ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO ELECTROMECÁNICO

DISEÑO E IMPLEMENTACIÓN DE UN JUEGO REAL PARA SALIR DE UN RECINTO

Autor: Laura Pallarés Portellano
Director: Álvaro Sánchez Miralles

Madrid
Julio 2016

Laura
Pallarés
Portellano

DISEÑO E IMPLEMENTACIÓN DE UN JUEGO REAL PARA SALIR DE UN RECINTO

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D._____**Laura Pallarés Portellano**_____
DECLARA ser el titular de los derechos de propiedad intelectual de la obra: _____**Diseño e implementación de un juego real para salir de un recinto**_____, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e

intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 19 de Julio de 2016

ACEPTA

Fdo……

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título "**Diseño e implementación de un juego real para salir de un recinto**" en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2015/2016 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Laura Pallarés Portellano Fecha: 19/ 07/ 2016

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Sánchez Miralles Fecha: ……/ ……/ ……

Vº Bº del Coordinador de Proyectos

Fdo.: Álvaro Sánchez Miralles Fecha: ……/ ……/ ……

Agradecimientos

En primer lugar, quiero agradecer a mi director de proyecto, Álvaro, su dedicación y apoyo, no sólo durante el tiempo que ha llevado terminar este proyecto, sino durante todo el pasado año.

Gracias, también, a mis padres y a mi hermano por su apoyo incondicional durante estos últimos años. Con ellos, todos los triunfos son compartidos. Gracias a mi abuela, por su fe ciega en mí, a pesar de no tener del todo claro para qué sirve todo esto.

Gracias a Rocío. Emocionalmente, una parte de este éxito es suyo.

Por último, gracias a todos mis amigos, compañeros, profesores, etc., que han aportado su granito de arena para hacer el día a día en ICAI más fácil.

DISEÑO E IMPLEMENTACIÓN DE UN JUEGO REAL PARA SALIR DE UN RECINTO

Autor: Pallarés Portellano, Laura
Director: Sánchez Miralles, Álvaro
Entidad colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Introducción

Gracias a la constante evolución de la tecnología, cada vez más se hace uso de la electrónica en la rutina del día a día de las personas. Pedir cita médica vía internet, evitar largas colas en los bancos pudiendo hacer pagos y transacciones desde un smartphone, tablet u ordenador, tener hogares inteligentes que, entre otras muchas funciones, facilitan su gestión y mantenimiento y se abastecen con energías renovables, son una mínima parte de los diferentes usos de la tecnología en la vida diaria.

Además, el entretenimiento y ocio en el siglo XXI es cada vez más tecnológico. Videojuegos, películas y series online, descarga de música, entre otras actividades, están ahora a la orden del día.

A pesar de las evidentes ventajas del impacto de las innovaciones tecnológicas en la sociedad, existen numerosas críticas acerca de su uso ya que, en ocasiones, conlleva la sustitución de actividades antes realizadas en grupo como hacer deporte, quedar con amigos y familia, etc. Este proyecto nace de la idea de integrar la electrónica con dichas actividades grupales de modo que un conjunto de personas pueda aprovechar el uso de las nuevas tecnologías para fomentar las relaciones personales, el trabajo en equipo y el ingenio a la par que disfrutan su tiempo de ocio.

Existe una idea preconcebida sobre que la relación de la tecnología y la electrónica con el tiempo libre se basa en la interacción de un solo individuo con la máquina. Con juegos como el que se propone, se pretende fomentar el trabajo en equipo al mismo tiempo que los jugadores se acercan al mundo de la electrónica interactuando con sensores, iPads y distintos elementos electrónicos que están presentes en nuestro día a día aunque no seamos del todo conscientes de ello.

La finalidad del juego es ir descubriendo las pistas que llevarán a la consecución del mismo. Dichas pistas pueden conseguirse gracias al ingenio, la observación y la resolución de los distintos enigmas propuestos a lo largo de la actividad.

En la actualidad, existen juegos con una base común a la del proyecto a realizar. Algunos de ellos tienen una base puramente tecnológica, como son los juegos de escape para iPhone/iPad. El objetivo de estos casos es salir de una habitación, investigando sus

objetos, interactuando con ellos y encontrando códigos y combinaciones ocultas. Todo ello en el marco de una historia.

Algunos de los rompecabezas interactivos de este tipo más famosos son: DOOORS, Diamond Penthouse Escape, The Room, y un largo etcétera.

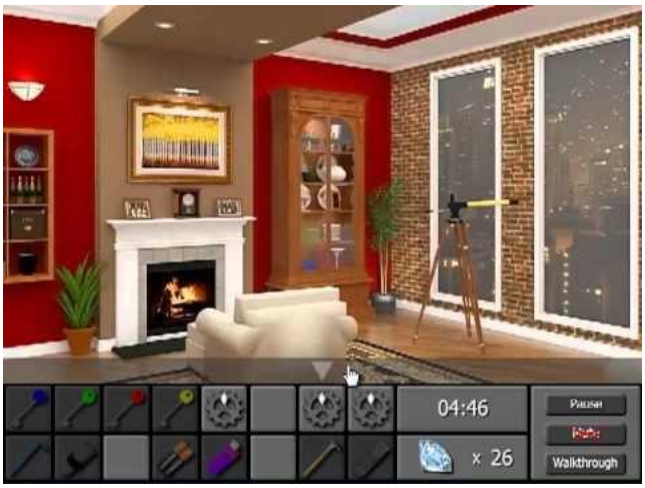


Figura 1. Pantalla del juego de escape Diamond Penthouse 2

Por otro lado, existen los juegos de escape en vivo que tratan de reproducir la esencia de los virtuales en el interior de una estancia real. En esta ocasión, desaparece la interacción hombre-máquina a la hora de desarrollar la actividad. Algunos ejemplos de estos juegos en vivo en Madrid son: Exit Game Madrid, Enigma Exprés Madrid y Escape Room Madrid.

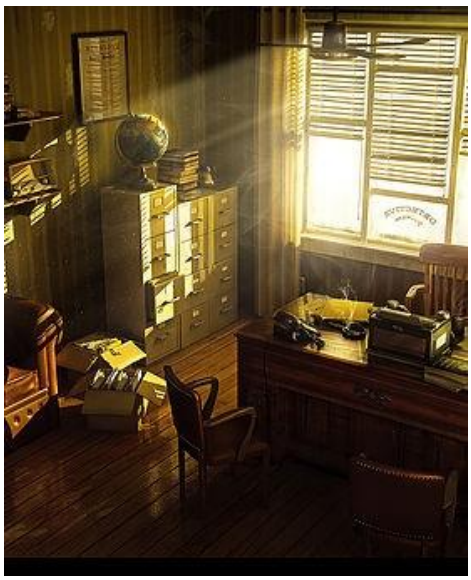


Figura 2. Sala de juego en Enigma Exprés Madrid

Metodología

El proyecto se compone de tres partes. Por un lado, el diseño del juego, por otro, el diseño de una tarjeta PCB y, por último, el diseño e implementación de cada uno de los rompecabezas a los que se tendrán que enfrentar los jugadores.

El juego consiste en una serie de pruebas y códigos que, al ser resueltas de forma exitosa, pondrán en marcha un servomotor que abrirá distintas cajas, cajones o puertas donde encontrarán pistas para la resolución de enigmas posteriores. Además, encontrarán una serie de figuras geométricas que deberán ir recogiendo para terminar el juego. No se trata de una actividad secuencial por lo que cada jugador o grupo de jugadores podrá recorrer caminos distintos para llegar al final.

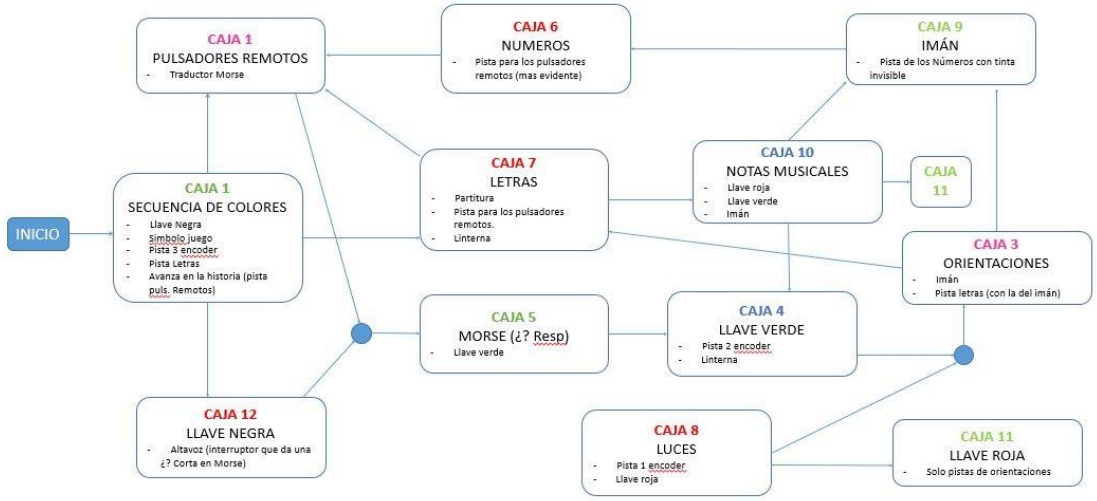


Figura 3. Esquema detallado de los posibles recorridos del jugador a lo largo de la actividad

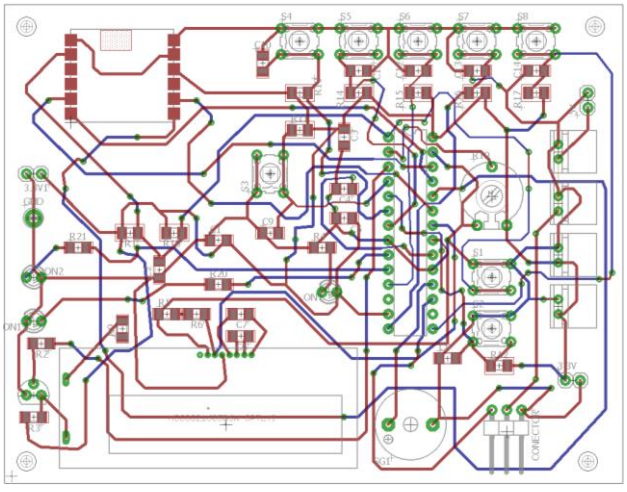


Figura 4. Diseño de la Printed Circuit Board

En el desarrollo de este proyecto se ha llevado a cabo el diseño e implementación de tres de los sistemas necesarios para completar el juego. Dichos sistemas son: detección

de secuencia de colores, detección de secuencia de letras y detección de proximidad de una persona para activar secuencia de luces.

- ➢ Detección de proximidad: El sistema consiste en un sensor de ultrasonido HC-SR04 que, al detectar a una persona a una distancia menor de 25 cm, envía una señal que activa una secuencia de luces. Además, se envía una señal sonora para alertar al jugador de la presencia de este sensor que forma parte del juego.

- ➢ Detección de secuencia de letras: El sistema consiste en una pantalla LCD en la que aparecen letras que van variando con cada pulso que se envía mediante un pulsador. Un segundo pulsador es el que se encarga de enviar una señal para fijar la letra escogida y, de este modo, poder empezar a variar la siguiente letra de la secuencia. Si la combinación resultara correcta, el sistema envía una señal que activa un servo. En caso de error, volvería al inicio.

- ➢ Detección de secuencia de colores: El sistema consiste en un led RGB que va variando su color en función de la posición de un potenciómetro. Un pulsador se encargará de hacer "recordar" al sistema el color escogido y permitirá la búsqueda del siguiente color de la secuencia. Si la combinación resultara correcta, el sistema envía una señal que activa un servo. En caso de error, volvería al inicio.

Resultados

El diseño del juego, sus normas y sus recorridos se han realizado de forma completa, quedando para el organizador del juego el marco de una historia o algunas variantes sobre cómo dar pistas de los rompecabezas.

Por otro lado, en el caso de la tarjeta electrónica, ésta presentaba algunos pequeños fallos de diseño tras la impresión. Estos fallos se centraban principalmente en la ausencia de pines para la conexión del programador del dsPIC. Se solucionó mediante un adaptador conectado a los pines del micro que llevaba incorporado la conexión del programador. Así, se solucionaba el problema de la programación del micro pero, para ello, hubo que sacrificar el uso de dos de los interruptores de la tarjeta.

Por último, se diseñaron correctamente los tres sistemas a implementar. En el caso del detector de proximidad, su implementación fue todo un éxito. En cambio, los otros dos sistemas no terminaron de funcionar de forma óptima. Los distintos bloques o sistemas más pequeños que los conformaban funcionaban por separado pero no se logró que funcionará el sistema de forma global.

Conclusiones

El proyecto no se ha completado en su totalidad ya que dos de los tres sistemas a implementar no han funcionado correctamente. Sin embargo, se podría considerar que el proyecto ha resultado un éxito ya que se han cubierto la mayoría de los objetivos propuestos y con una nueva depuración del código en un futuro podría llegar a funcionar a la perfección.

Bibliografía

[1] Real Academia Española. *Diccionario de la lengua española (22º ed.)* Madrid, España; 2001

[2] Sánchez Miralles, Álvaro. *Libro de Sistemas Electrónicos Digitales*. Universidad Pontificia Comillas.

[3] Microchip. *Datasheet del microcontrolador dsPIC33FJ32MC202*.

DESIGN AND IMPLEMENTATION OF A REAL GAME TO ESCAPE FROM A RECINT

Author: Pallarés Portellano, Laura
Director: Sánchez Miralles, Álvaro
Collaborating organization: ICAI – Universidad Pontificia Comillas

ABSTRACT

Introduction

Due to the constant evolution of technology, electronics is even more present in our day-to-day life. Some examples of technlogy uses in daily life are making an appointment with a doctor by Internet, avoiding queues at the bank by making payments and tradings from a Smartphone, Tablet or computer or owning Smart homes that facilitate its operations and maintenance and they supply themselves with renewable energies.

Furthermore, entertainment and leisure is increasingly technologic nowadays. Video-games, online films and series, music downloading, among others, are very frequent.

Despite the clear advantages of the technologic innovations impact on society, there are lots of bad reviews of their usage because, sometimes, it involves the substitution of group activities like doing sport, meeting friends, etc. This project stems from the idea of integrating electronics with those activities so that a group of people could take advantage of using new technologies to encourage personal relationships, teamwork and ingenuity, while at the same time enjoying their free time.

There is a preconceived idea about the relationship between technology and free time that is based on the interaction between the machine and a single individual. With games such as the one proposed, it is intended to encourage teamwork, while, at the same time, players find out more about electronics by interacting with sensors, iPads and different electronic elements that are present in our daily life.

The main objective of the game is discovering clues that help the player to complete the game. Obtaining those clues is due to ingenuity, observation and solving the many enigmas proposed.

Currently, there are some games which are based on the same idea. Some of them have only a technological basis, for example, escape mobile games. The objective in these games is to leave a place, making research work, interacting with the objects and founding hidden codes and combinations.

Some of the most famous interactive puzzles of this kind are: DOOORS, Diamond Penthouse Escape, The Room and so on.

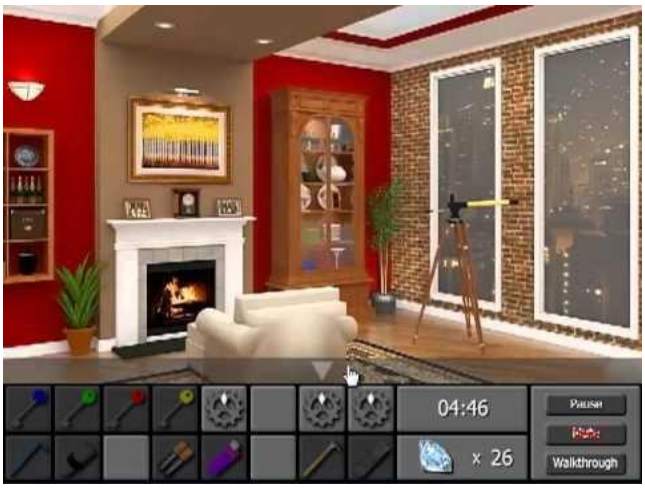


Figure 1. Diamond Penthouse 2 escape game screen

On the other hand, there are live escape games that try to reproduce the virtual one's essence inside a real room. In these, the interaction between the man and the machine disappears. Some examples of these live games in Madrid are: Exit Game Madrid, Enigma Exprés Madrid and Escape Room Madrid.

