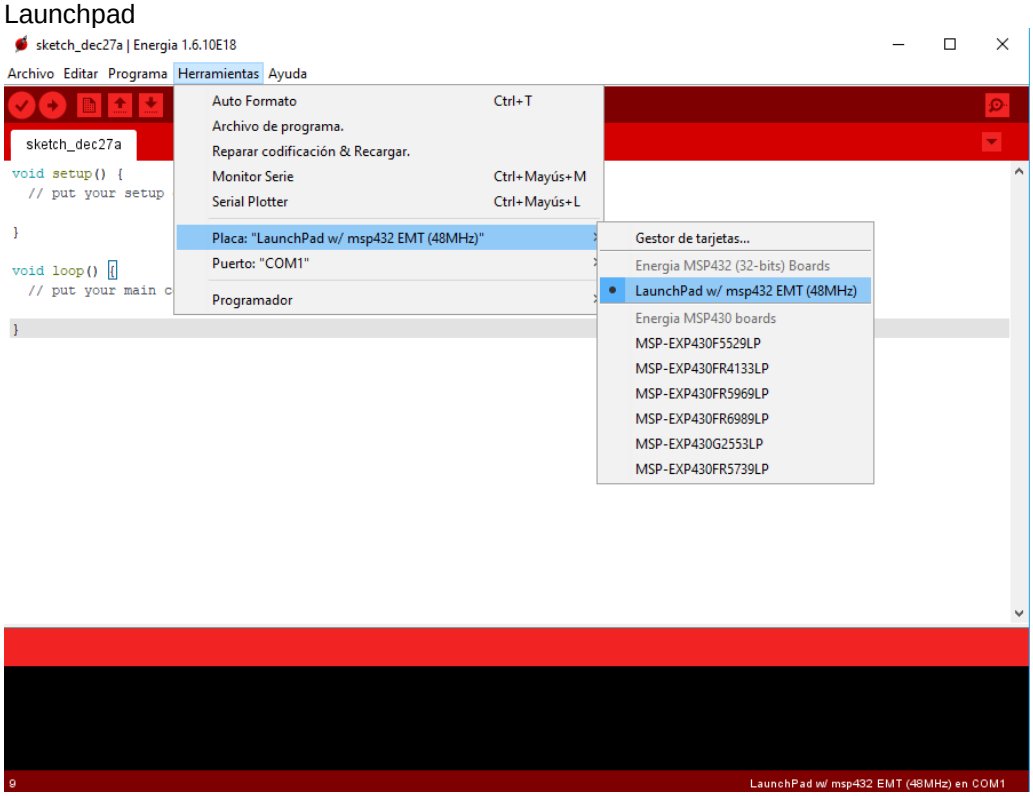


Ilustración 44 - Energia 12

Tras esto, solo queda abrir el proyecto, para ello desde el menú de Energia:
Archivo -> Abrir
Y buscando el archivo en la ubicación donde se haya descargado, sobre el fichero tetrisLite.ino pulsamos sobre abrir

