



Universitat Oberta  
de Catalunya

# Diseño e implementación de alarma antirrobo independiente, fiable y económica.

**Autor: Miguel Ángel Rosales Navarro.**

Máster Interuniversitario en Ingeniería de Telecomunicación.

Aplicaciones electrónicas.

**Consultor: Aleix López Antón.**

**Profesor responsable de la asignatura: Carlos Monzo Sánchez.**

23/06/2016



Esta obra está sujeta a una licencia de  
Reconocimiento-NoComercial-  
SinObraDerivada [3.0 España de Creative  
Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	Diseño e implementación de alarma antirrobo independiente, fiable y económica.
<b>Nombre del autor:</b>	<i>Miguel Ángel Rosales Navarro</i>
<b>Nombre del consultor/a:</b>	<i>Aleix López Antón</i>
<b>Nombre del PRA:</b>	<i>Carlos Monzo Sánchez</i>
<b>Fecha de entrega:</b>	<i>23/06/2016</i>
<b>Titulación:</b>	<i>Máster Interuniversitario en Ingeniería de Telecomunicación.</i>
<b>Área del Trabajo Final:</b>	<i>Aplicaciones electrónicas.</i>
<b>Idioma del trabajo:</b>	<i>Español</i>
<b>Palabras clave</b>	<i>Antirrobo, económico, STM32F4.</i>

### **Resumen del Trabajo**

Debido a la inseguridad que presentan los hogares frente a robos y el alto precio de los sistemas antirrobo que ofrecen las compañías especializadas (junto con su dependencia a estas) se ha diseñado e implementado una alarma antirrobo destinada a hogares de familias de bajo poder adquisitivo, robusta, fiable e independiente por un precio inferior a 70 euros.

El Sistema se ha desarrolla usando la plataforma STM32F4 y su familia de sensores y shields compatibles con el fin de adquirir nuevos conocimientos tecnológicos. Dispone de teclado (para arranque y parada manuales, y programación de arranque y parada automáticos), sensor RFID (para arranque y parada manuales), módulo RTC (para programación de horas de inicio y fin), pequeño visualizador de datos y leds de estado.

Para el desarrollo del proyecto se ha seguido una metodología tradicional de proyectos compuesta por toma de requisitos, especificaciones, diseño, implementación y pruebas que han ayudado al autor a completar la adquisición de todas competencias que ofrece el Máster Interuniversitario de Ingeniería de Telecomunicación UOC-URL.

Se ha conseguido finalizar la implementación completa del proyecto y tras pasar este por un amplio banco de pruebas, todas con resultado exitoso, se cumple el 100% de los objetivos del proyecto, dando el proyecto por

finalizado.

Las conclusiones de la realización de este proyecto son satisfactorias en todos los sentidos: plazos de planificación cumplidos, grata adquisición de nuevos conocimientos tecnológicos, resultados del proyecto exitosos y competencias del MIIT obtenidas.

**Abstract:**

Due to insecurity posed by homes against theft and the high price of anti-theft systems offered by specialized companies (along with their dependence on these) a burglar alarm has been designed and implemented to homes of families with low purchasing power, robust, reliable and independent for a lower price to 70 euros.

The System has been developed using the STM32F4 platform and its family of sensors and shields supported in order to acquire new technological knowledge. It has a keyboard (for manual start and stop, and scheduling automatic start and stop of the alarm), RFID sensor (manual start and stop of the alarm), RTC module (programming start and end), small display data and status LEDs.

To develop the project a traditional project methodology has been followed, that consists of collect requirements, specifications, design, implementation and testing that has helped the author to complete the acquisition of all skills offered by the Interuniversity Master in Telecommunication Engineering UOC-URL .

It has managed to complete the full implementation of the project and after passing this by a large bank of tests, all with successful result, 100% of the project objectives are met, giving the project completed.

The conclusions of the realization of this project are satisfactory in all respects: planning periods completed, grateful acquisition of new technological knowledge, successful project outcomes and competences of MIIT obtained.



# Resumen.

Debido a la inseguridad que presentan los hogares comunes frente a robos y el alto precio de los sistemas antirrobo que ofrecen las compañías especializadas (junto con su dependencia a estas) se pretende diseñar e implementar una alarma antirrobo destinada a hogares de familias de bajo poder adquisitivo, que sea robusta, fiable e independiente por un precio menor a 70 euros.

El Sistema dispondrá de los siguientes sensores:

- Sensor detector de movimiento.
- Sensor detector de apertura de puertas.

Los cuales al detectar movimiento o la apertura de una puerta/ventana dispararán una alarma cuya misión principal será:

- Activar una señal acústica.
- Activar una señal luminosa.

El Sistema además debe disponer de:

- Programación de periodo de arranque y parada del Sistema automáticos (con teclado).
- Arranque y parada del Sistema mediante teclado.
- Arranque y parada del Sistema mediante tarjeta RFID.
- Módulo RTC (Real Time Controller), para programación de calendario y horas.
- Pequeño visualizador de datos.
- Leds de estado.

El Sistema estará basado en el microprocesador STM32F4 y su familia de sensores y shields compatibles, ideales para cumplir con los requisitos de robustez, fiabilidad y bajo coste económico. El software se programará con el entorno de trabajo TrueStudio en su versión gratuita.

Para el desarrollo del proyecto se seguirá una metodología de trabajo tradicional compuesta por las siguientes etapas: estudio del Estado del Arte, toma de requisitos, definición de especificaciones, diseño, implementación y pruebas que ayudarán al autor a obtener todas las competencias que ofrece el Máster Interuniversitario en Ingeniería de Telecomunicación UOC-URL.

# Agradecimientos.

El autor quiere realizar una serie de agradecimientos públicos a las siguientes personas hablando en primera persona:

-A su padre José por su paciencia y sus inmensos sacrificios, eternos, sin él nada de esto habría sido posible. Simplemente un modelo a seguir.

-A su madre Encarnita por su transmisión constante de valores humanos, su alegría, su paciencia y su compresión que ha hecho que el autor se levante después de cada caída.

-A su hermano José Alberto por su ayuda, su transferencia constante de conocimientos y sus cuidados durante todo el camino recorrido en esta aventura.

-A su abuela Angustias por ese amor tan profundo y esa ternura que consigue arrancar una sonrisa incluso en el momento más triste.

El viaje por fin llega a su fin... ¿O continua en una nueva apasionante etapa?

GRACIAS.

# Índice.

<b>Resumen</b> .....	<b>2</b>
<b>Agradecimientos</b> .....	<b>3</b>
<b>Índice</b> .....	<b>4</b>
<b>Índice de figuras</b> .....	<b>5</b>
<b>Índice de tablas</b> .....	<b>6</b>
<b>Consideraciones importantes para la lectura de la memoria</b> .....	<b>8</b>
<b>Capítulo 1. Introducción</b> .....	<b>9</b>
1.1.- Palabras clave.....	9
1.2.- Objetivos principales del proyecto.....	9
1.3.- Motivación.....	10
1.4.- Entregables.....	11
<b>Capítulo 2. Estado del Arte</b> .....	<b>12</b>
2.1.- Soluciones comerciales dependientes.....	12
2.2.- Soluciones comerciales independientes .....	13
2.3.- Soluciones caseras básicas .....	14
2.4.- Soluciones caseras basadas en microprocesadores .....	15
2.4.1.- Soluciones caseras basadas en la plataforma Arduino.....	15
2.4.2.- Soluciones caseras basadas en la plataforma STM32F4	
Discovery.....	17
2.4.2.1- <i>La plataforma STM32F4 Discovery: manejo del</i>	
<i>procesador STM32F4</i> .....	17
2.4.2.2- <i>Éxitos de control de sensores y módulos entrada/salida</i>	
<i>con la plataforma STM32F4 Discovery</i> .....	18
2.5.- Soluciones software apoyadas en hardware existente.....	21
<b>Capítulo 3. Diseño</b> .....	<b>22</b>
3.1.- Estudio de la plataforma STM32F4 Discovery.....	22
3.2.- Especificaciones del Sistema.....	28
3.2.1.- Especificaciones generales.....	28
3.2.2.- Especificaciones técnicas.....	29
3.2.3.- Especificaciones funcionales.....	30
3.3.- Diseño de arquitectura.....	32
3.3.1.- Diagrama de bloques de alto nivel.....	32
3.3.2.- Diseño de fluujogramas funcionales.....	34
<b>Capítulo 4.- Implementación</b> .....	<b>45</b>
4.1.- Selección de componentes .....	46
4.2.- Interconexión de componentes .....	53
4.3.- Programación de librerías .....	57
4.4.- Plan de pruebas .....	65
<b>Capítulo 5. Mejoras futuras</b> .....	<b>81</b>
<b>Capítulo 6. Conclusiones</b> .....	<b>82</b>
<b>Anexos</b> .....	<b>83</b>
<b>Bibliografía</b> .....	<b>105</b>

# Índice de figuras.

Figura 01: diagrama WBS.....	10
Figura 02: ejemplo alarma comercial.....	13
Figura 03: ejemplo alarma casera.....	13
Figura 04: ejemplo 1 de alarma casera basada en arduino.....	14
Figura 05: ejemplo 2 de alarma casera basada en arduino.....	15
Figura 06: ejemplo 3 de alarma casera basada en arduino.....	15
Figura 07: Real Time Controller.....	17
Figura 08: Lector RFID.....	17
Figura 09: Motion Sensor Board.....	18
Figura 10: Matrix Keypad.....	18
Figura 11: LCD 16x2.....	19
Figura 12: Relé.....	19
Figura 13: diagrama de bloques de STM32F4 Discovery.....	23
Figura 14: diagrama de componentes de STM32F4 Discovery.....	24
Figura 15: foto real de la placa STM32F4 Discovery.....	24
Figura 16: diagrama de bloques del Sistema HomeAlarm.....	31
Figura 17: empaquetado del Sistema HomeAlarm.....	32
Figura 18: rutina “General”.....	33
Figura 19: rutina “Inicialización de Variables”.....	34
Figura 20: rutina “Encendido de LEDs”.....	35
Figura 21: rutina “Lectura Teclado (General)”.....	36
Figura 22: rutina “Lectura Teclado Menú A”.....	37
Figura 23: rutina “Lectura Teclado Menú B”.....	38
Figura 24: rutina “Lectura Teclado Menú C”.....	39
Figura 25: rutina “Lectura Teclado Menú D”.....	40
Figura 26: rutina “Leer RFID”.....	41
Figura 27: rutina “Lectura de Sensores”.....	42
Figura 28: rutinas de control de alarma.....	43
Figura 29: venta STM32F4 Discovery.....	45
Figura 30: venta fuente “MS-35-5”.....	46
Figura 31: venta de LEDs.....	46
Figura 32: venta de placa STM32F4 Discovery Shield.....	47
Figura 33: venta de LCD 16x2.....	48
Figura 34: venta de keypad 4x4.....	49
Figura 35: venta de RTC.....	49
Figura 36: venta de relé.....	50
Figura 37: venta de sensor de movimiento.....	50
Figura 38: venta de sensor magnético.....	51
Figura 38: venta de lector y tarjeta RFID.....	51
Figura 40: montaje Sistema HomeAlarm completo.....	56
Figura 41: captura “main.c” de HomeAlarm en entorno TrueStudio.....	57

# Índice de tablas.

Tabla 01: Precios de los Componentes del Sistema HomeAlarm.....	53
Tabla 02: Distribución de pines STM32F4 Discovery en puertos I2C.....	54
Tabla 02: Distribución de pines STM32F4 Discovery en puertos SPI.....	54
Tabla 04: Asignación de Pines del Sistema HomeAlarm.....	55
Tabla 05: Prueba HA-01.....	65
Tabla 06: Prueba HA-02.....	65
Tabla 07: Prueba HA-03.....	65
Tabla 08: Prueba HA-04.....	66
Tabla 09: Prueba HA-05.....	66
Tabla 10: Prueba HA-06.....	66
Tabla 11: Prueba HA-07.....	67
Tabla 12: Prueba HA-08.....	67
Tabla 13: Prueba HA-09.....	67
Tabla 14: Prueba HA-10.....	68
Tabla 15: Prueba HA-11.....	68
Tabla 16: Prueba HA-12.....	69
Tabla 17: Prueba HA-13.....	69
Tabla 18: Prueba HA-14.....	70
Tabla 19: Prueba HA-15.....	70
Tabla 20: Prueba HA-16.....	70
Tabla 21: Prueba HA-17.....	70
Tabla 22: Prueba HA-18.....	71
Tabla 23: Prueba HA-19.....	71
Tabla 24: Prueba HA-20.....	72
Tabla 25: Prueba HA-21.....	72
Tabla 26: Prueba HA-22.....	73
Tabla 27: Prueba HA-23.....	73
Tabla 28: Prueba HA-24.....	73
Tabla 29: Prueba HA-25.....	73
Tabla 30: Prueba HA-26.....	74
Tabla 31: Prueba HA-27.....	74
Tabla 32: Prueba HA-28.....	75
Tabla 33: Prueba HA-29.....	75
Tabla 34: Prueba HA-30.....	75
Tabla 35: Prueba HA-31.....	76
Tabla 36: Prueba HA-32.....	76
Tabla 37: Prueba HA-33.....	77
Tabla 38: Prueba HA-34.....	77
Tabla 39: Prueba HA-35.....	78
Tabla 40: Prueba HA-36.....	78
Tabla 41: Prueba HA-37.....	78

Tabla 42: Prueba HA-38.....	79
Tabla 43: Prueba HA-39.....	79
Tabla 44: Prueba HA-40.....	79

# Consideraciones importantes para la lectura de la memoria.

Durante esta memoria se irán realizando reseñas bibliográficas, las cuales serán marcadas con un número entre corchetes, [x] donde el número 'x' indica el número del recurso bibliográfico al que se está refiriendo. Los recursos bibliográficos serán listados por orden de aparición.

Ejemplo: la marca “[1]” indica que se está haciendo referencia al recurso bibliográfico “Recurso 1”.

# Capítulo 1. Introducción.

En la actualidad, debido a la crisis, en España el número de delitos por robo en hogares ha aumentado considerablemente. Según “El Confidencial” [1] sólo en 2014 se produjeron 120.000 robos en vivienda, un 70% más que 2007, antes del estallido de la crisis. Además del perjuicio económico que se produce con cualquier robo, se deben añadir los daños morales y psicológicos (pérdida de objetos de apego sentimental, sensación continua de inseguridad, agresiones, etc...). Debido a la inseguridad que presentan los hogares comunes frente a robos (“El Confidencial” [1] cifra en 11 millones de viviendas sin asegurar y más de cuatro millones de casas sin ningún tipo de dispositivo de seguridad instalado) y el alto precio de los sistemas antirrobo que ofrecen las compañías especializadas (junto con su dependencia a estas) se pretende diseñar e implementar una alarma antirrobo destinada a hogares de familias de bajo poder adquisitivo, que sea robusta, fiable e independiente por un precio menor a 70 euros.

El Sistema dispondrá de sensores (detector de movimiento y detector de apertura) los cuales dispararán una alarma para activar una señal luminosa y una señal acústica. El Sistema estará basado en el microprocesador STM32F4 y su familia de sensores y shields compatibles y dispondrá de teclado (arranque y parada manuales, y programación de arranque y parada automáticos), sensor RFID (arranque y parada manuales), módulo RTC (programación de horas de inicio y fin), pequeño visualizador de datos y leds de estado.

## 1.1.- Palabras clave.

Alarma, antirrobo, económica, independiente, microprocesador, STM32F4, sensores.

## 1.2.- Objetivos principales del proyecto.

Los objetivos principales del proyecto son los siguientes:

- Creación de alarma antirrobo fiable, independiente y económica (lo que a partir de ahora será llamado “el Sistema”).
- Diseñar e implementar el Sistema en un período de 16 semanas teniendo en cuenta que para ello se dispone de un solo recurso humano con una dedicación diaria máxima de 4 horas los 7 días de la semana.

- Diseñar e implementar el Sistema de forma que se cumplan el 100% de las funcionalidades que se describen en el resumen.
- Diseñar e implementar el Sistema de forma que integre el microprocesador STM32F4 y el 100% de sensores y módulos descritos en el resumen.
- Diseñar e implementar el Sistema de forma que se cumpla el requisito máximo de 70 euros de presupuesto.
- La consecución por parte del recurso humano dedicado a la realización del proyecto de las principales competencias de la asignatura “Trabajo Fin de Máster” del “Máster Interuniversitario en Ingeniería de Telecomunicación (UOC-URL)”, como pueden ser grosso modo la planificación y gestión íntegras de un proyecto, la modelización de un problema real, etc... Todas las competencias que se desean adquirir se encuentran descritas en el siguiente enlace [2]:

[http://cv.uoc.edu/tren/trenacc/web/GAT\\_EXP.PLANDOCENTE?any\\_academico=20152&cod\\_asignatura=M1.612&idioma=CAS&pagina=PD\\_PREV\\_SECRE&ache=S](http://cv.uoc.edu/tren/trenacc/web/GAT_EXP.PLANDOCENTE?any_academico=20152&cod_asignatura=M1.612&idioma=CAS&pagina=PD_PREV_SECRE&ache=S)

### **1.3.- Motivación.**

La principal motivación del autor de este documento para realizar este proyecto es la de profundizar en la programación y configuración de sistemas basados en microprocesadores de bajo coste ya introducida en la asignatura del máster “Instrumentación electrónica” mediante la plataforma “Arduino”. El autor de este documento considera esta rama de la electrónica un mundo fascinante mediante el cual se pueden dar soluciones de una forma simple, medianamente rápida, eficaz y sobre todo económica a problemas reales de diversa complejidad. La profundización sobre la plataforma STM32F4 aportará una visión más amplia de las opciones que presenta el mercado actual a la vez que una dificultad y reto añadidos al no ser una plataforma tan conocida como “Arduino”.

Además de ello, el autor de este documento siente una motivación especial por realizar un proyecto basado en electrónica puesto que a lo largo de su carrera (tanto formativa como laboral) ya se ha enfrentado a proyectos de otras índoles, principalmente software, en los que no existe de forma tan explícita la componente “hardware”, la componente física y se trata de un tipo nuevo de proyecto en el que desea formarse atendiendo a esa especial dualidad software-hardware.

Por último, el autor considera que la realización práctica de este proyecto le va aportar una experiencia que va a poder usar de manera

inmediata una vez finalizado el proyecto en la realización de más proyectos que den soluciones a problemas reales, algo que no adquiriría si por ejemplo realizase un trabajo fin de máster basado en un estudio teórico.

#### 1.4.- Entregables.

A continuación se especifican los entregables a nivel de producto en forma de “work breakdown structure” (wbs), o lo que es lo mismo, las partes y subpartes de las que constará el producto final (trabajo fin de máster). Se han tomado como las “partes” entregables, una por cada fase del proyecto, siendo la fase “Diseño e Implementación” la que dispone de más subpartes (las tres primeras para el diseño y las cuatro siguientes para la implementación, incluyendo la fase de pruebas como una parte de la implementación):

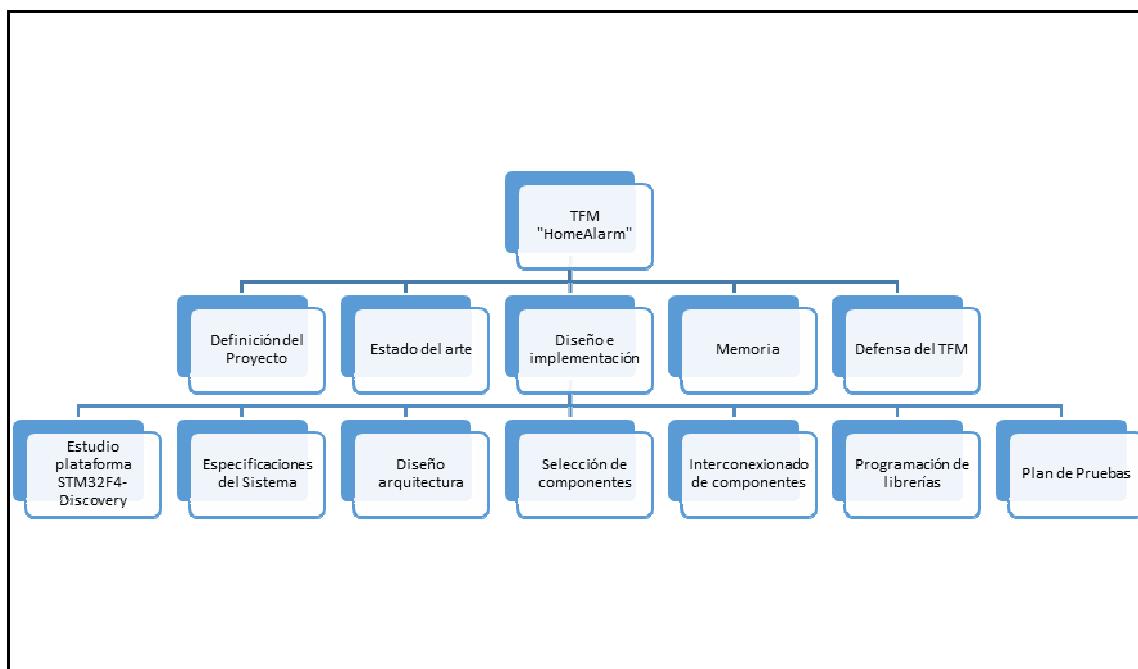


Figura 01: diagrama WBS.

# Capítulo 2. Estado del Arte.

El objetivo de este capítulo es presentar el Estado del Arte del proyecto que se va desarrollar: “Diseño e implementación de alarma antirrobo independiente, fiable y económica”.

Las soluciones existentes en la actualidad para el problema presentado, sistema antirrobo para hogares, son múltiples y de distinta naturaleza. Una posible clasificación de estas soluciones podría ser la siguiente:

- Soluciones comerciales basadas en dispositivos dependientes a empresas especializadas en seguridad.
- Soluciones comerciales basadas en dispositivos independientes (standalone).
- Soluciones caseras básicas (standalone).
- Soluciones caseras basadas en microprocesadores (dependientes y standalone).
- Soluciones software apoyadas en hardware existente

## **2.1.- Soluciones comerciales dependientes.**

En la actualidad son muchas las empresas que ofrecen servicios profesionales para la protección antirrobo de hogares. Dichos servicios están basados en dispositivos dependientes a servicios de la propia empresa, es decir, en dispositivos que ante la detección de una anomalía en la seguridad del hogar envían notificaciones a la propia empresa para que esta se encargue de actuar (de distintas maneras, como enviar un guardia de seguridad in situ, capturar imágenes, llamar a la policía, llamar al dueño del hogar, etc...). Estos dispositivos, aparte de contactar con la empresa de seguridad pueden activar alarmas acústicas y sonoras in situ, o no (si están configurados en modo silencioso).

La principal ventaja de este tipo de soluciones es su fiabilidad, al tratarse ya de un tipo de solución madura y testeada por la propia empresa. Su principal desventaja es el precio, elevado y unido a una cuota periódica a pagar mientras se desee mantener el servicio. Estas empresas suelen ofrecer distintos tipos de servicios ofreciendo mayor cantidad de protección aumentando el precio a pagar o pudiendo seleccionar servicios más económicos (y por tanto ofreciendo un nivel menor de seguridad).

A continuación se presentan enlaces a las páginas web de las principales empresas dedicadas a la implantación y mantenimiento de este tipo de soluciones antirrobo [3]:

Securitas Direct:

<http://www.securitasdirect.es/lp/g/alarmas-securitas-direct.php>

Tyco:

<http://alarmainteligente.tyco.es>

Prosegur:

<http://www.alarmahogarprosegur.com>

Verisure:

[https://www.verisure.es/alarma/landings/g/alarmas\\_verisure.html](https://www.verisure.es/alarma/landings/g/alarmas_verisure.html)

## **2.2.- Soluciones comerciales independientes.**

Existen soluciones comerciales basadas en dispositivos independientes, es decir, dispositivos que una vez instalados ante la detección de una anomalía en la seguridad del hogar, actúan in situ (normalmente activando una señal acústica y/o una señal luminosa), y al contrario que los anteriores, no envían nada a la empresa que los comercializa. En esta categoría estaría incluida la solución que se va a desarrollar en el presente proyecto.

En el siguiente enlace [4] se presenta una empresa que se encarga de comercializar este tipo de dispositivos. En su apartado “Nosotros” realizan una explicación adecuada del motivo por el que ofrecen sus soluciones:

<http://www.tualarmasincuotas.es/paginas/nosotros>

“Como en otras muchas ocasiones, esta tienda surge tras resolver y buscar una manera de solucionar una situación particular.... Estábamos HARTOS de pagar una cuota elevada por un servicio inexistente o fácilmente sustituible. Además, si queríamos convertir la alarma (conectada a Central) en un Sistema de Seguridad (incendio, inundación, etc) suponía una inversión muy elevada y con la consiguiente subida de cuotas adicional por cada elemento añadido. Entonces nos propusimos la necesidad de sustituir nuestra alarma conectada por una de NUESTRA propiedad y nuestro control, una alarma SIN cuotas mensuales....”

En el siguiente enlace [5] se muestra una solución concreta ofrecida por dicha empresa:

<http://www.tualarmasincuotas.es/productos/alarma-sin-cuotas-independiente>



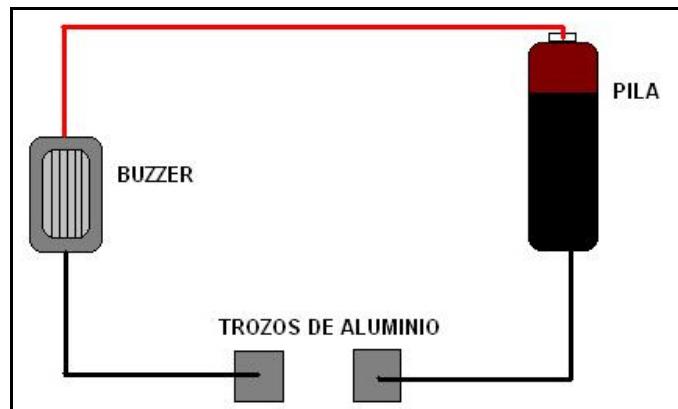
**Figura 02: ejemplo alarma comercial.**

Se trata de un sistema de alarma independiente con detección de movimiento, sirena (alarma acústica) y mando a distancia para programación arranque y parada. Se trata de un sistema similar al que se desea desarrollar, que si bien tiene buen precio (89€ el pack básico), su coste es más elevado y no proporciona todas las funcionalidades que se van a implementar.

### **2.3.- Soluciones caseras básicas.**

Existen soluciones caseras básicas (no basadas en microprocesadores) con las que se puede solucionar parte del problema presentado. Un ejemplo claro se presenta en el siguiente enlace [6], donde con un altavoz, una batería y un pequeño circuito eléctrico se puede simular la función de alarma acústica:

<http://comohacer.eu/como-hacer-una-alarma-casera/>



**Figura 03: ejemplo alarma casera.**

Este tipo de solución es barata y simple, y puede ser ampliada con facilidad para por ejemplo poder tener también una alarma luminosa. Sin embargo, no resulta robusta y fiable.

## 2.4.- Soluciones caseras basadas en microprocesadores.

Esta categoría será la que incluya la solución que se va a desarrollar, concretamente esta serán las soluciones caseras basadas en el procesador STM32F4. Existen soluciones basadas en multitud de familias diferentes de procesadores. Sin embargo, atendiendo a las motivaciones especificadas en la definición del proyecto, se considerará en el estudio del Estado del Arte diversas soluciones basadas en las plataformas de investigación y evaluación “Arduino” y “STM32F4 Discovery”.

### 2.4.1.- Soluciones caseras basadas en la plataforma Arduino.

Arduino es la plataforma inicial sobre la cual se deseaba realizar el desarrollo de la aplicación. Sin embargo, por motivos académicos fue descartada en favor de la plataforma STM32F4. Sobre la plataforma “Arduino” existen bastantes soluciones hoy día. A continuación se mostrarán diferentes ejemplos:

Ejemplo 1 [7]:

<http://descubrearduino.com/un-sistema-de-alarma-para-tu-hogar-creado-con-arduino/>

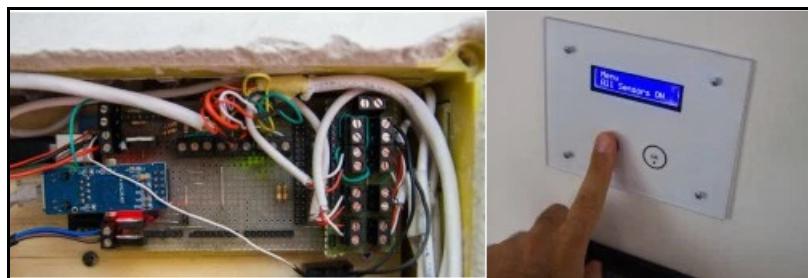
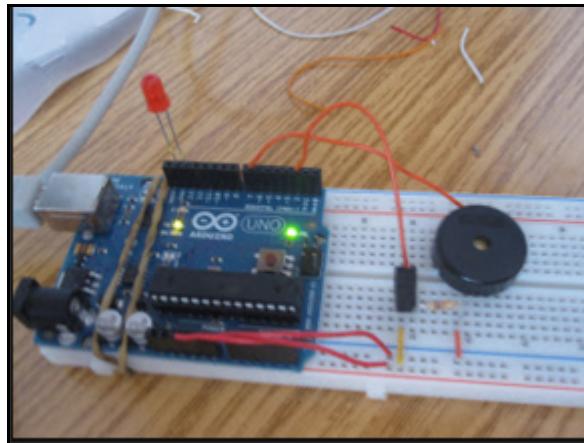


Figura 04: ejemplo 1 de alarma casera basada en arduino.

Se trata de un sistema donde la placa Arduino se conecta a una caja de control remoto montada discretamente cerca de la puerta principal, que cuenta con una pantalla LCD, un lector NFC y RFID para la autorización, un altavoz para las órdenes además de algunos botones capacitivos para armar y desarmar el sistema.

Ejemplo 2 [7]:

<http://arduinotesting.blogspot.com.es/2010/12/pequena-alarma-anti-robo.html>



**Figura 05: ejemplo 2 de alarma casera basada en arduino.**

Se trata de una página en la que muestran cómo se ha desarrollado una pequeña alarma usando la placa “Arduino” y los siguientes componentes:

- 1 tilt sensor.
- 1 resistencia de 10k Ohms.
- 1 zumbador piezo.
- 1 LED.

Ejemplo 3 [7]:

<https://www.youtube.com/watch?v=3UmN8CJlktI>



**Figura 06: ejemplo 3 de alarma casera basada en arduino.**

Se trata de un vídeo corto en el que se muestra una pequeña alarma que funciona con arduino y que consta de unos sensores que activan la alarma, el sensor de distancia y el sensor de luz (LDR).

