Universidad de La Laguna

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

TRABAJO DE FIN DE GRADO:

Sistema de alarma para vivienda unifamiliar gestionada telemáticamente.

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

Alumnos: Santiago Lima Concepción
Kevin Moisés Meneses González

Tutores: Alejandro José Ayala Alfonso
Beatriz Rodríguez Mendoza

Junio, 2017

Agradecimientos.

En primer lugar, queremos agradecer a nuestro tutor Alejandro Ayala por su compromiso, enseñanza, excelente predisposición y sobre todo por su implicación. De su mano, hemos aprendido y podido desarrollar un proyecto del cual nos sentimos muy orgullosos. Sus consejos, ideas y su pasión nos han brindado la energía necesaria para completar el proyecto de una manera que jamás hubiéramos imaginado.

En segundo lugar, agradecer a Beatriz Rodríguez por su colaboración y gentileza.

Queremos agradecer a Delfín Darías por su ayuda y múltiples enseñanzas en el campo del diseño en 3D. Su asesoramiento nos ha sido de gran utilidad y ha despertado en nosotros una gran curiosidad con todo relacionado con este mundo. Su interés, implicación y perfeccionismo reflejan su excelencia como persona.

Agradecer por su colaboración a los profesores del departamento: Francisco Llopis, Fernando de la Rosa, José Carlos Sanluís y Oswaldo B. González

Finalmente, queremos dar las gracias a nuestros familiares por su apoyo y ánimos, así como a los compañeros de nuestra carrera por sus enseñanzas y por los grandes momentos vividos.

Índice

Abstract.

The aim of this project is to design an alarm system for a house controlled by a GSM modem. It has been divided into four main parts: programming a keyboard based control, managing calls and messages through a GSM modem, development of a mobile application and design of a box built using a 3D printer.

Firstly, we have used an Arduino Mega microcontroller to manage the movement sensors, the matrix keyboard, the display, the real time clock and the relay.

Secondly, we have focused on communications with the GSM modem for the transmission and reception of information via both SMS's and calls.

Thirdly, we have designed a mobile application that simplifies the system management.

Finally, a box was manufactured using a 3D printer in order to assemble the whole system.

CAPÍTULO I: Introducción general. Objetivos

Capítulo I: Introducción general. Objetivos

I.1.-Introducción general.

La palabra alarma nace en el siglo XVI cuando los españoles e italianos la usaban en las guerras para comunicar la existencia de algún peligro. Con el desarrollo de la segunda revolución industrial su uso fue extendiéndose. Así en 1853 Augustus Russell Pope de Somerville inventó la primera instalación de alarma electromagnética. Su funcionamiento consistía en una unidad que activaba un circuito eléctrico cuando una puerta o ventana se abría, de tal manera que la corriente eléctrica generaba un campo electromagnético el cual producía una vibración, permitiendo así que un martillo golpeara una campanilla de latón[1].

Figura I.1. Primer sistema de alarma inventado en 1853

En la actualidad las alarmas se utilizan en multitud de ambientes: teléfonos móviles, coches, casas, electrodomésticos, etc. En todos ellos la finalidad es la misma: informar de un determinado evento mediante un algún tipo de estímulo, visual o sonoro en la mayoría de casos.
En el ámbito de la domótica es cada vez más común el uso de sistemas de alarma. Ante el aumento en los últimos años de los robos en viviendas, las alarmas se han convertido en un dispositivo imprescindible en la mayoría de casos. El objetivo fundamental es avisar a los propietarios sobre la situación de emergencia, consiguiendo llamar la atención de los vecinos y autoridades, evitando con ello males mayores.
El objetivo del presente proyecto se ha centrado en el diseño e implementación de un dispositivo electrónico, que haciendo uso de un microcontrolador y un modem GSM, posibilita, controlar la seguridad de una vivienda o recinto. El control del sistema se puede hacer de forma presencial o telemática.

I.2.-Objetivos del proyecto. Descripción general del sistema.

La Figura I.2 muestra el diagrama general de bloques del sistema implementado. Su funcionamiento está controlado por un microcontrolador ATmega2560 de Atmel presente en una tarjeta Arduino Mega 2560 que será el encargado de procesar la información que llega de los diferentes sensores, gestionar las comunicaciones del módem, etc.

Con el mismo, se pretende disuadir la entrada de intrusos en una vivienda o recinto y, en caso de que ésta tenga lugar, detectar su presencia e iniciar un protocolo de actuaciones de seguridad encaminadas a inhibir las acciones del intruso e informar del hecho al propietario o propietarios de la vivienda o local.

Como se ha indicado anteriormente, en todo momento el control del sistema es realizado por un microcontrolador Arduino, que se comunica con los usuarios mediante un módulo GSM, un teclado y un display.

El display permite mostrar la información al usuario o requerirla para que éste la introduzca mediante el teclado (Figura I.2). Al mismo tiempo, mediante la utilización de un dispositivo móvil y el uso de un modem GSM, podrá recibir mensajes de alerta o enviarlos (mediante una aplicación realizada al efecto) para activar o desactivar la alarma.

El módem empleado fue el SIM 900 de la casa Simcom, que dispone de una interfaz de comunicación estándar industrial y un lector de tarjetas SIM integrado.

Para los sensores de movimiento se empleó el modelo DC-SS502 de la casa Sure Electronics, que hace uso de un sensor infrarrojo para detectar el movimiento de objetos y personas.

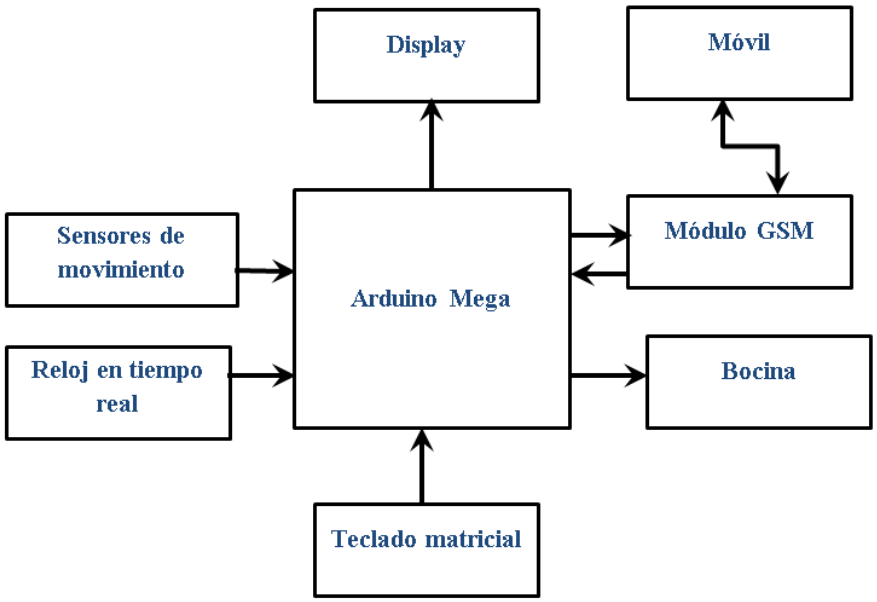


Figura I.2. Diagrama de bloques del sistema de alarma

I.3.-Estructura general del trabajo.

La memoria está dividida a lo largo de 8 capítulos en los cuales se exponen todos los aspectos relacionados con el proyecto. De esta manera se ha seguido la siguiente distribución:

El Capítulo I describe brevemente una visión global del proyecto, indica la finalidad del mismo y presentando la división general del trabajo.

El Capítulo II realiza una descripción general del microcontrolador empleado y de sus características más importantes.

Las herramientas de software utilizadas durante la ejecución del proyecto es realizada en el Capítulo III.

El Capítulo IV se centra en comentar el funcionamiento del software realizado de una manera sencilla mediante diagramas de flujo.

El Capítulo V detalla el funcionamiento de cada uno de los componentes que integran el sistema así como de sus características más importantes, para finalizar describiendo el funcionamiento global de la alarma.

A lo largo del Capítulo VI, se resumen el proceso de modelado y construcción de la carcasa exterior para la alarma.

Los resultados experimentales, verificaciones y demostraciones del funcionamiento del sistema se exponen en el Capítulo VII.

En el Capítulo VIII se presenta el presupuesto del proyecto.

CAPÍTULOII: Microcontrolador.

Capítulo II: Microcontrolador.

II.1.-Introducción.

A lo largo del siguiente capítulo se detallarán las características más importantes del microcontrolador usado, el ATmega2560 de Atmel.

II.2.-Microcontroladores.

Un microcontrolador es un circuito integrado programable y está constituido por tres unidades básicas: CPU, memoria y unidades de entrada-salida. En la Figura II.1 se muestra un esquema general del mismo.

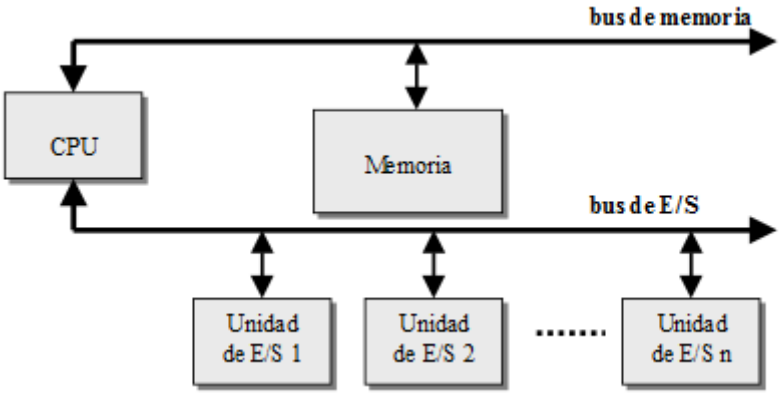


Figura II.1. Esquema general de un microcontrolador.

Los microcontroladores son dispositivos que presentan múltiples ventajas, entre ellas, que son baratos, versátiles y presentan un tamaño reducido.
En el proyecto realizado se ha empleado el microcontrolador ATmega2560 de Atmel.

II.3.-ATmega2560.

El ATmega2560 (Figura II.2) [2] es un microcontrolador de alto rendimiento basado en la arquitectura AVR (Arquitectura Harvard 8 bits RISC modificada) (Figura II.3) y de propósito general. Es de bajo consumo y pertenece a la subfamilia "megaAVR". A continuación se detallarán las características más relevantes del mismo.

Figura II.2. ATmega2560.

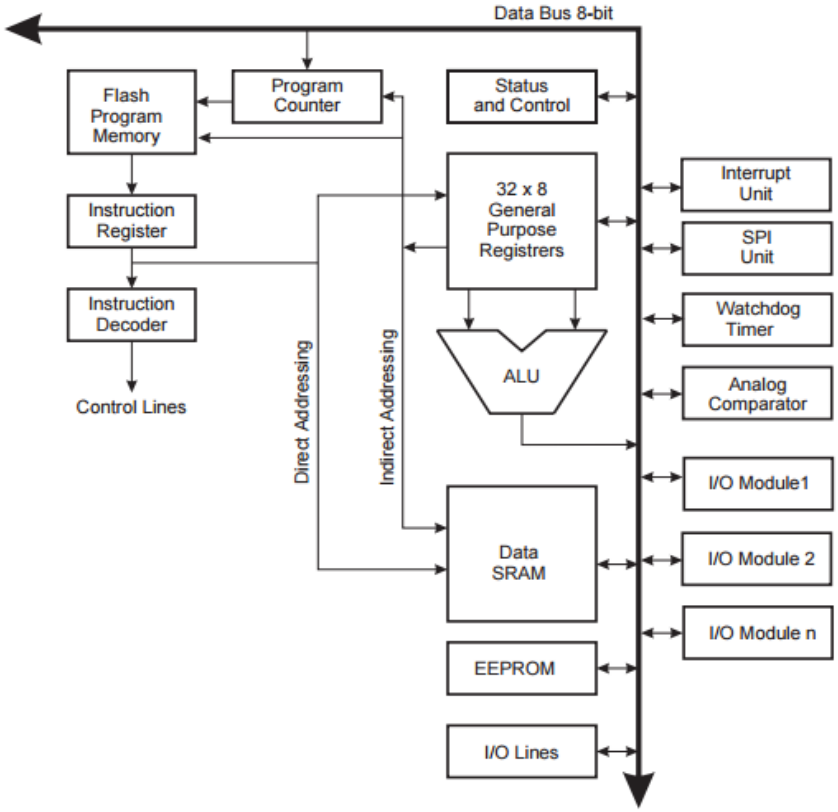


Figura II.3. Diagrama de Bloques de la arquitectura AVR.