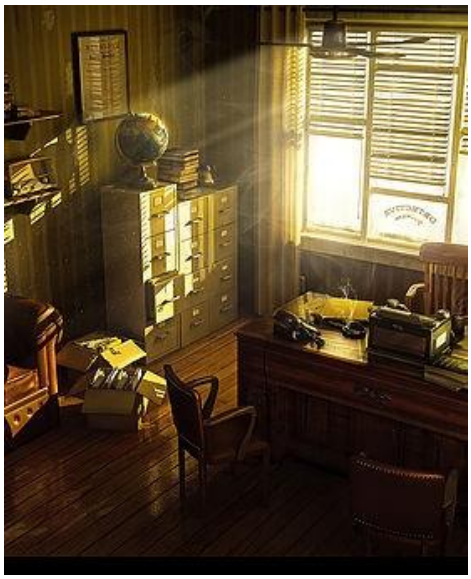


Figure 2. Enigma Exprés Madrid gaming room

Methodology

The Project has three different parts. First of all, the game design, then, a PCB design and, finally, the design and implementation of each and every one of the puzzles that players are going to face.

The game consist in a series of puzzles and codes that, after being solved, will activate a servomotor that will open different boxes, drawers or doors where the next clues will be found. Some kind of different geometric figures will also be discovered and they should be collected in order to finish the game. It is not a sequential activity so each player or group of players could go through different paths to reach the end.

.

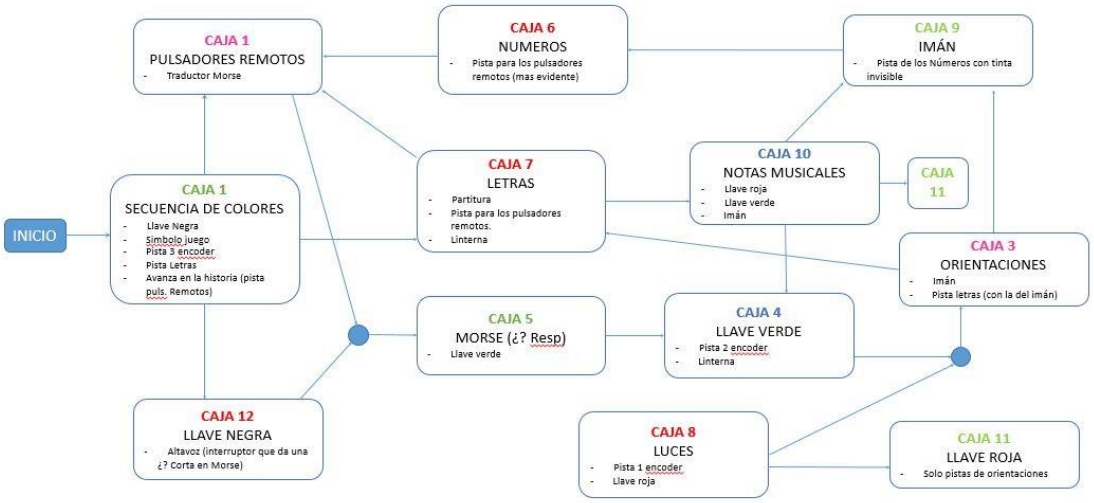


Figure 3. Detailed outline of the player possible routes

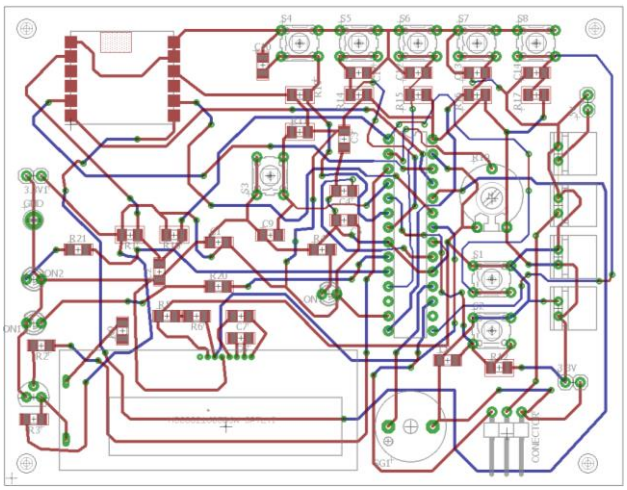


Figure 4. Printed Circuit Board design

In this project's development, three of the necessary systems have been designed and implemented. Those systems are: colour sequence detection, letter sequence detection and person closeness detection in order to activate a light sequence.

- ➢ Closeness detection: The system consist of a HC-SR04 ultrasonic sensor that, when a person is detected at a distance shorter than 25 cm, sends a

signal that activates a light sequence. A sound signal is also sent in order to alert the player of the presence of these sensors.

- ➢ Letter sequence detection: The system consist of a LCD display in which some letters appear and they change every time a switch is pressed. A second switch is the one that is in charge with setting the chosen letter so next letter can start to be changed. If the combination is correct, the system sends a signal that activates a servomotor. In the event of an error, it will be restarted.

- ➢ Colour sequence detection: The system consist in a RGB led that changes its colour depending on a potentiometer position. A switch will remember the chosen colour when pushed and the next colour sequence may be looked for. If the combination is correct, the system sends a signal that activates a servomotor. In the event of an error, it will be restarted.

Results

The game's design, its rules and routes have been completed. The game organizer could contribute by adding puzzle´s clues or story versions.

On the other hand, in the case of the electronic side, some small design failures have occured after the printing. Those failures were focused on the absence of pins for the dsPIC programmer. It was solved by an adaptor connected to the microprocessor's pins, which included the programmer connector. The microprocessor programming was solved but two switches had to be sacrificed for it.

Finally, the three systems were correctly designed. In the case of the proximity switch, its implementation was successful. By contrast, the other two systems did not perform satisfactorily. The different small parts of those systems worked separately but the operation of the global systems was not reached.

Conclusions

The project has not been completed due to the fact that two of the three systems to be implemented have not worked correctly. However, it should be considered that the project has been a success because most of the initial goals have been reached. With a new future code depuration, the project would work perfectly.

Bibliography

[1] Real Academia Española. *Diccionario de la lengua española (22º ed.)* Madrid, Spain; 2001

[2] Sánchez Miralles, Álvaro. *Libro de Sistemas Electrónicos Digitales*. Universidad Pontificia Comillas.

[3] Microchip. *Datasheet del microcontrolador dsPIC33FJ32MC202*.

Índice de la memoria

Índice de figuras

Índice de tablas Parte III

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

Gracias a la constante evolución de la tecnología, cada vez más se hace uso de la electrónica en la rutina del día a día de las personas. Pedir cita médica vía internet, evitar largas colas en los bancos pudiendo hacer pagos y transacciones desde un smartphone, tablet u ordenador, tener hogares inteligentes que, entre otras muchas funciones, facilitan su gestión y mantenimiento y se abastecen con energías renovables, son una mínima parte de los diferentes usos de la tecnología en la vida diaria.

Además, el entretenimiento y ocio en el siglo XXI es cada vez más tecnológico. Videojuegos, películas y series online, descarga de música, entre otras actividades, están ahora a la orden del día.

A pesar de las evidentes ventajas del impacto de las innovaciones tecnológicas en la sociedad, existen numerosas críticas acerca de su uso que, en ocasiones, conlleva la sustitución de actividades antes realizadas en grupo como hacer deporte, quedar con amigos y familia, etc. Este proyecto nace de la idea de integrar la electrónica con dichas actividades grupales de modo que un conjunto de personas pueda aprovechar el uso de las nuevas tecnologías para fomentar las relaciones personales, el trabajo en equipo y el ingenio a la par que disfrutan su tiempo de ocio.

1.1 ESTUDIO DE LOS TRABAJOS EXISTENTES / TECNOLOGÍAS EXISTENTES

En la actualidad, existen juegos con una base común a la del proyecto a realizar. La finalidad del juego es ir descubriendo las pistas que llevarán a la consecución del mismo. Dichas pistas pueden conseguirse gracias al ingenio, la

observación y la resolución de los distintos enigmas propuestos a lo largo de la actividad.

A continuación, se presentan los distintos juegos relacionados que existen en la actualidad.

1.1.1 JUEGOS DE ESCAPE PARA IPHONE/IPAD

Existen numerosos juegos de este tipo en el App Store. El objetivo de dichos juegos es salir de una habitación, investigando sus objetos, interactuando con ellos y encontrando códigos y combinaciones ocultas. Todo ello en el marco de una historia.

Algunos de los juegos de este tipo más famosos son:

- **DOOORS – Room escape game –**

Esta aplicación consta de un total de 80 niveles a completar. En ocasiones se requieren acciones intuitivas como pueden ser inclinar o agitar el dispositivo.

Figura 1. Dooors

- **Diamond Penthouse Escape 2**

Este juego se trata de la secuela de Diamond Penthouse Escape 1. La diferencia con otros juegos de este tipo reside en que, al mismo tiempo que se resuelven los distintos puzles y acertijos, se deben recoger unos diamantes que se encuentran escondidos por la habitación y que reportan puntuación extra. En la App Store se pueden encontrar, del mismo desarrollador, diversas versiones de este juego.

Figura 2. Diamond Penthouse Escape 2

- **The Room**

Esta aplicación presenta un rompecabezas interactivo envuelto en un juego de misterio. The Room no es compatible con todos los dispositivos iOS debido a los requisitos gráficos que precisa.

Figura 3. The Room

1.1.2 JUEGOS DE ESCAPE EN VIVO

Estos juegos están basados en los videojuegos mencionados en el punto anterior. En esta ocasión, los jugadores son encerrados en la habitación y poseen un tiempo límite para resolver el juego y salir de la habitación. Estos espacios están ambientados de acuerdo con el marco de la historia.

Algunos juegos de escape en vivo en Madrid son los siguientes:

- **Exit Game Madrid**

En Exit Game Madrid poseen varias misiones de diversa dificultad. Los equipos pueden ser de 2 a 5 personas y se pueden realizar competiciones entre dos equipos encerrados en habitaciones idénticas. Las misiones se pueden llevar a cabo tanto en español como en inglés y tienen una duración de 66 minutos. El precio oscila entre los 15 y los 22,50 €/persona.

Figura 4. Exit Game Madrid. Misión "El marchante de arte".

- **Enigma Exprés Madrid. Escape Room.**

En Enigma Exprés también se pueden encontrar varias aventuras de distintos niveles de dificultad. Aquí los equipos pueden ser de entre 2 y 6 personas y la duración de la misión es de 60 minutos. El precio ronda los 54 €/persona.

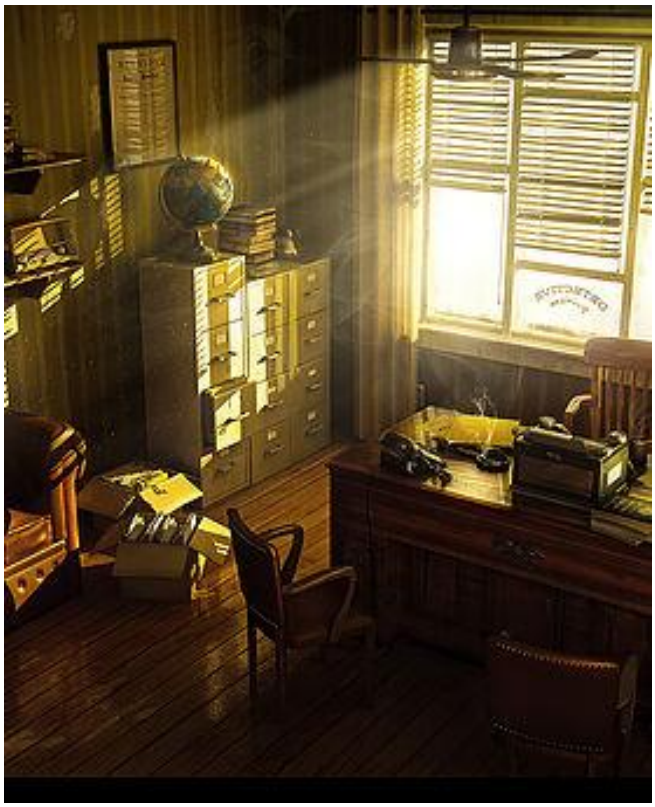


Figura 5. Enigma Exprés Madrid. Escape Room.

- **Escape Room Madrid**

Poseen 3 retos con dos habitaciones idénticas para cada uno de ellos de forma que dos equipos de jugadores puedan competir. Equipos de entre 2 y 5 personas y la duración de la sesión es de 60 minutos. El precio oscila entre los 20 y los 27,50 €/jugador.

Figura 6. Habitación de Escape Room Madrid.

1.2 MOTIVACIÓN DEL PROYECTO

Tal y como se ha comentado en los apartados anteriores, existe una idea preconcebida sobre que la relación de la tecnología y la electrónica con el tiempo libre se basa en la interacción de un solo individuo con la máquina. Con juegos como el que se propone, se pretende fomentar el trabajo en equipo al mismo tiempo que los jugadores se acercan al mundo de la electrónica interactuando con sensores, iPads y distintos elementos electrónicos que están presentes en nuestro día a día aunque no seamos del todo conscientes de ello.

Además de estar presentes en el campo del entretenimiento, cada vez más empresas están utilizando este tipo de juegos en sus procesos de selección. Durante las sesiones se puede valorar la capacidad de trabajo en equipo, dotes de liderazgo,

capacidad de organización de ideas y de búsqueda de soluciones de cada uno de los aspirantes en un ambiente con menor presión que la que se puede tener en las oficinas de la empresa.

En la actualidad, no existe ningún juego de escape en vivo en el que se reúnan la tecnología y la electrónica con las actividades mencionadas anteriormente. Por ello, consideramos que con este proyecto se va un paso más allá en la adaptación de los videojuegos de escape a la vida real.

1.3 OBJETIVOS

El objetivo del proyecto es el desarrollo y la implantación de un juego de escape electrónico en vivo de fácil montaje, de manera que éste pueda desarrollarse sin necesidad de contar con un emplazamiento fijo. Para ello se desarrollarán las distintas pruebas a las que se tendrá que enfrentar cada jugador durante el mismo.

1.4 RECURSOS / HERRAMIENTAS EMPLEADAS

Los recursos y herramientas utilizadas para la realización del proyecto son las siguientes:

1.4.1 HARDWARE

- *Microcontroladores dsPIC33FJ32MC202 [1]*
- *Sistema de desarrollo MPLAB ICD 3*
- *Un ordenador*
- *Servomotores Futaba S3003 [2]*
- *Sensores ultrasónicos HC-SR04 [3]*
- *Pantalla LCD alfanumérica Midas MCCOG21605B6W-SPTLYI [4]*
- *Módulo de comunicación Wi-Fi U3O-G2M5477 [5]*
- *Componentes electrónicos (tarjetas, cables, resistencias, pulsadores, condensadores, leds, zumbadores, potenciómetros, etc.)*

- *Equipo de soldadura*
- *Dispositivo con sistema operativo iOS (iPad)*
- *Baterías*

1.4.2 SOFTWARE

- *MPLAB X IDE v3.20, para programar los microcontroladores*
- *Eagle 7.5.0, para el diseño de las tarjetas*
- *Microsoft Office 2013*
- *Adobe Acrobat Reader*

Capítulo 2 DISEÑO DEL JUEGO

En este capítulo se va a realizar una descripción del juego a desarrollar. En la Figura 7 se puede observar un esquema de los posibles recorridos de cada jugador a lo largo del desarrollo del juego.