## 2.4.2.- Soluciones caseras basadas en la plataforma STM32F4 Discovery.

Cómo se comentaba en el apartado anterior la plataforma STM32F4 Discovery fue seleccionada para el desarrollo del proyecto por una motivación académica. Al no haber encontrado soluciones al problema planteado usando esta plataforma, se ha dividido el Estudio del Arte en los éxitos encontrados manejando con la plataforma el procesador STM32F4 y manejando mediante la plataforma los distintos módulos y sensores a usar.

### 2.4.2.1- *La plataforma STM32F4 Discovery: manejo del procesador STM32F4.*

Los éxitos acerca del manejo del procesador STM32F4 mediante la plataforma STM32F4 Discovery son múltiples.

- En el siguiente enlace [8] se puede encontrar el manual de usuario de la plataforma.

[http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user\\_manual/DM00039084.pdf](http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00039084.pdf)

- El siguiente enlace [9] contiene el “home” de la web oficial de la plataforma STM32F4-Discovery donde se alojan multitud de tutoriales y librerías para el trabajo con la plataforma. Es actualizado frecuentemente y será fuente fundamental de conocimiento para el desarrollo de las librerías que se implementarán en el proyecto.

<http://stm32f4-discovery.com/>

- Por último, en el siguiente enlace [10] contiene una web donde se encuentra alojado el “Trabajo Fin de Máster” del “Máster Interuniversitario en Ingeniería de Telecomunicación (UOC-URL)” en el que el alumno autor del trabajo ya ha inicializado el microprocesador STM32F4 y explica cómo funciona todo el entorno.

<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/42670/7/jvillalengaTFM0615memoria.pdf>

#### 2.4.2.2- Éxitos de control de sensores y módulos entrada/salida con la plataforma STM32F4 Discovery.

A continuación se presentan éxitos existentes en el ámbito de control de sensores y módulos entrada/salida con la placa STM32F4 Discovery.

##### RTC (Real Time Controller):

En el siguiente enlace [11] se presenta una librería con la que poder usar un módulo RTC. Se trata de un éxito importante para este proyecto ya que dicho módulo es usado para la programación de calendarios y horas:

<http://stm32f4-discovery.net/2014/05/library-15-ds1307-real-time-clock-for-stm32f429-discovery/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



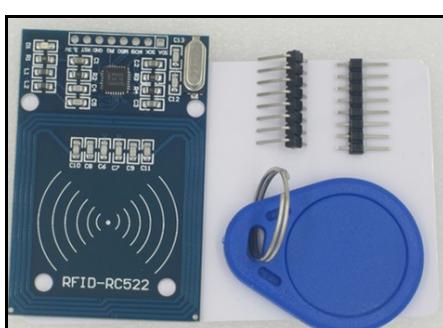
**Figura 07: Real Time Controller.**

##### Lector RFID:

En el siguiente enlace [12] se presenta una librería con la que poder usar un módulo RFID. Se utiliza para la activación y desactivación de la alarma:

<http://stm32f4-discovery.com/2014/07/library-23-read-rfid-tag-mfrc522-stm32f4xx-devices/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



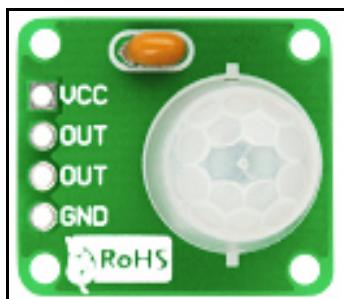
**Figura 08: Lector RFID.**

### PIR (Motion Sensor Board):

En el siguiente enlace [13] se encuentra la librería, ejemplos de uso y manual de usuario para la utilización de este sensor de movimiento o PIR. La función de este dispositivo es la de detectar el movimiento de un objeto, lo cual se utilizará lanzar el evento de alarma:

<http://www.mikroe.com/add-on-boards/various/motion-sensor/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



**Figura 09: Motion Sensor Board.**

### Teclado 4x4 (Matrix keypad):

En el siguiente enlace [14] se encuentra la librería y ejemplos de uso de un teclado de 4x4 teclas. La función de este teclado es la de programar el inicio y parada automática de la alarma así como el inicio y parada manual:

<http://stm32f4-discovery.com/2014/09/library-32-matrix-keypad-stm32f4xx/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



**Figura 10: Matrix Keypad.**

### LCD 16x2 Caracteres:

En el siguiente enlace [15] se encuentra la librería y ejemplos de uso de un LCD de 2 filas y 16 columnas de caracteres. La función de este LCD es la de mostrar datos de entrada y salida de la alarma:

<http://stm32f4-discovery.com/2014/06/library-16-interfacing-hd44780-lcd-controller-with-stm32f4/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



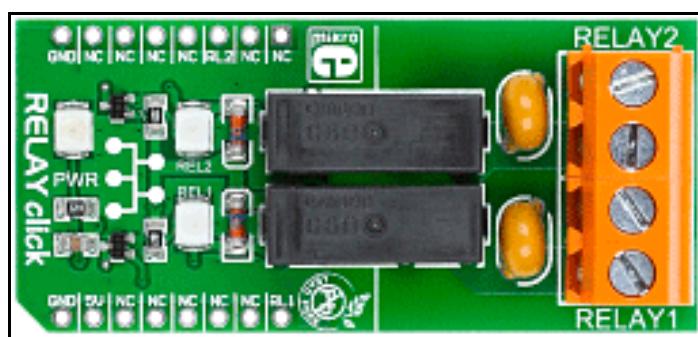
**Figura 11: LCD 16x2.**

### Relé:

En el siguiente enlace [16] se encuentra la librería, así como ejemplos y manual de instrucciones de relé. La función del relé es la de controlar el encendido y apagado de la señal luminosa:

<http://www.mikroe.com/click/relay/>

La siguiente imagen muestra un ejemplo de este tipo de módulo:



**Figura 12: Relé.**

#### Librería entrada/salida (GPIO):

En el siguiente enlace [17] se encuentra la librería y ejemplos de la configuración y uso de los puertos de entrada y salida del STM32F4 discovery. Las entradas y salidas se usan para leer eventos externos a la alarma y gestionar los distintos periféricos que se conectan a la alarma:

<http://stm32f4-discovery.com/2014/04/stm32f429-discovery-gpio-tutorial-with-onboard-leds-and-button/>

#### Generador de señales (DAC):

En el siguiente enlace [18] se encuentra la librería y ejemplos de la configuración y uso del generador de señales que la plataforma STM32F4 Discovery dispone. Este generador de señales será usado para generar las señales sonoras que se producirán cuando se produzca el evento de alarma:

<http://stm32f4-discovery.com/2014/09/library-36-dac-signal-generator-stm32f4/>

### **2.5.- Soluciones software apoyadas en hardware existente.**

Estas soluciones están basadas en aplicaciones software, normalmente para los S.O. Android o IOS que se instalan en móviles y tablets para proporcionar una solución al problema estudiado.

Para profundizar un poco en ello, el siguiente enlace [19] contiene un breve resumen de las principales aplicaciones existentes y las funcionalidades de las que disponen:

<http://articulos.softonic.com/protege-tu-casa-ladrones-con-apps>

# Capítulo 3. Diseño.

El objetivo de este capítulo es realizar el diseño del Sistema. En esta fase se decide dar un nombre corto al Sistema, para poder referirse a él con mayor comodidad. El nombre corto que se decide será “**HomeAlarm**”. Estará comprendido por tres apartados, que serán incluidos como independientes pero relacionados entre sí. Estos apartados serán “Estudio de la plataforma STM32F4 Discovery”, “Especificaciones del Sistema” y “Diseño de la Arquitectura”.

A modo de resumen-introducción decir que en el apartado “Estudio de la plataforma STM32F4 Discovery” se presentará dicha plataforma, describiendo sus especificaciones, entorno de trabajo, etc...

Por otra parte, el apartado “Especificaciones del Sistema” contendrá las especificaciones del Sistema “HomeAlarm” obtenidas a partir de los requisitos y objetivos definidos anteriormente.

Por último, el apartado “Diseño de la Arquitectura” usará los dos anteriores para definir la arquitectura del Sistema mediante un modelo de bloques de alto nivel (hardware) y una serie de flujogramas que describan las distintas funcionalidades del Sistema (software).

## 3.1.- Estudio de la plataforma STM32F4 Discovery.

En primer lugar, se realizará un resumen del manual de usuario oficial de la plataforma STM32F4 Discovery que puede ser obtenido en la siguiente enlace [8]:

[http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user\\_manual/DM00039084.pdf](http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00039084.pdf)

En ocasiones la web no está disponible y el manual puede ser obtenido del siguiente enlace no oficial [8]:

<https://docs.google.com/viewer?a=v&pid=sites&srcid=bXJ0LnVzdS5IZHV8Y3ViZS1zYXQtdGVhbS1yZXNvdXJjZXxneDozYmQwMWUwNTViNjE3NzU2>

La placa STM32F4 DISCOVERY es un kit de desarrollo de bajo costo y fácil de usar para poder evaluar e iniciar de forma rápida un desarrollo con el microcontrolador de alto rendimiento STM32F4.

La placa es compatible con Windows (XP, Vista, 7, 8 y 10) y necesita un cable USB tipo A a mini-USB para ser conectada a un PC. Es compatible con los siguientes entornos de desarrollo:

- Altium, TASKING™ VX-Toolset
- IAR, EWARM
- Keil™, MDK-ARM
- Atollic, TrueSTUDIO (que será el usado en el proyecto “HomeAlarm”).

Principales características:

- Microcontrolador STM32F407VGT6 con 1 MB de memoria flash, 192 KB de RAM en un paquete LQFP100.
- Incluye un puerto ST-LINK / V2 con interruptor de modo de selección para utilizar el kit como cd forma independiente o con conector SWD para la programación y depuración.
- Toma de fuente de alimentación: a través del bus USB o de una fuente de alimentación de 5V de tensión externa. En este caso, la placa STM32F4DISCOVERY debe ser alimentada con una unidad que cumpla con el estándar EN-60950-1: 2006+A11/2009..
- Ofrece fuente de alimentación para aplicaciones externas (como sensores) de 3V y 5V.
- LIS302DL, ST MEMS sensor de movimiento, acelerómetro de 3 ejes de salida digital.
- MP45DT02, sensor de audio ST MEMS, micrófono digital omnidireccional.
- CS43L22, conversor DAC de audio con controlador integrado de clase D para altavoz.
- Ocho LEDs:
  - LD1 (rojo / verde) para la comunicación USB.
  - LD2 (rojo) para la alimentación de 3,3 V
  - Cuatro LEDs de usuario: LD3 (naranja), LD4 (verde), LD5 (rojo) y LD6 (azul).
  - 2 USB OTG LED LD7 (verde) y LD8 (rojo).
- Dos pulsadores (de usuario y de reset).
- USB OTG con conector micro-AB.
- Extensión para LQFP100 E / S para la conexión rápida de placas de prototipo y fácil testeo.

La siguiente figura muestra un diagrama de alto nivel de los bloques hardware y las conexiones entre el procesador STM32F407VGT6 y sus periféricos (ST-LINK/V2, botones pulsadores, LED, DAC, USB, acelerómetro ST MEMS, micrófono ST MEMS y los distintos conectores:

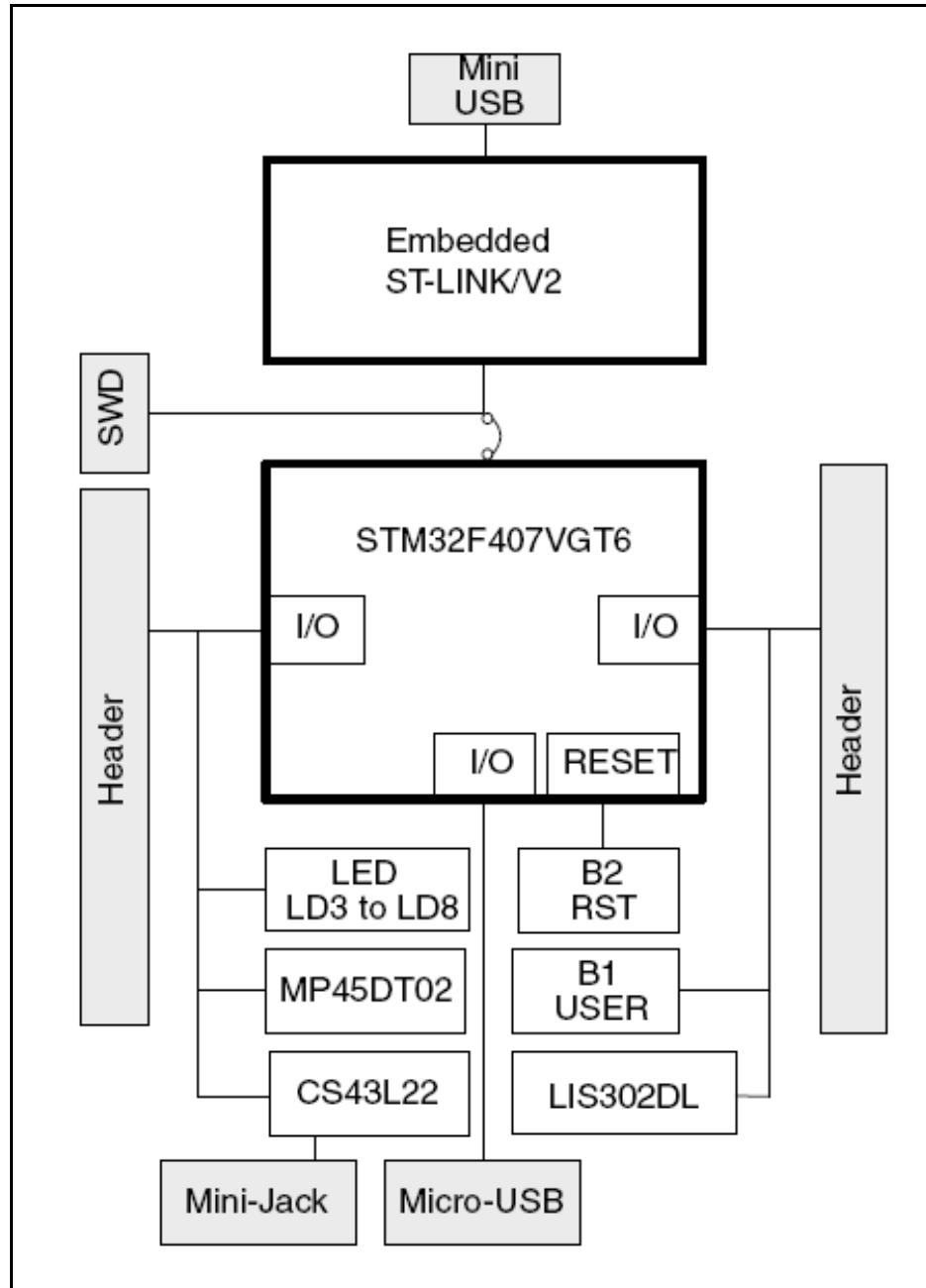


Figura 13: diagrama de bloques de STM32F4 Discovery.

La siguiente imagen muestra estos componentes en la placa real (cara superior de la placa):

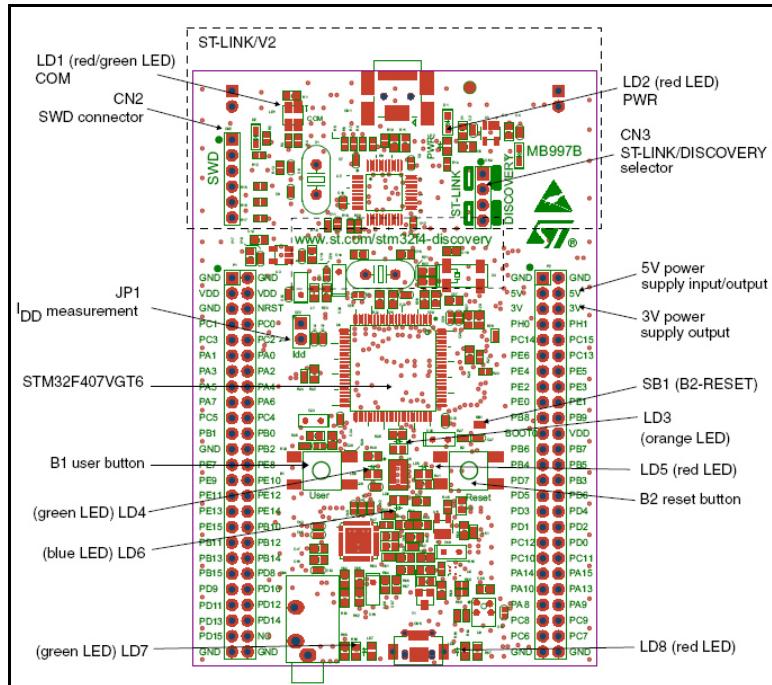


Figura 14: diagrama de componentes de STM32F4 Discovery.

Y la siguiente imagen muestra una foto real de la placa:

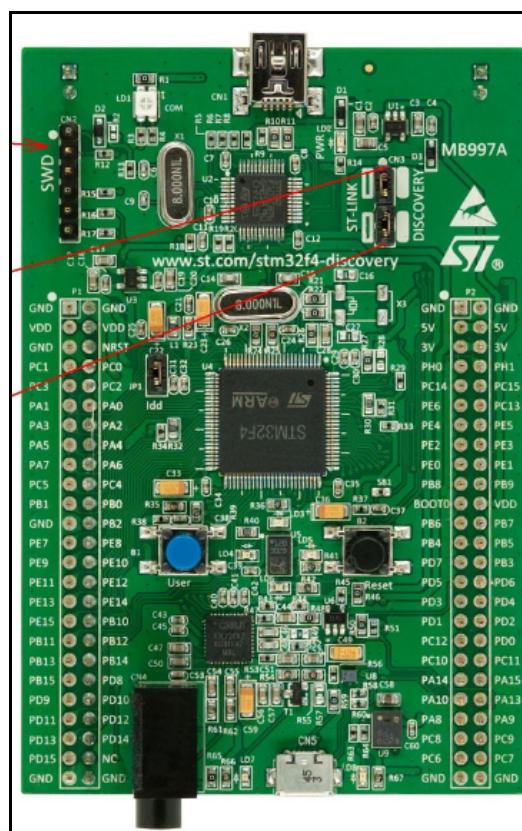


Figura 15: foto real de la placa STM32F4 Discovery.

Con respecto a los pines de conexión de la placa, esta dispone de 100 distribuidos del siguiente modo:

- 3 pines VDD.
- 10 pines GND.
- 2 pines que ofrecen alimentación de 5v a dispositivos externos.
- 2 pines que ofrecen alimentación de 3v a dispositivos externos.
- 1 pin NRST.
- 1 pin NC.
- 1 pin BOOT0.

El resto son pines configurables para entrada/salida:

- Puerto A: 16 pines nombrados de PA0 a PA15.
- Puerto B: 16 pines nombrados de PB0 a PB15.
- Puerto C: 16 pines nombrados de PC0 a PC15.
- Puerto D: 16 pines nombrados de PD0 a PD15.
- Puerto E: 16 pines nombrados de PE0 a PE15.
- Puerto H: 2 pines PH0 y PH1.

El microcontrolador STM32F407VGT6 es un ARM Cortex-M4 de 32 bits MCU con FPU que tiene 210 DMIPS, hasta 1 MB de memoria Flash / 192 + 4 KB RAM, USB OTG HS / FS, Ethernet, 17 TIM, 3 ADCs, 15 interfaces de comunicación y soporte para cámara.

Este dispositivo proporciona las siguientes ventajas:

- 168 MHz / 210 DMIPS Cortex-M4 con ciclo simple DSP MAC y unidad de coma flotante que ofrece:
  - Mejora en la ejecución de algoritmos de control.
  - Más características posibles para aplicaciones.
  - Facilidad de uso.
  - Mejor eficiencia de código.
  - Mayor rapidez de comercialización.
  - Fácil soporte para herramientas meta-lenguaje.
- Diseñado para un alto rendimiento y transferencias de datos ultra-rápida;
- Eficiencia energética excepcional; Ultra-baja potencia dinámica, baja potencia cuando funciona a baja tensión o con una batería recargable.
- Integración máxima: hasta 1 Mbyte de memoria flash en el chip y 192 Kbytes de SRAM.
- Periféricos superiores e innovadores que proporcionan nuevas posibilidades para conectar y comunicar los datos de alta velocidad y una mayor precisión debido a la alta resolución.

- Amplias herramientas y soluciones de software que proporcionan una amplia elección dentro del ecosistema STM32 para desarrollar aplicaciones.

Con respecto a la plataforma de desarrollo, la más extendida entre los tutoriales encontrados en internet es “Atollic, TrueSTUDIO” que puede ser descargado en el siguiente enlace [20]:

<http://timor.atollic.com/truestudio/>

Se trata de un IDE de desarrollo C/C++ profesional para ARM. Existen una versión de pago llamada “TrueSTUDIO Pro” y una versión gratuita llamada simplemente “TrueSTUDIO”. Esta será versión usada para el desarrollo del proyecto. Esta versión incluye lo siguiente:

- Enfocada para desarrollo ARM, es compatible con más de 2500 dispositivos y sobre 150 placas de evaluación (entre ellas STM32F4 Discovery).
- Construida con GCC y GDB usando un IDE eclipse simplificado para la fácil integración de plugins propios.
- Dispone compilador, ensamblador y lincador optimizados para construir aplicaciones de gran calidad en sistemas embebidos.
- Debugger mono-core y multi-core compatible con J-Link, ST-link, P&E Multilink, etc...

Para crear un proyecto con este IDE y debuggearlo o ejecutarlo en la placa STM32F4 Discovery, instalar el IDE descargado (instalará el driver de la placa), enchufar mediante USB la placa al PC (Windows la detecta automáticamente y la instala sin necesidad de software adicional), ejecutar el IDE y seguir los siguientes pasos (entre cada uno pulsar “Next”):

1. Pulsar “New” → “Project” y seleccionar “C Project”.
2. Seleccionar “Embedded Project” y escribir el nombre del proyecto.
3. En hardware settings seleccionar:
  - Vendor: STMicroelectronics.
  - Evaluation board: STM32F4\_Discovery.
  - Floating point: Hardware implementation.
  - Floating point unit: FPv4-SP-D16
  - Code location: FLASH
  - Instruction set: Thumb2
  - Endianess: Little endian
4. Seleccionar Library “Newlib-nano (Reduced functionality).
5. Debug probe selection: ST-LINK.
6. Select Configurations: marcar Debug y Release.

7. Esto creará un proyecto con una clase llamada main.c con un método “int main (void)” con un bucle infinito que será el método que se ejecute cuando se inicie el dispositivo.
8. Para compilar el proyecto hay que pulsar el botón “Build Project” (martillo).
9. Para ejecutarlo debuggeando basta con pulsar el botón “Debug” (insecto).

Esto comenzará a ejecutar el código en modo debug, por lo que se parará en la primera línea de código existente. Pulsando “Resume” la ejecución continuará. Para parar la ejecución, basta con pulsar “Stop” (y a partir de aquí bastará con modificar el código si es necesario y repetir los pasos 8 y 9).

### **3.2.- Especificaciones del Sistema.**

En el presente apartado se definirán las especificaciones del Sistema “HomeAlarm”. Serán la base que permita empezar a desarrollar el proyecto.

#### **3.2.1.- Especificaciones generales.**

El Sistema dispondrá de los siguientes sensores:

- Sensor detector de movimiento.
- Sensor detector de apertura de puertas (o ventanas).

Los cuales al detectar movimiento o la apertura de una puerta/ventana dispararán una alarma cuya misión principal será:

- Activar una señal acústica.
- Activar una señal luminosa.

El Sistema además debe disponer de:

- Programación de periodo de arranque y parada del Sistema automáticos (con teclado).
- Arranque y parada del Sistema mediante teclado.
- Arranque y parada del Sistema mediante tarjeta RFID.
- Módulo RTC (Real Time Controller), para programación de calendario y horas.
- Pequeño visualizador de datos.
- Leds de estado.

El Sistema estará basado en el microprocesador STM32F4 y su familia de sensores y shields compatibles, ideales para cumplir con los requisitos de robustez, fiabilidad y bajo coste económico.

### 3.2.2.- Especificaciones técnicas.

De acuerdo a las especificaciones generales, y el estudio de la plataforma "STM32F4 Discovery" se pueden desarrollar ya las especificaciones técnicas.

-El Sistema será alimentado por una fuente de alimentación externa cuyo conector sea un Mini-USB y entregue una tensión de 5v y una corriente de 7A para poder alimentar correctamente todos los dispositivos conectados a la placa sin necesidad de más fuentes adicionales.

-Dispondrá de 3 leds de alta intensidad para mostrar distintos estados (rojo, naranja y verde).

-Los detectores de movimiento y apertura podrán ser conectados a la placa mediante un cableado de hasta 2 metros de longitud sin que esto afecte a su correcto funcionamiento.

-El lector de tarjetas RFID podrá ser conectado a la placa mediante un cableado de hasta 2 metros de longitud sin que esto afecte a su correcto funcionamiento.

-El Sistema no tiene capacidad para programar tarjetas RFID, sólo para leerlas, por las que estás deben ser previamente grabadas con una password específica que el Sistema conocerá.

-El Sistema dispondrá de un visualizador de datos. Este debe poder mostrar al menos 32 caracteres.

- El Sistema no incluye los dispositivos que producen la señal luminosa ni la acústica, estos dispositivos se dejan a elección del usuario que deberá tener en cuenta que:

-El dispositivo luminoso tendrá su propia fuente de alimentación y debe funcionar a 220v.

-Para la señal acústica existen dos opciones:

1.-Señal no amplificada, obtenida del propio mini-jack de la placa STM32F4 Discovery teniendo en cuenta que se emitirá un sonido mono:

High Mono Output Power at 10% THD+N

1 x 1.90 W into 4 Ω @ 5.0 V

1 x 1.00 W into 4 Ω @ 3.7 V

1 x 350 mW into 4 Ω @ 2.5 V

- 2.- Si el volumen sonoro anterior es insuficiente el usuario puede optar por utilizar dicha señal como entrada a un amplificador externo cuyo encendido/apagado puede ser controlado enchufándolo al mismo relé que la señal luminosa.
- El empaquetado final será una caja de plástico de dimensiones 18cm x 14cm x 5cm.

### **3.2.3.- Especificaciones funcionales.**

Para acotar las especificaciones generales se ha decidido desarrollar una serie de especificaciones funcionales:

1. Los password de la alarma tendrán el siguiente formato: 4 dígitos numéricos.
2. El Sistema viene programado por defecto con un password cuyo valor será '0000'. Dicho valor debe ser indicado en el manual de usuario.
3. El password debe poder ser cambiado por el usuario por cualquiera que desee siempre que cumpla el formato establecido en el punto 1.
4. El Sistema no tiene la capacidad de guardar un password cambiado por lo que si pierde la alimentación, cuando la recupere tendrá el password por defecto.
5. Para realizar cualquier acción de configuración, el Sistema siempre solicitará el password actual. Si al introducirlo este no es correcto, no permitirá continuar mostrando un mensaje de error y saliendo de la acción de configuración.
6. El Sistema tendrá 4 opciones de configuración:
  - Programación de reloj general.
  - Cambio de password
  - Programación de hora de inicio y hora de fin de rango de alarma activa.
  - Inicio o parada de alarma activa.
7. El sistema puede tener 4 estados:
  - Alarma no configurada (sólo led rojo encendido): se da cuando o no está el reloj general programado o no está registrado la hora de inicio y de fin de rango de alarma activa.
  - Alarma configurada (sólo led naranja encendido): se da cuando o el reloj general está programado y está registrado la hora de inicio y de fin de alarma activa, pero la alarma no ha sido iniciada.
  - Alarma iniciada (sólo led verde encendido): se da cuando la alarma está configurada y además se ha solicitado el inicio de la alarma (bien con teclado bien con tarjeta RFID).
  - Alarma disparada (led verde encendido y led rojo parpadeando): se da cuando estando la alarma en estado iniciado, se detecta movimiento o apertura de puerta (o ventana) en un instante de

tiempo comprendido entre la hora de inicio y la hora de fin del rango de alarma activa.

8. Para todas las horas que solicite el Sistema, el formato para introducir estas serán 4 dígitos numéricos seguidos “hhmm” donde “hh” es la hora en formato 24 horas (de 00 a 23) y “mm” son los minutos (de 00 a 59). Esta información debe ser incluida en el manual de usuario.
9. El Sistema no tendrá la hora programada al iniciarla la primera vez por lo que será necesario realizarlo desde la opción de programación de reloj general. Si se realiza correctamente, el Sistema mostrará un mensaje de éxito y saldrá de la opción de configuración quedando el reloj general programado.
10. Si en el menú de programación del reloj general se introduce una hora errónea el Sistema mostrará un mensaje de error y saldrá de la acción de configuración.
11. El menú de cambio de password solicitará un nuevo password y a continuación la confirmación de ese password introduciéndolo de nuevo. Si no coinciden se mostrará un error y saldrá de la opción de configuración. Si coinciden, se mostrará un mensaje de éxito, saldrá de la opción de configuración y el password quedará cambiado por el nuevo.
12. El menú de programación de hora de inicio y hora de fin de rango de alarma activa, solicitará primero la hora de inicio. Si es incorrecta, mostrará un mensaje de error y saldrá. Si es correcta solicitará la hora de fin. Si se introduce una hora de fin incorrecta, mostrará un mensaje de error y saldrá de la opción de configuración. Si es correcta mostrará un mensaje de éxito y la hora de inicio y de fin de alarma activa quedarán registradas.
13. El menú inicio o para de alarma parará la alarma si está disparada o iniciada. Sin embargo, si la alarma está parada la iniciará.
14. Si el lector RFID lee una tarjeta que no contenga la password RFID correcta, el Sistema no hará nada.
15. Si el lector RFID lee una tarjeta que contenga la password RFID correcta, el Sistema parará la alarma si está disparada o iniciada. Sin embargo, si la alarma está parada la iniciará.

### 3.3.- Diseño de arquitectura.

A la hora de realizar el Diseño de la Arquitectura del Sistema se ha tenido en cuenta el apartado “Estudio de la plataforma STM32F4 Recovery” junto con el apartado “Especificaciones del Sistema HomeAlarm”. Así, el primero y el segundo han sido usados para crear un diagrama de bloques de alto nivel en el que se muestran todos los módulos funcionales que van a existir mientras que el segundo ha sido usado para crear los flujogramas del código que será posteriormente implementado y que deben cubrir todas las especificaciones del Sistema.