II.3.1.-Patillaje.

El patillaje del microcontrolador informa sobre las conexiones que se pueden realizar con otros dispositivos, así como la función que desempeña cada pin dentro del microcontrolador. En la Figura II.4 se puede apreciar su patillaje y su correspondencia con los pines de la placa Arduino.

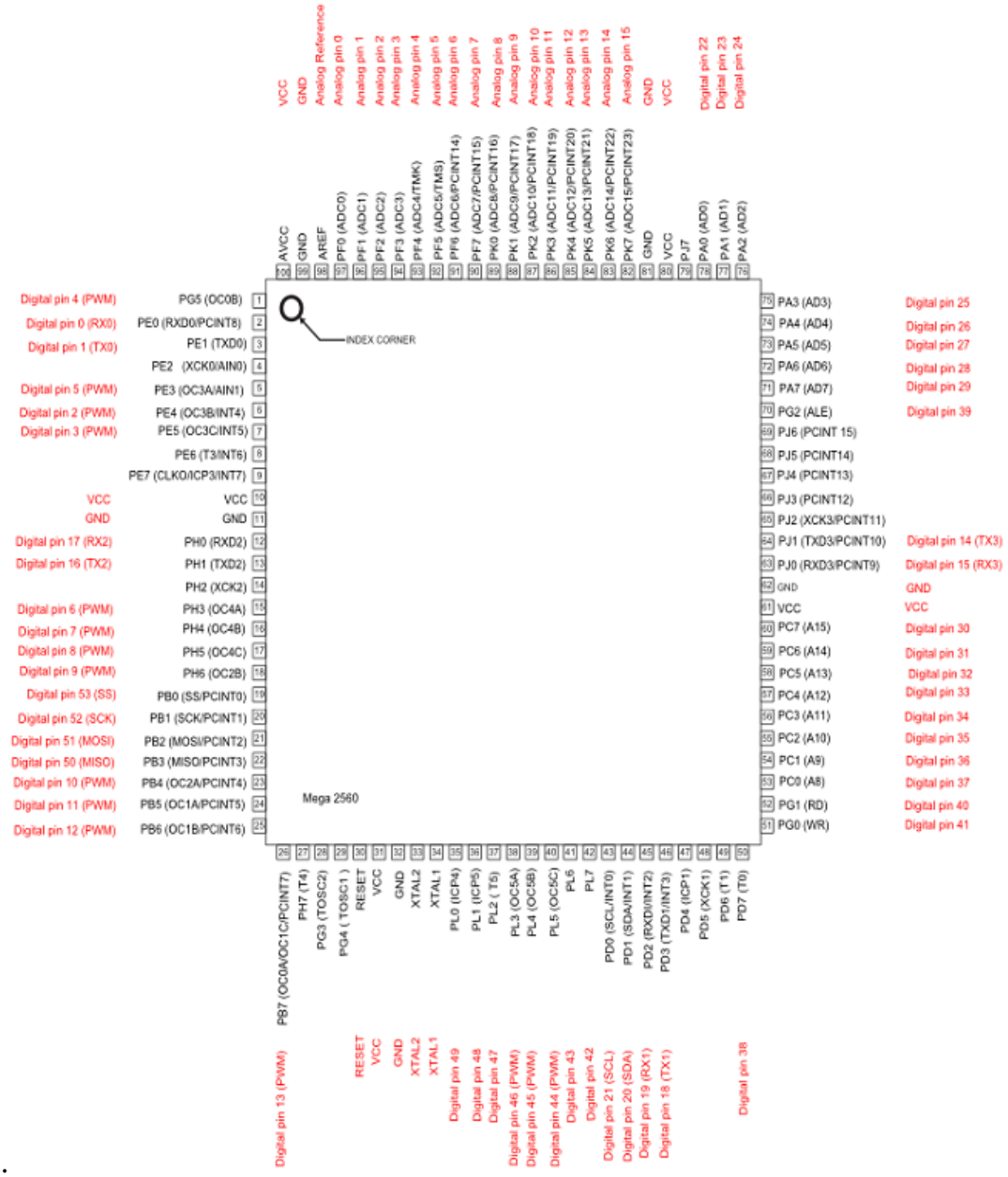


Figura II.4 Patillaje del microcontrolador.

II.3.2.-Memorias.

El ATmega2560 [3], dispone de tres memorias: Flash, SRAM y EEPROM.
La Flash, es una memoria de escritura y lectura que se programa eléctricamente. Es la encargada de almacenar el programa que ejecuta el microcontrolador y se puede reescribir cuando se desee. La capacidad de almacenamiento en el ATmega2560 es de 256KB.
La SRAM, es similar a Flash, con la principal diferencia de que cuando se le deja de alimentar se pierden todos los datos almacenados. Su misión principal es la de ir almacenando los datos en cada instante del programa. El ATmega2560 tiene una SRAM de 4KB.
La EEPROM, al igual que la SRAM y Flash es una memoria de escritura y lectura que se programa eléctricamente pero que a diferencia de las anteriores, permite almacenar datos cuando el microcontrolador no tiene alimentación. El ATmega2560 dispone de una EEPROM de 4KB.

II.3.3.-Registros.

Los registros son memorias de alta velocidad y poca capacidad, integradas dentro del microprocesador, estos permiten guardar momentáneamente datos que serán usados por el microcontrolador, generalmente para realizar operaciones aritméticas, así como guardar instrucciones en ejecución o recientemente ejecutadas. El microcontrolador ATmega2560 dispone de registros de 8 bits [4]

II.3.4.-Reloj.

El ATmega2560 dispone de una serie señales de reloj, a través de las cuales se controlan las velocidades de trasmisión y recepción de datos, ejecución del programa, etc. En la Figura II.5 se muestra un esquema de los diferentes relojes del microcontrolador.

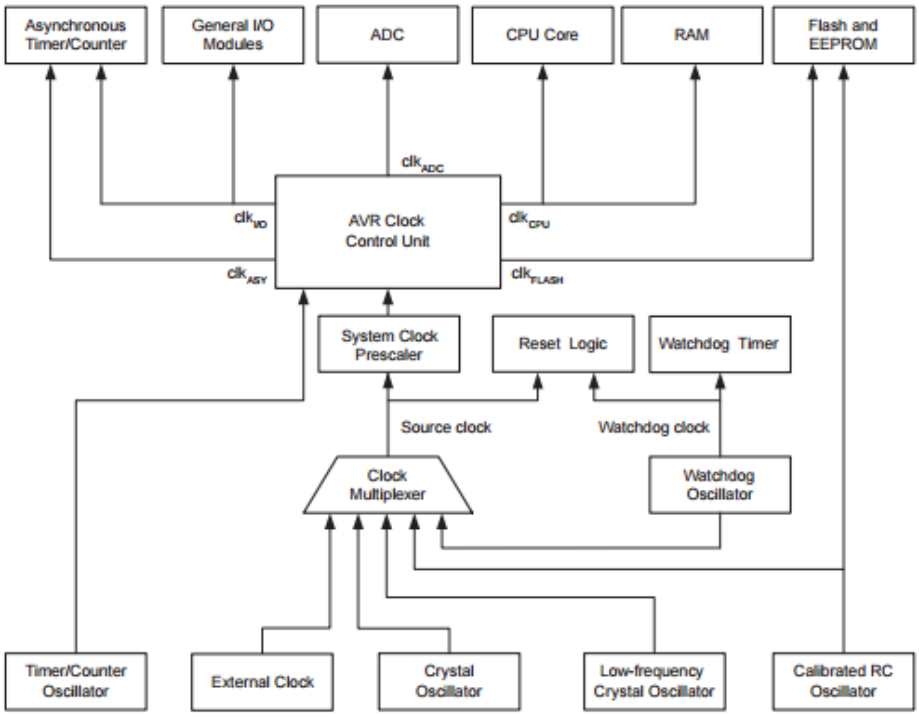


Figura II.5. Diagrama de bloques de los diferentes relojes del ATmega2560.

Del diagrama de bloques anterior podemos extraer la siguiente información:
CPU Clock – clkCPU (Reloj de la CPU): Es el reloj de la CPU del microcontrolador y marca el ritmo a la que se ejecutan las operaciones de la misma.

- I/O Clock – clkI/O (Reloj de entrada y salida): El reloj es usado por la mayoría de los módulos de entrada y salida, como temporizadores, contadores, SPI, etc.

- Flash Clock – clkFlash (Reloj Flash): Controla las operaciones de la interfaz Flash.
- Asynchronous Timer Clock – clkASY (Reloj asíncrono): Permite a los temporizadores y contadores asíncronos ser controlados por un reloj externo.

- ADC Clock – clkADC (Reloj del conversor analógico digital): Está dedicado únicamente al conversor analógico digital.

Además El ATmega2560 dispone de un oscilador de cuarzo de 16 MHz.

II.3.5.-Interrupciones externas.

El ATmega2560 dispone del uso de interrupciones. Gracias a ellas el microcontrolador es capaz de parar la ejecución normal del programa y saltar a una parte determinada del mismo para atenderla. En particular, las interrupciones externas ejecutan un fragmento de programa al producirse un determinado evento. Tal y como se indica en el datasheet del fabricante, los pines disponibles para el uso de interrupciones externas son el 2, 3, 18,19 y 21.

II.3.6.-Protocolos de comunicación.

Los buses integrados en el ATmega2560 permiten trasmitir información de manera serial (comunicación en serie) o paralela. La primera trasmite bit a bit, por lo que es más lenta que la trasmisión en paralelo. Sin embargo, la comunicación es más simple. Respecto de los protocolos seriales encontramos principalmente dos:

I^2C: Es un bus de datos serial, donde se establece una comunicación maestro-esclavo. Así el maestro inicia la comunicación y el maestro responde. La comunicación se realiza a través de dos líneas de datos, SCL y SDA donde la primera representa el reloj y la segunda incluye los datos. Durante la trasmisión se envía un byte con toda la información relacionada con la comunicación tal y como se indica en la Figura II.6 [5]

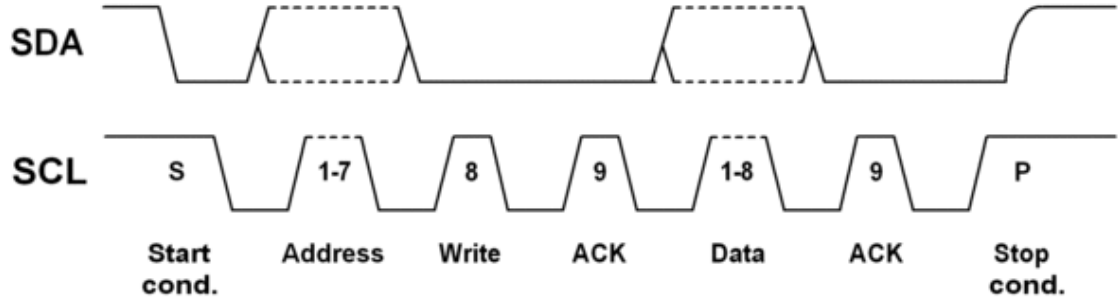


Figura II.6. Trasmisión de datos durante una comunicación vía I^2C.