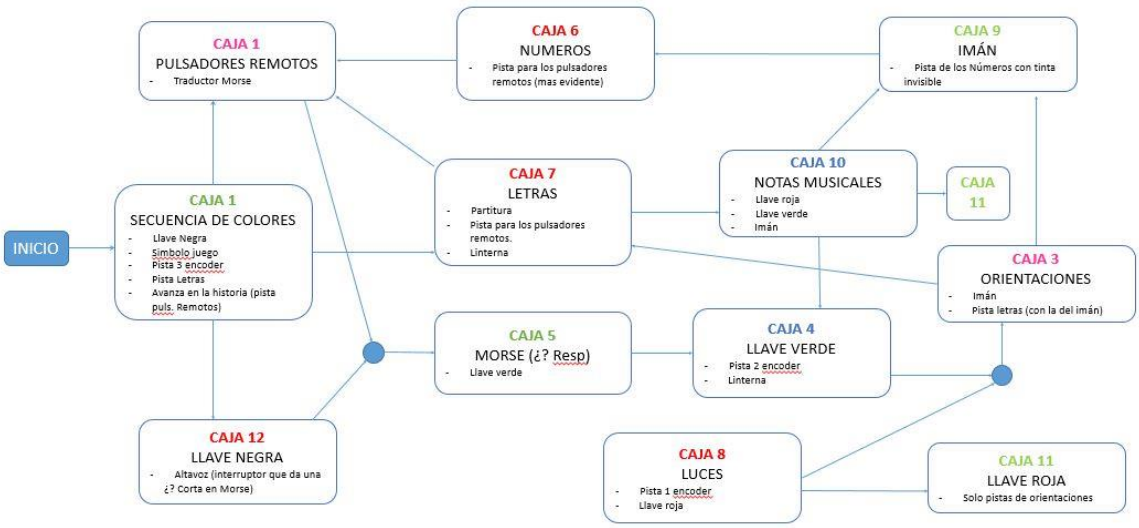


Figura 7. Esquema detallado de los posibles recorridos del jugador a lo largo de la actividad

2.1 PLANTEAMIENTO DEL JUEGO

El juego consiste en una serie de pruebas y códigos que, al ser resueltas de forma exitosa, pondrán en marcha un servomotor que abrirá distintas cajas, cajones o puertas donde encontrarán pistas para la resolución de enigmas posteriores. Además, encontrarán una serie de figuras geométricas que deberán ir recogiendo para terminar el juego.

El número de jugadores estará limitado en la medida en que el organizador ponga a su disposición un mayor o menor número de copias de las pistas y demás informaciones necesarias para la ejecución del mismo. El modo de juego se encuentra abierto a la posibilidad de participar de forma individual o en grupo, ya

que se trata de emplear la lógica para la resolución de los distintos enigmas propuestos.

Por otro lado, la duración del mismo también será variable y dependerá del organizador. Puede implantarse en cualquier lugar, de modo que el tiempo dependerá de la disponibilidad del emplazamiento donde se lleve a cabo.

A pesar de que las pruebas se encuentran numeradas para facilitar la explicación de cada una de ellas, el juego no es secuencial, es decir, no hay un solo camino para llegar al final del mismo, así como unos jugadores lo podrán resolver con mayor éxito que otros. Esto se explicará con mayor detenimiento en el Apartado 2.16.

2.2 INICIO DEL JUEGO

El inicio del juego es el común para todos los participantes. En primer lugar, cada jugador comenzará la prueba con el escaneo de un código QR similar al de la Figura 8 que pondrá a su disposición el organizador del juego. Este código los direccionará a una página web en la que recibirán las instrucciones necesarias así como el inicio de la historia en la que se encontrará enmarcado el juego.

Figura 8. Ejemplo de código QR

Además, recibirán las primeras pistas escondidas para la resolución de las primeras pruebas. Por ejemplo, el fondo de la página web llevará una determinada secuencia de colores que necesitarán para poner en marcha el servomotor que les

proporcionará los primeros elementos necesarios para el juego (ver esquema de la Figura 7).

2.3 PRIMERA PRUEBA

Como se puede observar en la Figura 7, la primera prueba es común a todos los jugadores. Esta prueba consiste en acertar la combinación de colores correcta. Para hallar dicha combinación, el jugador tendrá que haber observado con detenimiento la página web del inicio del juego (a la que podrá acceder siempre que quiera) y relacionarla con esta prueba.

Para el desarrollo de la misma, el jugador encontrará una rueda (potenciómetro), un led RGB y un pulsador. Al girar dicha rueda, el participante podrá observar cómo el led va cambiando su color. Cuando considere que se trata del color correcto, deberá apretar el pulsador para aceptar el color y podrá proceder a buscar el segundo. El método será idéntico para el resto de colores hasta finalizar la combinación.

Una vez introducidos todos los colores de la combinación, en caso de error, el sistema producirá un sonido que alertará al jugador del fallo y podrá volver a empezar.

En caso de acierto, el sistema producirá un sonido que avisará al jugador de su acierto y accionará el servomotor que permitirá el acceso al contenido de la caja, cajón o puerta (según montaje por parte del organizador).

2.4 SEGUNDA PRUEBA

La segunda prueba consiste en dos pulsadores localizados a cierta distancia el uno del otro y que deben ser accionados al mismo tiempo para poner en marcha el servomotor correspondiente.

2.5 TERCERA PRUEBA

La tercera prueba consiste en introducir una serie de giros con una rueda. Dicha rueda se corresponderá con un potenciómetro que, dependiendo de la posición del mismo, proporcionará una señal u otra. Una vez hallada la primera posición, se fijará la misma pulsando un interruptor de forma que, al mover de nuevo el potenciómetro, se estará buscando la siguiente posición.

En caso de introducir la secuencia correcta, se encenderá una luz verde y se activará el servo que bloquea el acceso a la información. En caso de error, se encenderá una luz roja y se activará un altavoz para alertar al jugador de su error [7].

2.6 CUARTA PRUEBA

La cuarta prueba consiste en una caja cuya apertura debe realizarse con una llave encontrada tras la consecución de alguna otra prueba.

2.7 QUINTA PRUEBA

La quinta prueba consiste en introducir una secuencia de símbolos morse que formarán la respuesta a una pregunta formulada en alguna pista anteriormente encontrada (véase Apartado 2.14). Estos símbolos se introducirán mediante sendos pulsadores que se corresponderán con el punto y la raya, respectivamente. En caso de acierto, se encenderá una luz verde y se activará el servo que bloquea la apertura y en caso de error, se encenderá una luz roja y se activará un altavoz que alertará al jugador de su error [7].

2.8 SEXTA PRUEBA

La sexta prueba consiste en introducir una secuencia de números. En esta ocasión, el jugador se encontrará con dos pulsadores. El primero de ellos tendrá como función ir variando el número que aparecerá en un display. Cuando el jugador crea haber encontrado el número correcto, usará el segundo pulsador para fijarlo en la pantalla del display. De este modo, al volver al primer pulsador, el número que variará será el siguiente en la secuencia.

En caso de acierto, se encenderá una luz verde y se activará el servo que bloquea el acceso a la información y, en caso de error, se encenderá una luz roja y se activará un altavoz que alertará al jugador de su error [7].

2.9 SÉPTIMA PRUEBA

La séptima prueba consiste introducir una secuencia de letras. En esta ocasión, el jugador se encontrará con dos pulsadores y un display LCD. Cada vez que se accione el primer pulsador, la primera letra irá variando en la pantalla del display. En el momento en el que el participante encuentre la letra que buscaba, deberá pulsar el segundo pulsador para aceptarla. Al aceptar cada letra, el primer pulsador variará la siguiente, manteniendo fijas las anteriores.

Una vez introducida la secuencia completa, en caso de error, el sistema alertará al jugador del fallo produciendo un determinado sonido y éste podrá volver a intentarlo de nuevo.

En caso de acierto, el sistema producirá un sonido distinto que avisará al jugador de su acierto y pondrá en marcha el servomotor correspondiente que le dará acceso a la nueva información.

2.10 OCTAVA PRUEBA

La octava prueba consiste en introducir una determinada secuencia de luces. En este caso, el jugador se encontrará con una serie de fotorresistencias que deberá iluminar siguiendo un orden concreto. Dicha iluminación se podrá realizar con cualquier fuente de luz (flash del móvil, linterna, etc.).

Una vez introducida la secuencia completa, en caso de error, el sistema alertará al jugador del fallo produciendo un determinado sonido y éste podrá volver a intentarlo.

En caso de acierto, el sistema producirá un sonido distinto que avisará al jugador de su acierto y pondrá en marcha el servomotor correspondiente que le dará acceso a la nueva información.

2.11 NOVENA PRUEBA

La novena prueba consiste en una caja de apertura con imán. Existen distintas variables para este caso ya que también podría utilizarse las comúnmente conocidas cajas mágicas.

2.12 DÉCIMA PRUEBA

La décima prueba consiste en una determinada secuencia de notas musicales. Esta combinación se introducirá mediante unos pulsadores, cada uno de los cuales se corresponderá con una nota distinta. En caso de acierto, se encenderá una luz verde y se activará el servo que bloquea la apertura y, en caso de error, se encenderá una luz roja y se activará un altavoz que alertará al jugador del fallo [7].

2.13 UNDÉCIMA PRUEBA

La undécima prueba consiste de nuevo en localizar la llave que posibilitará la apertura de la caja, cajón o puerta que impide el acceso a nueva información. Esta llave se encontrará junto a más pistas y objetos tras la consecución de alguna otra prueba (obsérvese el gráfico de la Figura 7).

2.14 DUODÉCIMA PRUEBA

La duodécima prueba consiste, una vez más, en localizar la llave que posibilita la apertura. En este caso, además, se encontrará en su interior un pulsador que activará un altavoz que emitirá una pregunta en código morse. El jugador deberá interpretar dicha pregunta para la resolución de la prueba número cinco.

2.15 PISTA SECUENCIA DE LUCES

Para poder realizar correctamente la octava prueba (véase apartado 2.10), el jugador deberá haber conseguido previamente la información correspondiente. Para ello, se encontrará a lo largo del juego, dos sensores ultrasónicos programados para detectar a una persona a menos de 25 cm. El participante deberá, con ayuda de otra persona, colocarse frente a ambos sensores al mismo tiempo para accionar una serie de leds que se encenderán y apagarán, otorgándole la secuencia correcta que deberá posteriormente introducir en los sensores de luz.

Cada vez que una persona se sitúe en el rango de distancia programado, el sistema emitirá un sonido para alertar al jugador de la presencia de un componente clave del juego.

2.16 PRUEBA FINAL

Durante el desarrollo del juego, los participantes encontrarán, junto con las pistas y el resto de informaciones, unas figuras geométricas que tendrán que ir recogiendo.

En la última etapa del juego, se encontrarán con una aplicación para iOS basada en la desarrollada en el proyecto de fin de carrera Gabriel, realizado por Ana Ruiz García [8]. En dicha aplicación aparecerán una serie de figuras que se corresponderán con las encontradas durante el juego. El jugador deberá colocarlas en su lugar correspondiente con una orientación determinada por pistas que habrá ido encontrando durante la consecución del juego.

Dependiendo del número de figuras diferentes encontradas por el participante y las colocadas de forma correcta, le corresponderá una puntuación u otra de forma que, a pesar de haber llegado a este punto, algunos jugadores lo habrán superado con mayor éxito que otros.

Capítulo 3 HARDWARE

En este capítulo se van a describir detalladamente los distintos componentes empleados en el desarrollo del proyecto así como el proceso seguido para el diseño de las tarjetas electrónicas (PCB[1]).

3.1 DISEÑO TARJETAS PCB

Para el desarrollo de este proyecto, se ha diseñado una PCB y se ha aprovechado el diseño de la tarjeta realizada por Ana Cogollos Baranda para su proyecto [7].

El diseño se ha realizado con el programa Eagle 7.5.0. En primer lugar, se ha llevado a cabo un diseño esquemático de todas las conexiones necesarias para el desarrollo del proyecto, englobando todos los componentes necesarios para cada una de las pruebas de forma que fuera suficiente con estos dos tipos de tarjeta.

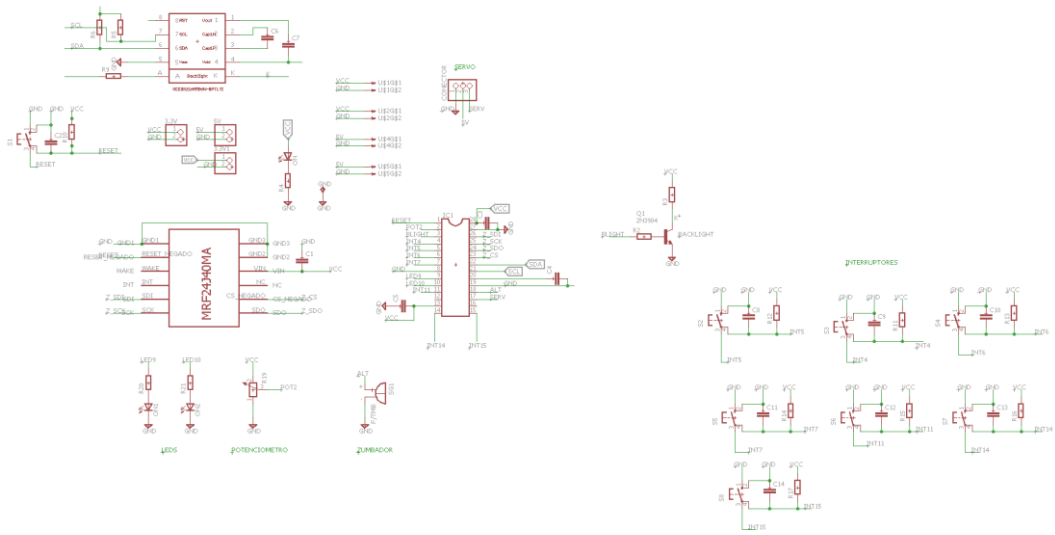


Figura 9. Diseño esquemático de la PCB (.sch)

[1] Printed Circuit Board

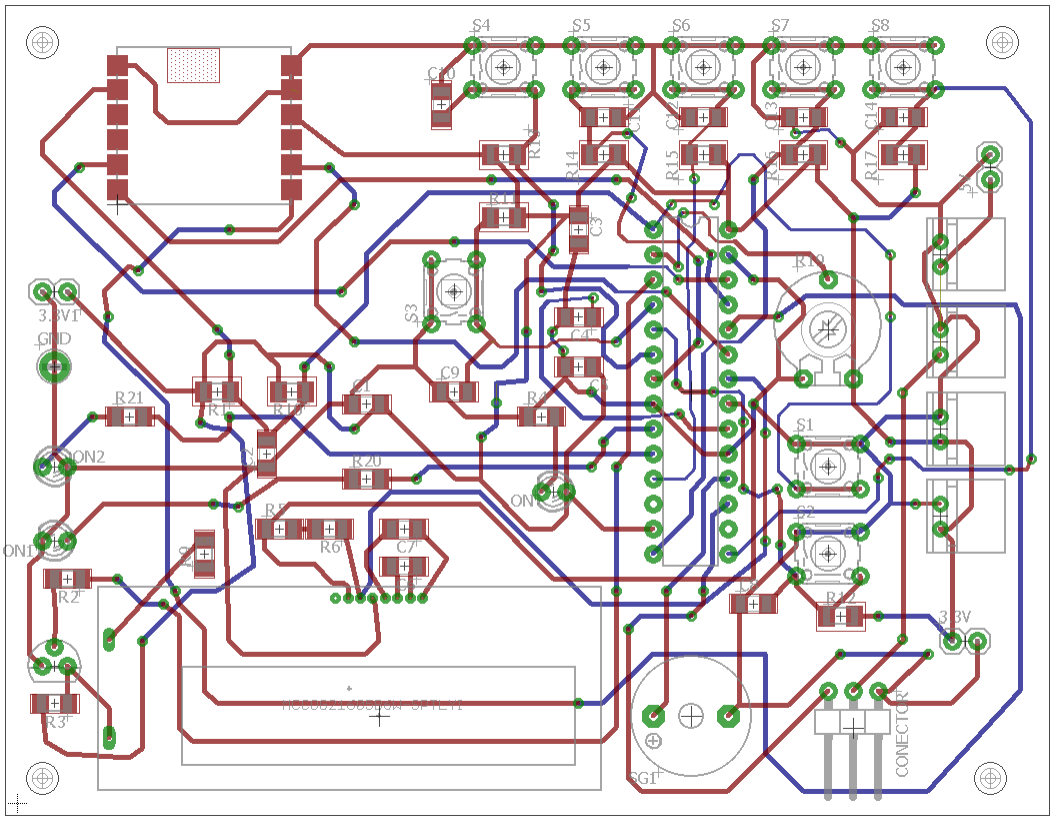


Figura 10. Diseño de las pistas de la tarjeta de componentes (.brd)

En la Parte II de este documento se detalla el plano de la tarjeta con sus componentes y conexiones.