#### 3.3.1.- Diagrama de bloques de alto nivel.

Tras analizar la plataforma STM32F4 Discovery y las especificaciones del Sistema, se ha obtenido el siguiente diagrama de bloques de alto nivel:

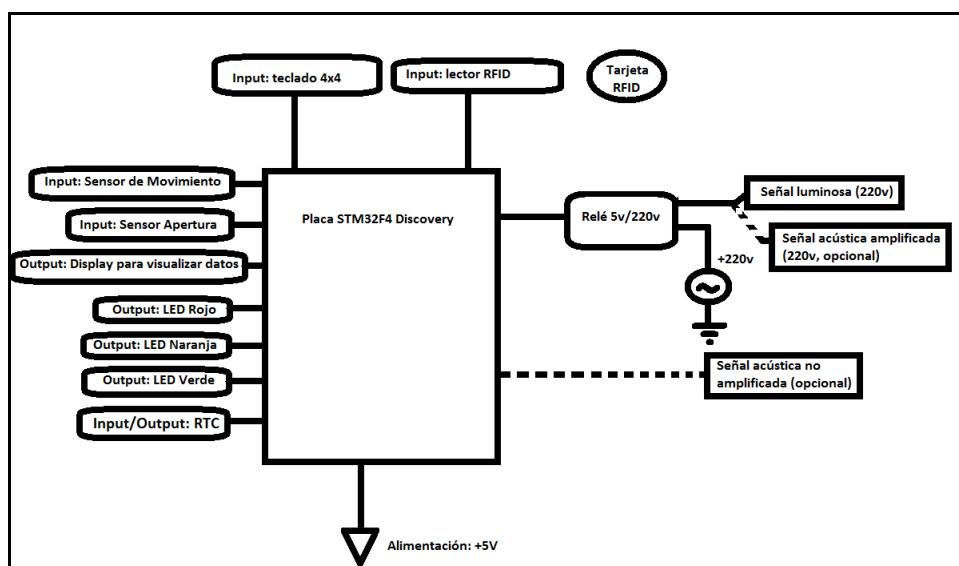


Figura 16: diagrama de bloques del Sistema HomeAlarm.

Como se puede observar el diagrama incluye todos los bloques funcionales necesarios para implementar las especificaciones del Sistema. Los componentes que se usarán para implementar cada bloque serán seleccionados más adelante, en la fase de implementación.

El usuario intervendrá con el Sistema mediante dos inputs: el teclado (en formato matriz 4x4) y el lector RFID (a través de la tarjeta RFID). El Sistema mostrará datos al usuario mediante un display de datos y 3 leds de diferentes colores. Un RTC (Real Time Controller) será programado (output) y leído (input) para conocer en qué rango de tiempo la alarma estará activa.

Los sensores de movimiento y apertura (a determinar más adelante qué tipo de sensor será usado para cada función) serán los inputs del entorno al Sistema encargados de activar las alarmas.

Por último, las alarmas activarán una señal luminosa activando un relé que alimenta a 220v un dispositivo luminoso. El mismo relé, opcionalmente, podrá activar una señal acústica amplificada que se alimenta a 220v. También se podrá activar opcionalmente una señal acústica no amplificada. Obligatoriamente el Sistema debe activar una de las dos señales, es elección del usuario decidir cuál conecta al Sistema (puede conectar las dos simultáneamente). Será necesario alimentar con 5V la placa principal del Sistema y con 220v los dispositivos conectados al relé.

A continuación se muestra el diseño de una versión final del Sistema, donde los componentes internos están integrados en un mismo empaquetado, quedando sólo exterior a este los dispositivos que físicamente deban estar lejos de este:

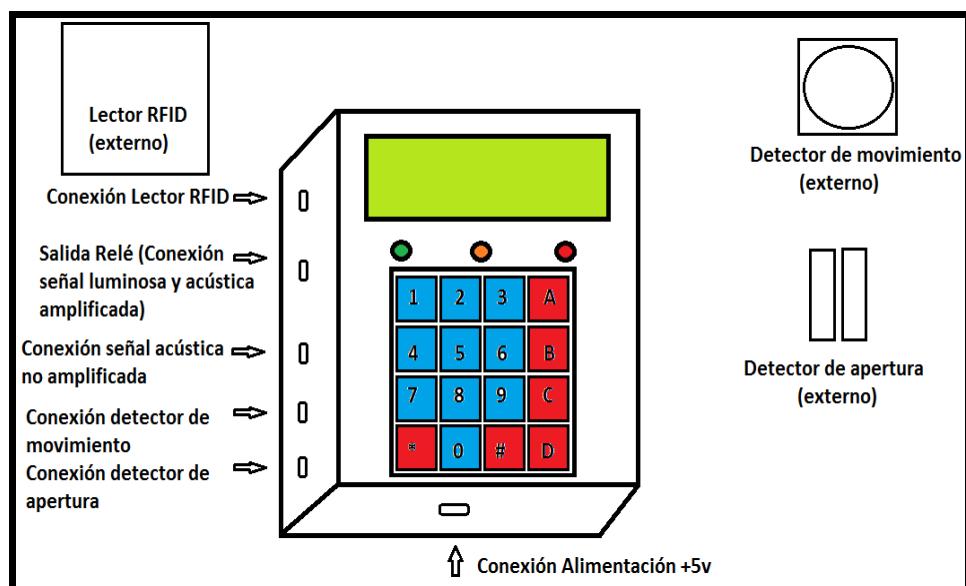


Figura 17: empaquetado del Sistema HomeAlarm.

Como se puede observar el empaquetado integrará el Display para mostrar los datos, los 3 leds de estado, el teclado, el RTC (en su interior) y el relé. La salida del relé estará disponible hacia el exterior para conectar a ella la señal luminosa y opcionalmente la señal acústica amplificada. Tendrá conexión también para la señal acústica no amplificada.

El empaquetado tendrá una conexión para el lector RFID y será el usuario quien decida donde ponerlo (normalmente si el empaquetado está dentro del edificio, el lector estará fuera de este). También dispondrá de conexiones para el detector de movimiento (que el usuario podrá colocar en la zona que desee vigilar) y para el detector de apertura (podrá ponerlo a una puerta o a una ventana). El empaquetado dispondrá de una conexión a alimentación +5v con formato Mini-USB.

### 3.3.2.- Diseño de flujogramas funcionales.

A continuación se mostrarán los flujogramas que describen todas las funcionalidades del Sistema.

#### Rutina General:

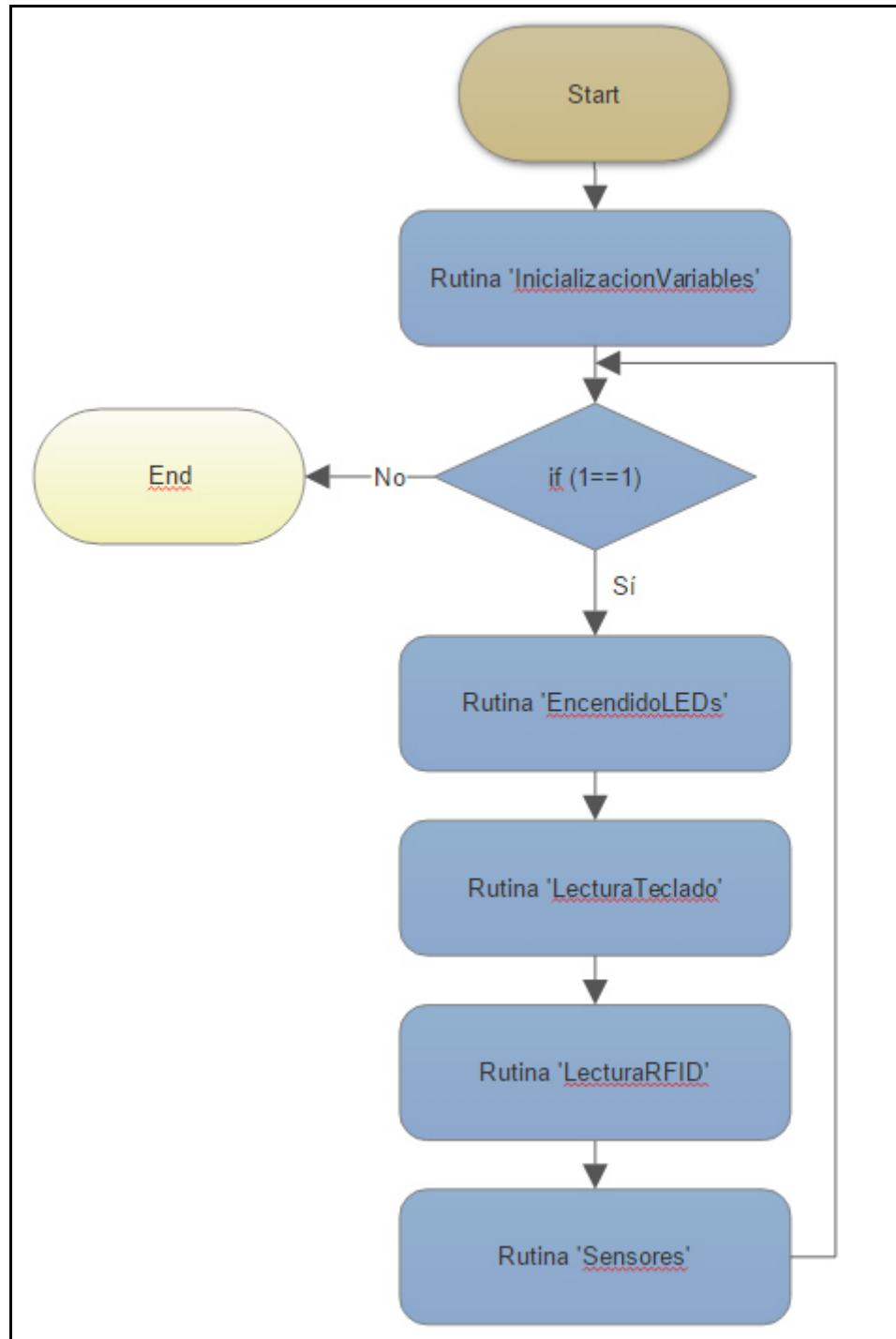


Figura 18: rutina “General”.

Este flujograma representa la rutina principal del Sistema, es decir, es la rutina que se ejecuta en cuanto se inicia el Sistema. En ella, tras la

inicialización de las variables globales que se van a usar a lo largo del resto de rutinas, se ejecuta un bucle infinito en el que se controla el encendido de LEDs de información, se lee el teclado, se lee el lector RFID y se leen los sensores de detección de movimiento y detección de apertura. Las distintas subrutinas, sd acorde al estado de todas las variables globales programarán la alarma, la iniciarán, la dispararán, la pararán, etc...

### Rutina Inicialización de Variables:

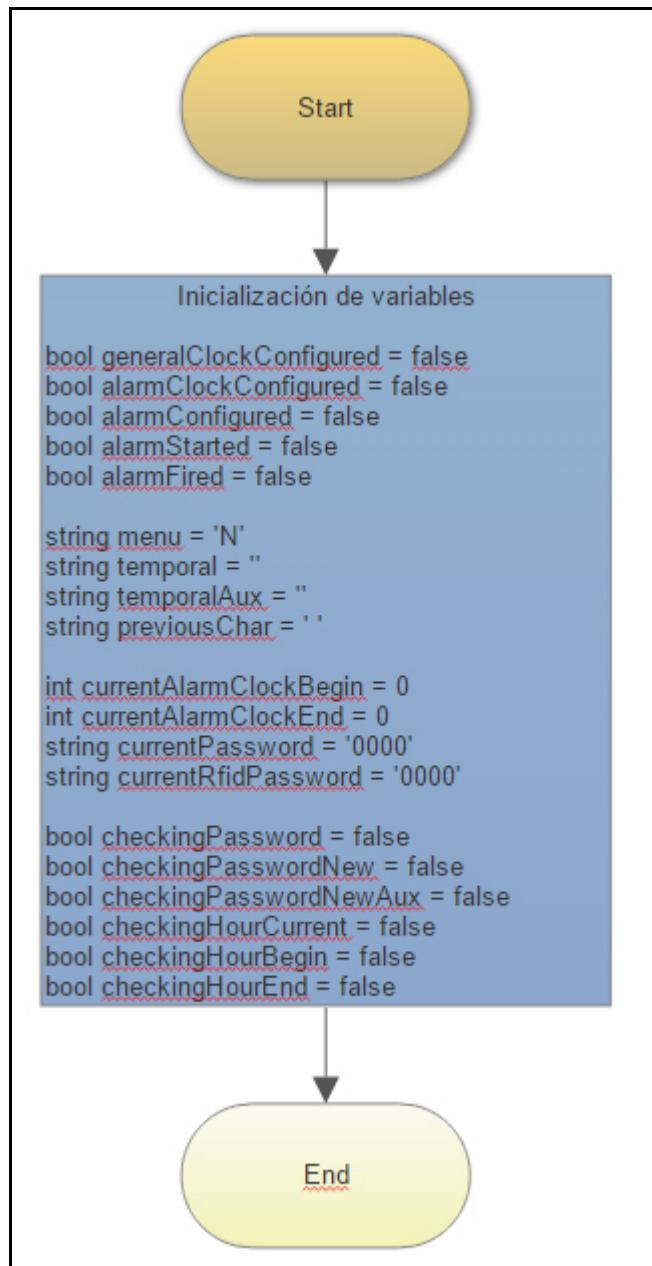


Figura 19: rutina “Inicialización de Variables”.

Este flujograma se encarga de definir e instanciar las variables globales que serán usadas (leídas y/o modificadas) por el resto de rutinas para el correcto funcionamiento global del Sistema. Se dispone de una serie de variables booleanas que controlan los distintos estados de la alarma: las

principales son “alarmConfigured” que indica si la alarma está configurada y puede ser arrancada, “alarmStarted” que indica si la alarma está iniciada o no y “alarmFired” que indica si la alarma ha sido disparada o no. Se dispone además de una serie de variables tipo string para el control del teclado y una serie de variables int en las que almacenar el password, el passwordRFID y las horas de inicio y fin de programación de alarma. Por último se inicializan una serie de variables booleanas cuyo nombre comienza por el prefijo “checking” que serán usadas en las distintas rutinas de teclado para conocer en qué paso concreto se encuentra cada uno de los menús (los nombres de las variables están escritos de tal manera que se agrupen las que tienen sentido, por ello por ejemplo en lugar de llamarse ‘checkingCurrentHour’, se llama ‘checkingHourCurrent’).

Nota: los tipos boolean, string e int son usados solo en pseudocódigo, en la implementación no coincidirán con estos.

### Rutina Encendido de LEDs:

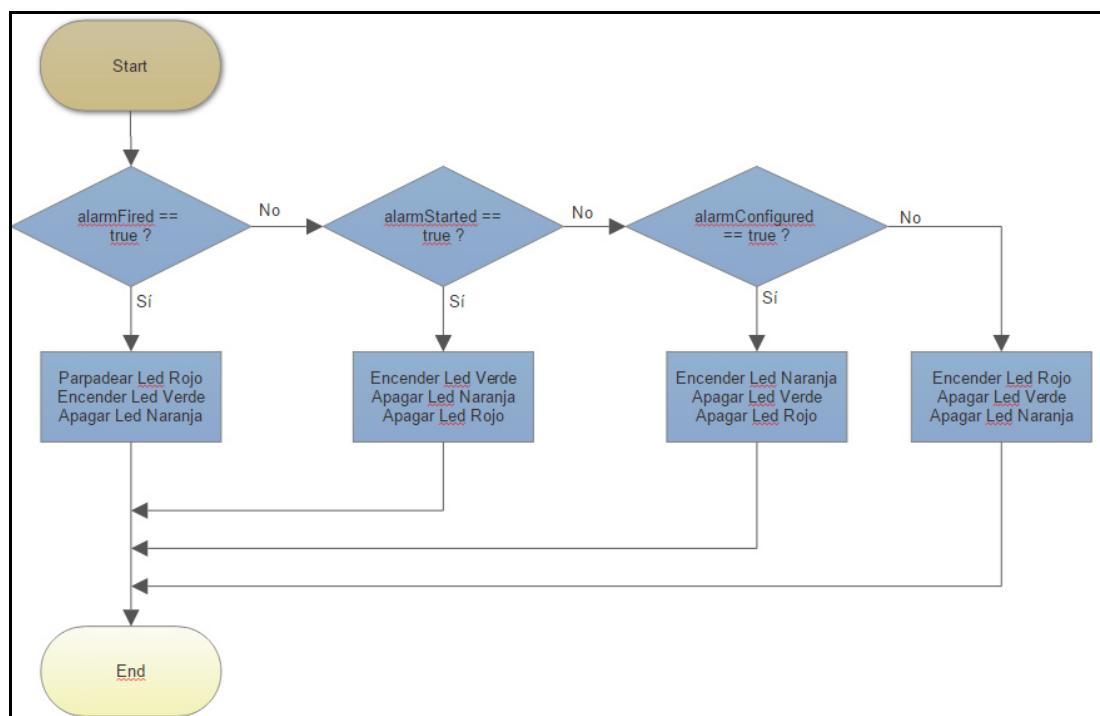


Figura 20: rutina “Encendido de LEDs”.

Como ya se comentó, una vez inicializadas las variables globales, la rutina General iniciará un bucle infinito y justo al inicio de este ejecutará la rutina de encendido de leds. Este flujograma representa una rutina sencilla que lee las variables de estado: si la alarma ha sido disparada encenderá los leds rojo y verde y apagará el resto, si no ha sido disparada pero está iniciada encenderá el led verde y apagará el resto, si sólo ha sido configurada encenderá el led naranja y apagará el resto, y si ni siquiera ha sido configurada encenderá el led rojo y apagará el resto.

## Rutina Lectura Teclado (General):

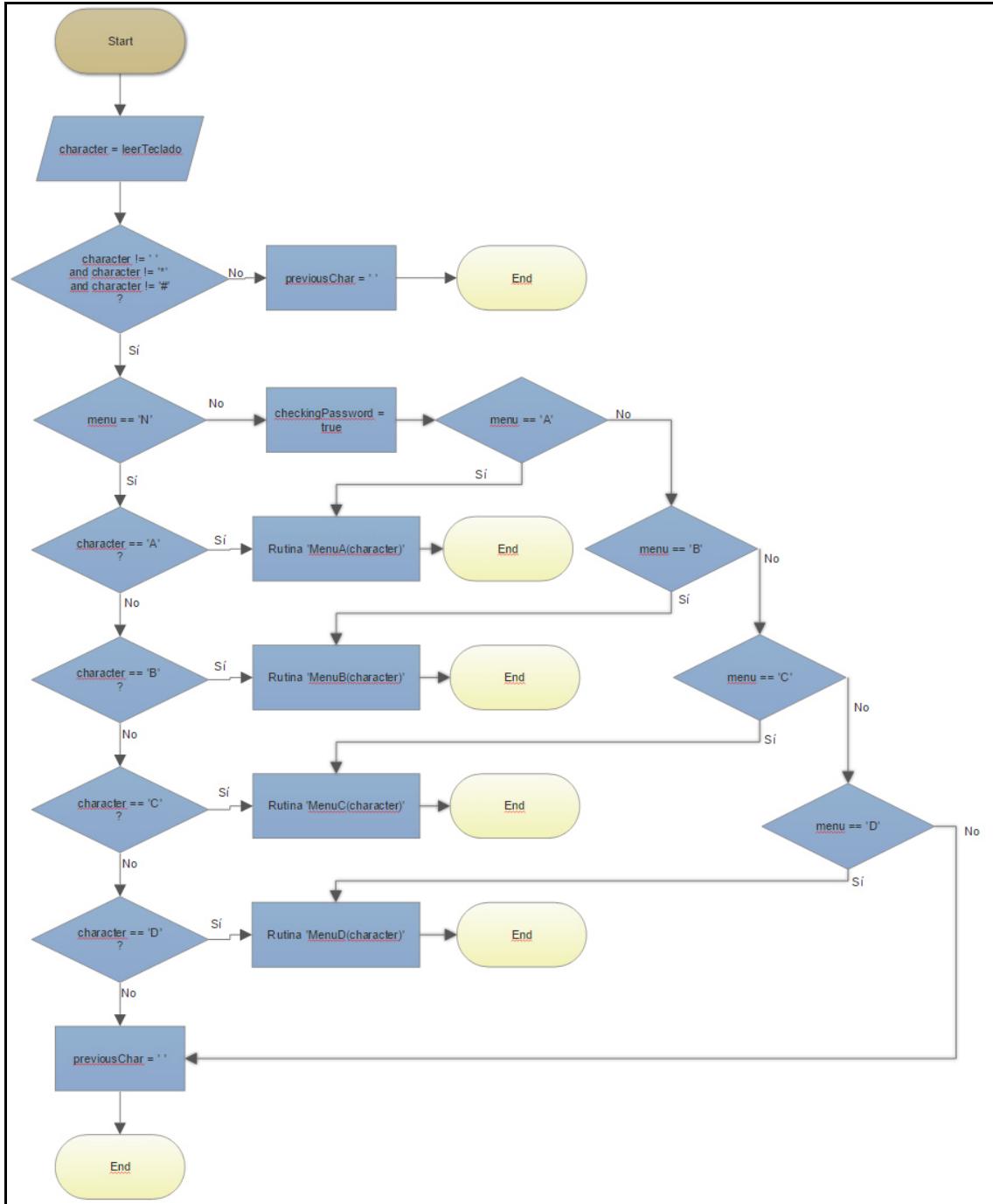


Figura 21: rutina “Lectura Teclado (General)”.

Este flujograma representa la rutina de lectura de teclado general, en la que se realiza una lectura del teclado y si se ha leído algo que no sea vacío, “\*\*” o “#” se elegirá a qué menú ir (A, B, C, D) si el Sistema no se encuentra en ninguno de ellos (menú == ‘N’) o ir al menú adecuado en el que se encuentra el Sistema, pasando el carácter leído. Aquí se ejecutarán una subrutina distinta (A, B, C o D).

## Rutina Leer Teclado Menú A:

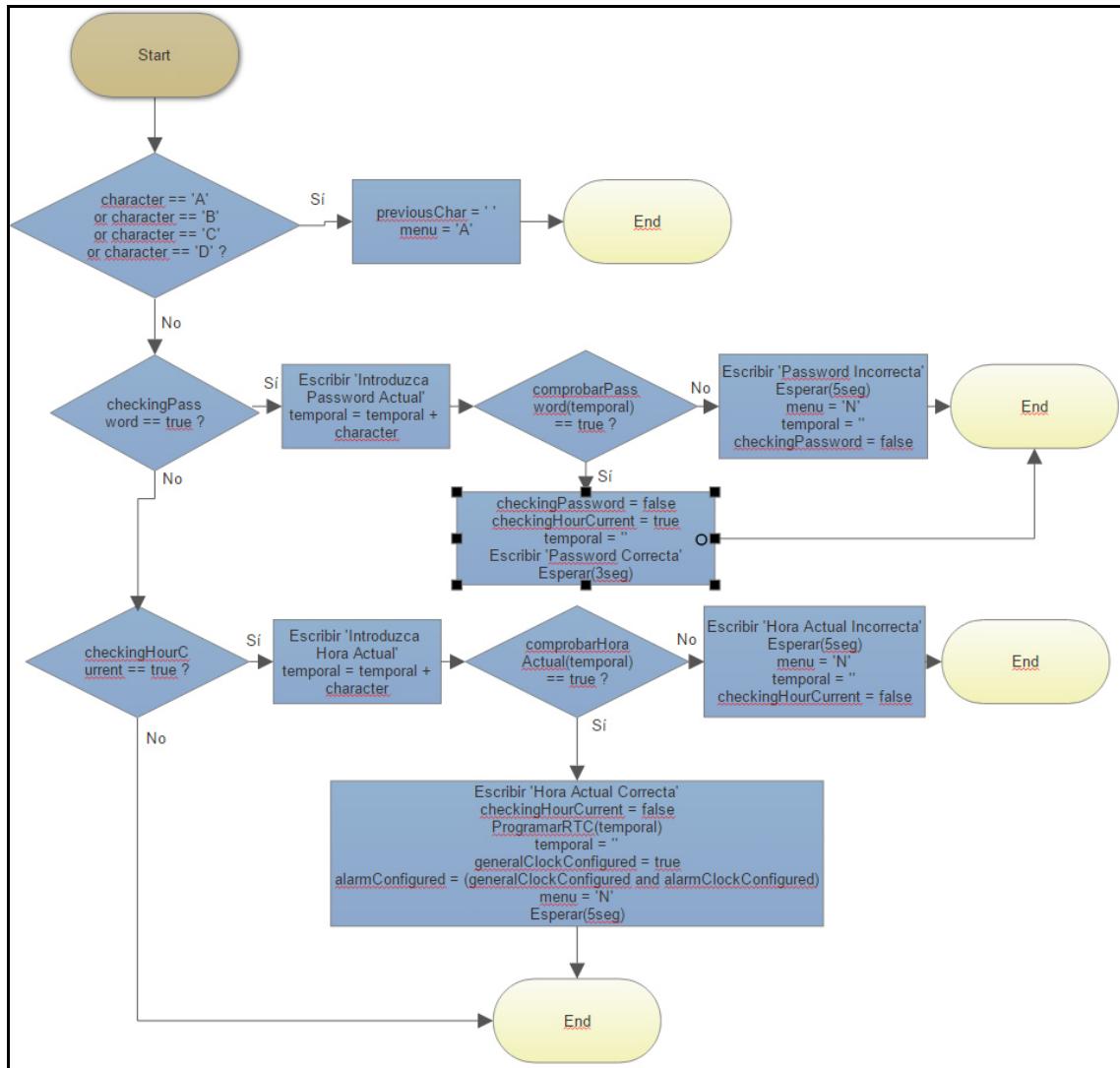
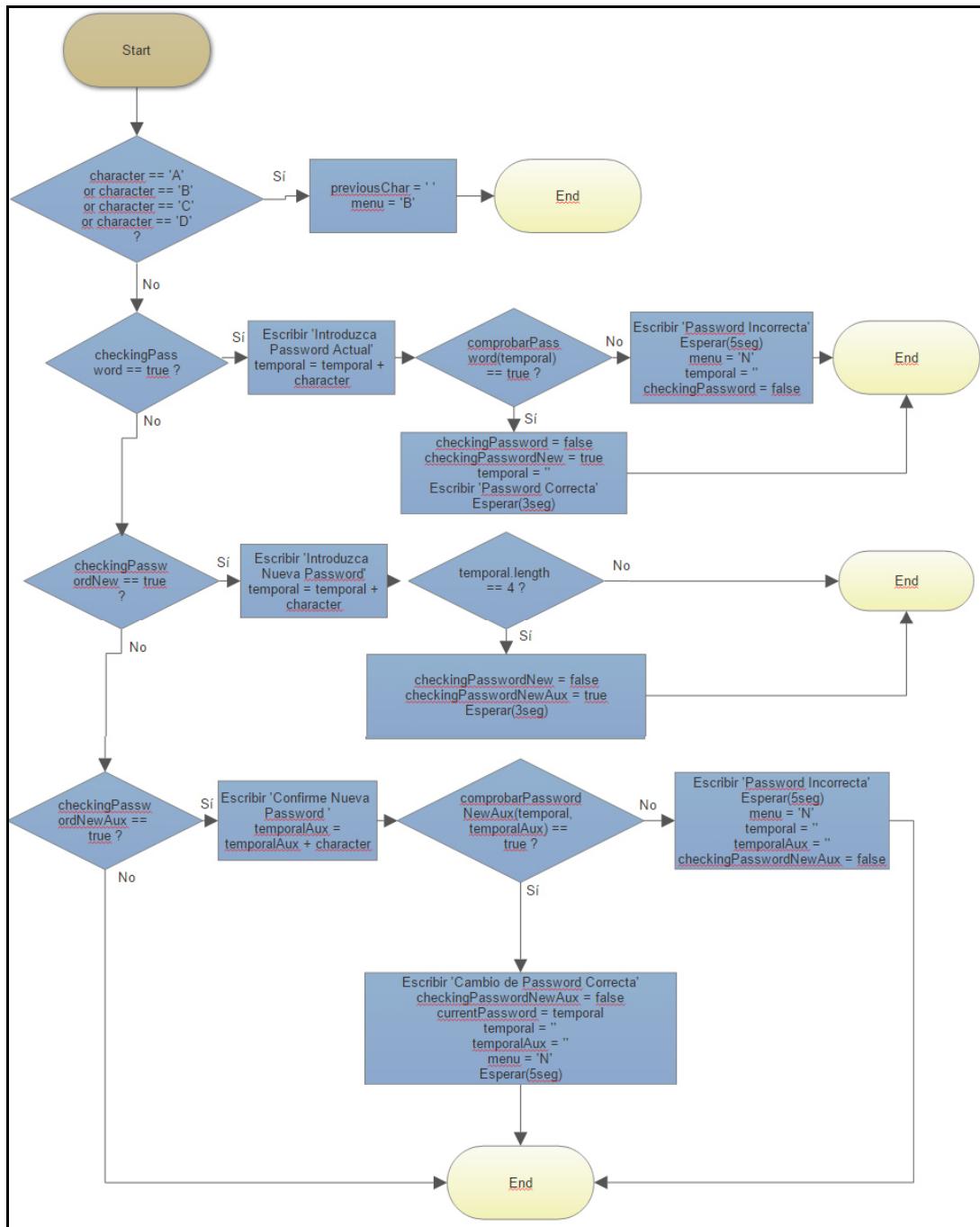


Figura 22: rutina “Lectura Teclado Menú A”.

Este flujograma será el que se ejecuta cuando se acceda al menú A. Dicho menú será usado para programar la hora del reloj interno de la alarma. La primera vez fijará la variable menú a 'A' y las siguientes veces hará lo siguiente:

- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe si coincide con la password guardada. Si no coincide mostrará un error y se saldrá del menú. Si coincide:
- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe que es una hora correcta. Si no es correcta se mostrará un error y se saldrá del menú. Si es correcta, se programará el RTC, se mostrará un mensaje de éxito y se saldrá del menú.

## Rutina Leer Teclado Menú B:



**Figura 23: rutina “Lectura Teclado Menú B”.**

Este flujo de rutina será el que se ejecuta cuando se acceda al menú B. Dicho menú será usado para cambiar la contraseña almacenada. La primera vez fijará la variable menú a 'B' y las siguientes veces hará lo siguiente:

- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe si coincide con la contraseña guardada. Si no coincide mostrará un error y se saldrá del menú. Si coincide:

-Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe que tiene una longitud 4. Una vez tiene longitud 4:

-Ir añadiendo el carácter leído a una variable temporal auxiliar hasta que se compruebe si coincide con la variable temporal anterior guardada. Si no coincide mostrará un error y se saldrá del menú. Si coincide: guardar variable temporal en variable global de “currentPassword”, mostrar mensaje de éxito y salir del menú.