SPI: Es un protocolo de comunicación serie que permite la comunicación de dispositivos electrónicos que dispongan de una señal de reloj, mediante un protocolo maestro-esclavo y permitiendo la comunicación simultánea y en ambas direcciones (full dúplex). Así, el protocolo está basado en el uso de 4 señales fundamentales: CLK (permite la sincronización entre los dispositivos), MOSI (Master Output Slave Input), MISO (Master Input Slave Output) y SS para indicar que el maestro desea comunicarse con el esclavo. El funcionamiento del protocolo se puede observar en la Figura II.7. Básicamente, cuando el maestro se desea comunicarse con el esclavo, se activa la señal SS, posteriormente se envía la trama de bits de manera síncrona, es decir, en cada flanco de subida de la señal CLK se envía un bit. [6].

Figura II.7. Funcionamiento del protocolo SPI.

CAPITULO: III Herramientas de diseño y desarrollo.

Capítulo III: Herramientas de diseño y desarrollo.

III.1.-Introducción.

En el dispositivo, para el esbozo de las diferentes partes del hardware y del software, se han utilizado diferentes herramientas de diseño y desarrollo. Las partes diseñadas con estas son: programación de la placa de Arduino Mega, programación del modem GSM SIM 900, diseño de la placa de circuito impreso, diseño de la carcasa del dispositivo y programación de la App para el control de la alarma a distancia.

En el módulo referente a la implementación del software podemos distinguir dos partes. La primera para la programación de las placas Arduino Mega y del modem GSM SIM 900 y la segunda para la programación de la App que se utilizará en el control a distancia de la alarma.
En la parte referida a la programación de la placa Arduino Mega hemos usado el IDE de Arduino, donde se ha hecho la programación la placa del Arduino Mega a través de funciones y estructuras de bucles así como comandos AT para la programación del modem GSM SIM 900.

Para el parte de programación de la App se ha utilizado el entorno de programación libre proporcionado por App Inventor 2, la estructura de software ha sido diseñada de forma que se pueda llevar a cabo el control de la alarma a distancia y desde el dispositivo móvil donde se descargue la misma.
En el módulo de implementación del hardware podemos distinguir dos partes la primera referida al diseño del circuito de la placa impresa a través del entorno de programación de Eagle y la segunda parte es aquella en la que interviene el diseño de la carcasa de la alarma en cuestión que se ha llevado a cabo a través del entorno del SolidWorks.
A continuación se explicara detalladamente cada uno de las herramientas de diseño utilizadas en la elaboración de cada una de las partes.

III.2.-IDE Arduino.

Arduino es la compañía conocida en Estados Unidos, Genuino es el nombre de la compañía a nivel internacional. Se trata de una comunidad tecnológica que manufactura

placas computadoras para el diseño del hardware y software, cuyos componentes son circuitos impresos que integran microcontroladores en un entorno IDE donde se lleva a cabo la programación de la placa. Dicho entorno puede apreciarse en la Figura III.1 [7].

El software del IDE de Arduino facilita de forma libre acceder a un entorno de programación pensado para programar microprocesadores en proyectos multidisciplinares. En nuestro caso, se ha programado el microprocesador ATmega 2560 que se encuentra alojado en la palca del Arduino Mega así como el módulo GSM SIM 900 que está inserto sobre la placa de Arduino Mega.

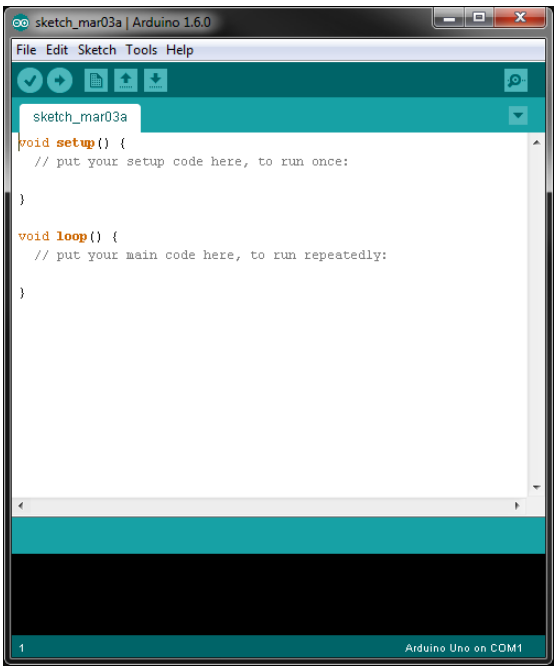


Figura III.1. Entorno IDE Arduino. Fuente : Arduino Web.

III.3.-Transmisión de datos. Comandos AT.

La elaboración del código para programar y configurar el módulo GSM (Global System For Mobile Comunication) es estándar y recibe los protocolos de comunicación de la segunda generación de telefonía móvil, por ello hemos utilizado el Shield específico para Arduino SIM 900; se puede visualizar el dispositivo en la Figura III.2. La programación se ha basado en el uso de los comandos AT que serán necesarios para recoger la información proporcionada por el Arduino y la enviará a la App del móvil.

Figura III.2. Módulo GSM/GPRS SIM900 TinySine . Fuente : DataSheet del fabricante.

A continuación se citaran los comandos AT utilizados en la programación y configuración del modem GSM:

AT
Este comando se ha utilizado para verificar que el modulo se conecta correctamente con la placa de Arduino Mega. Al enviar el comando AT el módulo contestara con OK, indicando que está listo para establecer la comunicación.

AT+IPR=XXXX
Con este comando se configura la frecuencia deseada para la transmisión de datos.

AT+CSCS= "XXX"
El comando configura el modem en modo texto.

AT+CMGF
Este comando configura el formato de los mensajes , ya sea PDU(0) o SMS(1).

AT+CMGS=XXXXXXXX
Envía un SMS , desplegándose el símbolo > (mayor que) Escribir el mensaje y al finalizar la escritura del mismo presionamos Ctrl+Z devolverá OK si el SMS se envió correctamente.

ATDXXXXXXXX;
En este caso el comando ha sido utilizado para enviar una llamada al teléfono móvil

deseado.

ATH;
Se ha utilizado para colgar la llamada realizada con el comando ATD.

AT+CMGR;
Este comando permite la lectura de los mensajes SMS enviados desde el teléfono móvil.

AT+CMGD;
El comando permite borrar mensajes de la memoria, consiguiendo así, borrar SMS de la tarjeta SIM y liberarla para recibir futuros mensajes.

III.4.-App Inventor 2.

Es un entorno de desarrollo de software diseñado y desarrollado por Google Labs para la elaboración de aplicaciones destinadas a sistemas operativos Android, tal y como se muestra en la Figura III.3. A partir de un conjunto de herramientas básicas y de forma visual se puede apreciar los enlaces entre bloques de programación creados para el desarrollo de la App pertinente. Las aplicaciones creadas con App Inventor 2 están limitadas en su desarrollo ya que sirven por simplicidad para cubrir necesidades elementales con nuestros dispositivos móviles [8].

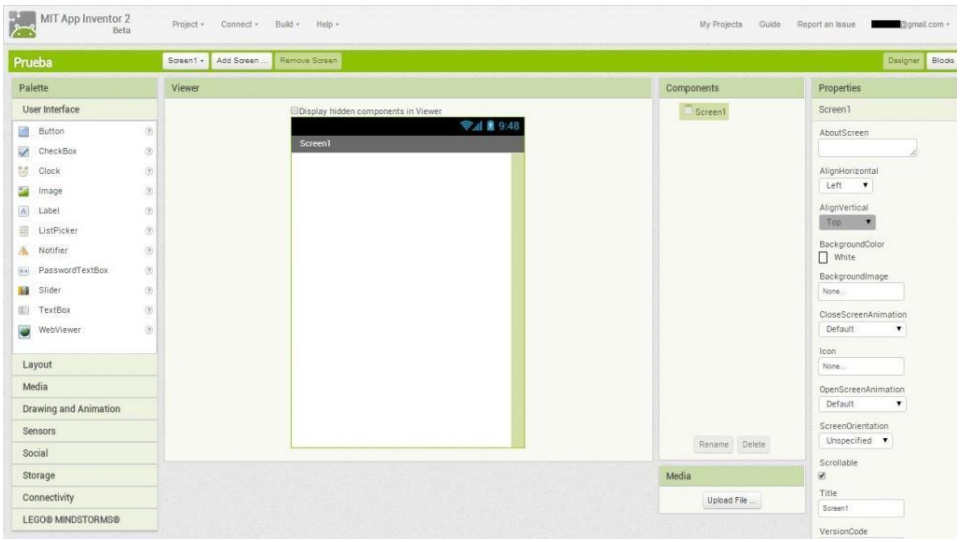


Figura III. 3. Entorno de App Inventor 2. Fuente: web, App- Inventor2.

III.5.- Entorno de desarrollo Eagle.

El diseño de la PCB o placa de circuito impreso ha sido ha sido elaborado con el software Eagle el cual permite la edición de PCB Layouts y Squematics(Figura III.4). Las siglas del software en cuestión son las correspondientes a los nombres (Easily Applicable Graphical Layout Editor), dispone de licencia Freeware compatibles con Window, MAC y Linux. Posee una extensa variedad de librerías y componentes en la red, nos permite editar diagramas electrónicos con un enrutador bastante eficiente, en el editor podemos encontrar la opción de convertir los archivos manejados a formato GERBER [9].

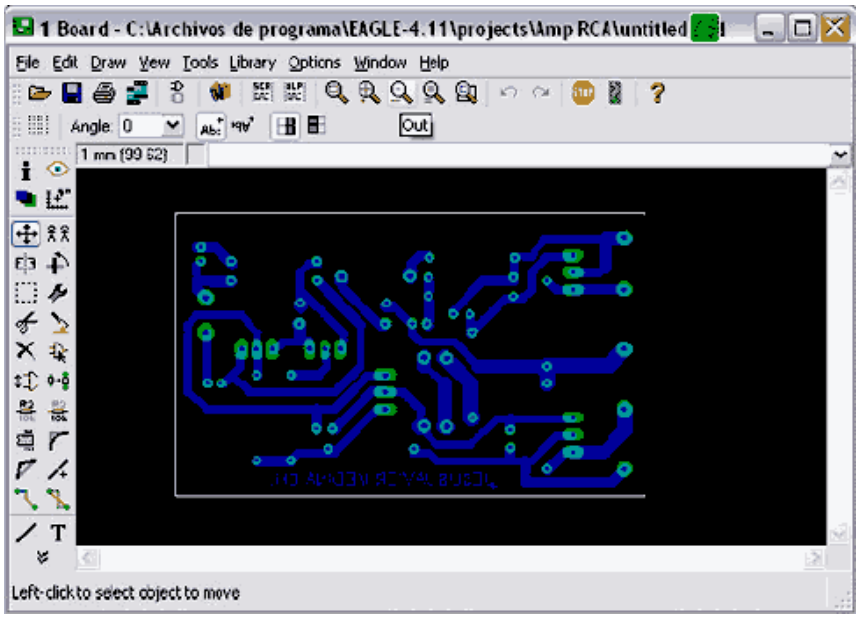


Figura III.4. Entorno del Eagle.

III.6.- SolidWorks.

SolidWorks es un software CAD, cuyo impulsor es Jon Hirschtick que en 1993, contrato a un grupo de ingenieros para el desarrollo de una App explicita para modelado en 3D que fuera más accesible. En 1997 fue adquirida por Dassault Systemes S.A. y esta sería la encargada del desarrollo de sus posteriores versiones. Permite el modelado mecánico en 3D , crear piezas así como conjuntos de piezas para extraer planos de los mismos haciéndolo de forma automatizada. La idea básica del SolidWorks es traspasar un modelo mental a un modelo virtual sólido el cual poder manipular, en la Figura III. 5 podemos apreciar el entorno de trabajo [10].