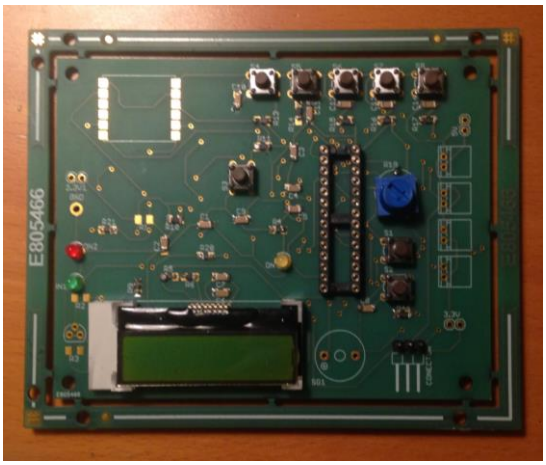


Figura 11. Tarjeta electrónica impresa

3.2 SENSOR DE ULTRASONIDOS

Este sensor se ha utilizado para la pista de la secuencia de luces (véase Capítulo 2, apartado 2.15). El modelo utilizado es el *HC-SR04*. Se puede ver un ejemplar en la Figura 11.

Figura 12. Sensor de ultrasonidos HC-SR04

Se ha elegido este tipo de sensor para detectar la presencia de un jugador puesto que el haz de ultrasonidos es cónico, lo que presenta una ventaja ya que éste será detectado en un área mayor que si se tratara de un sensor de tipo infrarrojo.

Entre la información a destacar de las especificaciones se encuentra su rango mínimo y máximo de medida, siendo estos 2 cm y 4 m, respectivamente. El resto de especificaciones pueden consultarse en la datasheet del fabricante [3].

El funcionamiento básico [6] de este tipo de sensores como medidores de distancia se puede observar en el esquema de la Figura 12. Se cuenta con un transmisor que emite un pulso de ultrasonido[2], cuyo campo de acción es de forma cónica, que rebotará sobre la superficie que encuentre en su camino. La reflexión de ese pulso será detectada por el receptor. Midiendo el tiempo que transcurre entre la emisión del pulso y la percepción del eco, se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora. Para ello, hay que tener en cuenta que el tiempo medido se trata del que transcurre durante el camino de ida y vuelta del sonido y por tanto, habrá que tener en cuenta únicamente la mitad de este tiempo.

[2] Sonidos con una frecuencia mayor que la máxima audible por el ser humano. En este caso, se utilizará sonido con una frecuencia de 40 kHz.

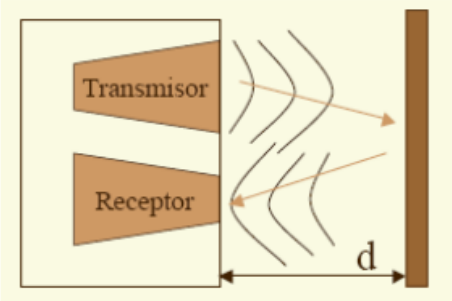


Figura 13. Funcionamiento de sensor ultrasónico

Para obtener la distancia recorrida se utilizará la siguiente fórmula:

$$distancia = \frac{1}{2} \times velocidad\ del\ sonido \times tiempo\ medido$$

Existen una serie de inconvenientes en el funcionamiento de los sensores ultrasónicos. Entre los más destacados se encuentran:

- El eco recibido nos indica que existe un obstáculo en el área cónica de acción del pulso pero no especifica la localización exacta del mismo. Además, la reflexión del sonido puede variar dependiendo de las características del material del obstáculo.
- Los factores ambientales, como la humedad y la temperatura, tienen una gran repercusión sobre las medidas que se realizan ya que las ondas de sonido se trasmiten por el aire y en la densidad de éste influyen dichos factores.
- Habrá que poner especial atención a la hora de situar los sensores de ultrasonidos ya que podrían aparecer *falsos ecos*. Estos ecos pueden deberse a la reflexión de la onda sonora en varias superficies antes de ser captada por el receptor. En este caso, el tiempo medido será mayor que el que transcurriría realmente al recorrer la distancia al obstáculo más cercano. Otra razón que podría provocar estos falsos ecos sería la comúnmente conocida como *crosstalk*. En este caso, varios sensores estarían trabajando al mismo tiempo y los pulsos emitidos por uno, serían recibidos por otro, y viceversa.

3.3 SERVOMOTORES

Los servomotores son empleados para el bloqueo y desbloqueo de la apertura de las distintas cajas, cajones o puertas del juego. Se han usado servomotores estándar marca Futaba, modelo s3003 [2].

Figura 14. Servomotor Futaba s3003

Capítulo 4 SOFTWARE

La programación de todas las pruebas se ha realizado en lenguaje C usando el programa MPLAB X IDE v3.20 y está basada en máquinas de estado.

En los siguientes apartados se explicarán brevemente los distintos estados y funciones que conforman cada uno de los códigos de las diferentes pruebas llevadas a cabo en este proyecto, a fin de aclarar al lector los códigos fuente adjuntados en la Parte IV de este documento.

4.1 PRIMERA PRUEBA: SECUENCIA DE COLORES

La máquina de estados de esta primera prueba está formada por cuatro estados principales, tal y como se puede observar en la Figura 15.

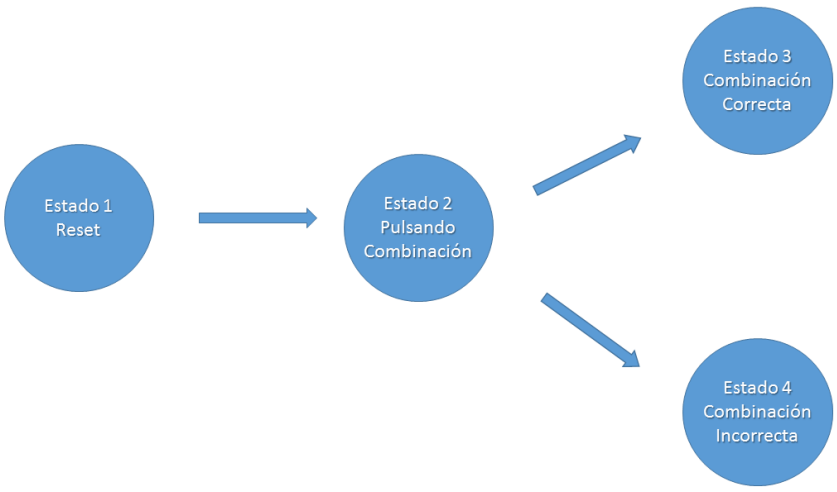


Figura 15. Máquina de estado primera prueba (secuencia de colores)

Dichos estados son:

- Estado 1: Reset.

  Este es el estado en el que entra el sistema cada vez que se reinicia. Al entrar en este estado, todas las variables vuelven a su estado inicial sin importar qué parte del código se estuviera ejecutando en el momento del reseteo. Una vez reiniciadas todas las variables, el programa salta automáticamente al estado 2.

- Estado 2: Pulsando combinación.

  Este estado, tal y como su nombre indica, es en el que se encuentra el sistema desde que se pone en marcha el dispositivo hasta que el jugador introduce los cuatro colores que conforman la secuencia completa. Dependiendo de si la combinación es correcta o no, el programa empezará a ejecutar el tercer estado o el cuarto, respectivamente.

  Mientras se está ejecutando este estado, el jugador podrá variar los colores del led RGB con el potenciómetro, así como ir aceptando los que crea correctos con el pulsador.

- Estado 3: Combinación correcta.

  En caso de que la combinación introducida en el estado 2 sea correcta, el programa entrará en este estado. Al ejecutarse este estado, se emitirá un sonido compuesto por dos frecuencias distintas y pondrá en marcha el servomotor que desbloqueará la apertura girando 90º.

  Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

- Estado 4: Combinación incorrecta

  En caso de que la combinación introducida sea incorrecta, el programa entrará en este estado. Mientras se ejecuta, el sistema producirá un sonido formado por una única frecuencia.

  Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

Por otro lado, se van a explicar brevemente las funciones más destacadas de este código. Las principales son:

- InicializaTimer1() e InicializaTimer2()

  En estas dos funciones, se inicializan los dos timers que se van a utilizar en el programa. En este caso, las interrupciones de los timers saltarán cada 1ms y 0.1ms, respectivamente.

- Init_AD (int canal)

  Esta función inicializa el conversor analógico/digital que se necesita para poder digitalizar la información que nos proporciona el potenciómetro. Al llamar a esta función, debemos enviarle a la misma el número del canal analógico en el que está conectado el potenciómetro.

- get_medida ()

  Esta función nos devuelve una variable que contiene la información recibida a través del potenciómetro.

- Pulsado()

  Se trata de una función que devuelve un 1 en caso de que haya habido un flanco de subida en el pulsador, es decir, en caso de que haya sido pulsado. En caso contrario, devolverá un 0.

- Servo_actua()

  En esta función es donde se gestiona el giro del servo. Se trata de un PWM en el que se fija un ancho de pulso igual a 2.5ms, que es el que produce un giro de 90º, y el periodo de la señal, que en el caso del servo ha de ser de 50 Hz.

- Estado2(), estado3() y estado4()

  En estas funciones se lleva a cabo lo correspondiente a cada uno de los estados, correspondiéndose con lo explicado al inicio de este apartado.

- Main()

  En esta función se gestiona la llamada a la función correspondiente al estado en que se encuentra el programa. Por ejemplo, en caso de encontrarse en el segundo estado, es aquí donde se llama a la función estado2() para que se ejecute. Además, es donde se declaran e inicializan los puertos y todas aquellas funciones que necesiten ser inicializadas.

- void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void) y void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void)

  Se trata de dos funciones que gestionan las interrupciones correspondientes a cada uno de los timers utilizados.

En el primer caso, se realiza la actualización de los puertos de entrada para la gestión del pulsador, así como las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para la reproducción de cada sonido por el altavoz.

En el caso de la segunda interrupción, se gestionan las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para activar el servomotor.

- void __attribute__((interrupt,no_auto_psv)) _ADC1Interrupt(void)

Se trata de la función que gestiona la interrupción del conversor A/D necesario para el manejo del potenciómetro. Cada vez que se ejecuta esta interrupción, se actualiza el valor de la variable medidas que contiene el valor de la posición del potenciómetro en ese momento.

4.2 SÉPTIMA PRUEBA: SECUENCIA DE LETRAS

La máquina de estados de esta primera prueba está formada por siete estados principales. Dichos estados son:

- Estado 1: Reset.

Este es el estado en el que entra el sistema cada vez que se reinicia. Al entrar en este estado, todas las variables vuelven a su estado inicial sin importar qué parte del código se estuviera ejecutando en el momento del reseteo. Una vez reiniciadas todas las variables, el programa salta automáticamente al estado 2.

- Estado 2: Primera letra.

Este estado, tal y como su nombre indica, es en el que se encuentra el sistema desde que se pone en marcha el dispositivo hasta que el jugador pulsa el pulsador que fija la primera letra en la pantalla.

Mientras se está ejecutando este estado, el jugador podrá variar la primera letra en la pantalla pulsando el pulsador correspondiente.

En el momento en que el jugador activa el pulsador para fijar la primera letra, el programa ejecuta el estado 3.

- Estado 3: Segunda letra.

Mientras se ejecuta este estado, la primera letra introducida por el usuario se mantiene fija en la pantalla del display, de forma que, al pulsar el pulsador correspondiente, la letra que varía es la segunda.

Una vez que el jugador pulsa el pulsador que fija la segunda letra en la pantalla, el programa ejecuta el estado 4.

- Estado 4: Tercera letra

En este caso, el funcionamiento es idéntico al del estado anterior pero, en esta ocasión, la dos primeras letras se mantienen fijas en la pantalla y se procede a variar la tercera letra con el pulsador.

Una vez que el jugador pulsa el pulsador que fija la tercera letra en la pantalla, el programa ejecuta el estado 5.

- Estado 5: Tercera letra

  En este caso, el funcionamiento es idéntico al del estado anterior pero, en esta ocasión, la tres primeras letras se mantienen fijas en la pantalla y se procede a variar la cuarta letra con el pulsador.

  Una vez que el jugador pulsa el pulsador que fija la tercera letra en la pantalla, el programa ejecuta el estado 6 en caso de que la combinación sea la correcta, y el estado 7 si la combinación es incorrecta.

- Estado 6: Combinación correcta.

  En caso de que la combinación introducida sea correcta, el programa entrará en este estado. Al ejecutarse este estado, se emitirá un sonido compuesto por dos frecuencias distintas y pondrá en marcha el servomotor que desbloqueará la apertura girando 90º.

  Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

- Estado 7: Combinación incorrecta

  En caso de que la combinación introducida sea incorrecta, el programa entrará en este estado. Mientras se ejecuta, el sistema producirá un sonido formado por una única frecuencia.

  Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

Por otro lado, se van a explicar brevemente las funciones más destacadas de este código. Las principales son:

- InicializaTimer1() e InicializaTimer2()

  En estas dos funciones, se inicializan los dos timers que se van a utilizar en el programa. En este caso, las interrupciones de los timers saltarán cada 1ms y 0.1ms, respectivamente.

- Pulsado_letra()

  Cada vez que se llama a esta función, se comprueba si el pulsador que varía la letra en la pantalla del display ha sido o no pulsado.

- Pulsado_ok()

  Cada vez que se llama a esta función, se comprueba si el pulsador que fija la letra en la pantalla del display ha sido o no pulsado.

- Servo_actua()

  En esta función es donde se gestiona el giro del servo. Se trata de un PWM en el que se fija un ancho de pulso igual a 2.5ms, que es el que produce un giro de 90º, y el periodo de la señal, que en el caso del servo ha de ser de 50 Hz.

- Estado2(), estado3(), estado4(), estado5(), estado6() y estado7()

  En estas funciones se lleva a cabo lo correspondiente a cada uno de los estados, correspondiéndose con lo explicado al inicio de este apartado.

- Main()

  En esta función se gestiona la llamada a la función correspondiente al estado en que se encuentra el programa. Por ejemplo, en caso de encontrarse en el segundo estado, es aquí donde se llama a la función estado2() para que se ejecute. Además, es donde se declaran e inicializan los puertos y todas aquellas funciones que necesiten ser inicializadas.

- InitDisplay(), DisplaySetLine() y DisplayPuts()

  Estas funciones gestionan la pantalla alfanumérica LCD. InitDisplay(), tal y como su nombre indica, es la que se encarga de inicializar el display.

  La función DisplaySetLine() se utiliza para indicar en cuál de las dos líneas que presenta la pantalla, se quiere escribir. En caso de enviarle un 0 a la función, el mensaje enviado aparecerá en la primera línea. En caso de enviar un 1, en la segunda.

  Por último, la función DisplayPuts() es la que se encarga de mostrar en la pantalla el mensaje que se le envía.

- void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void) y void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void)

  Se trata de dos funciones que gestionan las interrupciones correspondientes a cada uno de los timers utilizados.

  En el primer caso, se realiza la actualización de los puertos de entrada para la gestión de los pulsadores, así como las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para la reproducción de cada sonido por el altavoz.

En el caso de la segunda interrupción, se gestionan las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para activar el servomotor.