### Rutina Leer Teclado Menú C:

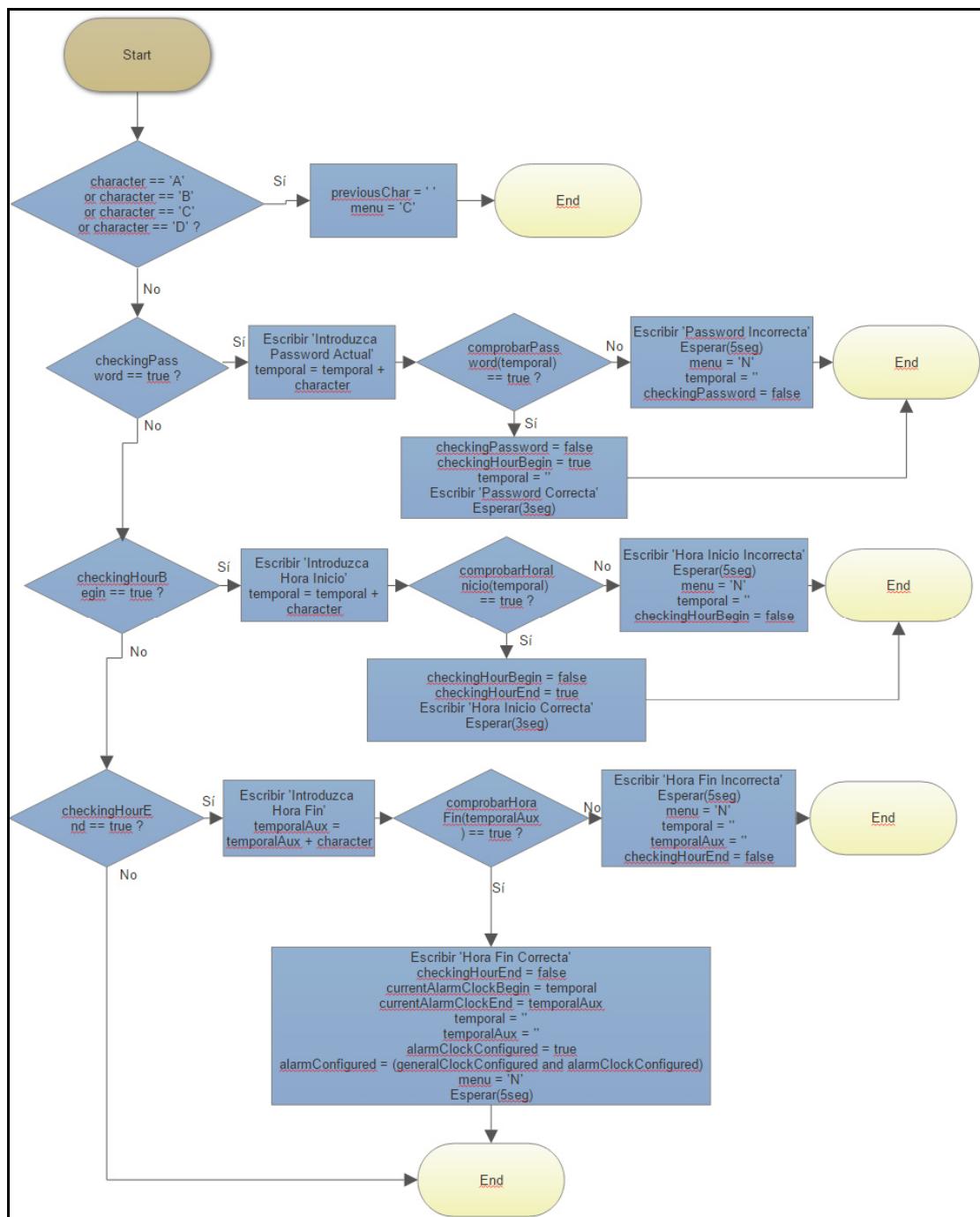


Figura 24: rutina “Lectura Teclado Menú C”.

Este flujograma será el que se ejecuta cuando se acceda al menú C. Dicho menú será usado para programar las horas de inicio y fin del rango de horas en el que la alarma estará activa. La primera vez fijará la variable menú a 'C' y las siguientes veces hará lo siguiente:

- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe si coincide con la contraseña guardada. Si no coincide mostrará un error y se saldrá del menú. Si coincide:
- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe que es una hora (inicio) correcta. Si no es correcta se mostrará un error y se saldrá del menú. Si es correcta:
- Ir añadiendo el carácter leído a una variable temporal auxiliar hasta que se compruebe que es una hora (fin) correcta. Si no es correcta se mostrará un error y se saldrá del menú. Si es correcta:
- Se guardará como hora de inicio de rango donde la alarma está activa el valor de la variable temporal y como hora de fin de rango donde la alarma está activa el valor de la variable temporal auxiliar. Se mostrará un mensaje de éxito y se saldrá del menú.

#### Rutina Leer Teclado Menú D:

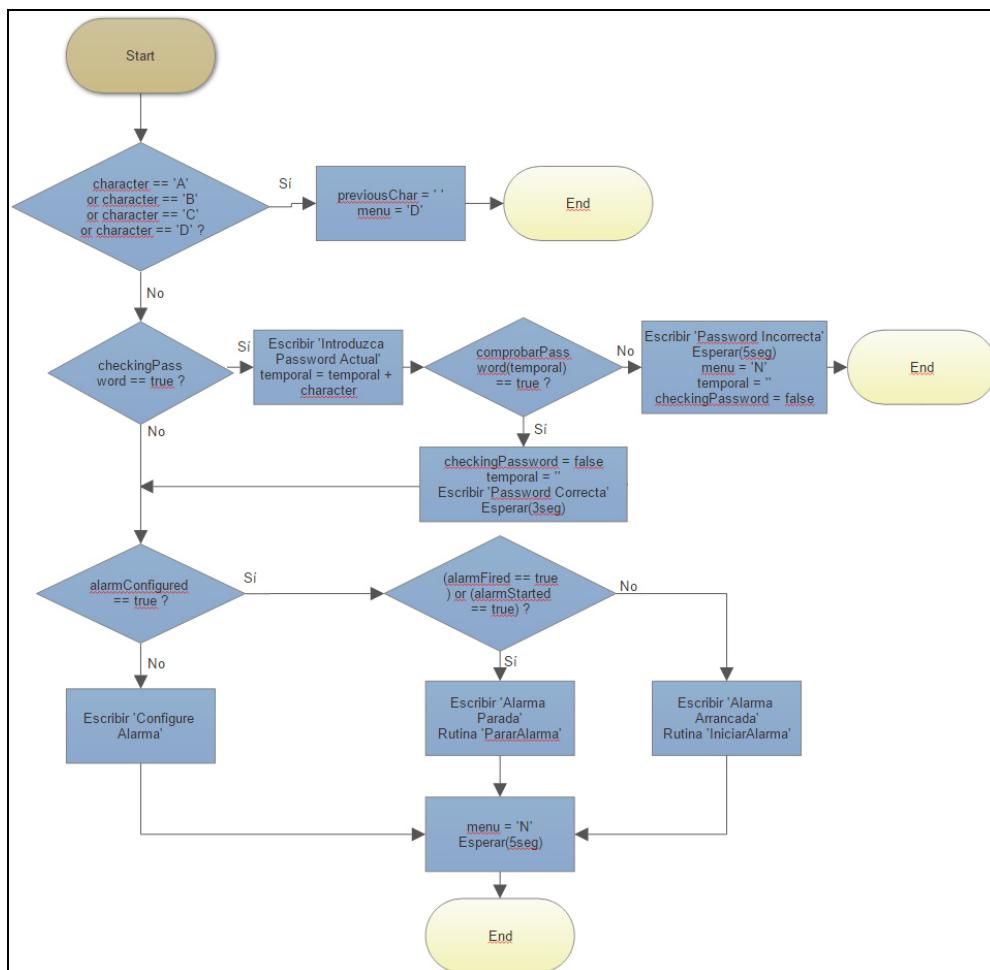


Figura 25: rutina “Lectura Teclado Menú D”.

Este flujograma será el que se ejecuta cuando se acceda al menú D. Dicho menú será usado iniciar la alarma si está parada o pararla si está iniciada o disparada. La primera vez fijará la variable menú a 'D' y las siguientes veces hará lo siguiente:

- Ir añadiendo el carácter leído a una variable temporal hasta que se compruebe si coincide con la contraseña guardada. Si no coincide mostrará un error y se saldrá del menú. Si coincide:
- Si la alarma no está configurada, escribirá un mensaje de error y se saldrá. Si la alarma está configurada:
- Si la alarma ha sido disparada o arrancada ejecutará la rutina para parar la alarma. Mostrará un mensaje de éxito y saldrá del menú.
- En otro caso, ejecutará la rutina para iniciar alarma, mostrará un mensaje de éxito y saldrá del menú.

#### Rutina Leer RFID:

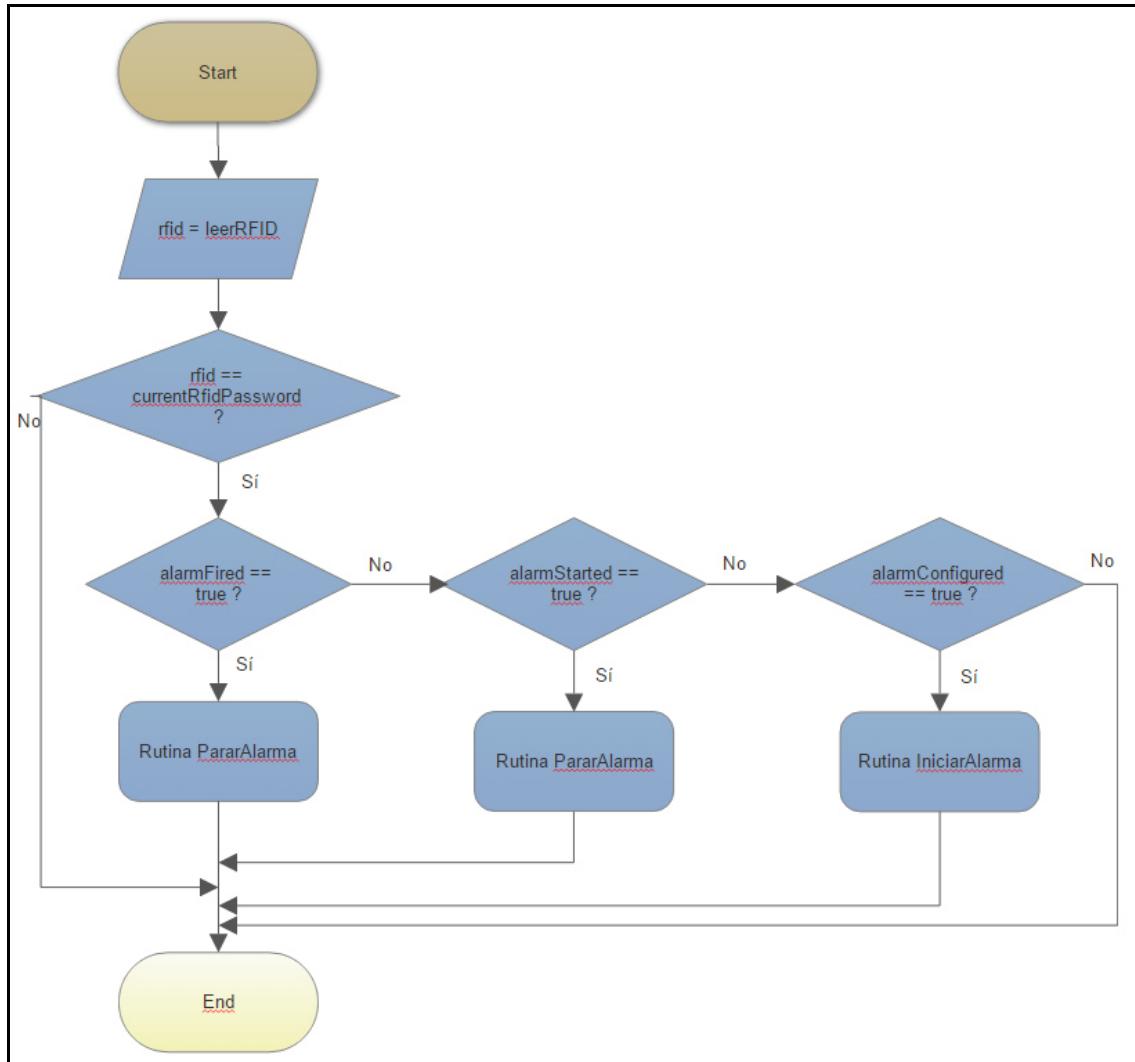


Figura 26: rutina “Leer RFID”.

Este flujograma corresponde con la rutina de lectura RFID. En primer lugar hace una lectura del dispositivo RFID. Si la lectura coincide con la contraseña RFID configurada se realizarán las siguientes configuraciones:

- Si la alarma ha sido disparada o iniciada, se ejecutará la rutina para parar la alarma.
- Si la alarma ha sido configurada, se ejecutará la rutina para iniciar la alarma. En caso contrario no se ejecutará nada.

#### Rutina Lectura de Sensores:

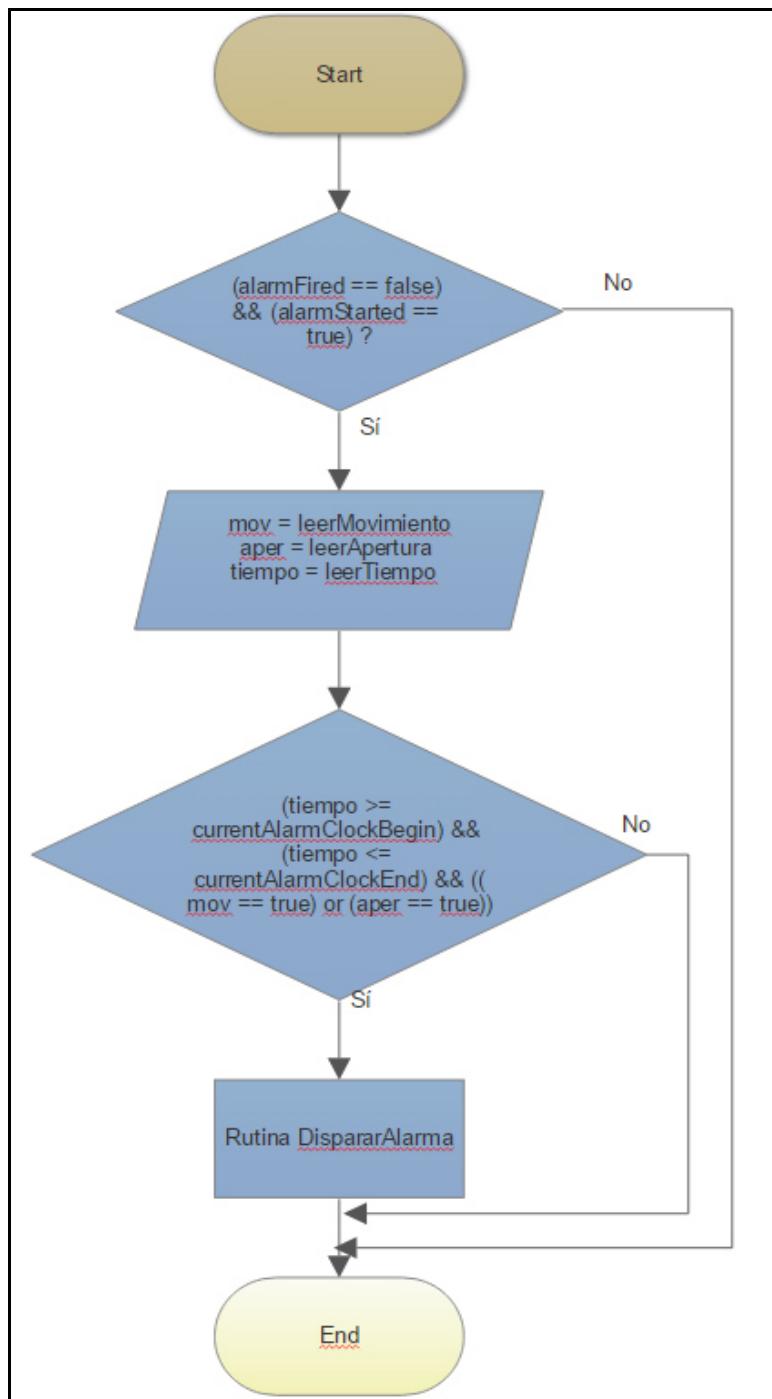


Figura 27: rutina “Lectura de Sensores”.

Este flujograma será el encargado de ejecutar la lectura de los sensores de movimiento y apertura, junto con el tiempo extraído del RTC. La lectura de estos tres dispositivos sólo se realizará si la alarma no ha sido disparada (si ha sido disparada, esta rutina no debe cambiar su estado) y la alarma ha sido iniciada. En tal caso, tras leer los valores, si se detecta movimiento o apertura, y el tiempo actual está dentro del rango en el que la alarma se encuentra activa, se ejecutará la rutina que dispara la alarma.

### Rutinas de control de alarma:

#### Rutina Disparar Alarma - Rutina Parar Alarma - Rutina Iniciar Alarma

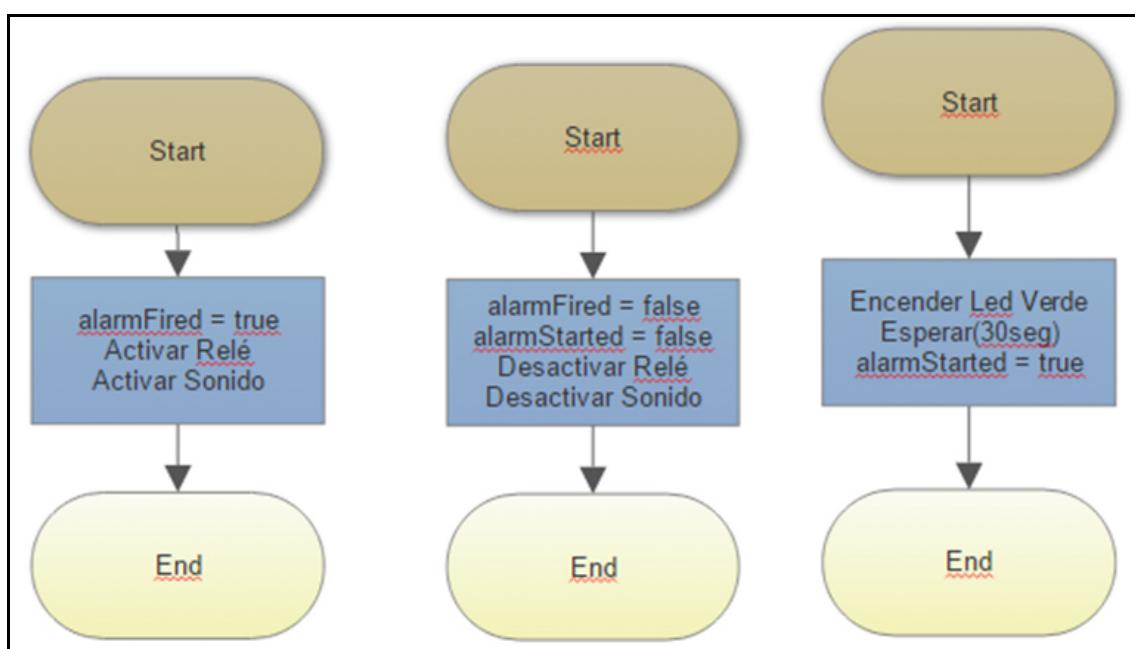


Figura 28: rutinas de control de alarma.

Estos tres flujogramas son utilizados para realizar el control de la alarma:

- La rutina “disparar alarma” marca el estado de la alarma como disparado, activa el relé que activará la señal luminosa y la señal acústica amplificada, y activa la salida para la señal acústica no amplificada.
- La rutina “parar alarma” marca el estado de la alarma como no disparado y no iniciado, desactiva el relé que activa la señal luminosa y la señal acústica amplificada, y desactiva la salida para la señal acústica no amplificada.
- La rutina “iniciar alarma” enciende automáticamente el led verde, detiene el Sistema 30 segundos (para que dé tiempo a salir del edificio por ejemplo) y marca el estado de la alarma como iniciado. El led es activado justo al comenzar para indicar que la alarma está iniciada incluso durante los 30 segundos de espera.

# Capítulo 4.- Implementación.

El objetivo de este capítulo es describir cómo se ha realizado el proceso de implementación del diseño creado para el Sistema conocido como “HomeAlarm”. Estará comprendido por cuatro apartados, que serán independientes pero relacionados entre sí. Estos apartados serán “Selección de componentes”, “Interconexionado de componentes”, “Programación de Librerías” y “Plan de Pruebas”.

A modo de resumen-introducción decir que en el apartado “Selección de componentes” se realizará un estudio sobre que componentes compatibles con la plataforma STM32F4 Discovery son los ideales para realizar todas las funcionalidades propuestas, atendiendo tanto a compatibilidad como a un criterio económico (se recuerda que uno de los objetivos es conseguir implementar el Sistema sin sobrepasar un presupuesto concreto).

Por otra parte, el apartado “Interconexionado de componentes” los componentes antes seleccionados serán conectados en los puertos adecuados de la placa STM32F4 Discovery atendiendo a la distinta naturaleza de cada uno de estos. Se presentará el layout físico final.

En el apartado “Programación de librerías” se describirá como han sido programados los flujogramas presentados en el apartado “Diseño de Arquitectura” para poder interactuar correctamente con los distintos componentes obteniendo finalmente las funcionalidades objetivo.

Por último, el apartado “Plan de Pruebas” describe las pruebas que han sido definidas al Sistema (objetivo, precondiciones, pasos y post condiciones) y el resultado de la ejecución de estas junto con comentarios acerca de este resultado.

#### 4.1.- Selección de componentes.

A continuación, se realizará un estudio sobre que componentes compatibles con la plataforma STM32F4 Discovery son los ideales para realizar todas las funcionalidades propuestas, atendiendo tanto a compatibilidad como a un criterio económico (se recuerda que uno de los objetivos es conseguir implementar el Sistema sin sobrepasar un presupuesto concreto). Como punto de partida, se calculará qué parte del presupuesto se ha consumido antes de añadir ningún componente.

Por un lado se tiene ya definido como corazón de la arquitectura la placa STM32F4 Discovery cuyo precio en el siguiente enlace [21] es 21,67€ gastos de envío incluidos:

[http://www.aliexpress.com/item/Free-Shipping-STM32-DISCOVERY-Board-STM32F401C-DISCO-STM32F4-kit-STM32F401VC-Learning-STM32-Development-Board/2051908613.html?spm=2114.01010208.3.2.o4J8Qf&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=810b41ef-3e6e-4053-af1b-7fce94d6ee76](http://www.aliexpress.com/item/Free-Shipping-STM32-DISCOVERY-Board-STM32F401C-DISCO-STM32F4-kit-STM32F401VC-Learning-STM32-Development-Board/2051908613.html?spm=2114.01010208.3.2.o4J8Qf&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=810b41ef-3e6e-4053-af1b-7fce94d6ee76)



Figura 29: venta STM32F4 Discovery.

El empaquetado del Sistema es una caja de plástico de dimensiones 18cm x 14cm x 5cm. La fabricación de dicha caja mediante impresión 3d puede ser solicitada a Mediamarkt Málaga por 3 euros. Es obvio que al por mayor sería más barata pero no es un mal precio por comprar una sola unidad.

Se necesita una fuente de alimentación independiente de 5v y que entregue una corriente de 7A. Tras una búsqueda exhaustiva en internet se llega a la conclusión de que la fuente "MS-35-5" [22] puede ser la adecuada:

<http://www.aliexpress.com/item/OSGD-high-quality-MS-35W-5V-7A-min-size-Small-scale-Switch-Power-supple-free->

[shopping/32613852436.html?spm=2114.01010208.3.182.14DCZA&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=f27e2dea-70be-491d-a62b-c7bd7f95fcab](http://www.aliexpress.com/item/32613852436.html?spm=2114.01010208.3.182.14DCZA&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=f27e2dea-70be-491d-a62b-c7bd7f95fcab)



Figura 30: venta fuente “MS-35-5”.

Necesitará de una caja para empaquetarla (1 euro), 1 cable para conectarla a la corriente eléctrica de 220v (50 céntimos) y un cable miniUSB que vaya de la fuente al empaquetado (50 céntimos), lo que hace un coste total de 9,47€. Se estudio integrarla dentro del empaquetado del Sistema, para ahorrar el coste de su propia caja, pero se desechó por dos motivos: puede que el calor que desprenda perjudique al Sistema y en caso de averiarse es más fácil sustituirla siendo externa.

Otra cosa fija ya seleccionada son los leds de colores, un total de 3 con un coste cada uno de 1 céntimo de euro aproximadamente. No se pueden usar los de la placa ya que esto implicaría dejar partes de la placa “al aire”, sin empaquetar. Se compra el siguiente pack [23] de 100 piezas por 87 céntimos:

[http://www.aliexpress.com/item/100pcs-3mm-White-Green-Red-Blue-Yellow-LED-Light-Bulb-Emitting-Diode-Lamps-Brand-new/32615755061.html?spm=2114.01010208.3.31.RCHhv8&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=fe349219-f5bd-4292-b131-7fd2cc74c574](http://www.aliexpress.com/item/100pcs-3mm-White-Green-Red-Blue-Yellow-LED-Light-Bulb-Emitting-Diode-Lamps-Brand-new/32615755061.html?spm=2114.01010208.3.31.RCHhv8&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=fe349219-f5bd-4292-b131-7fd2cc74c574)

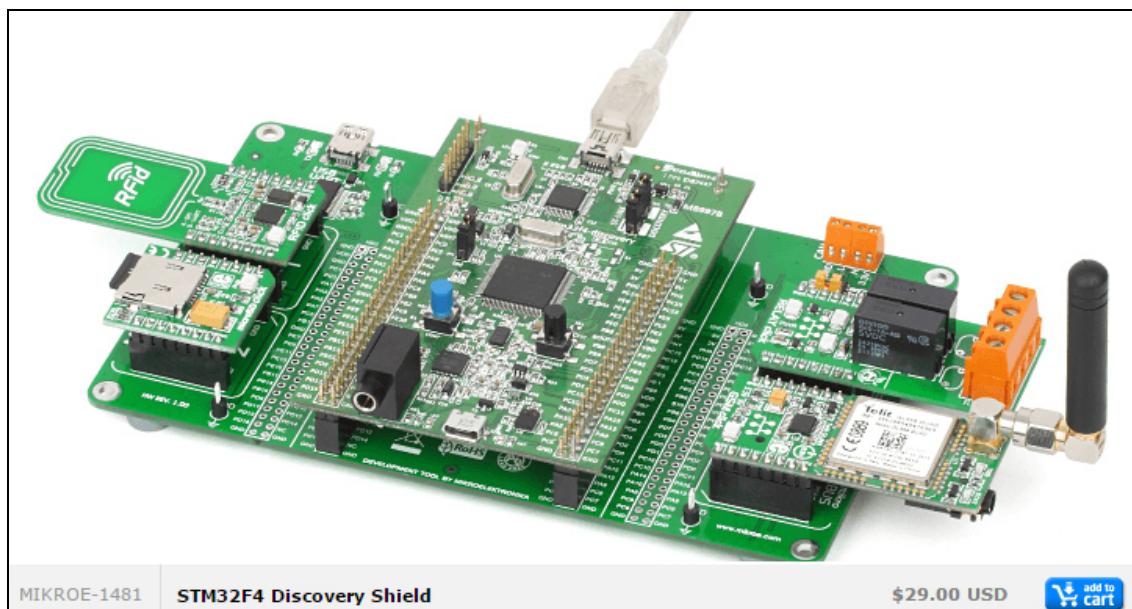


Figura 31: venta de LEDs.

Así, ya se tienen definidos los gastos fijos antes de seleccionar el resto de componentes (dispositivos que irán conectados a la placa principal):

Coste base =  $21,67\text{€} + 3\text{€} + 9,47\text{€} + 0,87 = 35,01\text{€}$ . Si se tiene en cuenta que el presupuesto total es de 70€, se tiene disponible 34,99€ para el resto de componentes.

Tras un primer vistazo a la página de componentes Mikroe [24], existe una opción muy sencilla en cuanto a desarrollo se refiere, que consistiría en unirle a la placa STM32F4 Discovery, la placa de investigación STM32F4 Discovery Shield, que añade una serie de conexiones Mikrobus, en las cuales se pueden conectar mediante simples clicks componentes de evaluación totalmente diseñados para trabajar con la plataforma, y de para los que existen multitud de librerías de control ya implementadas. Se ganaría con ello compatibilidad, fácil conexión y rápido desarrollo.



**Figura 32: venta de placa STM32F4 Discovery Shield.**

En la figura se puede ver la placa de ampliación conectada a la placa original y sobre la que se han conectado mediante simples clicks un lector RFID, un lector de tarjetas MicroSD, un transmisor/receptor GSM y un relé. La placa además ofrece un USB adicional y no se pierde ningún puerto original. Más puertos MikroBus Click pueden ser añadidos mediante un expansor de clicks.

Sin embargo, un simple vistazo a los precios de esta placa extensión y de los componentes de ese estilo hace que esta idea se abandone [24]:

STM32F4 Discovery Shield: 29 dólares → <http://www.mikroe.com/stm32/stm32f4-discovery-shield/>  
 RTC: 21 dólares → <http://www.mikroe.com/click/rtc/>  
 Relé: 18 dólares → <http://www.mikroe.com/click/relay/>  
 RFID: lector 29 dólares, tarjeta 2 dólares → <http://www.mikroe.com/click/rfid/>  
 Detector de movimiento: 13 dólares → <http://www.mikroe.com/click/motion/>  
 Detector de apertura (mediante detector magnético): 36 dólares → <http://www.mikroe.com/click/magneto/>

Como se puede apreciar claramente excede el presupuesto y eso que aún no se ha seleccionado una pantalla de datos y un teclado. Es una pena su alto precio ya que la idea es muy buena por lo fáciles que son de conectar este tipo de sensores y el soporte en forma de librerías que mikroe ofrece.

Tras este jarro de agua fría, se decide seguir una estrategia diferente para la selección de componentes: buscar dispositivos no oficiales y que sean compatibles con la placa o se puedan hacer de alguna forma compatible con ella.