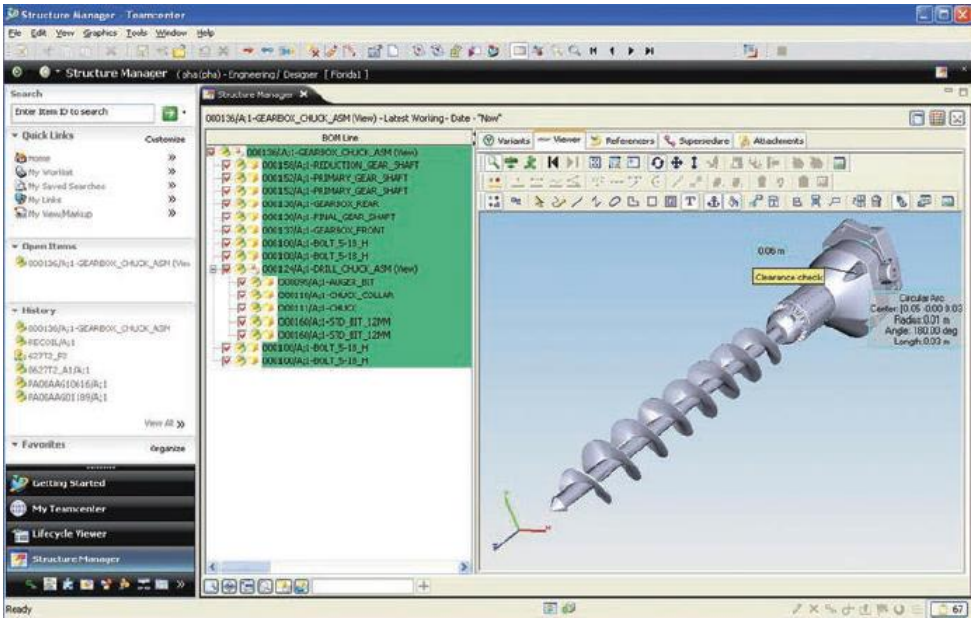


Figura III. 5. Entorno de trabajo del SolidWorks. Fuente web ,avantek

Capitulo IV: Descripción del sistema

Capitulo IV: Descripción del sistema.

IV.1.- Introducción.

En el presente capitulo se abordará con mayor profundidad cada una de las partes en las que se conforma el proyecto realizado, dividiéndolas en unidades del dispositivos, donde haremos mención de los componentes utilizados dentro de cada una de las mismas. Las unidades en las que se han clasificado han sido de entrada, salida y entrada – salida. Finalmente el capítulo dispone de una guía de uso para el dispositivo de alarma y de la APP desarrollada para el móvil.

IV.2.- Unidades de entrada.

Las unidades de entrada son todas aquellas que reciben información del medio exterior, siendo las encargadas de proporcionar datos al microcontrolador del Arduino Mega. En los siguientes apartados se muestran las diferentes unidades de entrada que forman parte del diseño del dispositivo de alarma, y que son son: teclado matricial, sensores de presencia y reloj en tiempo real.

IV.2.1.- Teclado.

El teclado que se ha utilizado es del tipo matricial pasivo, de dimensiones 3 x 4, donde se dispone de 3 columnas y 4 filas. Los teclados matriciales son interfaces de entrada cuyos datos se introducen de forma sencilla y manual. La introducción de datos se hace a través de las teclas situadas sobre los cruces de las filas y columnas conductoras. Los cruces se hace uso de un pulsador de tipo membrana o mecánico, que al ser pulsado establece el contacto eléctrico entre la fila y la columna correspondiente. El contacto que se establece entre ambas al pulsar una tecla genera un ‘0’ lógico y si no se ha pulsado, un ‘1’ lógico. Los valores se obtienen gracias a la puesta a masa (‘0’ voltios) de la resistencia Pull-up que corresponde a cada cruce de filas y columnas. En la Figura IV.1 se puede visualizar el circuito correspondiente a cada una de las partes mencionadas.

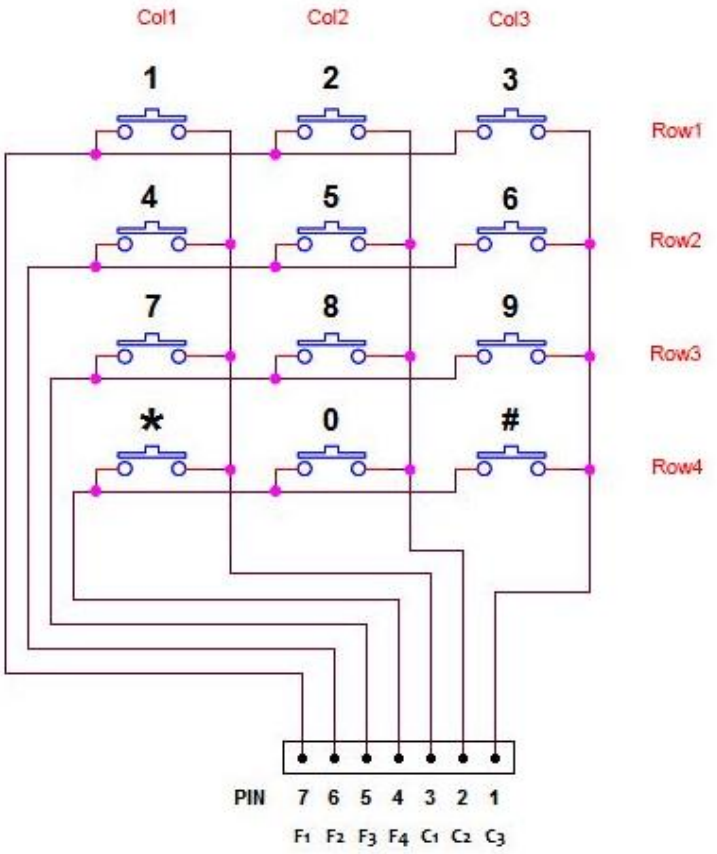


Figura IV. 1 Teclado matricial pasivo 3 x 4.

IV.2.2.- Sensores de movimiento / presencia.

Los sensores de presencia utilizados en el dispositivo alarma son del tipo PIR. Los PIR son sensores infrarrojos pasivos, cuyo principio de funcionamiento se basa en la medida de la radiación infrarroja existente entre un cuerpo y el sensor piezoeléctrico, que será capaz de transformar la radiación recibida en una señal eléctrica. Podemos distinguir, dentro de su funcionamiento, dos partes. La primera, es la que divide la visión del sensor en dos campos, realizando una medida por separado de la radiación, mientras, el circuito eléctrico que recibe las medidas proporcionadas por ellos, está destinado a compensar ambos resultados, de manera que, si los dos campos miden lo mismo en un determinado momento la resultante será nula, mientras que si es distinta, se genera una señal eléctrica diferencial cuya resultante es un '1' lógico. En la Figura IV.2 podemos apreciar la idea descrita anteriormente.

La segunda de las partes es la óptica del sensor que se compone de una cúpula de plástico recubierta de cristales de Fresnel cuya funcionalidad se basa en dividir el espacio en varias zonas y enfocar cada uno de los campos del sensor PIR. Se puede apreciar con más detalle el funcionamiento de la parte óptica del sensor en la Figura

IV.3, mientras la IV.4 muestra el patillaje del sensor utilizado.

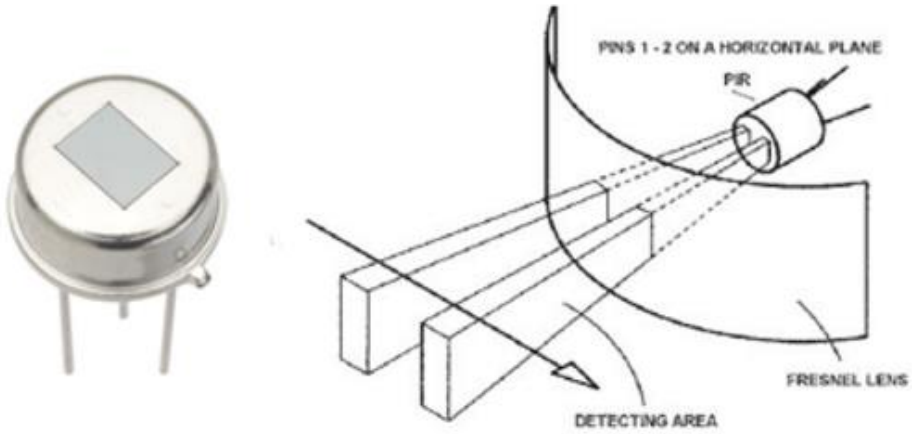


Figura IV. 2. Campos infrarrojos del sensor PIR.

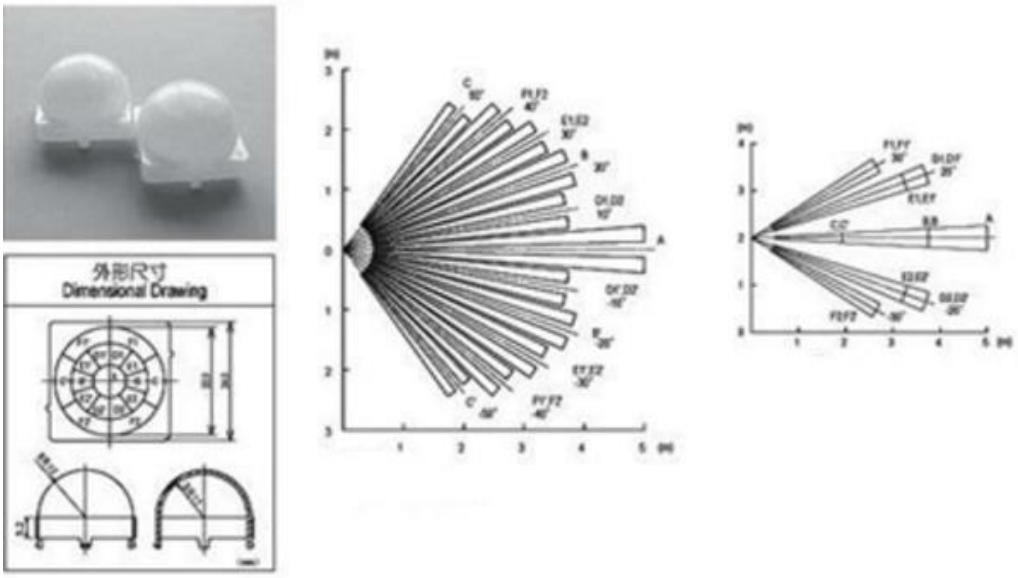


Figura IV. 3. Campo óptico generado por los cristales de Fresnel.

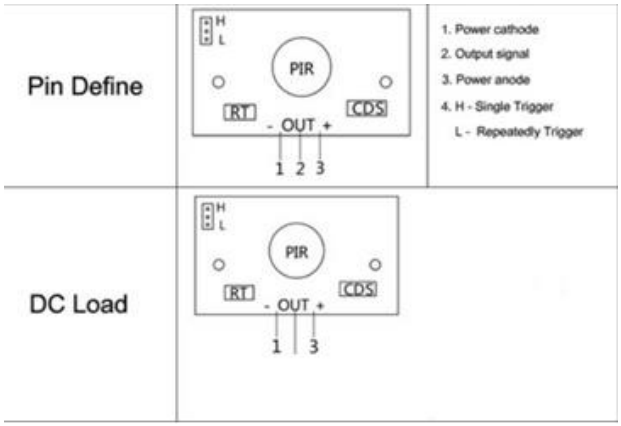


Figura IV. 4. Pines del sensor PIR.

IV.2.3.- Reloj.

Como reloj se utilizó el DS1307 de Dallas Semiconductor, que opera también como calendario en tiempo real válido hasta el año 2100, y que se comunica con el microcontrolador mediante protocolo I2C, Figura IV.5. Además, puede funcionar independientemente del sistema al disponer de su propia batería. La Figura IV.6 muestra sus terminales de conexión.