4.3 OCTAVA PRUEBA: SECUENCIA DE LUCES

Al igual que el primer sistema, la máquina de estados de esta octava prueba está formada por cuatro estados principales, tal y como se puede observar en la Figura 16.

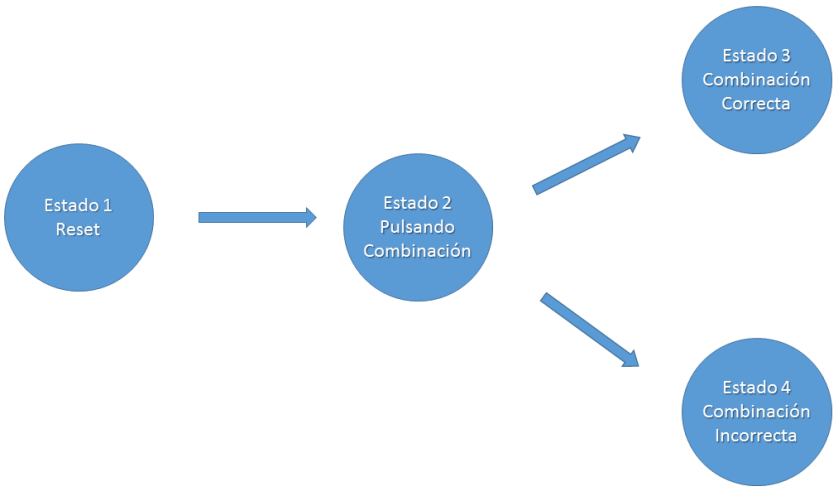


Figura 16. Máquina de estado octava prueba (secuencia de luces)

Dichos estados son:

- Estado 1: Reset.

Este es el estado en el que entra el sistema cada vez que se reinicia. Al entrar en este estado, todas las variables vuelven a su estado inicial sin importar qué parte del código se estuviera ejecutando en el momento del

reseteo. Una vez reiniciadas todas las variables, el programa salta automáticamente al estado 2.

- Estado 2: Introduciendo combinación.

Este estado, tal y como su nombre indica, es en el que se encuentra el sistema desde que se pone en marcha el dispositivo hasta que el jugador introduce la secuencia completa. Dependiendo de si la combinación es correcta o no, el programa empezará a ejecutar el tercer estado o el cuarto, respectivamente.

Mientras se está ejecutando este estado, el jugador podrá introducir de forma secuencial y ordenada la combinación mediante la incidencia con un foco de luz (linterna, flash del móvil, etc.) en las distintas fotorresistencias LDR.

- Estado 3: Combinación correcta.

En caso de que la combinación introducida en el estado 2 sea correcta, el programa entrará en este estado. Al ejecutarse este estado, se emitirá un sonido compuesto por dos frecuencias distintas y pondrá en marcha el servomotor que desbloqueará la apertura girando 90º.

Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

- Estado 4: Combinación incorrecta

En caso de que la combinación introducida sea incorrecta, el programa entrará en este estado. Mientras se ejecuta, el sistema producirá un sonido formado por una única frecuencia.

Una vez que ha terminado de ejecutarse, el programa vuelve al estado 2.

Por otro lado, se van a explicar brevemente las funciones más destacadas de este código. Las principales son:

- InicializaTimer1() e InicializaTimer2()

  En estas dos funciones, se inicializan los dos timers que se van a utilizar en el programa. En este caso, las interrupciones de los timers saltarán cada 1ms y 0.1ms, respectivamente.

- Servo_actua()

  En esta función es donde se gestiona el giro del servo. Se trata de un PWM en el que se fija un ancho de pulso igual a 2.5ms, que es el que produce un giro de 90º, y el periodo de la señal, que en el caso del servo ha de ser de 50 Hz.

- Estado2(), estado3() y estado4()

  En estas funciones se lleva a cabo lo correspondiente a cada uno de los estados, correspondiéndose con lo explicado al inicio de este apartado.

- Main()

  En esta función se gestiona la llamada a la función correspondiente al estado en que se encuentra el programa. Por ejemplo, en caso de encontrarse en el segundo estado, es aquí donde se llama a la función estado2() para que se ejecute. Además, es donde se declaran e inicializan los puertos y todas aquellas funciones que necesiten ser inicializadas.

- void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void) y void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void)

  Se trata de dos funciones que gestionan las interrupciones correspondientes a cada uno de los timers utilizados.

  En el primer caso, se realiza la actualización de los puertos de entrada para la gestión de las fotorresistencias LDR, así como las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para la reproducción de cada sonido por el altavoz.

  En el caso de la segunda interrupción, se gestionan las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para activar el servomotor.

4.4 PISTA SECUENCIA DE LUCES

En esta ocasión no existe máquina de estados del sistema. Por tanto, se procede a explicar brevemente las funciones más destacadas de este código. Las principales son:

- InicializaTimer2()

  En esta función, se inicializa el timer que se va a utilizar en el programa. En este caso, la interrupción del timer saltará cada 100 microsegundos.

- Main()

  En esta función se gestionan varias acciones. Por un lado, el ancho de pulso (100 microsegundos) y el periodo de la señal periódica (60 ms) a enviar por el pin de disparo correspondiente al sensor de ultrasonidos. Por

otro lado, las acciones a realizar en caso de que el sensor detecte a una persona a 25 cm o menos y, por último, las acciones a realizar en caso de no detectarse nada o hacerlo a más de dicha distancia.

En el caso en que el sensor detecta la presencia a una distancia adecuada, es en el interior de esta función donde se gestiona las señales a enviar al pin de conexión del altavoz. Dichas señales tienen diferente ancho de pulso (20ms y 500 ms, respectivamente) y periodos distintos (3s y 1s, respectivamente). Además, se encarga de dirigir la secuencia de señales a enviar a los leds.

En el caso en que no se cumpla la condición necesaria, se encarga de detener el envío de cualquier tipo de señal a los actuadores.

Además, es en el interior de esta función donde se declaran e inicializan los puertos entrada/salida y se inicializan las funciones que tengan que ser inicializadas

- void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void)

Se trata de la función que gestiona la interrupción correspondiente al timer utilizado.

Es aquí donde se realiza la actualización de los puertos de entrada para la recepción por el pin de *echo*[3] del sensor de ultrasonido, así como las cuentas necesarias para establecer el ancho de pulso y el periodo de la señal necesaria para la reproducción de cada sonido por el altavoz y de la señal que se envía por el pin de disparo del sensor de ultrasonido.

[3] Pin del sensor por el que se recibe el eco de la señal de ultrasonido enviada por el mismo.

Capítulo 5 RESULTADOS

En esta sección, se explicarán, con detalle, todos y cada uno de los experimentos realizados durante el desarrollo del proyecto, así como cada uno de los resultados obtenidos tras dichos experimentos.

5.1 DISEÑO DEL JUEGO

El diseño del juego, sus normas y los posibles recorridos de cada jugador, se han realizado de forma completa, quedando para el organizador del juego su contextualización en el marco de una historia así como algunas variantes sobre cómo dar las pistas de los distintos rompecabezas.

Uno de los objetivos de esta parte del proyecto se centraba en el diseño de sistemas portátiles, que pudieran ser trasladados de un lugar a otro con facilidad, así como instalables en lugares con distintas características: en el interior de una habitación cerrada, repartidos por el interior de un edificio de distintas proporciones, e incluso, en el exterior. Se podría afirmar que este objetivo ha sido cubierto puesto que los sistemas son totalmente independientes unos de otros y todos están pensados para poseer su propia fuente de alimentación, de manera que no tengan que estar conectados a la red durante su uso.

Otro de los objetivos acordados se correspondía con la sencillez de cada uno de los sistemas al mismo tiempo que suponían un reto para el jugador. En este caso, también se ha cubierto el objetivo ya que la dificultad de cada uno de los rompecabezas reside en la búsqueda de la solución del enigma y no en la manipulación de los componentes por parte del usuario.

Por último, se buscaba que existiese la posibilidad de ampliar fácilmente la duración del juego o el número de rompecabezas a enfrentar. Esto también se ha logrado ya que cualquier persona puede diseñar múltiples pruebas y su integración

en la totalidad del juego se resumiría en una reordenación de los posibles recorridos y de la información a encontrar tras superar cada reto.

5.2 DISEÑO E IMPRESIÓN DE LA TARJETA ELECTRÓNICA

En el caso de la tarjeta electrónica, ésta presentaba algunos pequeños fallos de diseño tras la impresión. Estos fallos se centraban principalmente en dos frentes.

En primer lugar, tras la impresión de la tarjeta, se descubrió la ausencia de pines para la conexión del programador del dsPIC. Se solucionó mediante un adaptador conectado a los pines del microprocesador que llevaba incorporado la conexión del programador. Así, se solucionaba el problema de la programación del micro pero, para ello, hubo que sacrificar el uso de dos de los interruptores de la tarjeta. Esto se debía a que el programador necesitaba utilizar los pines a los que se encontraban conectados sendos pulsadores y no había pines sobrantes en el micro.

El segundo frente se centraba en la alimentación de la tarjeta. Ésta se había diseñado para la alimentación mediante baterías pero, por falta de tiempo para su instalación, durante los experimentos se empleó una fuente de alimentación del laboratorio para proporcionar los 3.3V y 5V a la tarjeta.

Por tanto, en el momento de entrega de este proyecto, la tarjeta funcionaba correctamente a excepción de los dos pulsadores. Con suerte, estos dos pulsadores no fueron necesarios para la implementación de ninguna de las pruebas diseñadas y, por tanto, no supuso ningún atraso o perjuicio en la consecución del proyecto.

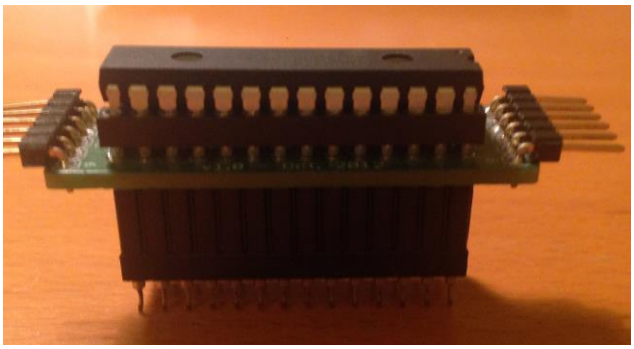


Figura 17. Adaptador usado para solucionar problema de programador

5.3 DISEÑO E IMPLEMENTACIÓN DEL PRIMER SISTEMA

En el caso de la primera prueba, consistente en un sistema de detección de una secuencia de colores, hubo que enfrentar dos problemas principalmente.

En primer lugar, a la hora del diseño del sistema, se pensó que cada color sería representado por un pulsador y con esa base se realizó el diseño de la tarjeta electrónica. Tras su impresión, para dotar al sistema de un atractivo mayor, se decidió que, en lugar de pulsadores, se usaría un led RGB que variaría su color en función de la posición de un potenciómetro. Como esto no había sido contemplado a la hora de diseñar la tarjeta, hubo que realizar pequeños ajustes para poder utilizar los pines de los leds de la tarjeta para el RGB. Finalmente, se consiguió que el led RGB funcionara como se esperaba.

En segundo lugar, cada uno de los estados por separado, funcionaban a la perfección pero a la hora de agruparlos para que el sistema funcionara de forma global, presentaba algunos fallos. La función del servomotor hacía que, efectivamente, éste girara 90º cuando se le ordenaba, al mismo tiempo que se reproducía una señal de dos frecuencias correspondiente al estado de combinación correcta (tercer estado). La combinación del led RGB y el potenciómetro, también funcionaban correctamente (acciones del segundo estado). Por último, las actuaciones correspondientes al estado de combinación incorrecta (cuarto estado), tampoco presentaban ningún problema. En cambio, en ocasiones, las transiciones entre estados sucedían antes de lo pretendido.

5.4 DISEÑO E IMPLEMENTACIÓN DEL SÉPTIMO SISTEMA

En el caso de la séptima prueba, consistente en la detección de una secuencia de letras, también hubo que enfrentar algunos problemas.

Por un lado, por falta de tiempo, no se comprobó el funcionamiento de la pantalla, por tanto, no hubo ocasión de probar si el código que se había programado era o no el correcto.

Por otro lado, al igual que en el caso anterior, las acciones correspondientes a los estados de combinación correcta e incorrecta (estados 6 y 7, respectivamente), funcionaban a la perfección.

5.5 DISEÑO E IMPLEMENTACIÓN DEL OCTAVO SISTEMA

En el caso de la octava prueba, consistente en la detección de una determinada secuencia de luces, también hubo que enfrentar algunos problemas.

En un principio, el sistema se había diseñado con el pensamiento de que, al iluminar las fotorresistencias en el orden preciso, se accionara el servomotor que bloqueaba el acceso a la información.

Sin embargo, al realizar el experimento correspondiente, se comprobó que la diferencia de la tensión que llegaba al pin del micro cuando incidía el haz de luz sobre una fotorresistencia era prácticamente la misma que cuando la luz que le llegaba era la natural. Esto suponía un problema puesto que la diferencia entre ambas tensiones era tan pequeña que el micro no era capaz de diferenciarlas y en ambos casos, el pin recibía un nivel alto.

En cambio, si en lugar de iluminar de cualquier manera la fotorresistencia, se tapaba la misma con un dedo, imposibilitando que cualquier foco de luz la alcanzara, el micro detectaba un flanco de bajada en el pin, es decir, pasaba a nivel bajo.

Esto se debe a un fallo en el diseño de la tarjeta electrónica que se aprovechó del proyecto de Ana Cogollos Baranda [7] a la hora de situar los componentes que correspondían a estos sensores. En la tarjeta, el esquema utilizado era el de la Figura 18.

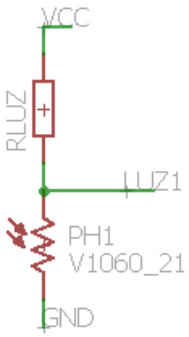


Figura 18. Esquema de conexión de la fotorresistencia

Las fotorresistencias aumentan el valor de su resistencia cuanto menor sea la luz que incida sobre ellas. Sabiendo esto y observando el esquema de la Figura 18, se puede concluir que, efectivamente, el pin se podrá a 0 cuando la resistencia del LDR sea muy grande y viceversa. La solución para esto sería intercambiar las posiciones de la alimentación (Vcc) y la tierra (GND).

El resultado final fue similar al de los sistemas anteriores. Los estados de combinación correcta e incorrecta funcionaban a la perfección (eso sí, como sistemas separados). La actividad del estado de introducir combinación, salvando la problemática de las fotorresistencias expresada anteriormente, también resultó exitosa. Al igual que en los casos anteriores, el sistema global daba problemas de transición de un estado a otro.

5.6 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE DETECCIÓN DE PROXIMIDAD

En el caso del sistema de detección de la presencia de una persona a una distancia igual o menor a 25 cm, el resultado fue un completo éxito.

En primer lugar, se comprobó que al enviar, por el pin de disparo del sensor de ultrasonido, una señal periódica de ancho de pulso 10 microsegundos y periodo 60 ms, efectivamente, se recibía una señal por el pin de *echo* cuyo ancho de pulso variaba en función de la distancia a la que se encontrase el obstáculo.

Tal y como se puede observar en las Figuras 19 y 20, a mayor distancia entre el sensor y el obstáculo, mayor es el ancho de pulso de la señal que se recibe. Esto es lógico ya que dicho ancho de pulso es proporcional al tiempo que tarda el ultrasonido en ir hasta el obstáculo, reflejarse y volver al sensor.