En primer lugar se ha seleccionado este display [25] para mostrar los datos. Se trata de un display LCD azul retroiluminado, que puede mostrar dos líneas de 16 caracteres cada una (cumple la especificación de 32 caracteres) cuyo precio es 1,23€, más 0,50€ para un plástico duro transparente para protegerlo, total 1,73€:

<http://www.aliexpress.com/item/1pcs-New-Character-LCD-Module-Display-LCM-1602-LCD1602-16X2-HD44780-Blue-Blacklight-Original-and-New/32573086114.html>



**Figura 33: venta de LCD 16x2.**

Como teclado se ha seleccionado el siguiente keypad [26] de 4 filas por 4 columnas, cuyo precio es 0,81€:

[http://www.aliexpress.com/item/New-4-4-Matrix-Array-Matrix-Keyboard-16-Key-Membrane-Switch-Keypad/32255922477.html?spm=2114.01010208.3.2.KAYzAG&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10](http://www.aliexpress.com/item/New-4-4-Matrix-Array-Matrix-Keyboard-16-Key-Membrane-Switch-Keypad/32255922477.html?spm=2114.01010208.3.2.KAYzAG&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10)

032 10009 10020 10008 10018 10019 101,searchweb201603 2&btsid=a52  
12096-e28f-498c-8a5a-9bf8f70cd8b0



Figura 34: venta de keypad 4x4.

Como Real Time Controller (RTC) ha sido seleccionado este componente [27] que puede ser conectado al bus I2C de la placa y cuyo precio es 0,60€:

[http://www.aliexpress.com/item/Free-shipping-20pcs-lot-The-Tiny-RTC-I2C-modules-24C32-memory-DS1307-clock-RTC-module-for/1876368739.html?spm=2114.01010208.3.2.QdefZn&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_1009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=979cf487-cf43-42f4-8f05-89c8eef63c7](http://www.aliexpress.com/item/Free-shipping-20pcs-lot-The-Tiny-RTC-I2C-modules-24C32-memory-DS1307-clock-RTC-module-for/1876368739.html?spm=2114.01010208.3.2.QdefZn&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_1009_10020_10008_10018_10019_101,searchweb201603_2&btsid=979cf487-cf43-42f4-8f05-89c8eef63c7)



Figura 35: venta de RTC.

Como relé ha sido seleccionado el siguiente componente [28] creado para Arduino Uno pero perfectamente compatible con la placa STM32F4 Discovery. Su precio es de 1,10€:

[http://www.aliexpress.com/item/Relay-module-for-arduino-uno/32400577838.html?spm=2114.01010208.3.139.I424rc&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=53747513-9b08-412a-8d26-cfa7f4512757](http://www.aliexpress.com/item/Relay-module-for-arduino-uno/32400577838.html?spm=2114.01010208.3.139.I424rc&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=53747513-9b08-412a-8d26-cfa7f4512757)



Figura 36: venta de relé.

Como sensor de movimiento se ha seleccionado el siguiente componente [29] compatible con Arduino y Raspperi Pi y por supuesto con STM32F4 Discovery. Su coste es de 1€, al que habría que añadir 1 euro adicional para encapsularlo en una pequeña caja de plástico fino, siendo el total 2€:

[http://www.aliexpress.com/item/1pcs-High-Quality-HC-SR501-Infrared-PIR-Motion-Sensor-Module-For-Arduino-Raspberry-pi/32558562655.html?spm=2114.01010208.3.11.flHkwp&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_1\\_0009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=323d6637-56aa-478d-b672-3ed74f74a62e](http://www.aliexpress.com/item/1pcs-High-Quality-HC-SR501-Infrared-PIR-Motion-Sensor-Module-For-Arduino-Raspberry-pi/32558562655.html?spm=2114.01010208.3.11.flHkwp&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_1_0009_10020_10008_10018_10019_101,searchweb201603_2&btsid=323d6637-56aa-478d-b672-3ed74f74a62e)



Figura 37: venta de sensor de movimiento.

Como detector de apertura, se decide que el tipo de sensor que mejor puede servir para monitorizar dicha función es un sensor magnético colocado en el marco de la puerta o ventana, que tenga un imán colocado en la puerta o la ventana. Cuando la puerta está cerrada detectará el campo magnético y si deja de detectarlo es que se encuentra abierta. Se ha seleccionado el siguiente componente [30] cuyo coste es 0,96€, al que habría que añadir 1 euro adicional para encapsularlo en una pequeña caja de plástico fino, y 0,50€ para el imán a colocar en la puerta/ventana siendo el total 2,46€:

<http://www.aliexpress.com/item/Standard-Hall-Magnetic-Sensor-Module-for-Arduino-AVR-PIC-Good-KY-003-New->

[Wholesale/32665617510.html?spm=2114.01010208.3.10.ltYyeu&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_1\\_0032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=b377c23e-56fb-4d04-a124-a6fce4858c6](http://www.aliexpress.com/item/Wholesale-32665617510.html?spm=2114.01010208.3.10.ltYyeu&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_1_0032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=b377c23e-56fb-4d04-a124-a6fce4858c6)



**Figura 38: venta de sensor magnético.**

Como lector RFID se ha seleccionado el siguiente componente [31] que va conectado al interfaz SPI. Tiene una tasa de transmisión máxima de 10Mbit/s y el paquete contiene el lector RFID, una tarjeta RFID compatible y un llavero RFID. Su precio es de 1,99€, al que habría que añadir 1 euro adicional para encapsularlo en una pequeña caja de plástico fino, siendo el total 2,99€:

<http://www.aliexpress.com/item/Free-shipping-MFRC-522-RC522-RFID-RF-IC-card-sensor-module-to-send-S50-Fudan-card/1623810751.html?spm=2114.01010208.8.64.FHSBMG>



**Figura 39: venta de lector y tarjeta RFID.**

A continuación se realizará una tabla resumen de todos los componentes seleccionados con su precio, para comprobar si se cumplen completamente los requisitos de presupuesto (coste total inferior a 70€). La tabla resumen es la siguiente:

Componente	Precio
Placa STM32F4Discovery + caja empaquetado	24,67€
Fuente de Alimentación MS-35-5 + caja + cable 220v + cable miniUSB	9,47€
LCD 16x2	1,73€
Keypad 4x4	0,81€
Real Time Controller RTC	0,60€
Relé	1,10€
Sensor de movimiento	2,00€
Sensor magnético	2,46€
Lector RFID + tarjeta RFID + llavero RFID	2,99€
Componentes: leds, cables, potenciómetro, miniboards de conexión, conectores hembra para chasis, conectores macho aéreos.	10,0€
Empaquetado y manual de usuario	4,0€
<b>Total</b>	<b>59,83€</b>
<b>Sobrante</b>	<b>10,17€</b>

**Tabla 01: Precios de los Componentes del Sistema HomeAlarm.**

Como se puede observar la opción seleccionada es mucho más económica que la opción de comprar componentes MikroBUS. Aunque tendrá más trabajo de desarrollo es más interesante porque ayuda a conseguir el objetivo de no incumplir el presupuesto. Es más, es seguro que esta opción puede ser abaratada más aun comprando grandes cantidades y ajustando las partidas de componentes y empaquetado y manual de usuario. El sobrante puede ser invertido en los dispositivos para la señal acústica y la luminosa, aunque no forme parte de las especificaciones del Sistema.

#### **4.2.- Interconexionado de componentes.**

Una vez seleccionados los componentes específicos que van a ser usados para desarrollar el proyecto, se deben decidir cómo van a ser conectados a la placa STM32F4 Discovery ya que dada la gran cantidad de pines entrada que tiene, existen múltiples posibilidades.

Por un lado, los componentes tienen diferentes tipos de conexión para intercambio de datos: unos usarán un simple pin de entrada o salida (leds, sensor de movimiento, sensor magnético, relé, teclado 8 pines independientes, display LCD), otros irán conectados al bus I2C (RTC), otros al bus SPI (lector RFID).

Para los puertos I2C se tienen las siguientes posibilidades:

I2Cx	Pinspack 1		Pinspack 2		Pinspack 3	
	SCL	SDA	SCL	SDA	SCL	SDA
I2C1	PB6	PB7	PB8	PB9	PB6	PB9
I2C2	PB10	PB11	PF1	PF0	PH4	PH5
I2C3	PA8	PC9	PH7	PH8	-	-

**Tabla 02: Distribución de pines STM32F4 Discovery en puertos I2C.**

Para los puertos SPI se tienen las siguientes posibilidades:

SPIx	Pinspack 1			Pinspack 2			Pinspack 3		
	MOSI	MISO	SCK	MOSI	MISO	SCK	MOSI	MISO	SCK
SPI1	PA7	PA6	PA5	PB5	PB4	PB3			
SPI2	PC3	PC2	PB10	PB15	PB14	PB13	PI3	PI2	PI0
SPI3	PB5	PB4	PB3	PC12	PC11	PC10			
SPI4	PE6	PE5	PE2	PE14	PE13	PE12			
SPI5	PF9	PF8	PF7	PF11	PH7	PH6			
SPI6	PG14	PG12	PG13						

**Tabla 03: Distribución de pines STM32F4 Discovery en puertos SPI.**

Por otro lado, todos los componentes usan un pin GND y uno de alimentación, pero unos serán alimentados a +5v (display LCD, sensor de movimiento, sensor magnético, relé) y otros a +3.3v (lector RFID, RTC) o no tienen alimentación (leds, keypad). Se observa que hay 4 componentes que usarán las salidas de +5v de la placa, pero esta solo dispone de dos salidas, por lo que tendrán que ser multiplicadas exteriormente (mediante cableado). Los componentes que funcionan a +3,3v no tienen dicho problema porque son sólo dos y pueden ir conectados directamente a las dos salidas +3,3v de la placa. Con respecto a GND, los 9 componentes que la necesitan (todos menos el keypad) pueden ir conectados directamente cada uno a una de los pines GND que ofrece la placa (hay 10 por lo que sobra 1).

Con respecto al tipo de cableado a usar para conectar los componentes a la placa, el lector RFID, el relé, el sensor de movimiento, el sensor magnético y los 3 leds usarán cables hembra-hembra. El RTC, el keypad 4x4 y el display LCD usarán cableado macho-hembra. En este punto cabe destacar que parece que la placa no está muy bien diseñada para esto, ya que los pines son mucho más cortos en la cara superior que en la cara inferior. Esto hace que conectarlos por la cara superior sea difícil ya que se sueltan con bastante facilidad, y tampoco sea fácil conectarlos por la cara inferior ya que aunque no se sueltan, en esta cara no están numerados y hay que comprobar muy bien que se está conectando el pin deseado.

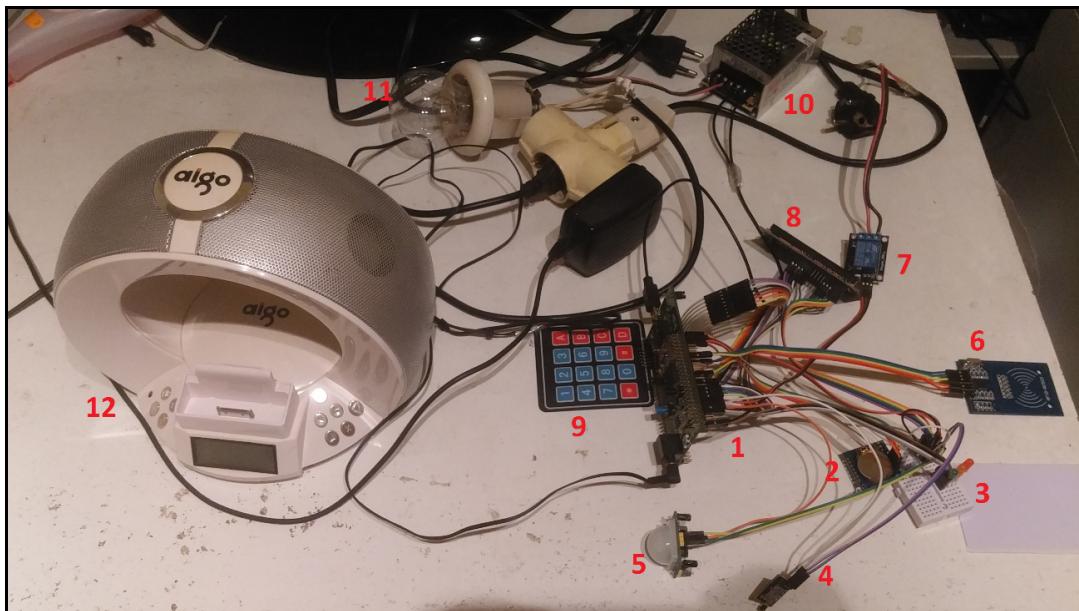
Una vez realizadas estas clasificaciones de los componentes, se asignarán pines a cada uno de ellos. Para numerar los pines se ha seguido el siguiente criterio: se contarán los pines verticalmente de menor a mayor, existiendo 4 columnas verticales en las que la menor es la de la izquierda y la mayor la de la derecha (con la placa puesta de tal forma que el serigrafiado se lee correctamente). De esta forma la primera columna contiene los pines del 1 al 25, la segunda del 26 al 50, la tercera del 51 al 75 y la cuarta del 76 al 100. La siguiente tabla muestra la asignación de pines:

Periférico	Pin Periférico	Pin STM32F4 Discovery
Display LCD 16X2	VSS	PIN12 (GND)
	VDD	PIN52 (5V)
	V0	PIN52 (5V)
	RS	PIN93 (PD0)
	RW	PIN92 (PD2)
	E	PIN91 (PD4)
	D0	NO CONECTADO
	D1	NO CONECTADO
	D2	NO CONECTADO
	D3	NO CONECTADO
	D4	PIN67 (PD1)
	D5	PIN66 (PD3)
	D6	PIN65 (PD5)
	D7	PIN64 (PD7)
Teclado keypad 4x4	A	PIN52 (5V)
	K	PIN12 (GND)
Teclado keypad 4x4	PIN0	PIN31 (PA0)
	PIN1	PIN06 (PA1)
	PIN2	PIN32 (PA2)
	PIN3	PIN07 (PA3)
	PIN4	PIN33 (PA4)
	PIN5	PIN08 (PA5)
	PIN6	PIN34 (PA6)
	PIN7	PIN09 (PA7)
Led Rojo	ÁNODO	PIN72 (PA08)
	CÁTODO	PIN00 (GND)
Led Naranja	ÁNODO	PIN97 (PA09)
	CÁTODO	PIN01 (GND)
Led Verde	ÁNODO	PIN71 (PA10)
	CÁTODO	PIN03 (GND)
RTC	GND	PIN25 (GND)
	+3.3V	PIN53 (3V)
	SDA	PIN62 (PB6)

	SCL	PIN87 (PB7)
	DS	NO CONECTADO
	BAT	NO CONECTADO
	SQ	NO CONECTADO
Relé	GND	PIN51 (GND)
	+5V	PIN52 (5V)
	DATA	PIN94 (PA11)
Sensor de movimiento	GND	PIN76 (GND)
	+5V	PIN77 (5V)
	DATA	PIN68 (PA12)
Sensor magnético	GND	PIN75(GND)
	+5V	PIN77 (5V)
	DATA	PIN96 (PA13)
Lector RFID	SDA	NO CONECTADO
	SCK	PIN58 (PE2)
	MOSI	PIN56 (PE6)
	MISO	PIN82 (PE5)
	IRQ	NO CONECTADO
	GND	PIN100 (GND)
	RST	NO CONECTADO
	+3.3V	PIN78 (3V)

**Tabla 04: Asignación de Pines del Sistema HomeAlarm.**

A continuación se muestra el montaje real durante la fase de implementación del Sistema HomeAlarm completo:

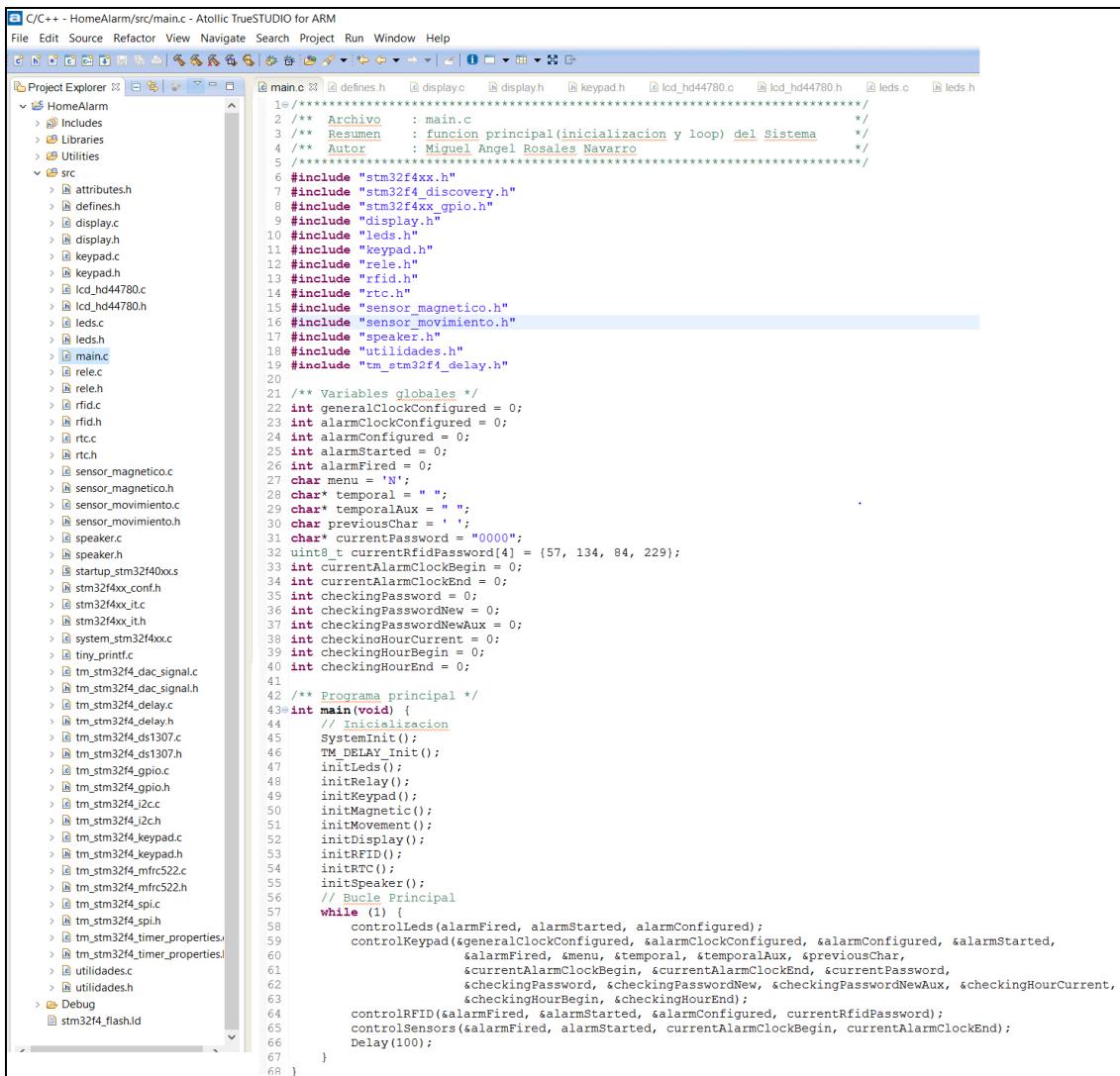


**Figura 40: montaje Sistema HomeAlarm completo.**

1→ STM32F4 Discovery; 2→ RTC; 3→ LEDs (rojo, verde, naranja); 4→ Sensor magnético (apertura); 5→ Sensor de movimiento; 6→ Lector RFID (tarjeta debajo); 7→ Relé; 8→ Display LCD 16x2; 9→ Keypad 4x4; 10→ Fuente de alimentación “MS-35-5”; 11→Simulador señal luminosa (conectado a relé); 12→ Simulador señal acústica (conectado al minijack de STM32F4 Discovery), amplificado mediante conexión eléctrica a relé.

#### 4.3.- Programación de librerías.

Una vez que se han escogido los componentes físicos específicos finales, y el layout final ya se ha definido y montado físicamente, se procede ahora a implementar el código que será cargado en el Sistema que se encargue de codificar las funcionalidades definidas en el apartado “Especificaciones del Sistema”. El código será implementado en una serie de librerías y será una transcripción en lenguaje en “C” de los flujoogramas definidos en el apartado “Diseño de Arquitectura”.



The screenshot shows the Atollic TrueSTUDIO for ARM IDE interface. The Project Explorer on the left lists the project structure for 'HomeAlarm'. The main.c file is open in the code editor, showing the following C code:

```

1 //***** Archivo : main.c ****
2 //** Resumen : funcion principal(inicializacion y loop) del Sistema ****
3 //** Autor : Miguel Angel Rosales Navarro ****
4 //***** ****
5 //***** ****
6 #include "stm32f4xx.h"
7 #include "stm32f4_discovery.h"
8 #include "stm32f4xx_gpio.h"
9 #include "display.h"
10 #include "leds.h"
11 #include "keypad.h"
12 #include "rele.h"
13 #include "rfid.h"
14 #include "rtc.h"
15 #include "sensor_magnetico.h"
16 #include "sensor_movimiento.h"
17 #include "speaker.h"
18 #include "utilidades.h"
19 #include "tm_stm32f4_delay.h"
20
21 /* Variables globales */
22 int generalClockConfigured = 0;
23 int alarmClockConfigured = 0;
24 int alarmConfigured = 0;
25 int alarmStarted = 0;
26 int alarmFired = 0;
27 char menu = 'N';
28 char* temporal = " ";
29 char* temporalAux = " ";
30 char previousChar = ' ';
31 char* currentPassword = "0000";
32 uint8_t currentRfidPassword[4] = {57, 134, 84, 229};
33 int currentAlarmClockBegin = 0;
34 int currentAlarmClockEnd = 0;
35 int checkingPassword = 0;
36 int checkingPasswordNew = 0;
37 int checkingPasswordNewAux = 0;
38 int checkingHourCurrent = 0;
39 int checkingHourBegin = 0;
40 int checkingHourEnd = 0;
41
42 /* Programa principal */
43 int main(void) {
44     // Inicializacion
45     SystemInit();
46     TM_DELAY_Init();
47     initLeds();
48     initRelay();
49     initKeypad();
50     initMagnetic();
51     initMovement();
52     initDisplay();
53     initRFID();
54     initRTC();
55     initSpeaker();
56     // Bucle Principal
57     while (1) {
58         controlLeds(alarmFired, alarmStarted, alarmConfigured);
59         controlKeypad(&generalClockConfigured, &alarmClockConfigured, &alarmConfigured, &alarmStarted,
60                      &alarmFired, &menu, &temporal, &temporalAux, &previousChar,
61                      &currentAlarmClockBegin, &currentAlarmClockEnd, &currentPassword,
62                      &checkingPassword, &checkingPasswordNew, &checkingPasswordNewAux, &checkingHourCurrent,
63                      &checkingHourBegin, &checkingHourEnd);
64         controlRFID(&alarmFired, &alarmStarted, &alarmConfigured, currentRfidPassword);
65         controlSensors(&alarmFired, &alarmStarted, currentAlarmClockBegin, currentAlarmClockEnd);
66         Delay(100);
67     }
68 }

```

Figura 41: captura “main.c” de HomeAlarm en entorno TrueStudio.

### **Librería “main.c”:**

La librería main.c será la encargada de ejecutarse automáticamente cada vez que se activa el Sistema. Comienza inicializando todas las variables necesarias para controlar los distintos estados y funcionalidades del Sistema. Como se puede observar en “C” no existe el tipo boolean o el tipo String usados en los fluogramas por lo que boolean ha sido sustituido por int siendo 0 equivalente a false y 1 equivalente a true y el tipo String sustituido por char\*.

Una vez inicializadas las variables globales del Sistema se ejecutará el método “**int main(void)**” que primero ejecuta una serie de funciones que inicializan los distintos componentes del Sistema, y tres ello comienza la ejecución de su bucle principal (infinito) en el que se ejecutan las cuatro funciones principales de la rutina general: “controlLeds”, “controlKeypad”, “controlRFID” y “controlSensors”. Cada una de ellas se mostrará subrayada al explicar la librería que lo contiene. Tras la ejecución de estas cuatro funciones se realizará un “delay” que se encarga de controlar la velocidad de ejecución del bucle principal.

Se ha creado una librería independiente para la gestión de cada componente del Sistema, que ofrecerán funciones públicas para que puedan ser usadas por “main” y funciones privadas para su uso interno dentro de la librería. Dependiendo del componente que se vaya a tratar, estas librerías realizarán por si mismas toda la gestión del componente o se apollarán en otras librerías descargadas de internet que ya contienen las funciones necesarias para inicializar y gestionar el componente. Se podría acceder a estas librerías descargadas de internet directamente desde “main” pero aún así se ha decidido dejar las librerías propias intermedias para que funcionen a modo de interfaz de forma que si en un futuro se encontrasen nuevas librerías de control de componentes más óptimas, con mayor rendimiento, etc... bastaría con cambiar las librerías en el proyecto y cambiar las llamadas internas que se realizan desde las librerías propias intermedias a las nuevas llamadas intermedias, sin tener que tocar nada de código en “main”.

### **Librería “defines.h”:**

El papel de esta librería, pese a no contener nada de código es muy importante ya que contiene todas las constantes que serán usadas por el resto de librerías referentes al mapeo de puertos y pines de los Componentes (y la configuración de estos). De esta forma, solo modificando los valores de estas constantes se puede variar completamente el pinout de la aplicación del modo que se desee sin modificar nada en el resto del código del proyecto.

### **Librerías “display.h” y “display.c”:**

Estas librerías se encargan de la gestión del display del Sistema. Para ello se apoyan en las librerías “lcd\_hd44780.h” y “lcd\_hd44780.c” [15] que disponen de los métodos necesarios para inicialización y escritura en el display. Los métodos públicos de “display.h” son:

**void initDisplay(void):** se encarga de la inicialización del display del Sistema apoyándose en “lcd\_hd44780.h” (mapeo de pines en defines.h).

**void cleanDisplay(void):** se encarga de borrar lo que este mostrando el display, apoyándose en “lcd\_hd44780.h”.

**void writeDisplay(char\* line1, char\* line2):** se encarga de mostrar en el display las dos líneas enviadas como argumento, apoyándose en “lcd\_hd44780.h”. Primero limpia la pantalla.

### **Librerías “keypad.h” y “keypad.c”:**

Estas librerías se encargan de la gestión del keypad del Sistema. Para ello se apoyan en las librerías “tm\_stm32f4\_keypad.h” y “tm\_stm32f4\_keypad.c” [14] que disponen de los métodos necesarios para inicialización y escritura en el keypad. Los métodos públicos de “keypad.h” son:

**void initKeypad(void):** se encarga de la inicialización del keypad del Sistema apoyándose en “tm\_stm32f4\_keypad.h” (mapeo de pines en defines.h).

**void controlKeypad(int \*generalClockConfigured, int \*alarmClockConfigured, int \*alarmConfigured, int \*alarmStarted, int \*alarmFired, char \*menu, char\* \*temporal, char\* \*temporalAux, char \*previousChar, int \*currentAlarmClockBegin, int \*currentAlarmClockEnd, char\* \*currentPassword, int \*checkingPassword, int \*checkingPasswordNew, int \*checkingPasswordNewAux, int \*checkingHourCurrent, int \*checkingHourBegin, int \*checkingHourEnd):**

Se trata de la función principal de control del keypad, llamada desde “main.c”, se encarga de leer el teclado y llamar a las distintas funciones privadas encargadas de los submenús del Sistema.

Las funciones privadas disponibles en “keypad.c” son:

**char readKeypad(void):** función encargada de la lectura de teclado, la cual lee los ocho pines del keypad 4x4, apoyándose en “tm\_stm32f4\_keypad.h.h”. El modo de funcionamiento de este componente es sencillo: se trata de una matriz 4x4 de tal forma que cada vez que se hace una lectura, se leen ocho pines, cuatro correspondientes a las cuatro filas del keypad y los otros cuatro correspondientes a las cuatro columnas del keypad. Si al leer los pines todos devuelven un cero lógico, es que ningún botón está pulsado. Si hay un botón pulsado, dos de los pines devolverán un 1 lógico, un pin corresponderá a la fila del botón pulsado y el otro 1 corresponderá a la columna del botón pulsado y con ello se obtiene el carácter del botón que se ha pulsado.

**void readKeypadMenuA(char character, int \*checkingPassword, int \*checkingHourCurrent, char \*menu, char \*previousChar, char\* \*temporal, int \*generalClockConfigured, int \*alarmConfigured, int alarmClockConfigured, char\* \*currentPassword):** función encargada de controlar el teclado cuando se ha seleccionado el menú A (encargado de programar el reloj del Sistema).

**void readKeypadMenuB(char character, int \*checkingPassword, int \*checkingPasswordNew, int \*checkingPasswordNewAux, char \*menu, char \*previousChar, char\* \*temporal, char\* \*temporalAux, char\* \*currentPassword):** función encargada de controlar el teclado cuando se ha seleccionado el menú B (encargado de cambiar el password del Sistema).

**void readKeypadMenuC(char character, int \*checkingPassword, int \*checkingHourBegin, int \*checkingHourEnd, char \*menu, char \*previousChar, char\* \*temporal, char\* \*temporalAux, int \*currentAlarmClockBegin, int \*currentAlarmClockEnd, int generalClockConfigured, int \*alarmConfigured, int \*alarmClockConfigured, char\* \*currentPassword):** función encargada de controlar el teclado cuando se ha seleccionado el menú C (encargado de programar las horas de inicio y fin de alarma activa del Sistema).

**void readKeypadMenuD(char character, int \*checkingPassword, char \*menu, char \*previousChar, char\* \*temporal, int \*alarmConfigured, int \*alarmStarted, int \*alarmFired, char\* \*currentPassword):** función encargada de controlar el teclado cuando se ha seleccionado el menú D (encargado de programar iniciar o parar la alarma del Sistema).

### Librerías “leds.h” y “leds.c”:

Estas librerías se encargan de la gestión de los leds del Sistema. No se apoyan en ninguna librería externa, salvo las propias de STM32F4 (métodos básicos de inicialización y escritura en pines GPIO). Sus métodos públicos son:

**void initLeds(void):** inicializa los leds del Sistema (mapeo de pines en defines.h).

**void activateLed(int activateLed, int numLed):** activa o desactiva un led. Si activateLed vale 1 se activa el led, si vale 0 se desactiva el led. El led se selecciona con el argumento numLed (usando las constantes de led definidas en “leds.h”).

**void controlLeds(int alarmFired, int alarmStarted, int alarmConfigured):** se trata de la función principal de control de los leds, llamada desde “main.c”, se encarga de su encendido o apagado en función del valor de las variables de estado globales.

Las funciones privadas disponibles en “leds.c” son:

**void blinkingLed(int numLed):** función que hace parpadear un led (seleccionado con constantes de led definidas en “leds.h”).