Figura IV.5. Módulo DS 1307

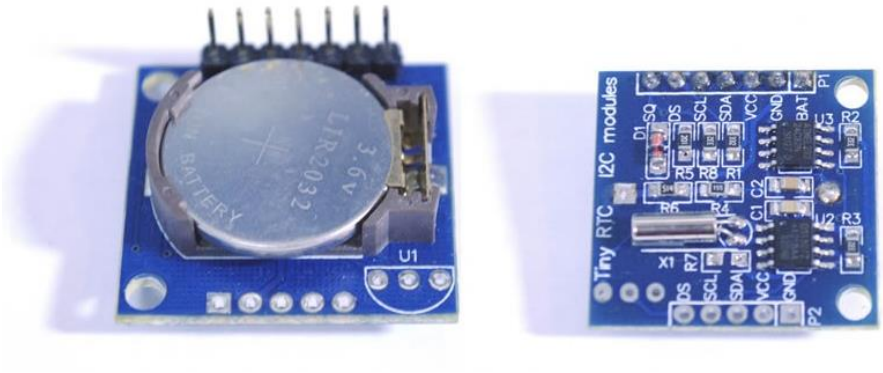


Figura IV. 6. Pines del reloj RTD

IV.3.- Unidades de salida.

Las unidades de salida del sistema son aquellas destinadas a mostrar los datos procesados por el microcontrolador. En el presente Capitulo se abordará como unidades de salida el display 20 x 4 y el conjunto relé/bocina.

IV.3.1.- Display 20 x 4.

El visualizador LCD (Liquid Cristal Display) de dimensiones 20 x 4 está destinado a mostrar caracteres alfanuméricos, permitiendo así la representación de la información de cualquier equipo electrónico de una forma sencilla y económica.

La pantalla del display se compone de una matriz de caracteres cuya distribución se dispone en una, dos, tres o cuatro líneas que se componen de entre 16 y 40 caracteres por línea dependiendo del modelo. El proceso de visualización de los datos es gobernado por un microcontrolador incorporado en la pantalla.

En relación a las características del dispositivo ponemos destacar las siguientes:

- Consumo del orden de 7,5mW.
- Caracteres de la pantalla ASCII, además de incorporarlos en japonés.
- Rotación de caracteres a lo largo de la pantalla, tanto hacia la izquierda como hacia la derecha.
- Memoria de 40 caracteres por línea de pantalla, pudiendo visualizarse 16 caracteres por línea.
- Está gobernado por un bus de 8 bits.

El módulo dispone de una zona de memoria RAM, llamada DDRAM (Data display RAM), donde almacena los caracteres que mostrará por pantalla.

Posee una capacidad de 80 bytes, 40 por cada línea, de los cuales 32 se pueden visualizan a la vez.

El módulo se conecta a través de los pines disponibles en la Figura IV.7 y los pines que se corresponden los el esquema de conexiones en la Figura IV.8.

PIN Nº	SÍMBOLO	DESCRIPCIÓN
1	V_{SS}	Patilla de tierra de alimentación
2	V_{DD}	Patilla de alimentación de 5 V
3	V_O	Patilla de contraste del cristal líquido.Normalmente se conecta a un potenciómetro a través del cual se aplica una tensión variable entre 0 y +5V que permite regualr el contraste del cristal líquido.
4	RS	Selección del registro de control/registro de datos: RS=0 Selección del registro de control RS=1 Selección del registro de datos
5	R/W	Señal de lectura/escritura R/W=0 El módulo LCD es escrito R/W=1 El módulo LCD es leido
6	E	Señal de activación del módulo LCD: E=0 Módulo desconectado E=1 Módulo conectado
7-14	D0-D7	Bus de datos bi-direccional. A través de estas líneas se realiza la transferencia de información entre el módulo LCD y el sistema informático que lo gestiona

Figura IV.7. Patillaje del display LCD 20 x 4.

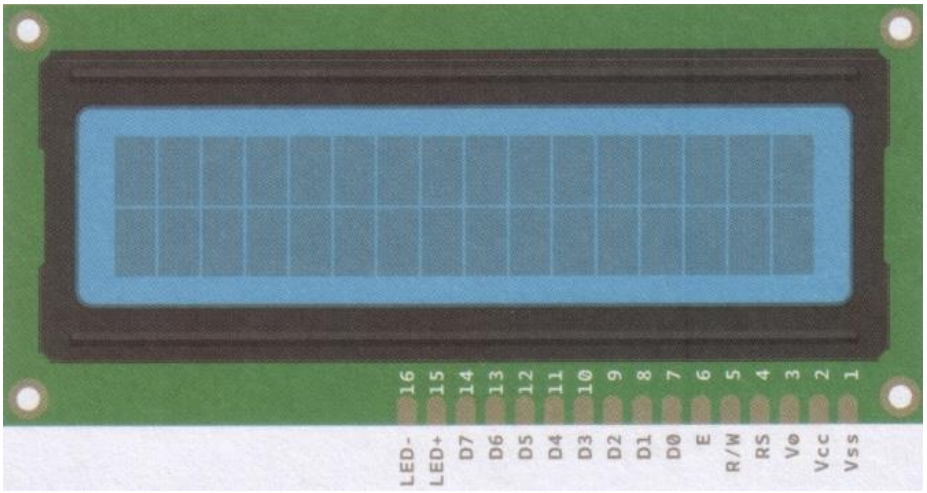


Figura IV.8. Esquema de conexiones de pines del display LCD 20 x 4.

El control del módulo de la pantalla LCD se ha llevado a cabo a través de un bus I2C el

cual se ha conectado a los pines mostrados en la Figura IV.7 mediante una interfaz paralelo/I2C (Figura IV.9). El bus I2C, llamado así por el acrónimo del inglés “inter-integrated circuit”,permite controlar la pantalla del display usando sólo dos líneas de señal y un cable común de tierra. Este módulo posibilita el intercambio de información a una velocidad de hasta 100 Kbits por segundo. La comunicación de datos del Bus I2C es serie y sincrónica. Una señal del bus se dedica a marcar el tiempo a través de los pulsos de reloj y la otra se usa para intercambiar datos. Para la descripción de las señales en detalle se ha usado la Tabla IV.1.

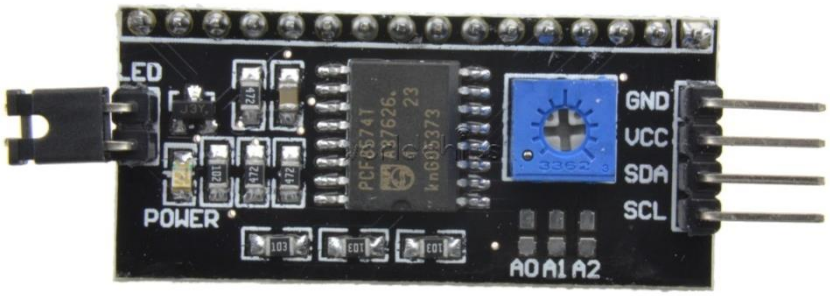


Figura IV.9. Conexión paralelo / I2C.

Sigla de la señal	Nombre de la señal	Descripción
SCL	System clock	Es la línea de pulsos de reloj que sincroniza el sistema
SDA	System Data	Es la línea por la que se transfieren los datos entre los dispositivos
GND	Masa	Es la señal común entre todos los dispositivos “enganchados” al bus

Tabla IV.1. Descripción de las señales del bus I2C.

Las líneas de señal referidas a las siglas SDA y SCL actúan de un modo similar a los colectores abiertos y están asociadas a un transistor de efecto campo. La polarización de

estas señales se hace en estado alto a través de la conexión de la alimentación a unas resistencias Pull-up lo que supone la conexión de múltiples entradas y salidas. A pesar de disponer de conexión para múltiples dispositivos, ha de tenerse en cuenta que la capacidad máxima de los dispositivos conectados no debe superar los 400pF. La Figura IV.10 muestra un esquema explicativo de las conexiones mediante bus I2C.

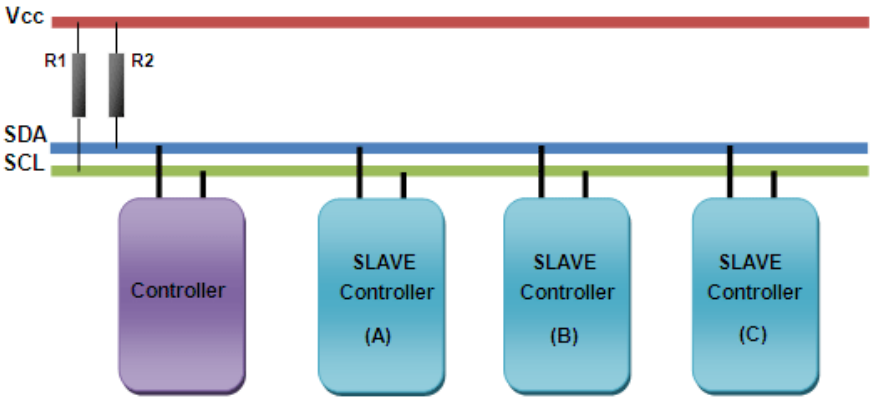


Figura IV.10. Esquema de conexiones disponibles en el bus I2C.

IV.3.2.- Relé – Bocina.

Una parte importante del sistema implementado lo constituye el conjunto relé-bocina, encargado de ahuyentar a los posibles intrusos que accedan a la propiedad en la que se encuentra la alarma.

Para accionar la bocina se ha recurrido a la utilización de un relé electromagnético controlado por el Arduino Mega a través de un transistor que opera en corte o en saturación (Figura IV.11).

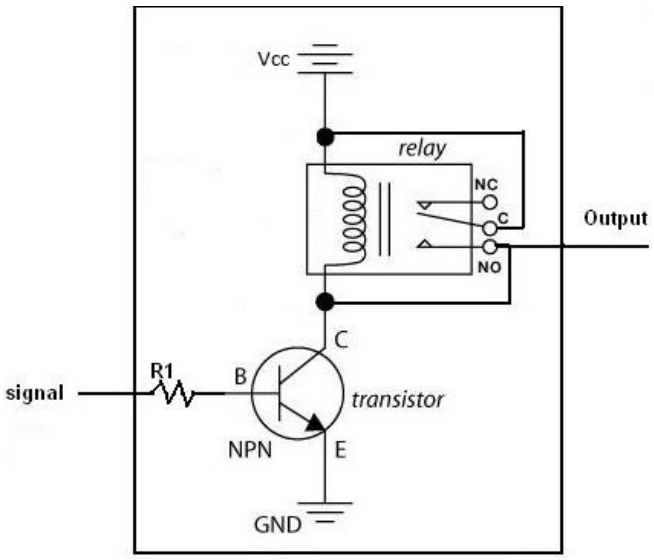


Figura IV.11 Relé contralado mediante transistor.

La base del transistor es llevada al terminal 53 del Arduino Mega, de tal manera que cuando en el mismo haya un "0" lógico (0 V) el transistor estaría en corte y su salida (o terminal "común") estaría conectada al terminal "Normalmente cerrado" (NC). Por el contrario, la presencia de un "1" lógico (5 V) en la misma base haría que el transistor pasara a saturación y, con ello, que su salida se conectara con el terminal "Normalmente abierto" (NA o NO).

En la Figura IV.12 podemos observar el relé empleado con su respectivo transistor SMD. En su parte derecha son visibles tres terminales que corresponden, respectivamente, a la alimentación del conjunto relé/transistor (Vcc, +5 V), tierra (GND) y la conexión a la base del transistor (IN). En su parte izquierda se encuentran tres clemas que corresponden a los terminales "Común", "Normalmente abierto" y "Normalmente cerrado".