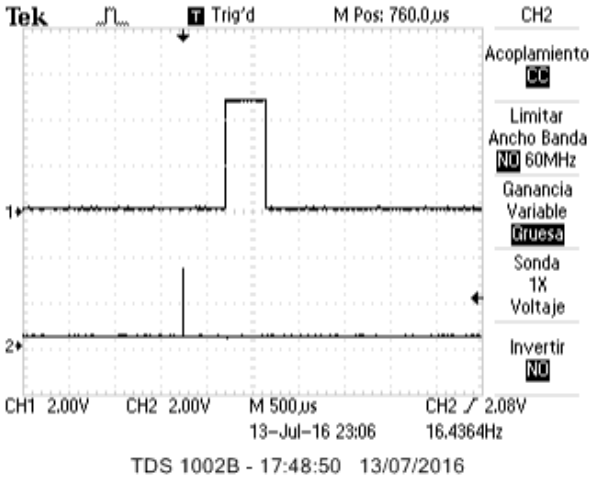


Figura 19. Pulso en el trigger y recepción por el pin echo con obstáculo cercano

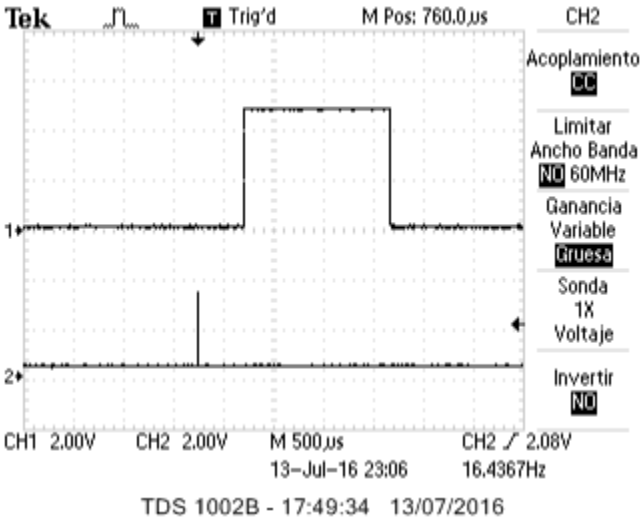


Figura 20. Pulso en el trigger y recepción por el pin echo con obstáculo lejano

Posteriormente, se calculó que para una distancia de 25 cm, el tiempo en que el pin de echo debía estar a nivel alto era de, como mucho, 1450 microsegundos la ida hasta el obstáculo y 1450 microsegundos, la vuelta al sensor.

Una vez que el sensor detectaba a una persona en el rango correcto de distancia, el resto de actuadores del sistema se activaban correctamente, así como detenían sus acciones cuando el sensor dejaba de detectar la presencia a la distancia correcta.

Capítulo 6 CONCLUSIONES

El proyecto no se ha completado en su totalidad, aún hay mucho margen de mejora y desarrollo en el mismo. Tal y como se ha explicado detalladamente en el Capítulo 5, sólo uno de los cuatros sistemas a desarrollar ha funcionado correctamente de forma completa.

Sin embargo, se podría considerar que el proyecto ha resultado exitoso ya que se han cubierto la mayoría de los objetivos propuesto. A saber:

- Diseño de un juego fácilmente transportable y atractivo para el usuario final, con gran versatilidad y facilidad para ampliar el número de retos a los que deberán enfrentarse los jugadores.

- Diseño de una tarjeta electrónica completa y comprobación de su correcto funcionamiento y,

- Diseño e implementación de cuatro sistemas que se correspondan con cuatro de los retos diseñados en el primer objetivo.

Capítulo 7 FUTUROS DESARROLLOS

Tal y como se ha ido diciendo a lo largo de todo el documento, los posibles desarrollos futuros son tan amplios como lo sea la imaginación de aquel que desee continuar desarrollando este proyecto.

Centrando estos desarrollos en los sistemas diseñados e implantados hasta el momento, las posibilidades también son muchas.

En primer lugar, podría modificarse el diseño de la tarjeta electrónica realizada incluyendo los pines adecuados para el programador. De esta forma, desaparecería la disyuntiva de mantener los pulsadores o poder programar el microprocesador. Aunque en este caso, la elección parece clara.

Por otro lado, en el caso del sistema de detección de proximidad, cabría la posibilidad de llevarlo a cabo con dos sensores de ultrasonido, colocados en distintos puntos del lugar donde se lleve a cabo el juego. Estos dos sensores habrían de detectar a una persona a determinada distancia de cada uno de ellos, activándose únicamente el resto del sistema en el caso en que dicha detección ocurriera de forma simultánea. Para ello, habría que incluir un módulo Wi-Fi en el sistema (esta opción de ha contemplado en el diseño de la tarjeta electrónica). La idea se basaría en usar una tarjeta en la que se instalaría todo el sistema tal y como se ha desarrollado en este proyecto y una segunda tarjeta en la que se instalaría el segundo sensor. De este modo, actuarían como tarjeta maestro y tarjeta esclavo. La tarjeta esclavo enviaría la información relativa al segundo sensor a la tarjeta maestro mediante Wi-Fi y sería ésta última la que gestionase el resto del sistema.

La prueba de los pulsadores remotos se diseñaría de una manera similar a lo anterior, con un pulsador en la tarjeta maestro y otro, en la tarjeta esclavo. En caso de que los pulsadores fueran activados al mismo tiempo, sería la tarjeta maestro la que activaría el servo que bloqueara el acceso a la información.

Por último, sería conveniente que se terminaran de depurar los sistemas con pequeños fallos y se implantaran con componentes más atractivos y llamativos para

el usuario final. Tal cómo lámparas de colores de mayor tamaño para sustituir a los leds, pulsadores más grandes y fáciles de manipular, etc.

Además, habría que montar las distintas cajas, cajones y puertas con algún sistema que permitiera a cada servomotor bloquear y desbloquear su apertura.

BIBLIOGRAFÍA

[1] **Microchip**, *Datasheet del microcontrolador dsPIC33FJ32MC202*. http://ww1.microchip.com/downloads/en/DeviceDoc/70283G.pdf

[2] **Futaba**, *Datasheet del servo s3003*. http://mech.vub.ac.be/teaching/info/mechatronica/finished_projects_2016/Wire_Guidance/images/et-servo-s3003.pdf

[3] **HC-SR04**, *Datasheet del sensor de ultrasonidos*. http://www.micropik.com/PDF/HCSR04.pdf

[4] **Midas**, *Datasheet de la pantalla LCD*. http://www.farnell.com/datasheets/2021773.pdf?_ga=1.11122996.1411361356.1467558870

[5] **Microchip**, *Datasheet del módulo Wi-Fi*. http://ww1.microchip.com/downloads/en/DeviceDoc/rn-131-ds-v3.2r.pdf

[6] **Diego Pérez de Diego**, *Sensores de distancia por ultrasonidos*. http://www.alcabot.com/alcabot/seminario2006/Trabajos/DiegoPerezDeDiego.pdf

[7] **Ana Cogollos Baranda**, *Diseño e implementación de un juego real para salir de un recinto. Proyecto Fin de Carrera, Escuela Técnica Superior de Ingeniería (ICAI)*. Universidad Pontificia Comillas.

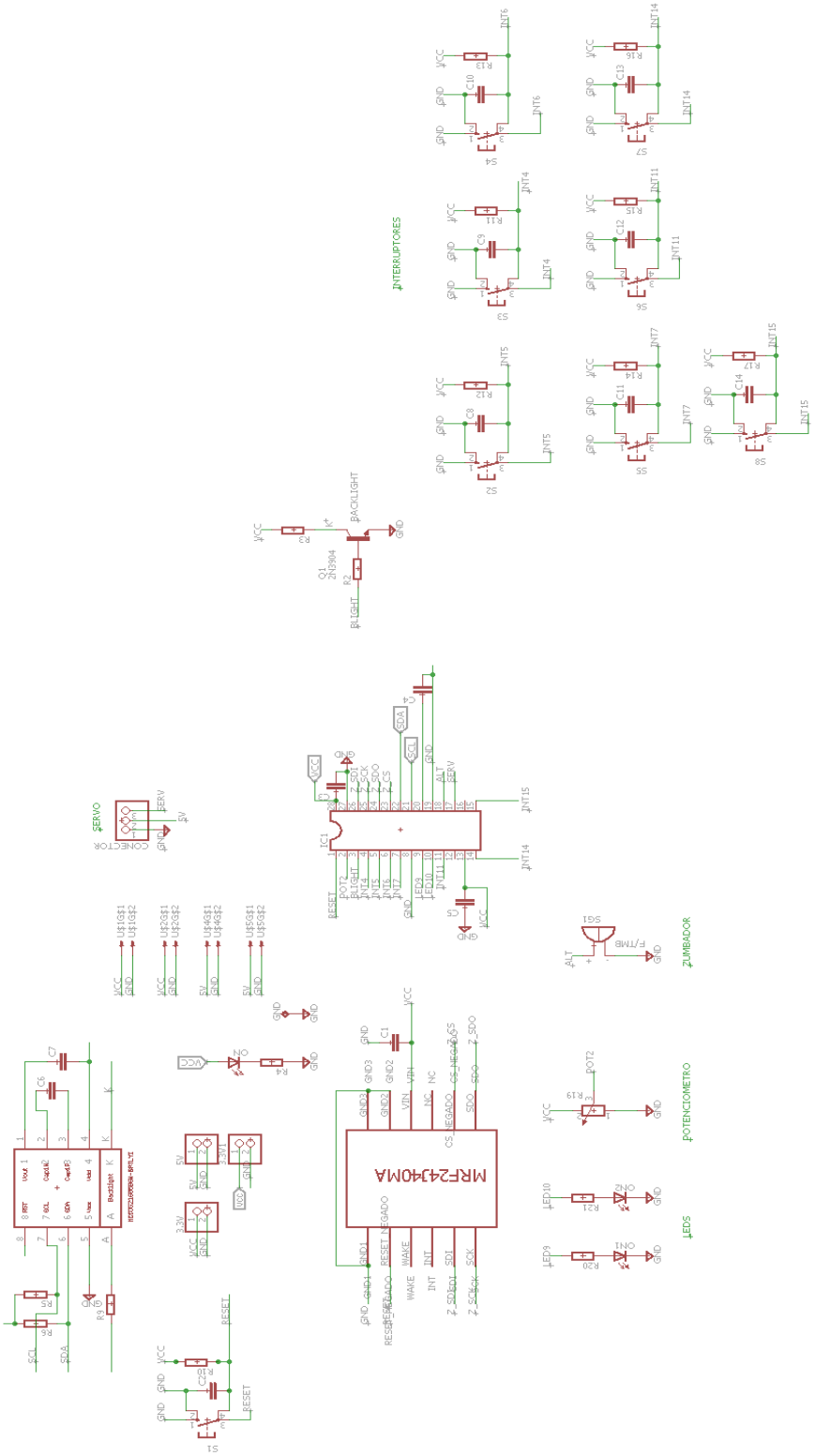
[8] **Ana Ruiz García**, *Gabriela. Proyecto Fin de Carrera, Escuela Técnica Superior de Ingeniería (ICAI)*. Universidad Pontificia Comillas.

[9] **Sánchez Miralles, Álvaro.** *Libro de Sistemas Electrónicos Digitales*. Universidad Pontificia Comillas.

[10] **Real Academia Española**. *Diccionario de la lengua española (22º ed.)* Madrid, España; 2001

Parte II PLANOS

Capítulo 1 CONEXIONADO PCB



Parte III PRESUPUESTO

ECONÓMICO

Capítulo 1 MEDICIONES

1.1 EQUIPO Y HERRAMIENTAS

Código	Descripción	Unidad
1.1.1	Instalación y licencia Eagle 7.5.0	1
1.1.2	Instalación y licencia de MPLAB X IDE	1
1.1.3	Instalación y licencia Microsoft Office	1
1.1.4	Instalación sensores de luminosidad y proximidad	5
1.1.6	Instalación servomotores	3
1.1.5	Unidad de ordenador	1

Tabla 1. Mediciones de equipo y herramientas

1.2 MANO DE OBRA DIRECTA

Código	Descripción	Unidad
1.2.1	Horas de programación	250
1.2.2	Horas de diseño de PCB	50
1.2.3	Horas de redacción de documentación	60
1.2.4	Horas de montaje	25
1.2.5	Horas de pruebas de funcionamiento de sensores y servomotores	20

Tabla 2. Mediciones mano de obra directa

1.3 CONSUMO ELÉCTRICO

Código	Descripción	Unidad
1.3.1	Consumo eléctrico en kWh Consumo suponiendo 405 horas de trabajo con una media de 300W por hora	121.5

Tabla 3. Mediciones consumo eléctrico

Capítulo 2 CUADRO DE PRECIOS Nº1

2.1 EQUIPO Y HERRAMIENTAS

Código	Unidad	Descripción	Importe en euros
2.1.1	Ud.	Instalación y licencia Eagle 7.5.0	5.00
2.1.2	Ud.	Instalación y licencia de MPLAB X IDE	7.00
2.1.3	Ud.	Instalación y licencia Microsoft Office	1.00
2.1.4	Ud.	Instalación sensores de luminosidad y proximidad	25.00
2.1.6	Ud.	Instalación servomotores	14.00
2.1.5	Ud.	Unidad de ordenador	925.00

Tabla 4. Cuadro de precios equipo y herramientas

2.2 MANO DE OBRA DIRECTA

Código	Unidad	Descripción	Importe en euros
2.2.1	Hora	Horas de programación	40.00
2.2.2	Hora	Horas de redacción de documentación	32.00
2.2.3	Hora	Horas de montaje	35.00

2.2.4	Hora	Horas de pruebas de funcionamiento de sensores y servomotores	40.00

Tabla 5. Cuadro de precios mano de obra directa

2.3 CONSUMO ELÉCTRICO

Código	Unidad	Descripción	Importe en euros
2.3.1	kWh	Consumo eléctrico en kWh Consumo suponiendo 405 horas de trabajo con una media de 300W por hora	0.20

Tabla 6. Cuadro de precios consumo eléctrico

Capítulo 3 CUADRO DE PRECIOS Nº2

3.1 EQUIPO Y HERRAMIENTAS

Código	Unidad	Descripción	Importe en euros
3.1.1	Ud.	Instalación y licencia Eagle 7.5.0	Mano de obra 5.00 Licencia 0.00 Total 5.00
3.1.2	Ud.	Instalación y licencia de MPLAB X IDE	Mano de obra 7.00 Licencia 0.00 Total 7.00
3.1.3	Ud.	Instalación y licencia Microsoft Office	Mano de obra 1.00 Licencia 0.00 Total 1.00
3.1.4	Ud.	Instalación sensores de luminosidad y proximidad	Mano de obra 6.00 Sensor 25.00 Total 31.00
3.1.6	Ud.	Instalación servomotores	Mano de obra 6.00 Servomotor 14.00 Total 20.00
3.1.5	Ud.	Unidad de ordenador	Precio ordenador 925.00 Total 925.00

Tabla 7. Cuadro de precios equipo y herramientas

3.2 MANO DE OBRA DIRECTA

Código	Unidad	Descripción	Importe en euros
3.2.1	Hora	Horas de programación	40.00
3.2.2	Hora	Horas de diseño PCB	40.00
3.2.3	Hora	Horas de redacción de documentación	32.00
3.2.4	Hora	Horas de montaje	35.00
3.2.5	Hora	Horas de pruebas de funcionamiento de sensores	40.00
3.2.6	Hora	Horas de pruebas de funcionamiento de servomotores	37.00

Tabla 8. Cuadro de precios mano de obra directa

3.3 CONSUMO ELÉCTRICO

Código	Unidad	Descripción	Importe en euros
3.3.1	kWh	Consumo eléctrico en kWh Consumo suponiendo 570 horas de trabajo con una media de 300W por hora	0.20

Tabla 9. Cuadro de precios consumo eléctrico

Capítulo 4 PRESUPUESTOS PARCIALES

4.1 EQUIPO Y HERRAMIENTAS

Código	Unidad	Descripción	Medición	Precio	Importe en euros
4.1.1	Ud.	Instalación y licencia Eagle 7.5.0	1	5.00	5.00
4.1.2	Ud.	Instalación y licencia de MPLAB X IDE	1	7.00	7.00
4.1.3	Ud.	Instalación y licencia Microsoft Office	1	1.00	1.00
4.1.4	Ud.	Instalación sensores de luminosidad y proximidad	5	31.00	155.00
4.1.5	Ud.	Instalación servomotores	3	20.00	60.00
4.1.6	Ud.	Unidad de ordenador	1	925.00	925.00