### Librerías “rele.h” y “rele.c”:

Estas librerías se encargan de la gestión del relé del Sistema. No se apoyan en ninguna librería externa, salvo las propias de STM32F4 (métodos básicos de inicialización y escritura en pines GPIO). Sus métodos públicos son:

**void initRelay(void):** inicializa el relé del Sistema (mapeo de pines en defines.h).

**void activateRelay(int activateRelay):** activa o desactiva el relé del Sistema. Si activateRelay vale 1 se activa el relé, si vale 0 se desactiva el relé.

### Librerías “rfid.h” y “rfid.c”:

Estas librerías se encargan de la gestión del RFID del Sistema. Para ello se apoyan en las librerías “tm\_stm32f4\_mfrc522.h” y “tm\_stm32f4\_mfrc522.c” [12] que disponen de los métodos necesarios para inicialización y lectura del RFID. Ya que se conecta a un puerto SPI, se ayuda también de las librerías “tm\_stm32f4\_spi.h” y “tm\_stm32f4\_spi.c” [12]. Los métodos públicos de “rfid.h” son:

**void initRFID(void):** inicializa el lector rfid del Sistema (mapeo de pines en defines.h).

**void controlRFID(int \*alarmFired, int \*alarmStarted, int \*alarmConfigured, uint8\_t\* currentRfidPassword):** se trata de la función principal de control del RFID, llamada desde “main.c”, se encarga de la lectura del RFID y disparar o parar la alarma en función del valor de las variables de estado globales.

Las funciones privadas disponibles en “rfid.c” son:

**int readRFID(uint8\_t\* currentRfidPassword):** comprueba si se hay una tarjeta cerca del lector y si está autorizada. Si devuelve 0 no está autorizada, si devuelve 1 está autorizada.

#### **Librerías “rtc.h” y “rtc.c”:**

Estas librerías se encargan de la gestión del RTC del Sistema. Para ello se apoyan en las librerías “tm\_stm32f4\_ds1307.h” y “tm\_stm32f4\_ds1307.c” [11] que disponen de los métodos necesarios para inicialización y lectura del RTC. Ya que se conecta a un puerto I2C, se ayuda también de las librerías “tm\_stm32f4\_i2c.h” y “tm\_stm32f4\_i2c.c” [11]. Los métodos públicos de “rfid.h” son:

**void initRTC(void):** inicializa el RTC del Sistema, apoyándose en “tm\_stm32f4\_ds1307.h” (mapeo de pines en defines.h).

**void programRTC(char\* \*currentHour):** programa el RTC del Sistema con la hora (HHMM) enviada, apoyándose en “tm\_stm32f4\_ds1307.h”

**int readRTC(void):** lee el RTC del Sistema devolviendo un int que representa la hora actual (HHMM), apoyándose en “tm\_stm32f4\_ds1307.h”.

#### **Librerías “sensor\_magntico.h” y “sensor\_magntico.c”:**

Estas librerías se encargan de la gestión del sensor magnético del Sistema. No se apoyan en ninguna librería externa, salvo las propias de STM32F4 (métodos básicos de inicialización y lectura en pines GPIO). Sus métodos públicos son:

**void initMagnetic(void):** inicializa el sensor magnético del Sistema (mapeo de pines en defines.h).

**int readMagnetic(void):** lee el sensor magnético del Sistema devolviendo 1 si detecta campo magnético o 0 si no lo detecta.

#### **Librerías “sensor movimiento.h” y “sensor movimiento.c”:**

Estas librerías se encargan de la gestión del sensor magnético del Sistema. No se apoyan en ninguna librería externa, salvo las propias de STM32F4 (métodos básicos de inicialización y lectura en pines GPIO). Sus métodos públicos son:

**void initMovement(void):** inicializa el sensor de movimiento del Sistema (mapeo de pines en defines.h).

**int readMovement(void):** lee el sensor de movimiento del Sistema devolviendo 1 si detecta movimiento o 0 si no lo detecta.

#### **Librerías “speaker.h” y “speaker.c”:**

Estas librerías se encargan de la gestión de la señal que se envía al speaker del Sistema. Se apoyan las librerías externas “tm\_stm32f4\_dac\_signal.h”, “tm\_stm32f4\_dac\_signal.c”, “tm\_stm32f4\_timer\_properties.h”, “tm\_stm32f4\_timer\_properties.c”, “tm\_stm32f4\_gpio.h” y “tm\_stm32f4\_gpio.c” [18]. Sus métodos públicos son:

**void initSpeaker(void):** inicializa el DAC que controla la señal que va al speaker del Sistema (mapeo de pines en defines.h).

**void activateRelay(int activateRelay):** activa o desactiva la señal que va al speaker del Sistema. Si activateSound vale 1 se activa la señal (un seno a 440Hz, es decir la nota “La”). Si activateSound vale 0 se desactiva la señal. En este último caso, lo que se ha hecho es activar una señal cuadrada modificada (modificando “tm\_stm32f4\_dac\_signal.c”), en la que tanto el valor alto como el valor bajo valen 0, por lo que a efectos prácticos es lo mismo que desactivar la señal sonora.

#### **Librerías “utilidades.h” y “utilidades.c”:**

Estas librerías contienen funciones de apoyo que serán usadas por el resto de librerías de tal manera que si es necesario cambiar u optimizar alguna de estas funciones auxiliares llamadas desde multitud de puntos, se pueda hacer cambiando el código de manera centralizada.

**int length(char\* arrayChar):** devuelve la longitud de la cadena argumento.

**int checkPassword(char\* password, char\* auxPassword):** comprueba que las dos cadenas argumento son iguales. Devuelve 0 si no son iguales; 1 = son iguales.

**int checkHour(char\* hour):** comprueba que la cadena argumento contiene una hora válida (número entre 0 y 2359). Devuelve 1 si la hora es válida; 0 si la hora no es válida.

**int alarmInProgrammedTime(int currentTime, int currentAlarmClockBegin, int currentAlarmClockEnd):** comprueba si una hora se encuentra entre otras dos horas. El argumento currentTime contiene la hora a comprobar, el argumento currentAlarmClockBegin contiene la hora de inicio del intervalo a comprobar, el argumento currentAlarmClockEnd contiene la hora fin del intervalo a comprobar. La función devuelve 1 si la hora está en el intervalo; 0 si la hora no está en el intervalo (tiene en cuenta incluso si la hora de inicio es mayor que la de fin porque se programa entre cambios de día).

Las siguientes tres funciones son las funciones generales de control de alarma del Sistema:

**void fireAlarm(int \*alarmFired):** función que dispara la alarma del Sistema actualizando la variable de estado alarmFired.

**void stopAlarm(int \*alarmFired, int \*alarmStarted):** función que detiene la alarma del Sistema actualizando las variables de estado alarmFired y alarmStarted.

**void startAlarm(int \*alarmStarted):** función que inicia la alarma del Sistema actualizando la variable de estado alarmStarted.

Por último, la última función de control general que queda por explicar:

**void controlSensors(int \*alarmFired, int alarmStarted, int currentAlarmClockBegin, int currentAlarmClockEnd):** se trata de la función principal de control de los sensores magnético y de movimiento del Sistema, llamada desde "main.c". Se ubicó en la librería de utilidades ya que controla ambos sensores a la vez. Lee ambos sensores y el RTC y en función de las variables de estado dispara la alarma.

#### 4.4.- Plan de pruebas.

En este apartado se definen las distintas pruebas a las que ha sido sometido el Sistema indicando además los resultados obtenidos de su ejecución. Cada prueba definida tendrá un código asociado cuyo formato será HA-XX, donde XX es un número de dos dígitos empezando en 01 y HA es elacrónico del nombre corto del proyecto **HomeAlarm**. El resultado de las pruebas será “**Pasado**” o “**Fallido**”, usando la celda “Comentarios” en caso de querer realizar aclaraciones adicionales acerca del resultado de la prueba. Existe una precondición común a todas las pruebas, por lo que no será incluida en cada una de ellas. Esta precondición es “El Sistema se encuentra iniciado.”

<b>Código de Prueba</b>	HA-01	
<b>Objetivo</b>	Mostrar el menú A para programar el reloj	
<b>Precondiciones</b>	No hay ningún menú seleccionado.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar tecla “A” en teclado.
<b>Postcondiciones</b>	-El Sistema se encontrará en el menú A mostrando la frase “Introduzca Password”	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 05: Prueba HA-01.**

<b>Código de Prueba</b>	HA-02	
<b>Objetivo</b>	Probar que dentro del menú A, si se pulsa cualquier tecla que no sea un número, el Sistema no hará nada.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú A.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar una tecla que no sea un número.
<b>Postcondiciones</b>	-El Sistema no se ve afectado (no hace nada).	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Ha sido probada varias veces pulsando A, B, C, D, * y #.	

**Tabla 06: Prueba HA-02.**

<b>Código de Prueba</b>	HA-03	
<b>Objetivo</b>	Probar que dentro del menú A, si se introduce una contraseña que no es correcta, el Sistema notifica un error y sale del menú.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú A. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña errónea.

<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Incorrecta”. -El Sistema sale del menú A.
<b>Resultado</b>	<b>Pasado</b>
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.

**Tabla 07: Prueba HA-03.**

<b>Código de Prueba</b>	HA-04	
<b>Objetivo</b>	Probar que dentro del menú A, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y solicita la hora actual.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú A. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Correcta”, espera 5 segundos y muestra la frase “Introduzca Hora Actual”.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 08: Prueba HA-04.**

<b>Código de Prueba</b>	HA-05	
<b>Objetivo</b>	Probar que dentro del menú A, si se introduce una hora actual errónea, el Sistema muestra un mensaje de error y sale del menú.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú A. La contraseña actual ha sido introducida correctamente. El Sistema está solicitando la hora actual.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora incorrecta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Hora Actual Incorrecta” y sale del menú A.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. La prueba ha sido pasada introduciendo como hora actual 9999.	

**Tabla 09: Prueba HA-05.**

<b>Código de Prueba</b>	HA-06	
<b>Objetivo</b>	Probar que dentro del menú A, si se introduce una hora actual correcta, el Sistema muestra un mensaje de éxito y se sale del menú.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú A. La contraseña actual ha sido introducida correctamente.	

	El Sistema está solicitando la hora actual.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora correcta.
<b>Postcondiciones</b>	<p>-El Sistema muestra la frase “Hora Actual Correcta” y sale del menú A.</p> <p>HA-05.1→Si el Sistema tiene el rango de hora de alarma activa configurado y la alarma no está iniciada se encenderá el led naranja.</p> <p>HA-05.2→Si el Sistema está en modo alarma iniciada, se mantendrá encendido el led verde.</p>	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	Ambas pruebas HA-05.1 y HA-05.2 han sido pasadas satisfactoriamente. Las pruebas han sido pasadas introduciendo como hora actual 1430.	

**Tabla 10: Prueba HA-06.**

<b>Código de Prueba</b>	HA-07	
<b>Objetivo</b>	Mostrar el menú B para cambiar contraseña.	
<b>Precondiciones</b>	No hay ningún menú seleccionado.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar tecla “B” en teclado.
<b>Postcondiciones</b>	-El Sistema se encontrará en el menú B mostrando la frase “Introduzca Password”	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 11: Prueba HA-07.**

<b>Código de Prueba</b>	HA-08	
<b>Objetivo</b>	Probar que dentro del menú B, si se pulsa cualquier tecla que no sea un número, el Sistema no hará nada.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar una tecla que no sea un número.
<b>Postcondiciones</b>	-El Sistema no se ve afectado (no hace nada).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Ha sido probada varias veces pulsando A, B, C, D, * y #.	

**Tabla 12: Prueba HA-08.**

<b>Código de Prueba</b>	HA-09	
<b>Objetivo</b>	Probar que dentro del menú B, si se introduce una	

	contraseña que no es correcta, el Sistema notifica un error y se sale del menú.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña errónea.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Incorrecta”. -El Sistema sale del menú B.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 13: Prueba HA-09.**

<b>Código de Prueba</b>	HA-10	
<b>Objetivo</b>	Probar que dentro del menú B, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y solicita una nueva contraseña.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Correcta”, espera 5 segundos y muestra la frase “Introduzca Nueva Password”.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 14: Prueba HA-10.**

<b>Código de Prueba</b>	HA-11	
<b>Objetivo</b>	Probar que dentro del menú B, una vez introducida la contraseña actual correctamente, si se introduce una nueva contraseña numérica de longitud 4 dígitos, el Sistema pide confirmación de la nueva contraseña.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B. La contraseña actual ha sido introducida correctamente. El Sistema está solicitando la nueva contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña de cuatro dígitos
<b>Postcondiciones</b>	-El Sistema muestra la frase “Confirme Nueva Password”.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 15: Prueba HA-11.**

<b>Código de Prueba</b>	HA-12	
<b>Objetivo</b>	Probar que dentro del menú B, una vez introducida la contraseña actual correctamente, si se introduce una nueva contraseña numérica de longitud 4 dígitos, y se confirma con una contraseña distinta a la anterior, el Sistema muestra un mensaje de error y sale del menú B.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B. La contraseña actual ha sido introducida correctamente. Se ha introducido una nueva contraseña. El Sistema está solicitando la confirmación de la nueva contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña de cuatro dígitos diferente a la justo anterior.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Incorrecta” y sale del menú B.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Para la nueva contraseña se introdujo 1111 y para su confirmación se introdujo 2222.	

**Tabla 16: Prueba HA-12.**

<b>Código de Prueba</b>	HA-13	
<b>Objetivo</b>	Probar que dentro del menú B, una vez introducida la contraseña actual correctamente, si se introduce una nueva contraseña numérica de longitud 4 dígitos, y se confirma con una contraseña igual a la anterior, el Sistema muestra un mensaje de éxito y sale del menú B.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú B. La contraseña actual ha sido introducida correctamente. Se ha introducido una nueva contraseña. El Sistema solicita la confirmación de nueva contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña de cuatro dígitos igual a la justo anterior.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Cambio de Password Correcta” y sale del menú B. -El estado de los leds no cambia.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Para la nueva contraseña se introdujo 1111 y para su confirmación se introdujo 1111.	

**Tabla 17: Prueba HA-13.**

<b>Código de Prueba</b>	HA-14
<b>Objetivo</b>	Mostrar el menú C para programar el rango horario donde la alarma se encuentra activa.
<b>Precondiciones</b>	No hay ningún menú seleccionado.
<b>Pasos a seguir</b>	<b>Paso</b>   <b>Acción</b>
	1   Pulsar tecla “C” en teclado.
<b>Postcondiciones</b>	-El Sistema se encontrará en el menú C mostrando la frase “Introduzca Password”
<b>Resultado</b>	Pasado
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.

**Tabla 18: Prueba HA-14.**

<b>Código de Prueba</b>	HA-15
<b>Objetivo</b>	Probar que dentro del menú C, si se pulsa cualquier tecla que no sea un número, el Sistema no hará nada.
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C.
<b>Pasos a seguir</b>	<b>Paso</b>   <b>Acción</b>
	1   Pulsar una tecla que no sea un número.
<b>Postcondiciones</b>	-El Sistema no se ve afectado (no hace nada).
<b>Resultado</b>	Pasado
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Ha sido probada varias veces pulsando A, B, C, D, * y #.

**Tabla 19: Prueba HA-15.**

<b>Código de Prueba</b>	HA-16
<b>Objetivo</b>	Probar que dentro del menú C, si se introduce una contraseña que no es correcta, el Sistema notifica un error y se sale del menú.
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. El Sistema está solicitando la contraseña.
<b>Pasos a seguir</b>	<b>Paso</b>   <b>Acción</b>
	1   Introducir una contraseña errónea.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Incorrecta”. -El Sistema sale del menú C.
<b>Resultado</b>	Pasado
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.

**Tabla 20: Prueba HA-16.**

<b>Código de Prueba</b>	HA-17
<b>Objetivo</b>	Probar que dentro del menú C, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y solicita la hora de inicio del rango.

<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Correcta”, espera 5 segundos y muestra la frase “Introduzca Hora Inicio”.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 21: Prueba HA-17.**

<b>Código de Prueba</b>	HA-18	
<b>Objetivo</b>	Probar que dentro del menú C, una vez introducida la contraseña actual correctamente, si se introduce una hora de inicio errónea, el Sistema muestra un mensaje de error y sale del menú C.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. La contraseña actual ha sido introducida correctamente. El Sistema está solicitando la hora de inicio.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora de inicio errónea.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Hora de Inicio Incorrecta” y sale del menú C.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada introduciendo como hora de inicio 9999.	

**Tabla 22: Prueba HA-18.**

<b>Código de Prueba</b>	HA-19	
<b>Objetivo</b>	Probar que dentro del menú C, una vez introducida la contraseña actual correctamente, si se introduce una hora de inicio correcta, el Sistema muestra un mensaje de éxito y solicita la hora de fin.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. La contraseña actual ha sido introducida correctamente. El Sistema está solicitando la hora de inicio.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora de inicio correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Hora de Inicio Correcta”, espera 5 segundos y muestra la frase “Introduzca Hora Fin”.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada introduciendo como hora de	

	inicio 1430.
--	--------------

**Tabla 23: Prueba HA-19.**

<b>Código de Prueba</b>	HA-20	
<b>Objetivo</b>	Probar que dentro del menú C, una vez introducida la contraseña actual correctamente, y una hora de inicio correcta, si se introduce una hora de fin errónea, el Sistema muestra un mensaje de error y sale del menú C.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. La contraseña actual ha sido introducida correctamente. La hora de inicio ha sido introducida correctamente El Sistema está solicitando la hora de fin.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora de fin errónea.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Hora de Fin Incorrecta” y sale del menú C.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada introduciendo como hora de fin 9999.	

**Tabla 24: Prueba HA-20.**

<b>Código de Prueba</b>	HA-21	
<b>Objetivo</b>	Probar que dentro del menú C, una vez introducida la contraseña actual correctamente, y una hora de inicio correcta, si se introduce una hora de fin correcta, el Sistema muestra un mensaje de éxito y sale del menú C.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú C. La contraseña actual ha sido introducida correctamente. La hora de inicio ha sido introducida correctamente El Sistema está solicitando la hora de fin.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una hora de fin correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Hora de Fin Correcta”, espera 5 segundos y sale del menú C. HA-21.1→Si el Sistema tiene el reloj general programado y la alarma no está iniciada se encenderá el led naranja. HA-22.2→Si el Sistema está en modo alarma iniciada, se mantendrá encendido el led verde.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada introduciendo como hora de fin 1530.	

**Tabla 25: Prueba HA-21.**

<b>Código de Prueba</b>	HA-22	
<b>Objetivo</b>	Mostrar el menú D para programar el rango horario donde la alarma se encuentra activa.	
<b>Precondiciones</b>	No hay ningún menú seleccionado.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar tecla “D” en teclado.
<b>Postcondiciones</b>	-El Sistema se encontrará en el menú D mostrando la frase “Introduzca Password”	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 26: Prueba HA-22.**

<b>Código de Prueba</b>	HA-23	
<b>Objetivo</b>	Probar que dentro del menú D, si se pulsa cualquier tecla que no sea un número, el Sistema no hará nada.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Pulsar una tecla que no sea un número.
<b>Postcondiciones</b>	-El Sistema no se ve afectado (no hace nada).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. Ha sido probada varias veces pulsando A, B, C, D, * y #.	

**Tabla 27: Prueba HA-23.**

<b>Código de Prueba</b>	HA-24	
<b>Objetivo</b>	Probar que dentro del menú D, si se introduce una contraseña que no es correcta, el Sistema notifica un error y se sale del menú.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña errónea.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Password Incorrecta”. -El Sistema sale del menú D.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 28: Prueba HA-24.**

<b>Código de Prueba</b>	HA-25	
<b>Objetivo</b>	Probar que dentro del menú D, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y si la alarma no está configurada, el Sistema muestra un mensaje de error y sale del menú D.	

<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Configure Alarma”, espera 5 segundos y sale del menú D.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 29: Prueba HA-25.**

<b>Código de Prueba</b>	HA-26	
<b>Objetivo</b>	Probar que dentro del menú D, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y si la alarma está configurada, iniciada y disparada, el Sistema muestra un mensaje de éxito, para la alarma y sale del menú D.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Alarma parada”, espera 5 segundos y sale del menú D. -Se encenderá el led naranja y el resto estarán apagados.	
<b>Resultado</b>	<b>Pasado</b>	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 30: Prueba HA-26.**

<b>Código de Prueba</b>	HA-27	
<b>Objetivo</b>	Probar que dentro del menú D, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y si la alarma está configurada, iniciada y no disparada, el Sistema muestra un mensaje de éxito, para la alarma y sale del menú D.	
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D. El Sistema está solicitando la contraseña.	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Introducir una contraseña correcta.
<b>Postcondiciones</b>	-El Sistema muestra la frase “Alarma parada”, espera 5 segundos y sale del menú D. -Se encenderá el led naranja y el resto estarán apagados.	
<b>Resultado</b>	<b>Pasado</b>	

<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.
--------------------	--

**Tabla 31: Prueba HA-27.**

<b>Código de Prueba</b>	HA-28				
<b>Objetivo</b>	Probar que dentro del menú D, si se introduce una contraseña correcta, el Sistema muestra un mensaje de éxito y si la alarma está configurada, y no iniciada, el Sistema muestra un mensaje de éxito, inicia la alarma y sale del menú D.				
<b>Precondiciones</b>	El Sistema se encuentra dentro del menú D. El Sistema está solicitando la contraseña.				
<b>Pasos a seguir</b>	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Introducir una contraseña correcta.</td> </tr> </tbody> </table>	Paso	Acción	1	Introducir una contraseña correcta.
Paso	Acción				
1	Introducir una contraseña correcta.				
<b>Postcondiciones</b>	<ul style="list-style-type: none"> <li>-El Sistema muestra la frase “Alarma Iniciada”, espera 5 segundos y sale del menú D.</li> <li>-Se encenderá el led verde y el resto estarán apagados.</li> </ul>				
<b>Resultado</b>	<b>Pasado</b>				
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.				

**Tabla 32: Prueba HA-28.**

<b>Código de Prueba</b>	HA-29						
<b>Objetivo</b>	Probar que si la alarma no está programada (led rojo encendido) el detector de movimiento y el detector de apertura no disparan la alarma.						
<b>Precondiciones</b>	El Sistema no está programado (led rojo encendido).						
<b>Pasos a seguir</b>	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Realizar un movimiento en una zona captada por el detector de movimiento.</td> </tr> <tr> <td>2</td> <td>Abrir una puerta o ventana controlada por el detector de apertura.</td> </tr> </tbody> </table>	Paso	Acción	1	Realizar un movimiento en una zona captada por el detector de movimiento.	2	Abrir una puerta o ventana controlada por el detector de apertura.
Paso	Acción						
1	Realizar un movimiento en una zona captada por el detector de movimiento.						
2	Abrir una puerta o ventana controlada por el detector de apertura.						
<b>Postcondiciones</b>	-El Sistema no se ve afectado. La alarma no es disparada.						
<b>Resultado</b>	<b>Pasado</b>						
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.						

**Tabla 33: Prueba HA-29.**

<b>Código de Prueba</b>	HA-30		
<b>Objetivo</b>	Probar que si la alarma está programada pero no iniciada (led naranja encendido) el detector de movimiento y el detector de apertura no disparan la alarma.		
<b>Precondiciones</b>	El Sistema está programado y no iniciado (led naranja encendido).		
<b>Pasos a seguir</b>	<table border="1"> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> </table>	Paso	Acción
Paso	Acción		

	1	Realizar un movimiento en una zona captada por el detector de movimiento.
	2	Abrir una puerta o ventana controlada por el detector de apertura.
<b>Postcondiciones</b>	-El Sistema no se ve afectado. La alarma no es disparada.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 34: Prueba HA-30.**

<b>Código de Prueba</b>	HA-31	
<b>Objetivo</b>	Probar que si la alarma está programada e iniciada (led verde encendido), el detector de movimiento y el detector de apertura no disparan la alarma si son excitados en una hora fuera del rango activo de alarma programado.	
<b>Precondiciones</b>	El Sistema está programado e iniciado (led verde encendido).	
<b>Pasos a seguir</b>		
	<b>Paso</b>	<b>Acción</b>
	1	Realizar un movimiento en una zona captada por el detector de movimiento en un horario fuera del rango activo programado.
	2	Abrir una puerta o ventana controlada por el detector de apertura en un horario fuera del rango activo programado.
<b>Postcondiciones</b>	-El Sistema no se ve afectado. La alarma no es disparada.	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 35: Prueba HA-31.**

<b>Código de Prueba</b>	HA-32	
<b>Objetivo</b>	Probar que si la alarma está programada e iniciada (led verde encendido), el detector de movimiento y el detector de apertura disparan la alarma si son excitados en una hora dentro del rango activo de alarma programado.	
<b>Precondiciones</b>	El Sistema está programado e iniciado (led verde encendido).	
<b>Pasos a seguir</b>		
	<b>Paso</b>	<b>Acción</b>
	1	Realizar un movimiento en una zona captada por el detector de movimiento en un horario dentro del rango activo programado.
	2	Abrir una puerta o ventana controlada por el detector de apertura en un horario dentro del

	rango activo programado.
<b>Postcondiciones</b>	-La alarma no es disparada (acústica y sonora). El led verde permanece encendido. El led rojo parpadea.
<b>Resultado</b>	<b>Pasado</b>
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.

**Tabla 36: Prueba HA-32.**

<b>Código de Prueba</b>	HA-33				
<b>Objetivo</b>	Probar que acercar una tarjeta RFID que no contenga el password RFID al lector RFID con la alarma en cualquier estado, no afecta al Sistema.				
<b>Precondiciones</b>	El Sistema se encuentra iniciado en cualquier estado.				
<b>Pasos a seguir</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th> <th><b>Acción</b></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Acercar una tarjeta RFID que no contenga el password RFID al lector RFID.</td> </tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Acercar una tarjeta RFID que no contenga el password RFID al lector RFID.
<b>Paso</b>	<b>Acción</b>				
1	Acercar una tarjeta RFID que no contenga el password RFID al lector RFID.				
<b>Postcondiciones</b>	<p>-El Sistema no se ve afectado.</p> <p>HA-33.1→ Si el Sistema no está configurado, permanece igual.</p> <p>HA-33.2→ Si el Sistema está configurado, permanece igual.</p> <p>HA-33.3→ Si el Sistema está configurado y la alarma iniciada, permanece igual.</p> <p>HA-33.4→ Si el Sistema está configurado, la alarma iniciada y disparada, permanece igual.</p>				
<b>Resultado</b>	<b>Pasado</b>				
<b>Comentarios</b>	Las cuatro sub-pruebas han sido pasadas satisfactoriamente.				

**Tabla 37: Prueba HA-33.**

<b>Código de Prueba</b>	HA-34				
<b>Objetivo</b>	Probar que si la alarma no está programada (led rojo encendido) acercar una tarjeta RFID que contenga el password RFID al lector RFID no afecta al Sistema.				
<b>Precondiciones</b>	El Sistema no está programado (led rojo encendido).				
<b>Pasos a seguir</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th> <th><b>Acción</b></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Acercar una tarjeta RFID que contenga el password RFID al lector RFID.</td> </tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Acercar una tarjeta RFID que contenga el password RFID al lector RFID.
<b>Paso</b>	<b>Acción</b>				
1	Acercar una tarjeta RFID que contenga el password RFID al lector RFID.				
<b>Postcondiciones</b>	-El Sistema no se ve afectado.				
<b>Resultado</b>	<b>Pasado</b>				
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.				

**Tabla 38: Prueba HA-34.**

<b>Código de Prueba</b>	HA-35	
<b>Objetivo</b>	Probar que si la alarma está programada pero no iniciada (led naranja encendido) acercar una tarjeta RFID que contenga el password RFID al lector RFID, inicia la alarma (led verde encendido).	
<b>Precondiciones</b>	El Sistema está programado pero no iniciado (led naranja encendido).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Acercar una tarjeta RFID que contenga el password RFID al lector RFID.
<b>Postcondiciones</b>	-El Sistema inicia la alarma (led verde encendido).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 39: Prueba HA-35.**

<b>Código de Prueba</b>	HA-36	
<b>Objetivo</b>	Probar que si la alarma está programada e iniciada (led verde encendido) acercar una tarjeta RFID que contenga el password RFID al lector RFID, para la alarma (led naranja encendido).	
<b>Precondiciones</b>	El Sistema está programado e iniciado (led verde encendido).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Acercar una tarjeta RFID que contenga el password RFID al lector RFID.
<b>Postcondiciones</b>	-El Sistema para la alarma (led naranja encendido).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.	

**Tabla 40: Prueba HA-36.**

<b>Código de Prueba</b>	HA-37	
<b>Objetivo</b>	Probar que si la alarma está programada, iniciada y disparada (led verde encendido y led rojo parpadeando) acercar una tarjeta RFID que contenga el password RFID al lector RFID, para la alarma (led naranja encendido).	
<b>Precondiciones</b>	La alarma está disparada (led verde encendido, led rojo parpadeando).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Acercar una tarjeta RFID que contenga el password RFID al lector RFID.
<b>Postcondiciones</b>	-El Sistema para la alarma (led naranja encendido).	
<b>Resultado</b>	Pasado	

<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente.
--------------------	--

**Tabla 41: Prueba HA-37.**

<b>Código de Prueba</b>	HA-38	
<b>Objetivo</b>	Probar que si la alarma está programada, iniciada y disparada (led verde encendido y led rojo parpadeando) y se corta el cable del detector de movimiento o el detector de apertura, el Sistema no se ve afectado (la alarma continua disparada).	
<b>Precondiciones</b>	La alarma está disparada (led verde encendido, led rojo parpadeando).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Desconectar el detector de movimiento y de apertura.
<b>Postcondiciones</b>	-El Sistema no se ve afectado, la alarma (acústica y sonora) continúa disparada (además, led verde encendido y rojo parpadeando).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. La alarma se puede parar con la opción D del teclado o con el lector RFID.	

**Tabla 42: Prueba HA-38.**

<b>Código de Prueba</b>	HA-39	
<b>Objetivo</b>	Probar que si la alarma está programada, iniciada y disparada (led verde encendido y led rojo parpadeando) y se corta el cable del lector RFID, el Sistema no se ve afectado (la alarma continua disparada).	
<b>Precondiciones</b>	La alarma está disparada (led verde encendido, led rojo parpadeando).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Desconectar el lector RFID.
<b>Postcondiciones</b>	-El Sistema no se ve afectado, la alarma (acústica y sonora) continúa disparada (además, led verde encendido y rojo parpadeando).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. La alarma se puede parar con la opción D del teclado.	