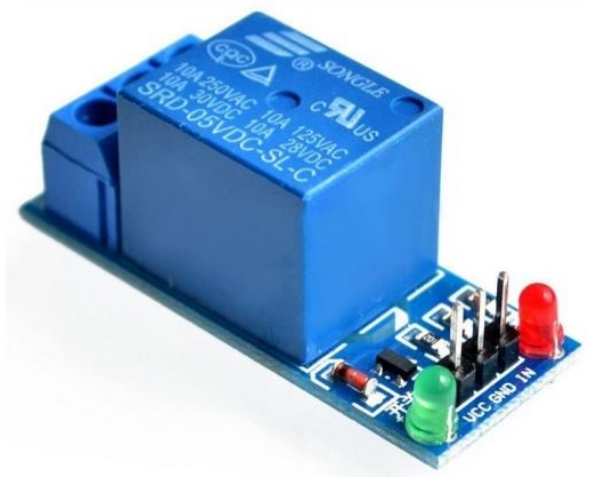


Figura IV.12 Sistema relé-transistor empleado.

Las características del relé son:

- Control mediante señal TTL.
- Alimentación: 5V, DC.
- Corriente máxima de carga 10A.

La Figura IV.13 muestra la bocina empleada, siendo sus características:

- Alimentación: 5-12V DC.
- Corriente: 300 mA.
- Salida: 110 dB.
- Frecuencia: 3.8 KHz.

Se eligió dicha bocina porque puede operar con tensiones comprendidas entre 5 y 12 V en DC. Ello posibilita que se pueda utilizar la propia fuente de alimentación de la alarma, capaz de proporcionar una corriente máxima de 6 A (a 5 V), u otra externa de diferente tensión. En ambos casos, la bocina se colocará en serie con la fuente de tensión elegida y uno de los terminales del conjunto resultante se llevará el terminal "Común" del relé mientras el otro lo hará al "Normalmente abierto".

Figura IV.13 Bocina utilizada

IV.4.- Unidades de entrada – salida.

En este apartado se mencionará el uso de los dispositivos de entrada / salida dentro del sistema, donde describiremos el uso del Modem GSM así como de del teléfono móvil usado para transmitir y recibir SMS´s.
Por otro lado se describirá el funcionamiento de la App encargada de gestionar la entrada y salida de los mensajes que activaran o desactivaran la alarma.

IV.4.1.- Modem GSM SIM900 Tinysine.

El módulo GSM permite conectarnos a todo tipo de redes móviles desde cualquier punto del planeta, utilizando servicios como SMS, MMS, GPRS y telefonía, además de configurarse mediante comandos AT. La placa incorpora 12 puertos GPIOs, 2 puertos PWM y un conversor AC/DC propios del módulo SIM900.
El Shield GSM/GPRS se basa en el controlador SIM900 diseñado para trabajar con Arduino y sus versiones compatibles.

IV.4.1.1.- Características generales.

En referencia a las características generales, citaremos aquellos aspectos técnicos del módulo entre los que se definirán banda de frecuencias de transmisión, comandos de control para programarlo, diferentes servicios que ofrecen, características energéticas, entre otras. En las Figuras IV.14 y IV.15 podemos apreciar las conexiones que nos ofrece el módulo SIM900, además de una tabla explicativa Tabla IV.3 donde se ha señalado el uso de los pulsadores dispuestos en la placa del módulo.

Las características técnicas del módulo son:

- Transmisión de información en 4 bandas de frecuencia: 850/900/1800/1900 MHz.
- GPRS multi-slot class 10/8.
- Estación móvil GPRS clase B.
- Cumple con GSM phase 2/2+.
- Controlable vía comandos AT estándar: GSM 07.07 & 07.05 y comandos AT mejorados: SIMCOM AT Comands.
- Dispone de servicio Short Massage (SMS) – para el envió de pequeños paquetes de datos a través del teléfono móvil.
- Está provisto de una pila tipo TCP/UDP, que permite enviar datos a un servidor web.
- Incorpora un Real Time Clock que requiere de una pila.
- El puerto serial es configurable para la comunicación con el Shield.
- Incorpora un soporte para la comunicación por software serial situada en los pines 6 y 7.
- Posee un conector externo para audífonos, micrófonos y alimentación.
- El rango de temperaturas que soporta está situado entre los -40 ºC y los +85ºC.

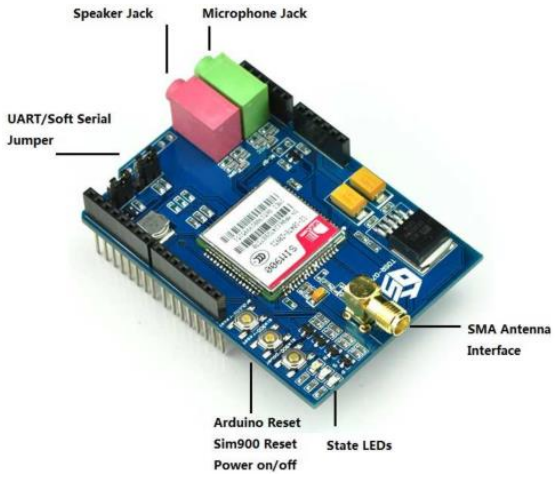


Figura IV.14. Conexiones módulo GSM SIM900.

Botón de la placa	Descripción
Power Button	Botón de activación de la placa del módulo SIM900
Reset Button	Reset del módulo SIM900
Arduino Reset Button	Botón de Reset del Arduino

Tabla IV.3. Botones del Módulo SIM900.

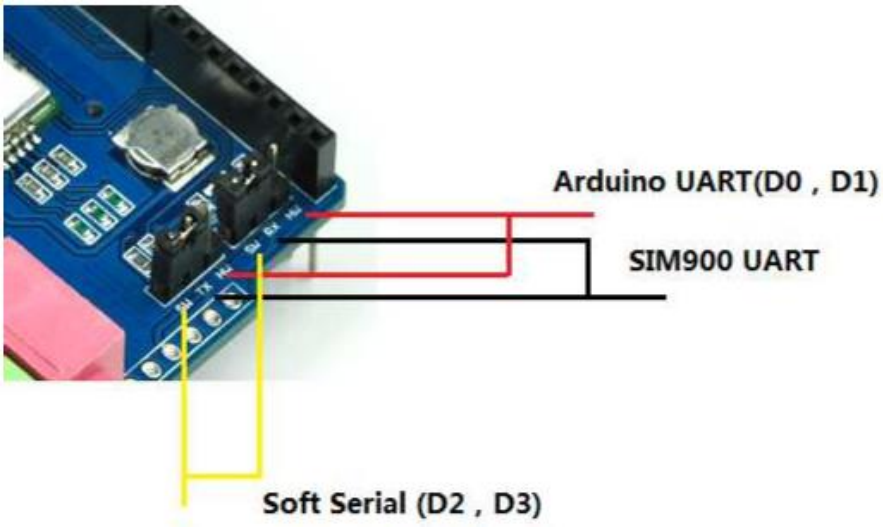


Figura IV.15. Jumpers utilizados para la comunicación mediante comandos AT.

IV.4.2.- Funcionamiento del sistema.

En el funcionamiento del sistema citaremos tanto las instrucciones de uso para la App que controla el dispositivo desde un teléfono móvil como las necesarias para el manejo por teclado de la alarma.

Hay que indicar que:

- Si bien puede ser aumentado, el diseño realizado está pensado para un máximo de cuatro usuarios, cada uno de los cuales dispondría de una clave de activación/desactivación propia.
- Cada vez que el usuario activa o desactiva la alarma a través del teclado, el resto de usuarios recibirán un SMS donde se indicará la fecha, hora y persona que ha llevado a cabo la acción.
- Si, de forma consecutiva, se introducen tres claves erróneas se remitirá un SMS a los 4 usuarios, que denominaremos administradores, indicando dicho incidente.
- Cada clave está compuesta por 4 dígitos que se introducen de manera consecutiva. Cada vez que uno de los dígitos es introducidos, aparece el símbolo * en el display.

  La alarma se disparará inmediatamente (haciendo sonar la bocina) cuando cualquiera de los sensores de presencia instantáneos detecte la presencia de un intruso. Sin embargo, si la detección es llevada a cabo por los sensores de presencia retardados, el usuario dispondrá de 60 s para proceder a la desactivación antes de que comience a sonar la bocina. Ambos tipos de sensores son llevados a las puertas OR de tal manera que se obtendrá un "1" lógico cuando se active cualquiera de los 4 sensores y otro "1" lógico diferentes cuando se activen para el caso de los retardados. Cuando la alarma se dispara el microcontrolador:
- Envía un SMS a los administradores con el texto "Atención, se ha disparado la alarma".
- Realiza una llamada perdida de diez tonos a uno de los administradores para asegurar que reciben la información.
- Se ha reservado el PIN 53 del microcontrolador como salida para activar la bocina de la alarma, que no dejará de sonar hasta que todos los sensores de presencia dejan de estar activos un mínimo de tres minutos.
- En cualquier momento los administradores podrán desactivar el sistema mediante una aplicación móvil diseñada al efecto, recibiendo el mensaje de confirmación de "Alarma desactivada".

IV.4.2.1.- Funcionamiento de la APP. Control mediante telefonía móvil.

A través del modem GSM, los administradores podrán controlar todo el sistema mediante SMS´s transmitidos desde el teléfono móvil, mediante una aplicación diseñada a tal fin, que ambos han de personalizar , Figura IV.16, introduciendo, sólo la primera vez, su nombre y número de teléfono. Así, cuando se desee activar la alarma, se remitirá un mensaje que inducirá la clave terminada en *, o en #, si se pretende su desactivación. El microcontrolador conoce la clave y teléfonos de los cuatro usuarios de tal manera que, al recibir el SMS de uno de los administradores, comprobará tanto el teléfono como su clave antes de activar o desactivar la alarma. Desde el punto de vista operativo, las acciones anteriores se limitan a hacer uso de los botones mostrados en las Figuras IV.17 y IV.18 Así, la primera de las figuras muestra que la alarma está "activada" (candado cerrado) y bastará pulsar la opción "Desactivar alarma" para obtener la pantalla de la Figura IV.17, donde ya aparece desactivada (candado abierto). La elección de la opción "Activar alarma" nos regresaría al caso anterior, pudiendo salir de la aplicación pulsando el botón "salir".

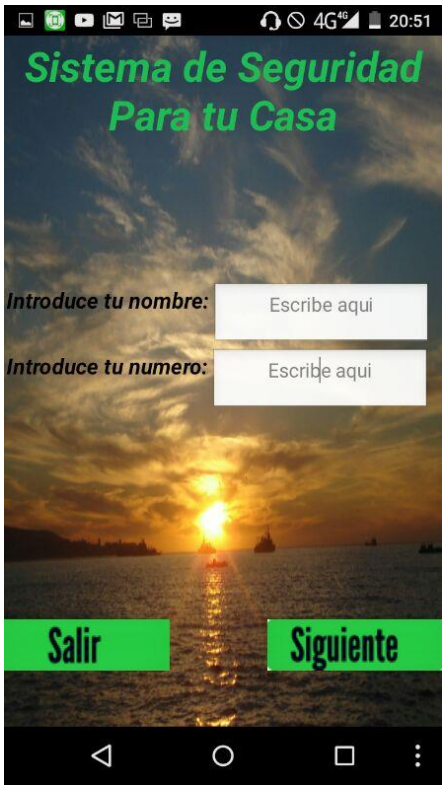


Figura IV.16. Primer menú de pantalla.

Figura IV.17. Segundo menú de pantalla.

Figura IV.18. Tercera menú de pantalla.

IV.4.2.2.- Funcionamiento de la alarma. Control mediante teclado matricial.

En el presente apartado se describirá el funcionamiento de la alarma desde el control por el teclado matricial dispuesto en la misma. Además, se citarán las figuras cuyas imágenes corresponden a los estados en los que se encuentre la alarma. El dispositivo consta de cuatro menús fundamentales en los que se muestran las etapas en las que se encuentra el usuario. Seguidamente se describirá con mayor detalle las diferentes fases de la alarma ilustrándolas con las figuras cuyas imágenes corresponden a cada estado.