Tabla 10. Presupuestos parciales equipo y herramientas

COSTE TOTAL DE EQUIPO Y HERRAMIENTAS	1153.00 €

4.2 MANO DE OBRA DIRECTA

Código	Unidad	Descripción	Medición	Precio	Importe en euros
4.2.1	Hora	Horas de programación	250	40.00	10000.00
4.2.2	Hora	Horas de diseño PCB	50	40.00	2000.00
4.2.3	Hora	Horas de redacción de documentación	60	32.00	1920.00
2.3	Hora	Horas de montaje	25	35.00	875.00
2.4	Hora	Horas de pruebas de funcionamiento de sensores y servos	20	40.00	800.00

Tabla 11. Presupuestos parciales mano de obra directa

COSTE MANO DE OBRA DIRECTA	15595.00 €

4.3 CONSUMO ELÉCTRICO

Código	Unidad	Descripción	Med.	Precio	Importe €
3.1	kWh	Consumo eléctrico en kWh Consumo suponiendo 405 horas de trabajo con una media de 300W por hora	121.5	0.20	24.3

Tabla 12. Presupuestos parciales consumo eléctrico

Capítulo 5 PRESUPUESTO TOTAL

5.1 PRESUPUESTO TOTAL

Descripción	Importe en euros
Equipo y herramientas	1153.00
Mano de obra directa	15595.00
Consumo eléctrico	24.3

Tabla 13. Presupuesto total

PRESUPUESTO TOTAL (€)	16772.30

5.2 PRESUPUESTO DE LA EJECUCIÓN

Descripción	Importe en euros
Presupuesto total	16772.30
13% gastos generales	2180.40
8% beneficio industrial	1341.78

Tabla 14. Presupuesto de la ejecución

TOTAL	20294.48
21% IVA	4261.84
PRESUPUESTO TOTAL DE EJECUCIÓN (€)	24556.32

Parte IV CÓDIGO FUENTE

Capítulo 1 CÓDIGO FUENTE

SECUENCIA DE COLORES

1.1 MAIN.C

```
int cont=0;
int cont_error=0;
int pulsos=0;
int medidas;
int estado=1;
int portB_act, portB_ant, portA_ant, portA_act;
int DC_alt=0;
int periodo_alt=0;
int nota=1;
int DC_servo=0;
int periodo_servo=0;

void InicializaTimer1(void){

    TMR1=0;
    PR1=40000;
    T1CON=0x8000; //cada 1ms

    IEC0bits.T1IE=1; //se habilitan interrupciones
    T1CONbits.TON=1; //se activa timer1
    IFS0bits.T1IF=0; //limpia el flag
}

void InicializaTimer2(void){

    TMR2=0;
    PR2=4000;
    T2CON=0x8000; //cada 0.1ms

    IEC0bits.T2IE=1; //se habilitan interrupciones
    T2CONbits.TON=1; //se activa timer1
    IFS0bits.T2IF=0; //limpia el flag

}

void Init_AD(int canal){

    AD1PCFGL = 0;
    TRISB |= 0x0F;
    TRISA |= 3;

    AD1CHS0 = 0;
```

```
    AD1CON3 = 0x109;
    AD1CON1 = 0x80E0;
    IFS0 &= ~ 0x2000;
    IEC0 |= 0x2000;
    AD1CON1 |=2;

}

int get_medida (int canal){

//    AD1CHS0 = canal;
    return medidas;
}


int pulsado(){

    if (portB_ant != portB_act && portB_ant == 0)
        return 1;

    return 0;
}


int servo_actua(){

    int PER_abierto=25; //ancho de pulso para 90 grados
    int duracion_servo=200; //periodo de la señal (50Hz=20ms)


    if(DC_servo<PER_abierto)
        PORTB |= ~0xFEFF; //Ponemos a 1 el RB8
    else
        PORTB &= 0xFEFF; //Ponemos a 0 el RB8

    if (DC_servo >= PER_abierto)
        DC_servo = 0;

    if (periodo_servo > duracion_servo)
        periodo_servo = 0;

    return 0;
}

int estado2 (){

    int i;
    int cont_ant;
    int resultado, suma;
    int v_medidas[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        for(i=0;i<7;i++)
            v_medidas[i]=v_medidas[i+1];

        resultado = (get_medida(0)>>7);
        v_medidas[7]= resultado;
        suma =0;

        for(i=0;i<8;i++)
            suma= suma+v_medidas[i];

        resultado=suma>>3;
```

```
        PORTA = (resultado<<2) & 0x001C; //actualizo el PORTA

        //los colores estan asociados a numeros del 0 al 7

        if(cont_error==0){

            if(pulsado()  &&  resultado  ==  3  &&  cont==0){  //color
correspondiente al 011 (RB4 off, RB3 y RB2 on)

                cont_ant=cont;
                cont++;
            }

            else if (pulsado() && resultado == 1 && cont==1){ //color 001
(RB4 y RB3 off y RB2 on)

                cont_ant=cont;
                cont++;
            }

            else if (pulsado() && resultado == 7 && cont==2){ //color 111
(todos on)
                cont_ant=cont;
                cont++;
            }
            else if (pulsado() && resultado == 5 && cont==3){ //color 101
(RB4 y RB2 on y RB3 off)
                cont_ant=cont;
                cont++;
            }
            else cont_error++;
        }
        //puedo poner el orden que quiera y repetido las veces que quiera
        else if(cont_error!=0 && pulsado()){
            cont_error++;
            pulsos=cont_error + cont;

        }


        if(cont==4){
            estado =3;
            cont=0;
        }
        else if (pulsos==4) {
            estado = 4;
            cont_ant=0;
            cont=0;
            cont_error=0;
            pulsos=0;
        }

    return 0;
}



int estado3(){

    unsigned int PER_correcto[2] = {20, 500};
    unsigned int duracion_correcto[2] ={3000,1000};
```

```
    //CONTROL ALTAVOZ

    if(DC_alt<PER_correcto[nota]/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)

    if (DC_alt >= PER_correcto[nota])
        DC_alt = 0;

    if (periodo_alt > duracion_correcto[nota]){
        periodo_alt = 0;
        nota++;
        if(nota>2) {
            nota=1;
            estado=2;
        }
    }

    //CONTROL SERVO
    servo_actua();

    return 0;
}

int estado4(){

    unsigned int PER_error = 3;
    unsigned int duracion_error = 2000;

    //CONTROL ALTAVOZ

    if(DC_alt<PER_error/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)


    if (DC_alt >= PER_error)
        DC_alt = 0;

    if (periodo_alt > duracion_error){
        periodo_alt = 0;
        estado =2;
    }
    return 0;

}



int main (void) {

     //servo: pin 17(RB8); potenciometro: pin 2(RA0); altavoz: pin 18
(RB9); LED tricolor: pines 9, 10 y 12 (RA2, RA3, RA4); interruptor: pin
4(RB0)

    //estado 1: reset
    //estado 2: combinacion
    //estado 3: correcto + activo servo + alt correcto
    //estado 4: incorrecto + alt error
    ADPCFG=0xFFFF;
```

```
    TRISA=0xFFE3;
    PORTA=0;
    TRISB=0xFCFF;
    PORTB=0;
    InicializaTimer1(); //1 ms
    InicializaTimer2(); //0.5 ms
    Init_AD(0x01); //canal 0

    while(1){


        if (estado==1)
            estado = 2;

        if (estado==2)
            estado2();

        if (estado==3)
            estado3();

        if (estado ==4)
            estado4();


    }



}


void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void){

    IFS0bits.T1IF=0; //limpia el flag

    portB_ant = portB_act;
    portB_act = PORTB & 1; //actualiza el port_actB
    portA_ant = portA_act;
    portA_act = PORTA & 0x03;
   if (estado==3 || estado == 4){
       DC_alt++;
       periodo_alt++;
   }

}

void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void){

    IFS0bits.T2IF=0; //limpia el flag
    if (estado == 3){
        periodo_servo++;
        DC_servo++;
    }

}

void __attribute__((interrupt,no_auto_psv)) _ADC1Interrupt(void) {

    IFS0 &= ~0x2000;
    medidas = ADC1BUF0;
    AD1CON1 |=2;
}
```

Capítulo 2 CÓDIGO FUENTE SECUENCIA DE LETRAS

2.1 MAIN.C

```
char Letra_uno=0;
char Letra_dos=0;
char Letra_tres=0;
char Letra_cuatro=0;
char mensaje[4];
int cont_letra=0;
int estado=1;
int portB_act, portB_ant;
int DC_alt=0;
int periodo_alt=0;
int nota=1;
int DC_servo=0;
int periodo_servo=0;

void InicializaTimer1(void){

    TMR1=0;
    PR1=40000;
    T1CON=0x8000; //cada 1ms

    IEC0bits.T1IE=1; //se habilitan interrupciones
    T1CONbits.TON=1; //se activa timer1
    IFS0bits.T1IF=0; //limpia el flag
}

void InicializaTimer2(void){

    TMR2=0;
    PR2=4000;
    T2CON=0x8000; //cada 0.1ms

    IEC0bits.T2IE=1; //se habilitan interrupciones
    T2CONbits.TON=1; //se activa timer1
    IFS0bits.T2IF=0; //limpia el flag

}

int pulsado_letra(){

    if (portB_ant!= portB_act && portB_act == 1) //RB0
        return 1;

    return 0;
}

int pulsado_ok(){
```

```
    if (portB_ant!=portB_act && portB_act==2) //RB1
        return 1;

    return 0;
}

int servo_actua(){

    int PER_abierto=25; //ancho de pulso para 90 grados
    int duracion_servo=200; //periodo de la señal (50Hz=20ms)


    if(DC_servo<PER_abierto)
        PORTB |= ~0xFEFF; //Ponemos a 1 el RB8
    else
        PORTB &= 0xFEFF; //Ponemos a 0 el RB8

    if (DC_servo >= PER_abierto)
        DC_servo = 0;

    if (periodo_servo > duracion_servo)
        periodo_servo = 0;

    return 0;
}

int estado2(){
    Letra_dos=0;
    Letra_tres=0;
    Letra_cuatro=0;


    if(pulsado_letra() && cont_letra==0){
        Letra_uno='A';
        cont_letra++;
    }
   if(pulsado_letra() && cont_letra==1){
       Letra_uno='B';
       cont_letra++;
    }
    if(pulsado_letra() && cont_letra==2){
        Letra_uno='C';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==3){
        Letra_uno='D';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==4){
        Letra_uno='E';
        cont_letra++;
    }
    //todas las letras que quiera
    mensaje[0]=Letra_uno;
    mensaje[1]=Letra_dos;
    mensaje[2]=Letra_tres;
    mensaje[3]=Letra_cuatro;
    DisplaySetLine(0);
    DisplayPuts(mensaje);

    if(pulsado_ok()){
        cont_letra=0;
        estado=3;
```

```
    }
    return 0;
}

int estado3(){

    Letra_tres=0;
    Letra_cuatro=0;

    if(pulsado_letra() && cont_letra==0){
        Letra_dos='A';
        cont_letra++;
    }
   if(pulsado_letra() && cont_letra==1){
       Letra_dos='B';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==2){
        Letra_dos='C';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==3){
        Letra_dos='D';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==4){
        Letra_dos='E';
        cont_letra++;
    }
    //todas las letras que quiera
    mensaje[0]=Letra_uno;
    mensaje[1]=Letra_dos;
    mensaje[2]=Letra_tres;
    mensaje[3]=Letra_cuatro;
    DisplaySetLine(0);
    DisplayPuts(mensaje);

    if(pulsado_ok()){
        cont_letra=0;
        estado=4;
    }
    return 0;
}

int estado4(){

    Letra_cuatro=0;

    if(pulsado_letra() && cont_letra==0){
        Letra_tres='A';
        cont_letra++;
    }
   if(pulsado_letra() && cont_letra==1){
       Letra_tres='B';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==2){
        Letra_tres='C';
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==3){
        Letra_tres='D';
        cont_letra++;
```

```
    }
    if(pulsado_letra() && cont_letra==4){
        Letra_tres='E';
        cont_letra++;
    }
    //todas las letras que quiera

    mensaje[0]=Letra_uno;
    mensaje[1]=Letra_dos;
    mensaje[2]=Letra_tres;
    mensaje[3]=Letra_cuatro;
    DisplaySetLine(0);
    DisplayPuts(mensaje);

    if(pulsado_ok()){
        cont_letra=0;
        estado=5;
    }
    return 0;
}

int estado5(){


    if(pulsado_letra() && cont_letra==0){
        Letra_cuatro='A';
        DisplaySetLine(0);
        DisplayPuts(mensaje);
        cont_letra++;
    }
   if(pulsado_letra() && cont_letra==1){
       Letra_cuatro='B';
        DisplaySetLine(0);
        DisplayPuts(mensaje);
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==2){
        Letra_cuatro='C';
        DisplaySetLine(0);
        DisplayPuts(mensaje);
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==3){
        Letra_cuatro='D';
        DisplaySetLine(0);
        DisplayPuts(mensaje);
        cont_letra++;
    }
    if(pulsado_letra() && cont_letra==4){
        Letra_cuatro='E';
        DisplaySetLine(0);
        DisplayPuts(mensaje);
        cont_letra++;
    }
    //todas las letras que quiera

    if(pulsado_ok()){
        if(Letra_uno=='A'   &&   Letra_dos=='B'   &&   Letra_tres=='C'   &&
Letra_cuatro =='D')
            cont_letra=0;
            estado=6;
    }else{
            cont_letra=0;
```

```
            estado=7;
    }

    return 0;
}

int estado6(){

    unsigned int PER_correcto[2] = {20, 500};
    unsigned int duracion_correcto[2] ={3000,1000};


    //CONTROL ALTAVOZ

    if(DC_alt<PER_correcto[nota]/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)

    if (DC_alt >= PER_correcto[nota])
        DC_alt = 0;

    if (periodo_alt > duracion_correcto[nota]){
        periodo_alt = 0;
        nota++;
        if(nota>2) {
            nota=1;
            estado=1;
        }
    }

    //CONTROL SERVO
    servo_actua();

    return 0;
}

int estado7(){

    unsigned int PER_error = 3;
    unsigned int duracion_error = 2000;

    //CONTROL ALTAVOZ

    if(DC_alt<PER_error/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)


    if (DC_alt >= PER_error)
        DC_alt = 0;

    if (periodo_alt > duracion_error){
        periodo_alt = 0;
        estado =2;
    }
    return 0;

}
```

```
int main(void){

   InicializarReloj();
   InitDisplay();
   ADPCFG=0xFFFF;

   //servo: pin 17(RB8); display LCD: pines 21 y 22 (RB10 y RB11); altavoz:
pin 18 (RB9); pulsador cambio letra: pin 4(RB0), pulsador ok letra:pin 5
(RB1);
   TRISB=0xFFF0;
   PORTB=0;


    //Estado 1: reset
    //Estado 2: primera letra
    //Estado 3: segunda letra
    //Estado 4: tercera letra
    //Estado 5: cuarta letra
    //Estado 6: correcto + servo
    //Estado 7: incorrecto

       while(1){

         if (estado == 1)
             estado = 2;

         if (estado == 2)
             estado2();

         if (estado == 3)
             estado3();

         if (estado == 4)
             estado4();

         if (estado == 5)
             estado5();

         if (estado == 6)
             estado6();

         if (estado == 7)
             estado7();


    }
         return 0;

}


void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void){

    IFS0bits.T1IF=0; //limpia el flag

    portB_ant = portB_act;
    portB_act = PORTB & 3; //actualiza el port_actB
   if (estado==3 || estado == 4){
       DC_alt++;
       periodo_alt++;
   }

}
```

```
void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void){

    IFS0bits.T2IF=0; //limpia el flag
    if (estado == 3){
        periodo_servo++;
        DC_servo++;
    }

}
```