**Tabla 43: Prueba HA-39.**

<b>Código de Prueba</b>	HA-40
<b>Objetivo</b>	Probar que si la alarma está programada, iniciada y disparada (led verde encendido y led rojo parpadeando)

	y se corta el cable del teclado, el Sistema no se ve afectado (la alarma continua disparada).	
<b>Precondiciones</b>	La alarma está disparada (led verde encendido, led rojo parpadeando).	
<b>Pasos a seguir</b>	<b>Paso</b>	<b>Acción</b>
	1	Desconectar el teclado.
<b>Postcondiciones</b>	-El Sistema no se ve afectado, la alarma (acústica y sonora) continua disparada (además, led verde encendido y rojo parpadeando).	
<b>Resultado</b>	Pasado	
<b>Comentarios</b>	La prueba ha sido pasada satisfactoriamente. La alarma se puede parar con el lector RFID.	

**Tabla 44: Prueba HA-40.**

## Capítulo 5. Mejoras futuras.

El objetivo de este capítulo es presentar una serie de mejoras futuras que se pueden incorporar al Sistema. El proyecto cuenta con un alcance concreto cerrado y acotado inicialmente. Durante las distintas fases por las que ha pasado el proyecto se han encontrado distintas mejoras/líneas de trabajo que se podían incluir en el Sistema pero que debido al propio alcance del proyecto no podían ser incluidas por distintos motivos (tiempo de diseño e implementación, aumento de costes, etc...).

Las principales mejoras que se podrían introducir en el Sistema en una posible ampliación de este serían las siguientes:

- Uso de memoria MicroSD para almacenar datos (programaciones, passwords, datos provenientes de nuevas funcionalidades), de tal forma que estos datos no se perdiessen tras dejar de alimentar el Sistema.
- Agregar cámara fotográfica/video que captase el elemento que origina la alarma (el que capta el detector de movimiento o de apertura) y guardase dichos datos en la memoria interna.
- Agregar módulo GSM para avisar por SMS a quien se deseé en caso de que la alarma sea disparada.
- Agregar módulo Ethernet o 3G (o ambos) para poder manejar alarma a distancia desde un dispositivo conectado a internet. Esto junto con la cámara amplia infinitamente el número de posibilidades: envío de fotografías, vídeos, visión en directo, programación de alarma a distancia, arranque y parada a distancia. Evidentemente el coste económico aumenta.
- Preparar el Sistema para que funcione con arrays de detectores de movimiento (aumenta el área de cobertura) y con arrays de detectores de apertura (aumenta el número de puertas/ventanas a monitorizar).
- Añadir al Sistema la posibilidad de programar tarjetas RFID con el password que se deseé.
- Realizar una versión del Sistema de bajo consumo de forma que pueda ser alimentada con baterías durante cortes de corriente eléctrica (ideal si se combina con sistemas GSM o 3G/4G) para que el hogar no quede desprotegido si se realiza un corte intencionado de la corriente eléctrica general).

Como se puede observar, cualquiera de estas opciones aumenta la robustez y fiabilidad del Sistema, aunque complican su implementación y aumentan sus costes.

## Capítulo 6. Conclusiones.

El número de robos en viviendas ha aumentado en los últimos años considerablemente debido a la crisis económica que vive España. Además, ésta impide a muchas familias poder proteger su hogar con algún tipo de dispositivo antirrobo debido a los altos precios que estos tienen en el mercado y a la posterior dependencia mediante cuotas de alto valor económico a la empresa que instala el dispositivo. Debido a ello las familias tienen problemas de seguridad que desean solucionar.

El objetivo de este proyecto ha sido desarrollar un Sistema que dé una solución a esta necesidad de manera robusta, fiable y económica y que además tenga el valor añadido de poder ser instalado y usado fácilmente por perfiles de usuario ajenos al mundo de la seguridad.

La solución implementada cubre un hueco importante ya que existen soluciones similares pero o bien tienen un coste económico muy elevado o bien no son lo suficientemente robustas y fiables y ofrecen funcionalidades muy básicas.

El diseño y desarrollo de la solución, aparte de los problemas intrínsecos a cualquier tipo de desarrollo hardware-software (tiempo, necesidad de I+D, etc...) ha sido todo un éxito al conseguir implementar un Sistema que cumple con todos y cada uno de los objetivos propuestos aparte de contener todos los requisitos funcionales detectados.

A nivel académico el desarrollo de este proyecto fin de máster ha sido muy beneficioso para el autor al haber obtenido nuevos conocimientos sobre una tecnología que desconocía, el microprocessador STM32F4 y su familia de sensores y shields compatibles, y poder familiarizarse con ella para futuros desarrollos en un mercado que en la actualidad se ha convertido en todo un gran nicho por descubrir, en el que los consumidores tienen cada vez más necesidad de gadgets que ejecuten funciones complicadas manteniendo un precio económico. Con ello el autor aumenta su base de conocimientos de programación para microprocesadores sumando a Arduino y Motorola 68020, el microprocessador STM32F4 lo cuál ha resultado francamente muy satisfactorio.

A esto hay que sumarle los beneficios que el desarrollo integral de un proyecto de esta índole le ha aportado: la adquisición de las competencias necesarias para poder trabajar en proyectos hardware-software de manera independiente al haber estado involucrado en todas las fases que han compuesto el ciclo de vida de este proyecto.

## Anexos.

Como anexos a continuación se ofrece el código fuente de todas las librerías programadas:

```
/*****************************************************************************  
** Archivo  : defines.h                                         */  
** Resumen  : define los puertos y pines usados en el Sistema      */  
** Autor    : Miguel Angel Rosales Navarro                         */  
*****  
#include "stm32f4xx.h"  
#include "stm32f4_discovery.h"  
#include "stm32f4xx_gpio.h"  
#include "tm_stm32f4_i2c.h"  
#include "tm_stm32f4_spi.h"  
  
#ifndef TM_DEFINES_H  
#define TM_DEFINES_H  
  
/** */  
  
/** KEYPAD */  
#define GPIO_KEYPAD      GPIOA  
//ROWS  
#define PIN_KEYPAD_ROW_1 GPIO_Pin_0  
#define PIN_KEYPAD_ROW_2 GPIO_Pin_1  
#define PIN_KEYPAD_ROW_3 GPIO_Pin_2  
#define PIN_KEYPAD_ROW_4 GPIO_Pin_3  
//COLUMNS  
#define PIN_KEYPAD_COLUMN_1 GPIO_Pin_4  
#define PIN_KEYPAD_COLUMN_2 GPIO_Pin_5  
#define PIN_KEYPAD_COLUMN_3 GPIO_Pin_6  
#define PIN_KEYPAD_COLUMN_4 GPIO_Pin_7  
  
/** LED ROJO */  
#define GPIO_LED_RED GPIOA  
#define CLK_LINE_LED_RED RCC_AHB1Periph_GPIOA  
#define PIN_LED_RED GPIO_Pin_8  
  
/** RED NARANJA */  
#define GPIO_LED_ORANGE GPIOA  
#define CLK_LINE_LED_ORANGE RCC_AHB1Periph_GPIOA  
#define PIN_LED_ORANGE GPIO_Pin_9  
  
/** LED VERDE */  
#define GPIO_LED_GREEN GPIOA  
#define CLK_LINE_LED_GREEN RCC_AHB1Periph_GPIOA  
#define PIN_LED_GREEN GPIO_Pin_10  
  
/** RELE */  
#define GPIO_RELAY GPIOA  
#define CLK_LINE_RELAY RCC_AHB1Periph_GPIOA  
#define PIN_RELAY GPIO_Pin_11  
  
/** SENSOR DE MOVIMIENTO */  
#define GPIO_SENSOR_MOVEMENT GPIOA  
#define CLK_LINE_SENSOR_MOVEMENT RCC_AHB1Periph_GPIOA  
#define PIN_SENSOR_MOVEMENT GPIO_Pin_12
```



```

/*****
** Archivo : main.c
** Resumen : funcion principal(inicializacion y loop) del Sistema */
** Autor : Miguel Angel Rosales Navarro */
****

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include "display.h"
#include "leds.h"
#include "keypad.h"
#include "rele.h"
#include "rfid.h"
#include "rtc.h"
#include "sensor_magnetico.h"
#include "sensor_movimiento.h"
#include "speaker.h"
#include "utilidades.h"
#include "tm_stm32f4_delay.h"

/** Variables globales */
int generalClockConfigured = 0;
int alarmClockConfigured = 0;
int alarmConfigured = 0;
int alarmStarted = 0;
int alarmFired = 0;
char menu = 'N';
char* temporal = " ";
char* temporalAux = " ";
char previousChar = ' ';
char* currentPassword = "0000";
uint8_t currentRfidPassword[4] = {57, 134, 84, 229};
int currentAlarmClockBegin = 0;
int currentAlarmClockEnd = 0;
int checkingPassword = 0;
int checkingPasswordNew = 0;
int checkingPasswordNewAux = 0;
int checkingHourCurrent = 0;
int checkingHourBegin = 0;
int checkingHourEnd = 0;

/** Programa principal */
int main(void) {
    // Inicializacion
    SystemInit();
    TM_DELAY_Init();
    initLeds();
    initRelay();
    initKeypad();
    initMagnetic();
    initMovement();
    initDisplay();
    initRFID();
    initRTC();
    initSpeaker();
    // Bucle Principal
    while (1) {
        controlLeds(alarmFired, alarmStarted, alarmConfigured);
        controlKeypad(&generalClockConfigured, &alarmClockConfigured,
&alarmConfigured, &alarmStarted,

```

```

        &alarmFired, &menu, &temporal, &temporalAux,
&previousChar,
        &currentAlarmClockBegin, &currentAlarmClockEnd,
&currentPassword,
        &checkingPassword, &checkingPasswordNew,
&checkingPasswordNewAux, &checkingHourCurrent,
        &checkingHourBegin, &checkingHourEnd);
        controlRFID(&alarmFired, &alarmStarted, &alarmConfigured,
currentRfidPassword);
        controlSensors(&alarmFired, alarmStarted, currentAlarmClockBegin,
currentAlarmClockEnd);
        Delay(100);
    }
}

/** Callback used by stm32f4_discovery_audio_codec.c. Refer to
stm32f4_discovery_audio_codec.h for more info. */
void EVAL_AUDIO_TransferComplete_CallBack(uint32_t pBuffer, uint32_t Size) {
    return;
}

/** Callback used by stm324xg_eval_audio_codec.c. Refer to stm324xg_eval_audio_codec.h
for more info. */
uint16_t EVAL_AUDIO_GetSampleCallBack(void) {
    return -1;
}
//********************************************************************

//********************************************************************

/** Archivo : display.h */
/** Resumen : control del display del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#ifndef _DISPLAY_H_
#define _DISPLAY_H_

/** Inicializacion del display del Sistema (mapeo de pines en defines.h) */
void initDisplay(void);

/** Funcion que borra lo que este mostrando el display */
void cleanDisplay(void);

/** Funcion que muestra en el display las dos lineas enviadas */
void writeDisplay(char* line1, char* line2);

#endif
//********************************************************************

//********************************************************************

/** Archivo : display.c */
/** Resumen : control del display del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "display.h"
#include "lcd_hd44780.h"

void initDisplay(void) {

```

```

        // Funcion incluida en "lcd_hds44780.h"
        lcd_init();
    }

void cleanDisplay(void) {
    // Funcion incluida en "lcd_hds44780.h"
    lcd_strxy("                ",0,0);
    lcd_strxy("                ",0,1);
}

void writeDisplay(char* line1, char* line2) {
    cleanDisplay();
    lcd_strxy(line1,0,0);
    lcd_strxy(line2,0,1);
}
//********************************************************************

/** Archivo : keypad.h */
/** Resumen : control del keypad del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#ifndef _KEYPAD_H_
#define _KEYPAD_H_

/** Inicializa el keypad (mapeo de pines en defines.h) */
void initKeypad(void);

/** Funcion principal de control del teclado. se encarga de leer el teclado y llamar a los distintos
submenus */
void controlKeypad(int *generalClockConfigured, int *alarmClockConfigured, int
*alarmConfigured, int *alarmStarted,
                    int *alarmFired, char *menu, char* *temporal, char* *temporalAux, char
*previousChar,
                    int *currentAlarmClockBegin, int *currentAlarmClockEnd, char*
*currentPassword,
                    int *checkingPassword, int *checkingPasswordNew, int
*checkingPasswordNewAux, int *checkingHourCurrent,
                    int *checkingHourBegin, int *checkingHourEnd);

#endif
//********************************************************************

/** Archivo : keypad.c */
/** Resumen : control del keypad del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include "keypad.h"
#include "rtc.h"
#include "utilidades.h"
#include "display.h"
#include "tm_stm32f4_delay.h"
#include "tm_stm32f4_keypad.h"

```

```

void initKeypad(void) {
    // Funcion incluida en "tm_stm32f4_keypad.h"
    TM_KEYPAD_Init(TM_KEYPAD_Type_Large);
}

/** Funcion encargada de la lectura de teclado
* devuelve el caracter correspondiente al boton que se esta pulsando
* o espacio en blanco si no esta pulsando nada
*/
char readKeypad(void) {
    char returnVal = ' ';
    // Crear instancia de keypad
    // Funcion incluida en "tm_stm32f4_keypad.h"
    TM_KEYPAD_Button_t Keypad_Button = TM_KEYPAD_Read();
    // Si se ha presionado un boton
    if (Keypad_Button != TM_KEYPAD_Button_NOPRESSED) {
        switch (Keypad_Button) {
            case TM_KEYPAD_Button_0:           /* Boton 0 presionado */
                returnVal = '0';
                break;
            case TM_KEYPAD_Button_1:           /* Boton 1 presionado */
                returnVal = '1';
                break;
            case TM_KEYPAD_Button_2:           /* Boton 2 presionado */
                returnVal = '2';
                break;
            case TM_KEYPAD_Button_3:           /* Boton 3 presionado */
                returnVal = '3';
                break;
            case TM_KEYPAD_Button_4:           /* Boton 4 presionado */
                returnVal = '4';
                break;
            case TM_KEYPAD_Button_5:           /* Boton 5 presionado */
                returnVal = '5';
                break;
            case TM_KEYPAD_Button_6:           /* Boton 6 presionado */
                returnVal = '6';
                break;
            case TM_KEYPAD_Button_7:           /* Boton 7 presionado */
                returnVal = '7';
                break;
            case TM_KEYPAD_Button_8:           /* Boton 8 presionado */
                returnVal = '8';
                break;
            case TM_KEYPAD_Button_9:           /* Boton 9 presionado */
                returnVal = '9';
                break;
            case TM_KEYPAD_Button_STAR:        /* Boton * presionado */
                returnVal = '*';
                break;
            case TM_KEYPAD_Button_HASH:        /* Boton # presionado */
                returnVal = '#';
                break;
            case TM_KEYPAD_Button_A:           /* Boton A presionado */
                returnVal = 'A';
                break;
            case TM_KEYPAD_Button_B:           /* Boton B presionado */
                returnVal = 'B';
                break;
            case TM_KEYPAD_Button_C:           /* Boton C presionado */
                returnVal = 'C';
                break;
        }
    }
}

```

```

        returnVal = 'C';
        break;
    case TM_KEYPAD_Button_D:           /* Boton D presionado */
        returnVal = 'D';
        break;
    default:
        returnVal = '';
        break;
    }
}

return returnVal;
}

/** Funcion encargada de controlar el teclado cuando se ha seleccionado el menu A
 * (Menu encargado de programar el reloj del Sistema)
 */
void readKeypadMenuA(char character, int *checkingPassword, int *checkingHourCurrent,
char *menu, char *previousChar,
        char* *temporal, int *generalClockConfigured, int *alarmConfigured,
int alarmClockConfigured,
        char* *currentPassword) {
    if ((character == 'A') || (character == 'B') || (character == 'C') || (character == 'D')) {
        *previousChar = '';
        *menu = 'A';
    } else if (*checkingPassword == 1) {
        writeDisplay("Introduzca", "Password Actual");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkPassword(*temporal, *currentPassword) == 1) {
                *checkingHourCurrent = 1;
                writeDisplay("Password", "Correcta");
                Delay(3000);
                writeDisplay("Introduzca", "Hora Actual");
            } else {
                *menu = 'N';
                writeDisplay("Password", "Incorrecta");
                Delay(5000);
                cleanDisplay();
            }
            *temporal = "";
            *checkingPassword = 0;
        }
    } else if (*checkingHourCurrent == 1) {
        writeDisplay("Introduzca", "Hora Actual");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkHour(*temporal) == 1) {
                programRTC(temporal);
                *generalClockConfigured = 1;
                if (alarmClockConfigured == 1) {
                    *alarmConfigured = 1;
                } else {
                    *alarmConfigured = 0;
                }
                writeDisplay("Hora Actual", "Configurada");
            } else {
                writeDisplay("Hora Actual", "Incorrecta");
            }
        }
        *checkingHourCurrent = 0;
    }
}

```

```

        *menu = 'N';
        *temporal = "";
        Delay(5000);
        cleanDisplay();
    }
}

/** Funcion encargada de controlar el teclado cuando se ha seleccionado el menu B
 * (Menu encargado de cambiar el password del Sistema)
 */
void readKeypadMenuB(char character, int *checkingPassword, int *checkingPasswordNew,
int *checkingPasswordNewAux,
        char *menu, char *previousChar, char* *temporal, char*
*temporalAux, char* *currentPassword) {
    if ((character == 'A') || (character == 'B') || (character == 'C') || (character == 'D')) {
        *previousChar = ' ';
        *menu = 'B';
    } else if (*checkingPassword == 1) {
        writeDisplay("Introduza", "Password Actual");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkPassword(*temporal, *currentPassword) == 1) {
                *checkingPasswordNew = 1;
                writeDisplay("Password", "Correcta");
                Delay(3000);
                writeDisplay("Introduza", "Nueva Password");
            } else {
                *menu = 'N';
                writeDisplay("Password", "Incorrecta");
                Delay(5000);
                cleanDisplay();
            }
            *checkingPassword = 0;
            *temporal = "";
        }
    } else if (*checkingPasswordNew == 1) {
        writeDisplay("Introduza", "Nueva Password");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            *checkingPasswordNew = 0;
            *checkingPasswordNewAux = 1;
            writeDisplay("Confirme", "Nueva Password");
        }
    } else if (*checkingPassword == 1) {
        writeDisplay("Confirme", "Nueva Password");
        *temporalAux = *temporalAux + character;
        if (length(*temporalAux) == 4) {
            if (checkPassword(*temporal, *temporalAux) == 1) {
                *currentPassword = *temporal;
                writeDisplay("Cambio de Password", "Correcto");
            } else {
                writeDisplay("Cambio Password", "Incorrecto");
            }
            *checkingPasswordNewAux = 0;
            *temporal = "";
            *temporalAux = "";
            *menu = 'N';
            Delay(5000);
            cleanDisplay();
        }
    }
}

```

```

        }
    }

/** Funcion encargada de controlar el teclado cuando se ha seleccionado el menu C
 * (Menu encargado de programar las horas de inicio y fin de alarma activa del Sistema)
 */
void readKeypadMenuC(char character, int *checkingPassword, int *checkingHourBegin, int
*checkingHourEnd, char *menu, char *previousChar,
char* *temporal, char* *temporalAux, int *currentAlarmClockBegin,
int *currentAlarmClockEnd,
int generalClockConfigured, int *alarmConfigured, int
*alarmClockConfigured, char* *currentPassword) {
    if ((character == 'A') || (character == 'B') || (character == 'C') || (character == 'D')) {
        *previousChar = ' ';
        *menu = 'C';
    } else if (*checkingPassword == 1) {
        writeDisplay("Introduzca", "Password Actual");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkPassword(*temporal, *currentPassword) == 1) {
                *checkingHourBegin = 1;
                writeDisplay("Password", "Correcta");
                Delay(3000);
                writeDisplay("Introduzca", "Hora Inicio");
            } else {
                *menu = 'N';
                writeDisplay("Password", "Incorrecta");
                Delay(5000);
                cleanDisplay();
            }
            *checkingPassword = 0;
            *temporal = "";
        }
    } else if (*checkingHourBegin == 1) {
        writeDisplay("Introduzca", "Hora Inicio");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkHour(*temporal) == 1) {
                *checkingHourEnd = 1;
                writeDisplay("Hora Inicio", "Correcta");
                Delay(3000);
                writeDisplay("Introduzca", "Hora Fin");
            } else {
                writeDisplay("Hora Inicio", "Incorrecta");
                *menu = 'N';
                Delay(5000);
                cleanDisplay();
            }
            *checkingHourBegin = 0;
            *temporal = "";
        }
    } else if (*checkingHourEnd == 1) {
        writeDisplay("Introduzca", "Hora Fin");
        *temporalAux = *temporalAux + character;
        if (length(*temporalAux) == 4) {
            if (checkHour(*temporalAux) == 1) {
                *currentAlarmClockBegin = (atoi(temporal[0])*1000) +
(atoi(temporal[1])*100)

```

```

        + (atoi(temporal[2])*10) +
atoi(temporal[3]);
        *currentAlarmClockEnd = (atoi(temporalAux[0])*1000) +
(atoi(temporalAux[1])*100)
        + (atoi(temporalAux[2])*10) +
atoi(temporalAux[3]);
        *alarmClockConfigured = 1;
        if (generalClockConfigured == 1) {
            *alarmConfigured = 1;
        } else {
            *alarmConfigured = 0;
        }
        writeDisplay("Hora Fin", "Correcta");
        Delay(3000);
        writeDisplay("Alarma", "Programada");
    } else {
        writeDisplay("Hora Fin", "Incorrecta");
    }
    *checkingHourEnd = 0;
    *temporal = "";
    *temporalAux = "";
    *menu = 'N';
    Delay(5000);
    cleanDisplay();
}
}

/** Funcion encargada de controlar el teclado cuando se ha seleccionado el menu D
 * (Menu encargado de programar iniciar o parar la alarma del Sistema)
 */
void readKeypadMenuD(char character, int *checkingPassword, char *menu, char
*previousChar, char* *temporal,
int *alarmConfigured, int *alarmStarted, int *alarmFired, char*
*currentPassword) {
    if ((character == 'A') || (character == 'B') || (character == 'C') || (character == 'D')) {
        *previousChar = ' ';
        *menu = 'D';
    } else if (*checkingPassword == 1) {
        writeDisplay("Introduzca", "Password Actual");
        *temporal = *temporal + character;
        if (length(*temporal) == 4) {
            if (checkPassword(*temporal, *currentPassword) == 1) {
                writeDisplay("Password", "Correcta");
                Delay(3000);
                if (*alarmConfigured == 1) {
                    if ((*alarmFired == 1) || (*alarmStarted == 1)) {
                        writeDisplay("Alarma", "Parada");
                        stopAlarm(alarmFired, alarmStarted);
                    } else {
                        writeDisplay("Alarma", "Iniciada");
                        startAlarm(alarmStarted);
                    }
                } else {
                    writeDisplay("Configure", "Alarma");
                }
            } else {
                writeDisplay("Password", "Incorrecta");
            }
        }
        *menu = 'N';
    }
}

```

```

        *temporal = "";
        *checkingPassword = 0;
        Delay(5000);
        cleanDisplay();
    }
}

void controlKeypad(int *generalClockConfigured, int *alarmClockConfigured, int
*alarmConfigured, int *alarmStarted,
                    int *alarmFired, char *menu, char* *temporal, char* *temporalAux,
char *previousChar,
                    int *currentAlarmClockBegin, int *currentAlarmClockEnd, char*
*currentPassword,
                    int *checkingPassword, int *checkingPasswordNew, int
*checkingPasswordNewAux, int *checkingHourCurrent,
                    int *checkingHourBegin, int *checkingHourEnd) {

    char character = readKeypad();
    if ((character != ' ') && (character != '*') && (character != '#')) {
        if (*menu == 'N') {
            if (character == 'A') {
                readKeypadMenuA(character, checkingPassword,
                checkingHourCurrent, menu, previousChar, temporal,
                generalClockConfigured, alarmConfigured,
                *alarmClockConfigured, currentPassword);
            } else if (character == 'B') {
                readKeypadMenuB(character, checkingPassword,
                checkingPasswordNew, checkingPasswordNewAux,
                menu, previousChar, temporal,
                temporalAux, currentPassword);
            } else if (character == 'C') {
                readKeypadMenuC(character, checkingPassword,
                checkingHourBegin, checkingHourEnd, menu, previousChar,
                temporal, temporalAux,
                currentAlarmClockBegin, currentAlarmClockEnd,
                *generalClockConfigured,
                alarmConfigured, alarmClockConfigured, currentPassword);
            } else if (character == 'D') {
                readKeypadMenuD(character, checkingPassword, menu,
                previousChar, temporal,
                alarmConfigured, alarmStarted,
                alarmFired, currentPassword);
            }
        } else {
            *checkingPassword = 1;
            if (*menu == 'A') {
                readKeypadMenuA(character, checkingPassword,
                checkingHourCurrent, menu, previousChar, temporal,
                generalClockConfigured, alarmConfigured,
                *alarmClockConfigured, currentPassword);
            } else if (*menu == 'B') {
                readKeypadMenuB(character, checkingPassword,
                checkingPasswordNew, checkingPasswordNewAux,
                menu, previousChar, temporal,
                temporalAux, currentPassword);
            } else if (*menu == 'C') {
                readKeypadMenuC(character, checkingPassword,
                checkingHourBegin, checkingHourEnd, menu, previousChar,

```

```

temporal, temporalAux,
currentAlarmClockBegin, currentAlarmClockEnd,
*generalClockConfigured,
alarmConfigured, alarmClockConfigured, currentPassword);
} else if (*menu == 'D') {
    readKeypadMenuD(character, checkingPassword, menu,
previousChar, temporal,
alarmConfigured, alarmStarted,
alarmFired, currentPassword);
} else {
    *previousChar = ' ';
}
} else {
    *previousChar = ' ';
}
}
//********************************************************************

/** Archivo : leds.h */
/** Resumen : control de los leds del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#ifndef _LEDS_H_
#define _LEDS_H_


/** Inicializa los leds del Sistema (mapeo de pines en defines.h) */
void initLeds(void);

/** Activa o desactiva un led:
* activateLed: 1 = activar el led; 0 = desactivar el led
* numLed: selecciona el led a activar/desactivar (usar constantes de led)
*/
void activateLed(int activateLed, int numLed);

/** Funcion principal para controlar los leds del Sistema en funcion de las variables de estado */
void controlLeds(int alarmFired, int alarmStarted, int alarmConfigured);

#endif
//********************************************************************

/** Archivo : leds.c */
/** Resumen : control de los leds del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include "leds.h"
#include "tm_stm32f4_delay.h"
#include "defines.h"

/** Constante para referencia del led rojo */
int LED_RED = 0;

```

```

/** Constante para referencia del led naranja */
int LED_ORANGE = 1;
/** Constante para referencia del led verde */
int LED_GREEN = 2;

void initLeds(void) {
    // LED ROJO
    GPIO_InitTypeDef GPIO_InitDefRed;
    RCC_AHB1PeriphClockCmd(CLK_LINE_LED_RED, ENABLE);
    GPIO_InitDefRed.GPIO_Pin = PIN_LED_RED;
    GPIO_InitDefRed.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitDefRed.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefRed.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitDefRed.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_LED_RED, &GPIO_InitDefRed);
    GPIO_ResetBits(GPIO_LED_RED, PIN_LED_RED);

    // LED NARANJA
    GPIO_InitTypeDef GPIO_InitDefOrange;
    RCC_AHB1PeriphClockCmd(CLK_LINE_LED_ORANGE, ENABLE);
    GPIO_InitDefOrange.GPIO_Pin = PIN_LED_ORANGE;
    GPIO_InitDefOrange.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitDefOrange.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefOrange.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitDefOrange.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_LED_ORANGE, &GPIO_InitDefOrange);
    GPIO_ResetBits(GPIO_LED_ORANGE, PIN_LED_ORANGE);

    // LED VERDE
    GPIO_InitTypeDef GPIO_InitDefGreen;
    RCC_AHB1PeriphClockCmd(CLK_LINE_LED_GREEN, ENABLE);
    GPIO_InitDefGreen.GPIO_Pin = PIN_LED_GREEN;
    GPIO_InitDefGreen.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitDefGreen.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefGreen.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitDefGreen.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_LED_GREEN, &GPIO_InitDefGreen);
    GPIO_ResetBits(GPIO_LED_GREEN, PIN_LED_GREEN);
}

void activateLed(int activateLed, int numLed) {
    if (numLed == LED_RED) {
        if (numLed == 1) {
            GPIO_SetBits(GPIO_LED_RED, PIN_LED_RED);
        } else {
            GPIO_ResetBits(GPIO_LED_RED, PIN_LED_RED);
        }
    } else if (numLed == LED_ORANGE) {
        if (numLed == 1) {
            GPIO_SetBits(GPIO_LED_ORANGE, PIN_LED_ORANGE);
        } else {
            GPIO_ResetBits(GPIO_LED_ORANGE, PIN_LED_ORANGE);
        }
    } else if (numLed == LED_GREEN) {
        if (numLed == 1) {
            GPIO_SetBits(GPIO_LED_GREEN, PIN_LED_GREEN);
        } else {
            GPIO_ResetBits(GPIO_LED_GREEN, PIN_LED_GREEN);
        }
    }
}

```

```

/** Funcion que hace parpadear un led (seleccionado con constantes de led *)
void blinkingLed(int numLed) {
    activateLed(numLed,1);
    Delay(100);
    activateLed(numLed,0);
    Delay(100);
    activateLed(numLed,1);
    Delay(100);
    activateLed(numLed,0);
}

void controlLeds(int alarmFired, int alarmStarted, int alarmConfigured) {
    if (alarmFired == 1) {
        blinkingLed(LED_RED);
        activateLed(LED_GREEN, 1);
        activateLed(LED_ORANGE, 0);
    } else if (alarmStarted == 1) {
        activateLed(LED_GREEN, 1);
        activateLed(LED_ORANGE, 0);
        activateLed(LED_RED, 0);
    } else if (alarmConfigured == 1) {
        activateLed(LED_ORANGE, 1);
        activateLed(LED_GREEN, 0);
        activateLed(LED_RED, 0);
    } else {
        activateLed(LED_RED, 1);
        activateLed(LED_GREEN, 0);
        activateLed(LED_ORANGE, 0);
    }
}
//********************************************************************

/** Archivo : rele.h */
/** Resumen : control del rele del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#ifndef _RELE_H_
#define _RELE_H_

/** Inicializa el rele del Sistema (mapeo de pines en defines.h) */
void initRelay(void);

/** Activa o desactiva el rele del Sistema:
 *  activateRelay: 1 = activar rele; 0 = desactivar rele
 */
void activateRelay(int activateRelay);

#endif
//********************************************************************

/** Archivo : rele.c */
/** Resumen : control del rele del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>