El menú principal muestra un mensaje de bienvenida donde se cita en la misma pantalla la fecha, la hora actual y el estado inicial de la alarma (inactiva). Si se quiere activar, se deberá pulsar '*' tal y como nos indica el mensaje situado en la parte inferior de la pantalla del display. La Figura IV.19 nos ofrece una vista de lo mencionado anteriormente.

Figura IV.19. Menú principal de alarma.

En el segundo menú nos encontramos con la segunda fase donde se muestra el mensaje para introducir la contraseña y activar la alarma desde el teclado (Figura IV.20). En la parte inferior de la pantalla se dispone de un espacio donde se puede introducir la contraseña. Ésta contiene cuatro dígitos que permiten activar la alarma en cuestión.

Figura IV.20. Segundo menú de la alarma.

Seguidamente cuando aparezca el menú citado en la Figura IV.20 se podrá introducir la contraseña por teclado apareciendo en la pantalla los caracteres pulsados en el teclado matricial en forma de asteriscos '*'. Como se mencionó antes, estos caracteres son cuatro e indican los lugares que ocupan en el número secreto. En el caso de no acertar en la contraseña introducida aparecerá un mensaje de error (Figura IV.21).

Figura IV.21. Mensaje de error. Contraseña errónea

En caso de acertar en la contraseña introducida podremos visualizar el mensaje correspondiente al estado de alarma activada (Figura IV. 23).

Figura IV. 22. Mensaje de acierto contraseña correcta

Una vez aceptada la contraseña por el sistema; se entrará en el tercer menú (Figura IV.23), donde podremos apreciar del mismo modo los detalles de la fecha/hora así como el estado en el que se encuentra la alarma (“alarma activada”).

Figura IV. 23. Tercer menú de la alarma.

Ya en este punto, para desactivar la alarma bastará con pulsar la opción “#” para que aparezca la pantalla mostrada en la Figura IV.24. Siguiendo un procedimiento similar al realizado durante la activación, se logrará desactivarla (Figura IV.25).

Figura IV.24. Cuarto menú. Desactivación de la alarma

Figura IV.25. Retorno al menú principal

Capitulo V: Software implementado

Capítulo V: Software implementado.

V.1.- Introducción.

El presente capítulo tiene como objetivo fundamental explicar, mediante diagramas de flujo, el software que ha sido desarrollado para el control de las diferentes partes del sistema: rutina principal, envío y recepción de SMS, control de la fecha y hora, activación y desactivación del sistema mediante contraseñas, activación del relé, etc.

V.2.- Rutina principal.

Es una rutina que siempre se está ejecutando, de esta manera, el programa verifica en cada ciclo el estado de todas las variables, actualizando los valores de las mismas cada vez que comienza el ciclo, además, encargándose de ejecutar todas las acciones de manera cíclica. En la siguiente figura se puede apreciar su comportamiento específicamente para el programa desarrollado (Figura V.1).

V.3.- Actualización fecha y hora.

Es una función que se ha implementado con el objetivo de mantener actualizada la fecha y hora en el display. Se encarga, fundamentalmente, de ir comprobando si ha transcurrido un minuto, una hora, un día, un mes o un año para recoger los datos del reloj y mostrarlos en la pantalla. Su comportamiento se puede analizar en la Figura V.2.

V.4.-Activación/desactivación por teclado y envío de mensajes.

El diseño propuesto está pensado para que un máximo de cuatro usuarios, cada uno de los cuales dispondría de una clave de activación/desactivación de tal manera que:

- Cada vez que un usuario activa o desactiva la alarma a través del teclado, se envía un SMS, siguiendo la siguiente regla, si el usuario que activa es un administrador, se enviara un mensaje al otro administrador, si por el contrario el usuario que activa o desactiva no es un administrador, se enviaran un mensaje a cada uno de los administradores.
- Si, de forma consecutiva, se introducen tres claves erróneas se realizara una llamada a alguno de los dos administradores del sistema.
- Cada clave está compuesta por cuatro dígitos que se introducen de manera consecutiva, apareciendo * en el display con cada digito introducido.

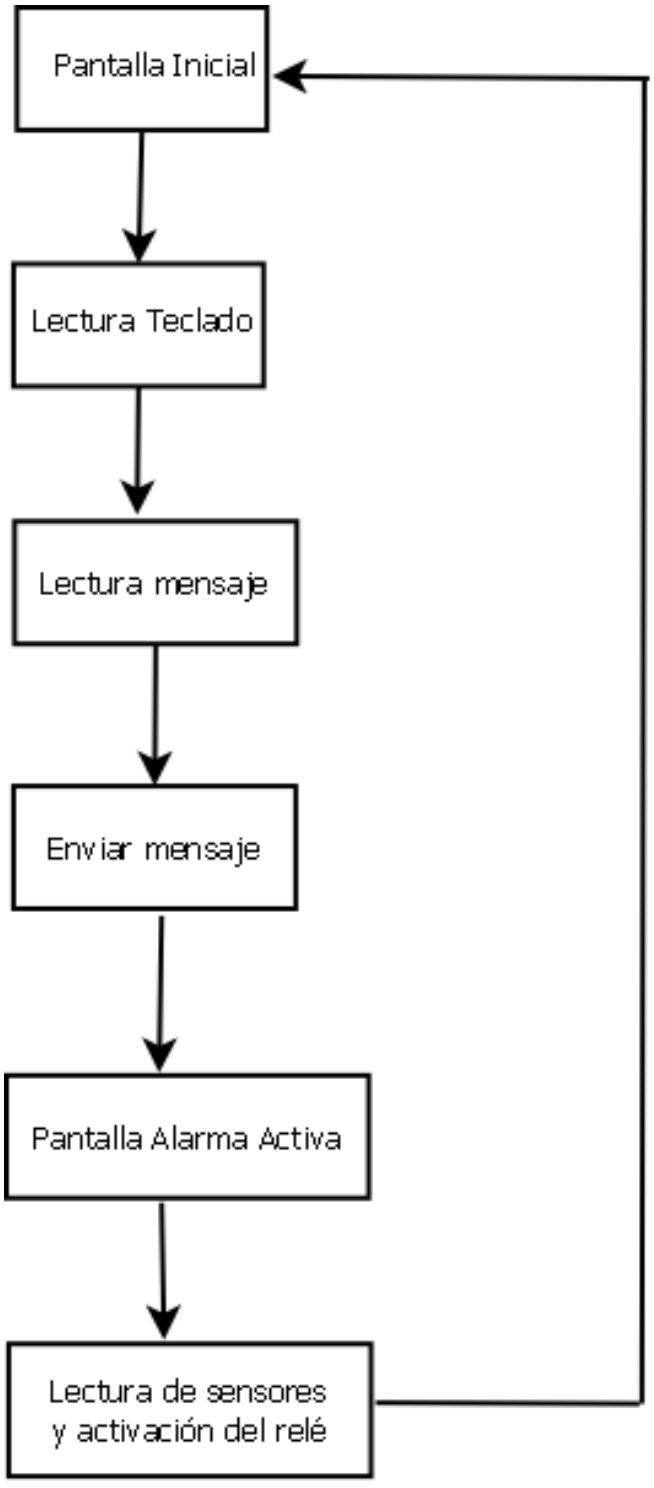


Figura V.1. Diagrama de flujo de la rutina principal.

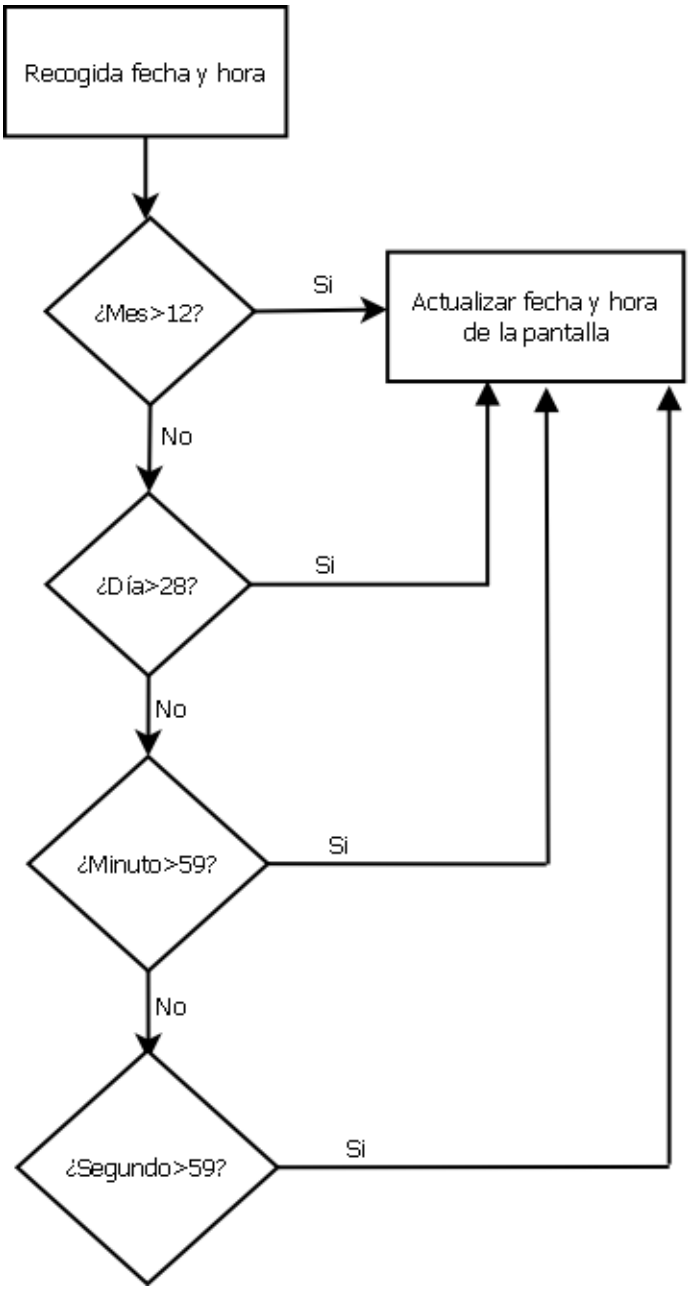


Figura V.2 Diagrama de flujo para la actualización de la hora.