2.2 MIDASDISPLAYDRIVER.C

```
#include "MidasDisplayDriver.h"
#include <p33FJ32MC202.h>

// El baud rate generator se calcula según la fórmula:
// I2CBRG = ((1/frec_i2c-130ns)*Fcy)-2

#define BAUD_400 91  // Baud rate generator a 400 kHz (reloj Fcy = 39,61375
MHz)
#define BAUD_100 289 // Baud rate generator a 100 kHz (reloj Fcy = 39,61375
MHz)

// Funciones privadas
static void Retardo(unsigned int ms);
static void I2CEsperaIdle(void);
static void I2CEnviaStart(void);
static void I2CEnviaStop(void);
static int I2CEnviaByte(unsigned char byte_env);

/* Función: InitDisplay
 *
 * Inicializa el bus I2C del dsPIC33FJMC202 y envía los comandos de
 * inicialización del display.
 * ATENCIÓN: Esta función ha de llamarse antes de configurar los puertos,
ya que
 * si al configurarlos se genera un pulso espúreo en las líneas del I2C,
el display
 * no interpretará los comandos de esta función correctamente.
 *
 * Retorna: 0 si display inicializado OK, 1 si el display envía NACK durante
 *          la inicialización.
 */
int InitDisplay(void)
{
    int ret;
    /* Primero se inicializa el periférico I2C */
    I2C1BRG = BAUD_100; // Se inicializa el Baud Rate generator
    I2C1CON = 0x1200; /* SCLREL (bit 12) tiene que estar a 1, pues de lo
                       * contrario mantiene SCL = 0 (clock stretch).
                       * DISSLW (bit 9) se pone a 1 para inhabilitar el
control
                       * de slew-rate
                       */
    I2C1RCV = 0x0000; // Se inicializan los registros del transmisor y
receptor
```

```
    I2C1TRN = 0x0000;
    I2C1CON = 0x9200; // I2CEN (bit 15) a 1 para habilitar el I2C

    /* Según el manual hay que esperar al menos 40 ms después del reset
       para que se estabilice el display. Espero 50 por si acaso */
    Retardo(500);

    /* Ahora se envían los comandos de inicialización al display MIDAS */

       I2CEsperaIdle();
       I2CEnviaStart();
    ret = I2CEnviaByte(0x7C); // dirección
    //Retardo(1);
    ret = I2CEnviaByte(0x00); // Se va a enviar un comando al display
    //Retardo(1);
    ret = I2CEnviaByte(0x39); // Function Set: Interface 8 bits, 2 líneas,
                              // Extended Instruction Set
    //Retardo(1);
    ret = I2CEnviaByte(0x14);
    //Retardo(1);
    ret = I2CEnviaByte(0x79);
    //Retardo(1);
    ret = I2CEnviaByte(0x50);
    //Retardo(1);
    ret = I2CEnviaByte(0x6C);
    //Retardo(1);
    ret = I2CEnviaByte(0x0C);
    //Retardo(1);
    ret = I2CEnviaByte(0x01);
    I2CEnviaStop();
    Retardo(30); // Para que le de tiempo a borrarse al display

    /* Se imprime el mensaje de bienvenida */

    DisplayPuts(" Laboratorio de");
    DisplaySetLine(1);
    DisplayPuts(" Micros ICAIdea");
    DisplaySetLine(0); // Se deja el cursor en el inicio}

    return ret;

}
/* Función: I2CEsperaIdle
 *
 * Espera en un bucle sin fin a que el bus I2C esté inactivo
 *
 */

static void I2CEsperaIdle(void)
{
    while(I2C1STATbits.TRSTAT) // Bit a 0 cuando no hay transmisión en el
bus
        ;
}

/* Función: I2CEnviaStart
 *
 * Envía el bit de start al bus. La función se bloquea hasta que termine
dicho
 * bit de start.
 *
 */
```

```
static void I2CEnviaStart(void)
{
    I2C1CONbits.SEN = 1;    // Genera la condición de start

    while(I2C1CONbits.SEN) // El HW borra este bit cuando finaliza el envío
de
        ;                    // la condición de start.
}

/* Función: I2CEnviaStop
 *
 * Envía el bit de stop al bus. La función se bloquea hasta que termine
dicho
 * bit de stop.
 *
 */
static void I2CEnviaStop(void)
{
    I2C1CONbits.PEN = 1;    // Inicia la condición de stop en el bus
    while(I2C1CONbits.PEN) // El HW borra este bit cuando finaliza el envío
de
        ;                  // la condición de stop.
}

/* Función: I2CEnviaByte
 *
 * Envía un byte por el bus. La función se bloquea hasta que termine dicho
 * envío y devuelve 0 si el esclavo ha respondido con un ACK o 1 si el
esclavo
 * ha respondido con un NACK a la transmisión del byte.
 *
 */

static int I2CEnviaByte(unsigned char byte_env)
{
    I2C1TRN = byte_env; // Comienza el envío.
    I2CEsperaIdle();
    return I2C1STATbits.ACKSTAT; // Devuelve el ACK/NACK retornado por el
esclavo
}
/* Función: Retardo
 *
 * Genera un retardo de n milisegundos usando el timer 1 en modo polling.
 *
 * Argumentos:
 *   unsigned int n_ms; número de milisegundos a esperar
 *
 */

static void Retardo(unsigned int n_ms)
{
    int pr1_bak, t1con_bak;

    // Guardo el estado del timer 1.
    pr1_bak = PR1;
    t1con_bak = T1CON;

    // Inicializo el timer
    TMR1 = 0;
    PR1 = 39614; // Timer a 1 milisegundo. Como estamos a 39.61375 MHZ,
hay que
                 //contar 39614
    IFS0bits.T1IF = 0; // Borra el flag
```

```
    T1CON = 0x8000; // Timer on, Prescaler = 0 while(IFS0bits.T1IF == 0)
    while(n_ms != 0){
        while(IFS0bits.T1IF == 0)
            ; // Espera fin del timer
        IFS0bits.T1IF = 0; // Borra el flag
        n_ms--;
    }

    // Dejamos el timer 1 como estaba
    T1CON = t1con_bak;
    PR1 = pr1_bak;
}
/* Función: DisplayPuts
 *
 * Envía una cadena de caracteres (terminada con \0) al display.
 *
 * Argumentos:
 *   char *pcad: Cadena a enviar.
 *
 */

void DisplayPuts(char *pcad)
{
    int ret;

    I2CEsperaIdle();
    I2CEnviaStart();
    ret = I2CEnviaByte(0x7C); // dirección
    ret = I2CEnviaByte(0x40); // Se van a enviar datos al display
    while(*pcad != '\0'){
      ret = I2CEnviaByte(*pcad);
      pcad++;
    }
    I2CEnviaStop();
}

/* Función: DisplaySetLine
 *
 * Coloca el cursor al principio de la línea cuyo número se pasa
 * como argumento
 *
 * Argumentos:
 *   int linea: Número de la línea en la que se desea colocar el cursor.
 *              Si vale 0 se pone el cursor en la línea 0. Cualquier otro
valor
 *              lo pone en la línea 1.
 */

void DisplaySetLine(int linea)
{
    I2CEsperaIdle();
    I2CEnviaStart();
    I2CEnviaByte(0x7C); // dirección
    I2CEnviaByte(0x00); // Va un comando a continuación
    if(linea == 0)
      I2CEnviaByte(0x80); /* Pone la dirección de memoria en 0, que es
donde empieza la línea 0 */
    else{
      I2CEnviaByte(0xC0); /* Pone la dirección de memoria en 0x40, que es
                             donde empieza la línea 1 */
    }
    I2CEnviaStop();
}
```

Capítulo 3 CÓDIGO FUENTE SECUENCIA DE LUCES

3.1 MAIN.C

```
int cont=0;
int cont_ant=0;
int cont_error=0;
int pulsos=0;
int estado=1;
int portB_act, portB_ant, portA_ant, portA_act;
int DC_alt=0;
int periodo_alt=0;
int nota=1;
int DC_servo=0;
int periodo_servo=0;

void InicializaTimer1(void){

    TMR1=0;
    PR1=40000;
    T1CON=0x8000; //cada 1ms

    IEC0bits.T1IE=1; //se habilitan interrupciones
    T1CONbits.TON=1; //se activa timer1
    IFS0bits.T1IF=0; //limpia el flag
}

void InicializaTimer2(void){

    TMR2=0;
    PR2=4000;
    T2CON=0x8000; //cada 0.1ms

    IEC0bits.T2IE=1; //se habilitan interrupciones
    T2CONbits.TON=1; //se activa timer1
    IFS0bits.T2IF=0; //limpia el flag

}



int servo_actua(){

    int PER_abierto=25; //ancho de pulso para 90 grados
    int duracion_servo=200; //periodo de la señal (50Hz=20ms)
```

```
    if(DC_servo<PER_abierto)
        PORTB |= ~0xFEFF; //Ponemos a 1 el RB8
    else
        PORTB &= 0xFEFF; //Ponemos a 0 el RB8

    if (DC_servo >= PER_abierto)
        DC_servo = 0;

    if (periodo_servo > duracion_servo)
        periodo_servo = 0;

    return 0;
}

int estado2 (){

    if(cont_error==0){
        if(portB_ant!=portB_act  && portB_act  ==  2  && cont==0){  //esta
pulsado el RB1 (segundo sensor)
            cont_ant=cont;
            cont++;
        }
        else if (portA_ant!=portA_act && portA_act == 1 && cont==1){ //se
pulsa el RA0 (tercer sensor)
            cont_ant=cont;
            cont++;
        }
        else if (portB_ant!=portB_act && portB_act == 1 && cont==2){ //se
pulsa el RB0 (primer sensor)
            cont_ant=cont;
            cont++;
        }
        else if (portA_ant!=portA_act && portA_act == 2 && cont==3){ //se
pulsa el RA1 (cuarto sensor)
            cont_ant=cont;
            cont++;
        }
        //puedo poner el orden que quiera y repetido las veces que quiera
        else cont_error++;
    }
    else if (cont_error!=0){


        if(portA_ant!=portA_act && portA_act!=0){
            cont_error++;
            pulsos=cont_error + cont;
        }
        else if(portB_ant!=portB_act && portB_act!=0){
            cont_error++;
            pulsos=cont_error + cont;
        }
    }

    if(cont==4){
        estado =3;
        cont=0;
    }
    else if (pulsos==4) {
        estado =4;
        cont_ant=0;
        cont=0;
        cont_error=0;
```

```
        pulsos=0;
    }

    return 0;
}



int estado3(){

    unsigned int PER_correcto[2] = {20, 500};
    unsigned int duracion_correcto[2] ={3000,1000};


    //CONTROL ALTAVOZ

    if(DC_alt<PER_correcto[nota]/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)

    if (DC_alt >= PER_correcto[nota])
        DC_alt = 0;

    if (periodo_alt > duracion_correcto[nota]){
        periodo_alt = 0;
        nota++;
        if(nota>2) {
            nota=1;
            estado=1;
        }
    }

    //CONTROL SERVO
    servo_actua();

    return 0;
}

int estado4(){

    unsigned int PER_error = 3;
    unsigned int duracion_error = 2000;

    //CONTROL ALTAVOZ

    if(DC_alt<PER_error/2)
        PORTB |= ~0xFDFF; //Ponemos a 1 el bit 18(RB9)
    else
        PORTB &= 0xFDFF; //Ponemos a 0 el bit 18 (RB9)


    if (DC_alt >= PER_error)
        DC_alt = 0;

    if (periodo_alt > duracion_error){
        periodo_alt = 0;
        estado =2;
    }
    return 0;

}
```

```
int main (void) {

     //servo: pin 17(RB8); sensores luz: RB0, RB1, RA0, RA1; altavoz: pin
18 (RB9)

    //estado 1: reset
    //estado 2: pulsando combinacion
    //estado 3: correcto + activo servo + alt correcto
    //estado 4: incorrecto + alt error
    ADPCFG=0xFFFF;
    TRISA=0xFF;
    TRISB=0xFCFF;
    PORTB=0;
    PORTA=0;
    InicializaTimer1(); //1 ms
    InicializaTimer2(); //0.5 ms


    while(1){


        if (estado==1)
            estado = 2;

        if (estado==2)
            estado2();

        if (estado==3)
            estado3();

        if (estado ==4)
            estado4();

    }
}


void __attribute__ ((interrupt, no_auto_psv)) _T1Interrupt(void){

    IFS0bits.T1IF=0; //limpia el flag
    portB_ant = portB_act;
    portB_act = PORTB & 0x0003; //actualiza el port_actB
    portA_ant = portA_act;
    portA_act = PORTA & 0x03;
   if (estado==3 || estado == 4){
       DC_alt++;
       periodo_alt++;
   }

}

void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void){

    IFS0bits.T2IF=0; //limpia el flag
    if (estado == 3){
        periodo_servo++;
        DC_servo++;
    }

}
```

Capítulo 4 CÓDIGO FUENTE DETECTOR DE PROXIMIDAD

4.1 MAIN.C

```
int medida  = 0;
int sensor=0;
int ticks=0;
int cont=0;
int DC_alt=0;
int periodo_alt=0;
int nota=0;
int periodo_sens=0;
int DC_sens=0;
int cont_detecta=0;


void InicializaTimer2(void){

    TMR2=0;
    PR2=4000;
    T2CON=0x8000; //cada 100us

    IEC0bits.T2IE=1; //se habilitan interrupciones
    T2CONbits.TON=1; //se activa timer1
    IFS0bits.T2IF=0; //limpia el flag
}


int main(void){

    //sensor: pines 21(R) y 22(T)(RB7 y RB8); leds: pines 6,7,11 y 14
(RB15, RB14, RB13, RB12); zumbador: pin 18 (RB2)

    unsigned int PER_correcto[2] = {200, 5000};
    unsigned int duracion_correcto[2] ={30000,10000};
   unsigned int PER_sens=600;


        InicializarReloj();
        RemapeaPerifericos();
        InicializaTimer2();
        ADPCFG=0xFFFF;
        TRISB=0x0F7B;
        PORTB=0xF000;




       while(1){
```

```
            if(DC_sens==0)
                PORTB |= 0x0080; //Pongo a 1 el trig
            else
                PORTB &= ~0x0080; //pongo a 0 el trig

            if (DC_sens >= PER_sens)
                DC_sens = 0;


//            PORTB = (medida/58) << 12;

            if(medida<=28){


            if(DC_alt<PER_correcto[nota]/2)
                PORTB |= ~0xFFFB; //Ponemos a 1 el bit 18(RB2)
            else
                PORTB &= 0xFFFB; //Ponemos a 0 el bit 18 (RB2)

            if (DC_alt >= PER_correcto[nota])
                DC_alt = 0;

            if (periodo_alt > duracion_correcto[nota]){
                periodo_alt = 0;
                nota++;
            if(nota>2) {
                nota=1;

                }
            }

            if (ticks<10000 && cont==0){
                PORTB&= ~0x4000;             //se enciende el 2° led
                PORTB|=0x1000;
            }
            else if (ticks<10000 && cont==1){
                PORTB&= ~0x2000;        //se enciende el 3° led
                PORTB|=0x4000;
            }
            else if (ticks<10000 && cont==2){
                PORTB&= ~0x8000;             //se enciende el 1° led
                PORTB|=0x2000;
            }
            else if (ticks<10000 && cont==3){
                PORTB&= ~0x1000;        //se enciende el 4° led
                PORTB|= 0x8000;
            }
         }
        else{

         PORTB = 0xF00B; //apago leds y altavoz
         ticks=0;
         cont=0;
         DC_alt=0;
         periodo_alt=0;
    }

    }

    return 0;

}
```

```
void __attribute__ ((interrupt, no_auto_psv)) _T2Interrupt(void){

    IFS0bits.T2IF=0; //limpia el flag

    DC_sens++;

  if(medida<=28){ //1450 us son 25 cm
       DC_alt++;
       periodo_alt++;
       ticks++;
       if (ticks>=10000){
           ticks=0;
           cont++;
        }
       if (cont>=4)
           cont=0;

    }

    int echo_ant=0;
    int echo_act;



    echo_ant=echo_act;
    echo_act=PORTB&0x40;

    if(echo_ant!=echo_act && echo_ant==0)
        cont_detecta=0;
    cont_detecta++;
   if(echo_ant!=echo_act && echo_act==0)
       medida = cont_detecta;

}
```