```

```

#include "rele.h"
#include "defines.h"

void initRelay(void) {
    GPIO_InitTypeDef GPIO_InitDefRelay;
    RCC_AHB1PeriphClockCmd(CLK_LINE_RELAY, ENABLE);
    GPIO_InitDefRelay.GPIO_Pin = PIN_RELAY;
    GPIO_InitDefRelay.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitDefRelay.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefRelay.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitDefRelay.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_RELAY, &GPIO_InitDefRelay);
    GPIO_ResetBits(GPIO_RELAY, PIN_RELAY);
}

void activateRelay(int activateRelay) {
    if (activateRelay == 1) {
        GPIO_SetBits(GPIO_RELAY, PIN_RELAY);
    } else {
        GPIO_ResetBits(GPIO_RELAY, PIN_RELAY);
    }
}
//********************************************************************

/** Archivo : rfid.h */ 
/** Resumen : control del rfid del Sistema */ 
/** Autor : Miguel Angel Rosales Navarro */ 
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"

#ifndef _RFID_H_
#define _RFID_H_

/** Inicializa el rfid del Sistema (mapeo de pines en defines.h) */
void initRFID(void);

/** Funcion principal para controlar el rfid del Sistema en funcion de las variables de estado */
void controlRFID(int *alarmFired, int *alarmStarted, int *alarmConfigured, uint8_t* currentRfidPassword);

#endif
//********************************************************************

/** Archivo : rfid.c */ 
/** Resumen : control del rfid del Sistema */ 
/** Autor : Miguel Angel Rosales Navarro */ 
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include "rfid.h"
#include "utilidades.h"
#include "defines.h"
#include "tm_stm32f4_mfrc522.h"

void initRFID(void) {

```

```

        // Funcion incluida en "tm_stm32f4_mfrc522.h"
        TM_MFRC522_Init();
    }

/** Comprueba si se hay una tarjeta cerca del lector y si esta autorizada
 * Si devuelve 0 no esta autorizada, si devuelve 1 esta autorizada.
 */
int readRFID(uint8_t* currentRfidPassword) {
    int returnVal = 0;
    uint8_t cardRead[5];
    // Funcion incluida en "tm_stm32f4_mfrc522.h"
    if (TM_MFRC522_Check(cardRead) == MI_OK) {
        // Funcion incluida en "tm_stm32f4_mfrc522.h"
        if (TM_MFRC522_Compare(cardRead, currentRfidPassword) == MI_OK) {
            returnVal = 1;
        }
    }
    return returnVal;
}

void controlRFID(int *alarmFired, int *alarmStarted, int *alarmConfigured, uint8_t*
currentRfidPassword) {
    if (readRFID(currentRfidPassword) == 1) {
        if (*alarmFired == 1) {
            stopAlarm(alarmFired, alarmStarted);
        } else if (*alarmStarted == 1) {
            stopAlarm(alarmFired, alarmStarted);
        } else if (*alarmConfigured == 1) {
            startAlarm(alarmStarted);
        }
    }
}
//********************************************************************

/** Archivo : rtc.h */ 
/** Resumen : control del rtc del Sistema */ 
/** Autor : Miguel Angel Rosales Navarro */ 
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"

#ifndef _RTC_H_
#define _RTC_H_

/** Inicializa el rtc del Sistema (mapeo de pines en defines.h) */
void initRTC(void);

/** Programa el rtc del Sistema con la hora (HHMM) enviada */
void programRTC(char* currentHour);

/** Lee el rtc del Sistema devolviendo un int que representa la hora actual (HHMM) */
int readRTC(void);

#endif
//********************************************************************

/** Archivo : rtc.c */ 
/** Resumen : control del rtc del Sistema */ 
*/

```

```


/** Autor : Miguel Angel Rosales Navarro */
/*****************************************/
#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include <stdlib.h>
#include "rtc.h"
#include "defines.h"
#include "display.h"
#include "tm_stm32f4_ds1307.h"

void initRTC(void) {
    // Funcion incluida en "tm_stm32f4_ds1307.h"
    TM_DS1307_Init();
}

void programRTC(char* *currentHour){
    TM_DS1307_Time_t time;
    /* Establecer fecha dummy 2 de mayo de 2016 (".day" guarda el dia de la semana que
    es 1, lunes) */
    time.year = 16;
    time.month = 5;
    time.date = 2;
    time.day = 1;
    int hours = (atoi(currentHour[0])*10) + atoi(currentHour[1]);
    int minutes = (atoi(currentHour[2])*10) + atoi(currentHour[3]);
    time.hours = hours;
    time.minutes = minutes;
    time.seconds = 0;
    // Funcion incluida en "tm_stm32f4_ds1307.h"
    TM_DS1307_SetDateTime(&time);
}

int readRTC(void){
    int returnVal = 0;
    TM_DS1307_Time_t time;
    // Funcion incluida en "tm_stm32f4_ds1307.h"
    TM_DS1307_EnableOutputPin(TM_DS1307_OutputFrequency_4096Hz);
    // Funcion incluida en "tm_stm32f4_ds1307.h"
    TM_DS1307_GetDateTime(&time);
    returnVal = (time.hours*100) + time.minutes;
    return returnVal;
}

/*****************************************/
/** Archivo : sensor_magntico.h */
/** Resumen : control del sensor magnetico del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
/*****************************************/

#ifndef _SENSORMAGNETICO_H_
#define _SENSORMAGNETICO_H_

/** Inicializa el sensor magnetico del Sistema (mapeo de pines en defines.h) */
void initMagnetic(void);
/** Lee el sensor magnetico del Sistema devolviendo 1 si hay campo magnetico o 0 si no hay */
int readMagnetic(void);


```

```

#endif
/*****
** Archivo : sensor_magntico.c
** Resumen : control del sensor magntico del Sistema
** Autor : Miguel Angel Rosales Navarro
****/



#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include "sensor_magntico.h"
#include "defines.h"

void initMagnetic(void) {
    GPIO_InitTypeDef GPIO_InitDefMagnetic;
    RCC_AHB1PeriphClockCmd(CLK_LINE_SENSOR_MAGNETIC, ENABLE);
    GPIO_InitDefMagnetic.GPIO_Pin = PIN_SENSOR_MAGNETIC;
    GPIO_InitDefMagnetic.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitDefMagnetic.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefMagnetic.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitDefMagnetic.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_SENSOR_MAGNETIC, &GPIO_InitDefMagnetic);
}

int readMagnetic(void) {
    int returnVal = 0;
    if(GPIO_ReadInputDataBit(GPIO_SENSOR_MAGNETIC, PIN_SENSOR_MAGNETIC))
    {
        returnVal = 1;
    }
    return returnVal;
}
/*****



** Archivo : sensor_movimiento.h
** Resumen : control del sensor de movimiento del Sistema
** Autor : Miguel Angel Rosales Navarro
****/



#ifndef _SENSORMOVIMIENTO_H_
#define _SENSORMOVIMIENTO_H_

/* Inicializa el sensor de movimiento del Sistema (mapeo de pines en defines.h) */
void initMovement(void);
/* Lee el sensor de movimiento del Sistema devolviendo 1 si hay movimiento o 0 si no hay */
int readMovement(void);

#endif
/*****



** Archivo : sensor_movement.c
** Resumen : control del sensor de movimiento del Sistema
** Autor : Miguel Angel Rosales Navarro
****/



#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"

```

```

#include <stdio.h>
#include "sensor_movimiento.h"
#include "defines.h"

void initMovement(void) {
    GPIO_InitTypeDef GPIO_InitDefMovement;
    RCC_AHB1PeriphClockCmd(CLK_LINE_SENSOR_MOVEMENT, ENABLE);
    GPIO_InitDefMovement.GPIO_Pin = PIN_SENSOR_MOVEMENT;
    GPIO_InitDefMovement.GPIO_Mode = GPIO_Mode_IN;
    GPIO_InitDefMovement.GPIO_OType = GPIO_OType_PP;
    GPIO_InitDefMovement.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_InitDefMovement.GPIO_Speed = GPIO_Speed_100MHz;
    GPIO_Init(GPIO_SENSOR_MOVEMENT, &GPIO_InitDefMovement);
}

int readMovement(void) {
    int returnVal = 0;
    if(GPIO_ReadInputDataBit(GPIO_SENSOR_MOVEMENT,
    PIN_SENSOR_MOVEMENT)) {
        returnVal = 1;
    }
    return returnVal;
}
//********************************************************************

//********************************************************************

/** Archivo : speaker.c */
/** Resumen : control de la senyal para el speaker del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"

#ifndef _SPEAKER_H_
#define _SPEAKER_H_

/** Inicializa el DAC que controla la senyal que va al speaker del Sistema
 * (mapeo de pines en defines.h)
 */
void initSpeaker(void);

/** Activa o desactiva la senyal que va al speaker del Sistema:
 * activateSound: 1 = activar la senyal; 0 = desactivar la senyal.
 */
void activateSound(int activateSound);

#endif
//********************************************************************

/** Archivo : speaker.c */
/** Resumen : control de la senyal para el speaker del Sistema */
/** Autor : Miguel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include "stm32f4xx_gpio.h"
#include <stdio.h>
#include "speaker.h"
#include "defines.h"

```

```

#include "tm_stm32f4_dac_signal.h"

void initSpeaker(void) {
    // Funcion incluida en "tm_stm32f4_dac_signal.h"
    TM_DAC_SIGNAL_Init(TM_DAC1, TIM4);
}

void activateSound(int activateSound) {
    if (activateSound == 1) {
        // Se activa el sonido con un seno de 440Hz (nota "La")
        // Funcion incluida en "tm_stm32f4_dac_signal.h"
        TM_DAC_SIGNAL_SetSignal(TM_DAC2, TM_DAC_SIGNAL_Signal_Sinus, 440);
    } else {
        // Se desactiva el sonido con una senyal cuadrada modificada para que sus dos
        valores sean 0
        // Funcion incluida en "tm_stm32f4_dac_signal.h"
        TM_DAC_SIGNAL_SetSignal(TM_DAC2, TM_DAC_SIGNAL_Signal_Square,
        440);
    }
}
//********************************************************************

/** Archivo : utilidades.d */
/** Resumen : libreria con funciones utiles a usar por el Sistema */
/** Autor : Miquel Angel Rosales Navarro */
//********************************************************************

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"

#ifndef _UTILIDADES_H_
#define _UTILIDADES_H_

/** Devuelve la longitud de la cadena argumento */
int length(char* arrayChar);

/** Comprueba que las dos cadenas argumento son iguales:
 * Devuelve: 0 = no son iguales; 1 = son iguales
 */
int checkPassword(char* password, char* auxPassword);

/** Comprueba que la cadena argumento contiene una hora valida: numero entre 0 y 2359
 * Devuelve: 1 = la hora es valida; 0 = la hora no es valida
 */
int checkHour(char* hour);

/** Comprueba si una hora se encuentra entre otras dos horas:
 * currentTime: hora a comprobar
 * currentAlarmClockBegin: hora inicio del intervalo a comprobar
 * currentAlarmClockEnd: hora fin del intervalo a comprobar
 * Devuelve: 1 = la hora esta en el intervalo; 0 = la hora no esta en el intervalo
 */
int alarmInProgrammedTime(int currentTime, int currentAlarmClockBegin, int
currentAlarmClockEnd);

/** Dispara la alarma del Sistema actualizando la variable de estado alarmFired */
void fireAlarm(int *alarmFired);

/** Para la alarma del Sistema actualizando las variables de estado alarmFired y alarmStarted
 */
void stopAlarm(int *alarmFired, int *alarmStarted);

```

```

/** Inicia la alarma del Sistema actualizando la variable de estado alarmStarted */
void startAlarm(int *alarmStarted);

/** Funcion principal para controlar los sensores (movimiento y magnetico) del Sistema
 * y en funcion de las variables de estado disparar la alarma
 */
void controlSensors(int *alarmFired, int alarmStarted, int currentAlarmClockBegin, int
currentAlarmClockEnd);

#endif
/***** */

/***** */
/** Archivo : utilidades.c */
/** Resumen : libreria con funciones utiles a usar por el Sistema */
/** Autor : Miguel Angel Rosales Navarro */
/***** */

#include "stm32f4xx.h"
#include "stm32f4_discovery.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "utilidades.h"
#include "rele.h"
#include "speaker.h"
#include "leds.h"
#include "sensor_magnetico.h"
#include "sensor_movimiento.h"
#include "rtc.h"
#include "tm_stm32f4_delay.h"

int length(char* arrayChar) {
    int returnVal = strlen(arrayChar);
    return returnVal;
}

int checkPassword(char* password, char* auxPassword) {
    int returnVal = 0;
    if (strcmp(password, auxPassword) == 0) {
        returnVal = 1;
    }
    return returnVal;
}

int checkHour(char* hour) {
    int returnVal = 0;
    int intHour = atoi(hour);
    if ((intHour >=0) && (intHour < 2400)) {
        returnVal = 1;
    }
    return returnVal;
}

int alarmInProgrammedTime(int currentTime, int currentAlarmClockBegin, int
currentAlarmClockEnd) {
    int returnVal = 0;

    if (currentAlarmClockBegin < currentAlarmClockEnd) {

```

```

        if ((currentTime >= currentAlarmClockBegin) && (currentTime <=
currentAlarmClockEnd)) {
            returnVal = 1;
        }
    } else if (currentAlarmClockBegin > currentAlarmClockEnd) {
        if (((currentTime >= currentAlarmClockBegin) && (currentTime <= 0))
||((currentTime >= 0) && (currentTime <= currentAlarmClockEnd))) {
            returnVal = 1;
        }
    }

    return returnVal;
}

void fireAlarm(int *alarmFired) {
    *alarmFired = 1;
    activateRelay(1);
    activateSound(1);
}
void stopAlarm(int *alarmFired, int *alarmStarted) {
    *alarmFired = 0;
    *alarmStarted = 0;
    activateRelay(0);
    activateSound(0);
}
void startAlarm(int *alarmStarted) {
    activateLed(0, 0);
    activateLed(1, 0);
    activateLed(2, 1);
    Delay(30000);
    *alarmStarted = 1;
}

void controlSensors(int *alarmFired, int alarmStarted, int currentAlarmClockBegin, int
currentAlarmClockEnd) {
    if ((*alarmFired == 0) && (alarmStarted == 1)) {
        int mov = readMovement();
        int mag = readMagnetic();
        int currentTime = readRTC();
        if (alarmInProgrammedTime(currentTime, currentAlarmClockBegin,
currentAlarmClockEnd) == 1)
            && ((mov == 1) || (mag == 0))) {
            fireAlarm(alarmFired);
        }
    }
}
//*****************************************************************/

```

### Librerías externas:

Se han usado las siguientes librerías externas (si no se especifica es porque incluye el archivo ".h" y el archivo ".c"):

“attributes.h”, “lcd\_hd44780”, “tm\_stm32f4\_dac\_signal”, “tm\_stm32f4\_delay”,  
“tm\_stm32f4\_ds1307”, “tm\_stm32f4\_gpio”, “tm\_stm32f4\_i2c”,  
“tm\_stm32f4\_keypad”, “tm\_stm32f4\_mfrc522”, “tm\_stm32f4\_spi”,  
“tm\_stm32f4\_timer\_properties”.

Todas se pueden descargar desde [9]: <http://stm32f4-discovery.com/>

# Bibliografía.

A continuación se muestran los recursos bibliográficos que han sido usados para la elaboración de este documento, listados en orden de aparición:

## **Recurso 1:**

Título del recurso: "Los ladrones lo tienen fácil: cuatro millones de casas sin una sola medida de seguridad"

Tipo de recurso: página web.

Autor/es: Elena Sanz.

Fecha: 17-abril-2015; Fecha de consulta: 01-mar-2016

Fuente de información:

[http://www.elconfidencial.com/vivienda/2015-04-17/los-ladrones-lo-tienen-facil-cuatro-millones-de-casas-sin-una-sola-medida-de-seguridad\\_760262/](http://www.elconfidencial.com/vivienda/2015-04-17/los-ladrones-lo-tienen-facil-cuatro-millones-de-casas-sin-una-sola-medida-de-seguridad_760262/)

## **Recurso 2:**

Título del recurso: Plan de Estudios de la assignatura Trabajo Fin de Máster.

Tipo de recurso: página web.

Autor/es: Universitat Oberta de Catalunya.

Fecha de consulta: 01-mar-2016

Fuente de información:

[http://cv.uoc.edu/tren/trenacc/web/GAT\\_EXP.PLANDOCENTE?any\\_academico=20152&cod\\_asignatura=M1.612&idioma=CAS&pagina=PD\\_PREV\\_SECRE&ache=S](http://cv.uoc.edu/tren/trenacc/web/GAT_EXP.PLANDOCENTE?any_academico=20152&cod_asignatura=M1.612&idioma=CAS&pagina=PD_PREV_SECRE&ache=S)

## **Recurso 3:**

Título del recurso: Páginas web de empresas especializadas en seguridad.

Tipo de recurso: página web.

Autor/es: N/A

Fecha de consulta: 02-abril-2016

Fuente de información:

Securitas Direct → <http://www.securitasdirect.es/Ip/g/alarmas-securitas-direct.php>

Tyco → <http://alarmainteligente.tyco.es>

Prosegur → <http://www.alarmahogarprosegur.com>

Verisure → [https://www.verisure.es/alarma/landings/g/alarmas\\_verisure.html](https://www.verisure.es/alarma/landings/g/alarmas_verisure.html)

## **Recurso 4:**

Título del recurso: "Sitio web de la empresa SINcuotas, S.L."

Tipo de recurso: página web.

Autor/es: SINcuotas, S.L.

Fecha de consulta: 02-abril-2016

Fuente de información:

<http://www.tualarmasincuotas.es/paginas/nosotros>

**Recurso 5:**

Título del recurso: “Alarma Sin Cuotas Independiente”

Tipo de recurso: página web.

Autor/es: SINcuotas, S.L.

Fecha de consulta: 02-abril-2016

Fuente de información:

<http://www.tualarmasincuotas.es/productos/alarma-sin-cuotas-independiente>

**Recurso 6:**

Título del recurso: “Cómo hacer una alarma casera”

Tipo de recurso: página web.

Autor/es: Jorge Prudencio.

Fecha de consulta: 02-abril-2016

Fuente de información:

<http://comohacer.eu/como-hacer-una-alarma-casera/>

**Recurso 7:**

Título del recurso: Varias alarmas basadas en la plataforma Arduino.

Tipo de recurso: página web.

Autor/es: N/A

Fecha de consulta: 02-abril-2016

Fuente de información:

Ejemplo 1 → <http://descubrearduino.com/un-sistema-de-alarma-para-tu-hogar-creado-con-arduino/>

Ejemplo 2 → <http://arduinotesting.blogspot.com.es/2010/12/pequena-alarma-anti-robo.html>

Ejemplo 3 → <https://www.youtube.com/watch?v=3UmN8CJlktI>

**Recurso 8:**

Título del recurso: “UM1472 User manual”

Tipo de recurso: manual de usuario alojado en web.

Autor/es: “ST”.

Fecha: 04-feb-2016; Fecha de consulta: abril y mayo de 2016.

Fuente de información:

[http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user\\_manual/DM00039084.pdf](http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00039084.pdf)  
<https://docs.google.com/viewer?a=v&pid=sites&srcid=bXJ0LnVzdS5IZHV8Y3ViZS1zYXQtdGVhbS1yZXNvdXJjZXxneDozYmQwMWUwNTViNjE3NzU2>

**Recurso 9:**

Título del recurso: “Libraries and tutorials for STM32Fxxx”

Tipo de recurso: página web.

Autor/es: Tilen Majerle.  
Fecha de consulta: 02-abril-2016  
Fuente de información:  
<http://stm32f4-discovery.com/>

#### **Recurso 10:**

Título del recurso: “Uso de acelerómetros para el control de dispositivos mediante captura de movimiento”  
Autor/es: Villaluenga Morán, José Luis  
Tipo de recurso: trabajo fin de máster.abril y mayo de 2016  
Fuente de información:  
<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/42670/7/jvillaluengaTFM0615memoria.pdf>

#### **Recurso 11:**

Título del recurso: “Library 15- DS1307 Real Time Clock for STM32F4”  
Tipo de recurso: página web.  
Autor/es: “tilz0R”.  
Fecha: 23-ago-2015; Fecha de consulta: abril, mayo de 2016.  
Fuente de información:  
<http://stm32f4-discovery.net/2014/05/library-15-ds1307-real-time-clock-for-stm32f429-discovery/>

#### **Recurso 12:**

Título del recurso: “Library 23- Read RFID tag with MF RC522 on STM32F4”  
Tipo de recurso: página web.  
Autor/es: “tilz0R”.  
Fecha: 13-jul-2014; Fecha de consulta: abril, mayo de 2016.  
Fuente de información:  
<http://stm32f4-discovery.com/2014/07/library-23-read-rfid-tag-mfrc522-stm32f4xx-devices/>

#### **Recurso 13:**

Título del recurso: “Motion Sensor Board”  
Tipo de recurso: página web con ejemplos y manual de usuario.  
Autor/es: “MikroElektronika”.  
Fecha de consulta: abril y mayo de 2016.  
Fuente de información:  
<http://www.mikroe.com/add-on-boards/various/motion-sensor/>

#### **Recurso 14:**

Título del recurso: “Library 32- Matrix keypad on STM32F4”  
Tipo de recurso: página web.  
Autor/es: “tilz0R”.

Fecha: 01-sep-2014; Fecha de consulta: abril, mayo de 2016.

Fuente de información:

<http://stm32f4-discovery.com/2014/09/library-32-matrix-keypad-stm32f4xx/>

### **Recurso 15:**

Título del recurso: “Library 16- Interfacing HD44780 LCD controller with STM32F4”

Tipo de recurso: página web.

Autor/es: “tilz0R”.

Fecha: 01-jun-2014; Fecha de consulta: abril, mayo de 2016.

Fuente de información:

<http://stm32f4-discovery.com/2014/06/library-16-interfacing-hd44780-lcd-controller-with-stm32f4/>

### **Recurso 16:**

Título del recurso: “Relay Click”

Tipo de recurso: página web con ejemplos y manual.

Autor/es: MikroElektronika.

Fecha de consulta: abril y mayo de 2016.

Fuente de información:

<http://www.mikroe.com/click/relay/>

### **Recurso 17:**

Título del recurso: “Library 02- STM32F429 Discovery GPIO tutorial with onboard leds and button”

Tipo de recurso: página web.

Autor/es: “tilz0R”.

Fecha: 06-abr-2014; Fecha de consulta: abril, mayo de 2016.

Fuente de información:

<http://stm32f4-discovery.com/2014/04/stm32f429-discovery-gpio-tutorial-with-onboard-leds-and-button/>

### **Recurso 18:**

Título del recurso: “Library 36- DAC Signal generator for STM32F4”

Tipo de recurso: página web.

Autor/es: “tilz0R”.

Fecha: 06-sep-2014; Fecha de consulta: abril, mayo de 2016.

Fuente de información:

<http://stm32f4-discovery.com/2014/09/library-36-dac-signal-generator-stm32f4/>

### **Recurso 19:**

Título del recurso: “Protege tu casa de los ladrones con estas apps antirrobo”

Tipo de recurso: página web.

Autor/es: Iván Ramírez.

Fecha: 14-ago-2014; Fecha de consulta: abril, mayo de 2016.

Fuente de información:

<http://articulos.softonic.com/protege-tu-casa-ladrones-con-apps>

**Recurso 20:**

Título del recurso: “Atollic, TrueSTUDIO”

Tipo de recurso: página web.

Autor/es: “Atollic”.

Fecha de consulta: mayo de 2016.

Fuente de información:

<http://timor.atollic.com/truestudio/>

**Recurso 21:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

STM32F4 Discovery → [http://www.aliexpress.com/item/Free-Shipping-STM32-DISCOVERY-Board-STM32F401C-DISCO-STM32F4-kit-STM32F401VC-Learning-STM32-Development-Board/2051908613.html?spm=2114.01010208.3.2.04J8Qf&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=810b41ef-3e6e-4053-af1b-7fce94d6ee76](http://www.aliexpress.com/item/Free-Shipping-STM32-DISCOVERY-Board-STM32F401C-DISCO-STM32F4-kit-STM32F401VC-Learning-STM32-Development-Board/2051908613.html?spm=2114.01010208.3.2.04J8Qf&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=810b41ef-3e6e-4053-af1b-7fce94d6ee76)

**Recurso 22:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

Fuente de alimentación MS-35-4 → [http://www.aliexpress.com/item/OSGD-high-quality-MS-35W-5V-7A-min-size-Small-scale-Switch-Power-supple-free-shopping/32613852436.html?spm=2114.01010208.3.182.14DCZA&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=f27e2dea-70be-491d-a62b-c7bd7f95fcab](http://www.aliexpress.com/item/OSGD-high-quality-MS-35W-5V-7A-min-size-Small-scale-Switch-Power-supple-free-shopping/32613852436.html?spm=2114.01010208.3.182.14DCZA&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=f27e2dea-70be-491d-a62b-c7bd7f95fcab)

**Recurso 23:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

LEDs → [http://www.aliexpress.com/item/100pcs-3mm-White-Green-Red-Blue-Yellow-LED-Light-Bulb-Emitting-Diode-Lamps-Brand-new/32615755061.html?spm=2114.01010208.3.31.RCHhv8&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10017\\_10021\\_507\\_10022\\_10020\\_10009\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=fe349219-f5bd-4292-b131-7fd2cc74c574](http://www.aliexpress.com/item/100pcs-3mm-White-Green-Red-Blue-Yellow-LED-Light-Bulb-Emitting-Diode-Lamps-Brand-new/32615755061.html?spm=2114.01010208.3.31.RCHhv8&ws_ab_test=searchweb201556_0,searchweb201602_4_10017_10021_507_10022_10020_10009_10008_10018_10019_101,searchweb201603_2&btsid=fe349219-f5bd-4292-b131-7fd2cc74c574)

### **Recurso 24:**

Título del recurso: “Páginas de venta de componentes Mikroe”

Tipo de recurso: página web.

Autor/es: “Mikroe”.

Fuentes de información:

STM32F4 Discovery Shield → <http://www.mikroe.com/stm32/stm32f4-discovery-shield/>

RTC → <http://www.mikroe.com/click/rtc/>

Relé → <http://www.mikroe.com/click/relay/>

RFID → <http://www.mikroe.com/click/rfid/>

Detector de movimiento → <http://www.mikroe.com/click/motion/>

Detector de apertura (magnético) → <http://www.mikroe.com/click/magneto/>

### **Recurso 25:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

Display LCD 16x2 → <http://www.aliexpress.com/item/1pcs-New-Character-LCD-Module-Display-LCM-1602-LCD1602-16X2-HD44780-Blue-Blacklight-Original-and-New/32573086114.html>

### **Recurso 26:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

Keypad 16x2 → [http://www.aliexpress.com/item/New-4-4-Matrix-Array-Matrix-Keyboard-16-Key-Membrane-Switch-Keypad/32255922477.html?spm=2114.01010208.3.2.KAYzAG&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=a5212096-e28f-498c-8a5a-9bf8f70cd8b0](http://www.aliexpress.com/item/New-4-4-Matrix-Array-Matrix-Keyboard-16-Key-Membrane-Switch-Keypad/32255922477.html?spm=2114.01010208.3.2.KAYzAG&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=a5212096-e28f-498c-8a5a-9bf8f70cd8b0)

**Recurso 27:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: "Aliexpress".

Fecha de consulta: mayo de 2016.

Fuentes de información:

RTC → [http://www.aliexpress.com/item/Free-shipping-20pcs-lot-The-Tiny-RTC-I2C-modules-24C32-memory-DS1307-clock-RTC-module-for-1876368739.html?spm=2114.01010208.3.2.QdefZn&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=979cf487-cf43-42f4-8f05-89c8eff63c7](http://www.aliexpress.com/item/Free-shipping-20pcs-lot-The-Tiny-RTC-I2C-modules-24C32-memory-DS1307-clock-RTC-module-for-1876368739.html?spm=2114.01010208.3.2.QdefZn&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=979cf487-cf43-42f4-8f05-89c8eff63c7)

**Recurso 28:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: "Aliexpress".

Fecha de consulta: mayo de 2016.

Fuentes de información:

Relé → [http://www.aliexpress.com/item/Relay-module-for-arduino-uno/32400577838.html?spm=2114.01010208.3.139.I424rc&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=53747513-9b08-412a-8d26-cfa7f4512757](http://www.aliexpress.com/item/Relay-module-for-arduino-uno/32400577838.html?spm=2114.01010208.3.139.I424rc&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=53747513-9b08-412a-8d26-cfa7f4512757)

**Recurso 29:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: "Aliexpress".

Fecha de consulta: mayo de 2016.

Fuentes de información:

Sensor de movimiento → [http://www.aliexpress.com/item/1pcs-High-Quality-HC-SR501-Infrared-PIR-Motion-Sensor-Module-For-Arduino-Raspberry-pi/32558562655.html?spm=2114.01010208.3.11.flHkwp&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_10032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=323d6637-56aa-478d-b672-3ed74f74a62e](http://www.aliexpress.com/item/1pcs-High-Quality-HC-SR501-Infrared-PIR-Motion-Sensor-Module-For-Arduino-Raspberry-pi/32558562655.html?spm=2114.01010208.3.11.flHkwp&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_10032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=323d6637-56aa-478d-b672-3ed74f74a62e)

**Recurso 30:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: "Aliexpress".

Fecha de consulta: mayo de 2016.

Fuentes de información:

Sensor magnético → [http://www.aliexpress.com/item/Standard-Hall-Magnetic-Sensor-Module-for-Arduino-AVR-PIC-Good-KY-003-New-Wholesale/32665617510.html?spm=2114.01010208.3.10.ltYyeu&ws\\_ab\\_test=searchweb201556\\_0,searchweb201602\\_4\\_10037\\_10017\\_10021\\_507\\_10022\\_1\\_0032\\_10009\\_10020\\_10008\\_10018\\_10019\\_101,searchweb201603\\_2&btsid=b377c23e-56fb-4d04-a124-a6fce4858c6](http://www.aliexpress.com/item/Standard-Hall-Magnetic-Sensor-Module-for-Arduino-AVR-PIC-Good-KY-003-New-Wholesale/32665617510.html?spm=2114.01010208.3.10.ltYyeu&ws_ab_test=searchweb201556_0,searchweb201602_4_10037_10017_10021_507_10022_1_0032_10009_10020_10008_10018_10019_101,searchweb201603_2&btsid=b377c23e-56fb-4d04-a124-a6fce4858c6)

**Recurso 31:**

Título del recurso: Página de venta de componentes

Tipo de recurso: página web.

Autor/es: “Aliexpress”.

Fecha de consulta: mayo de 2016.

Fuentes de información:

Lector RFID → <http://www.aliexpress.com/item/Free-shipping-MFRC-522-RC522-RFID-RF-IC-card-sensor-module-to-send-S50-Fudan-card/1623810751.html?spm=2114.01010208.8.64.FHSBMG>