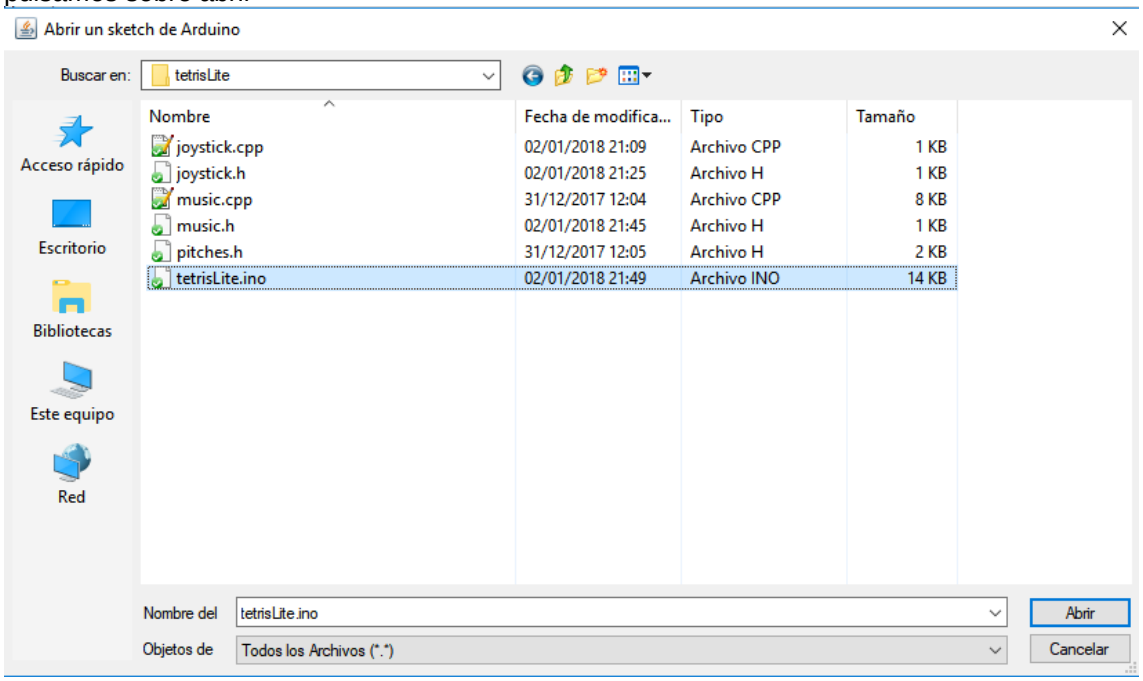


Ilustración 45 - Energia 13

Ya solo quedara lanzarlo en el Launchpad pulsando sobre

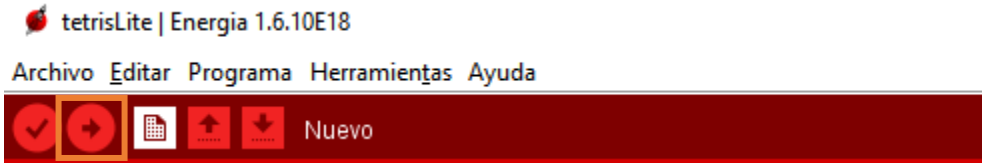


Ilustración 46 - Energia 14

10.2. Esquemas técnicos del Launchpad

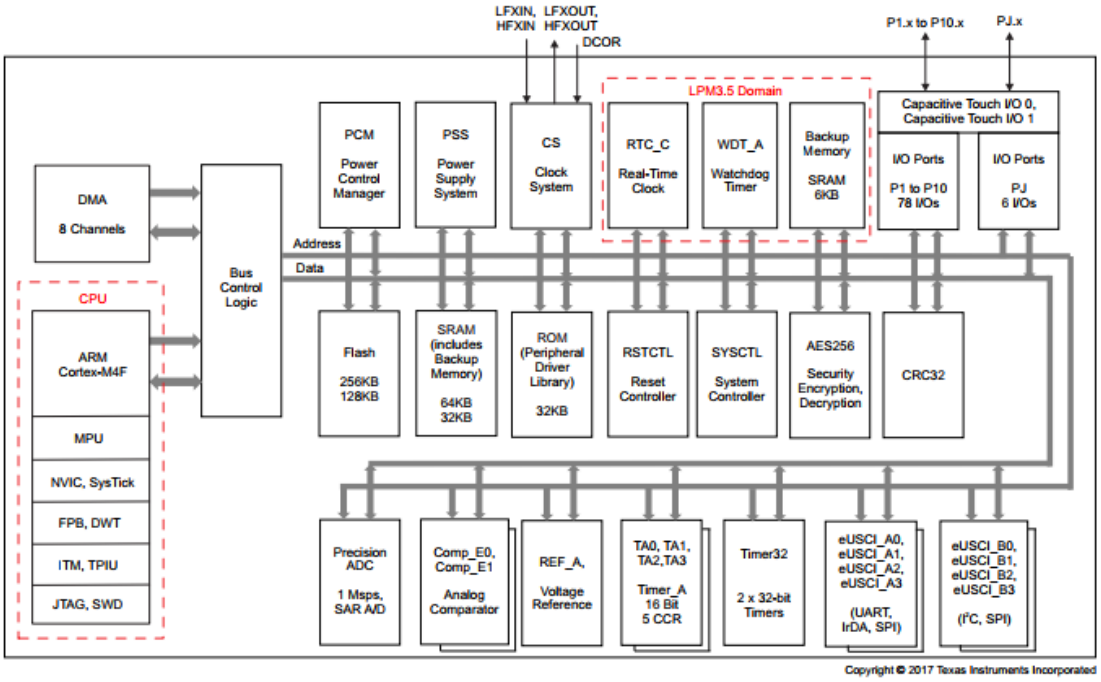


Ilustración 47 - Esquema bloques Launchpad

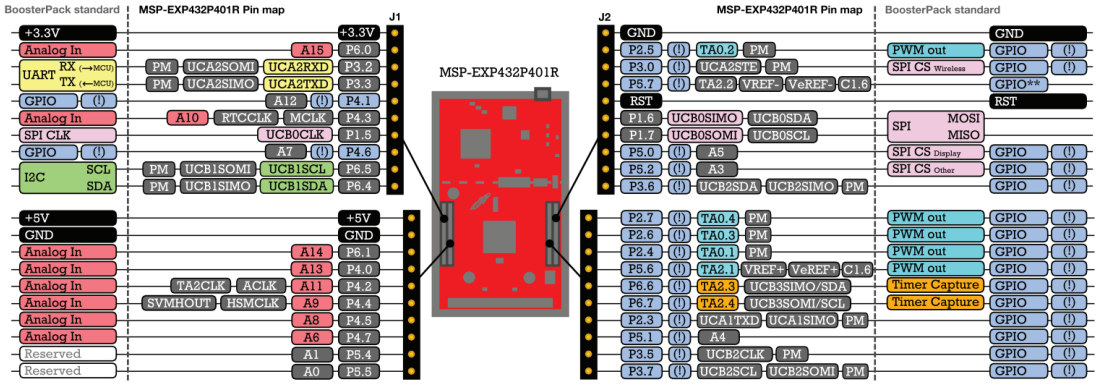


Ilustración 48 - Esquema pineado Launchpad

10.3. Licencias empleadas

Para reproducir el tema del Tetris se utilizó el código que tenemos en el repositorio:

https://github.com/electricmango/Arduino-Music-Project/tree/master/A_Theme___Tetris_with_Bass

Con la siguiente licencia:

Note: This license OVERRIDES any other license present in this project.

The MIT License (MIT)

Copyright (c) 2016 Minhyeok Jang

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.