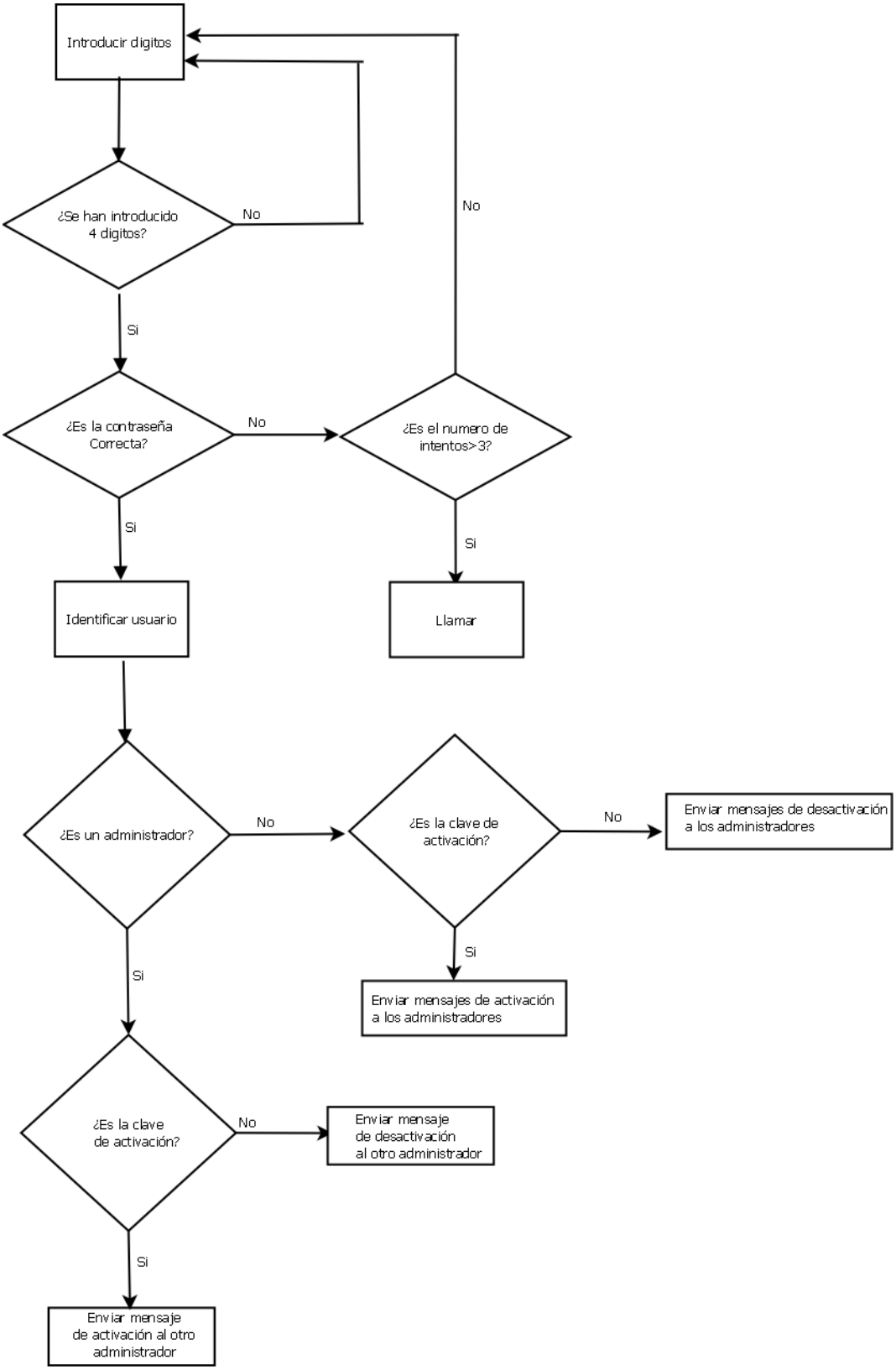


Figura V.3 Diagrama de flujo. Activación/desactivación por teclado y envío de mensaje.

V.5.-Lectura de mensajes de activación/desactivación y envío de mensajes.

La rutina se ha diseñado con el objetivo de poder activar y desactivar la alarma mediante SMS. Como se ha comentado anteriormente, cada usuario dispone de una contraseña única mediante la cual puede activar o desactivar la alarma y además los administradores reciben una notificación vía SMS cuando se ha producido este evento. En la Figura V.4 se puede apreciar la implementación realizada.

V.6.- Lectura de los sensores y activación/desactivación del relé.

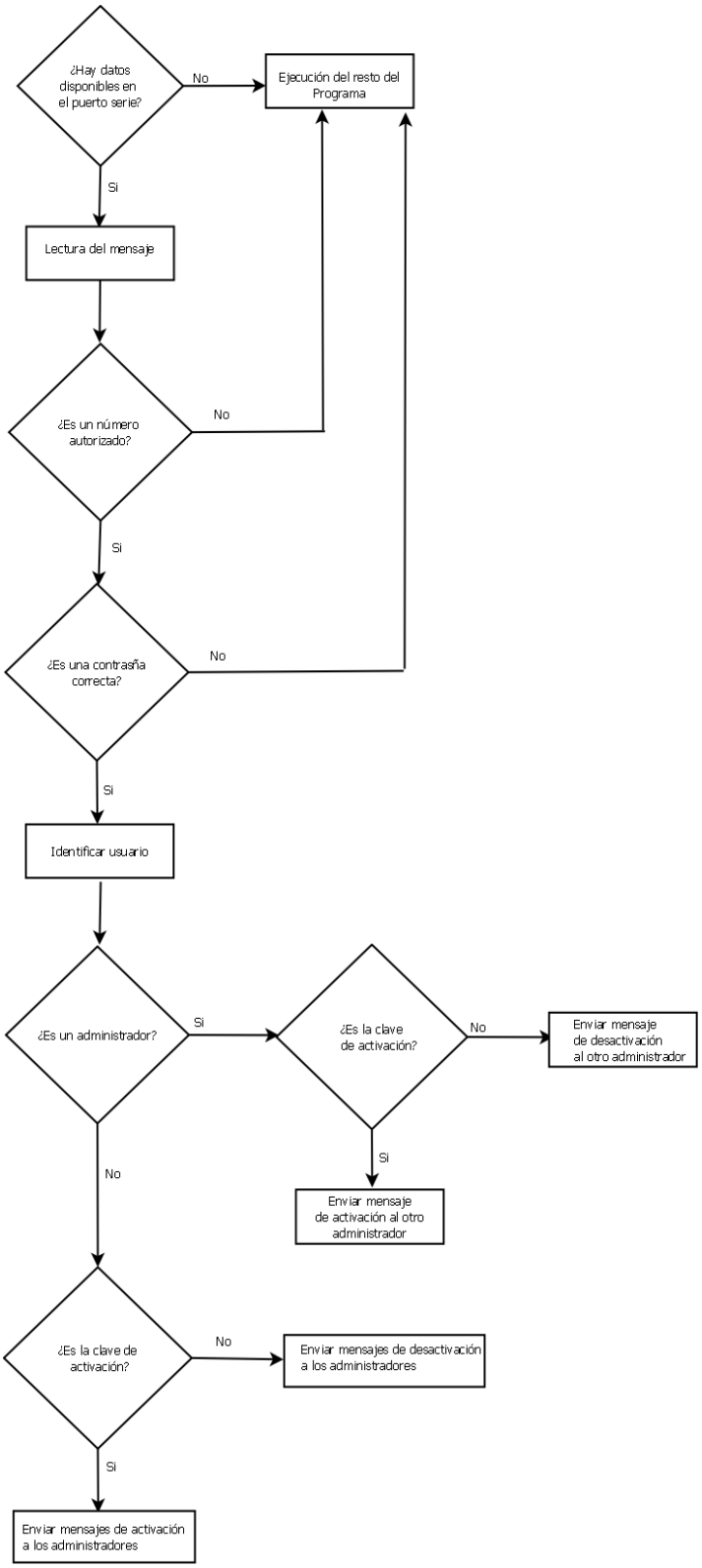
Esta función será la encargada de gestionar la activación y desactivación del relé que pone en funcionamiento para ahuyentar a los posibles intrusos (Figura V.5).

El sistema permite un total de seis sensores, cuatro configurados en modo instantáneo y dos retardados. La alarma se disparará inmediatamente (haciendo sonar una bocina) cuando cualquiera de los sensores de presencia instantáneos detecte presencia de un intruso. Sin embargo, si la detección se lleva es llevada a cabo por uno de los sensores retardados, el usuario dispondrá de 60 para proceder a su desactivación antes de que comience a sonar la bocina. Ambos tipos de sensores son llevados a una puerta OR de tal manera que se obtendrá un "1" lógico cada vez que se active cualquiera de los cuatro sensores instantáneos y otro "1" lógico diferente para el caso de los retardados. Ambas salidas están conectadas al microcontrolador. De esta manera, cuando la alarma se dispara el microcontrolador:

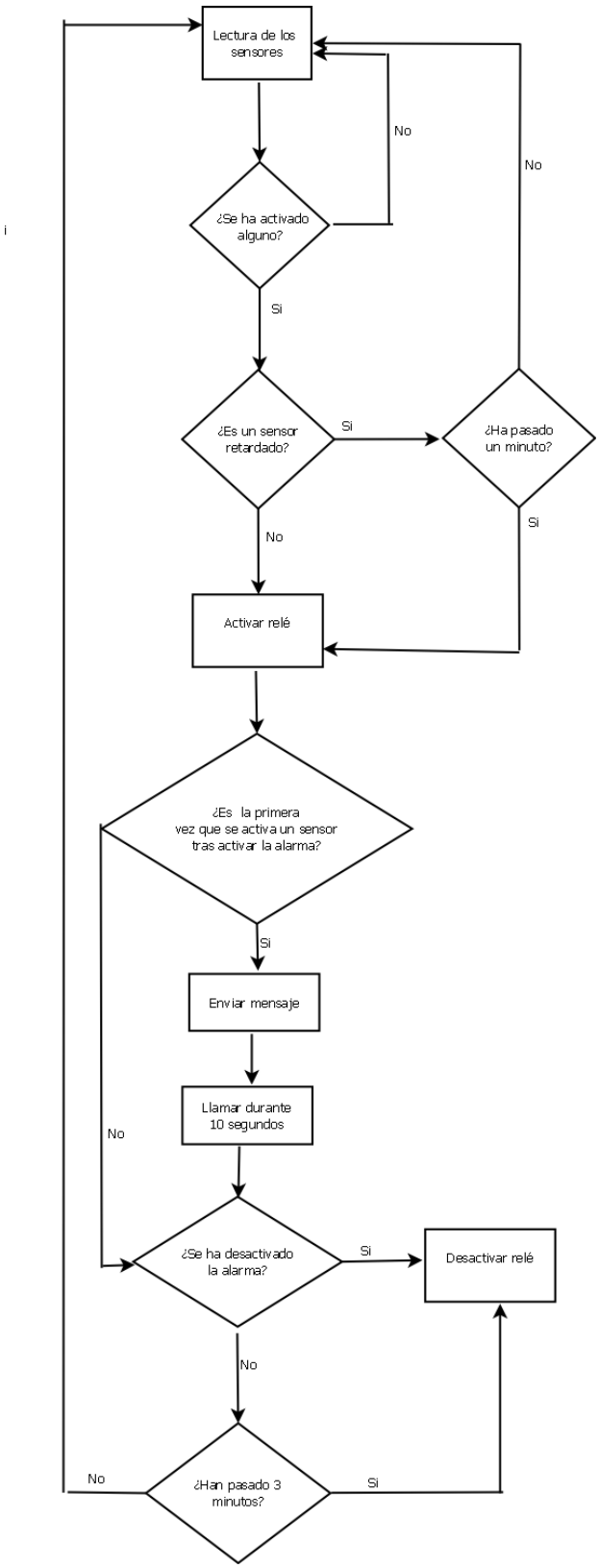
-Envía un SMS a los administradores con el texto" Atención, se ha disparado la alarma".

- Realizar una llamada perdida de diez tonos a cada uno de los administradores para asegurar que reciben la información.

- Mantendrá la señal de activación del relé activa durante tres minutos, siempre y cuando no se active nuevamente alguno de los sensores instantáneos. En este caso se reiniciará la cuenta y la bocina estará activa tres minutos más.



FiguraV.4 Diagrama de flujo activación/desactivación de alarma mediante SMS.



FiguraV.5 Diagrama de flujo. Lectura de sensores y activación /desactivación del relé.

V.7.- Programación de la aplicación para teléfonos móviles

En el presente proyecto, se ha pensado que sería de gran utilidad para el usuario el disponer de una aplicación a través de la cual pueda gestionar la activación y desactivación de la alama de forma telemática mediante mensajería SMS. Dentro de la misma, el usuario dispone de un primer menú donde deberá introducir su nombre y su número de teléfono. Posteriormente mediante el accionamiento del botón "*siguiente"* pasará a la segunda pantalla, siempre y cuando, haya introducido los campos solicitados anteriormente, donde dispondrá de dos botones principales para *"activar*" y *"desactivar*" la alama. En este mismo menú tiene la posibilidad de salir de alarma, pulsando el botón "*salir"* o de volver al menú anterior, presionando el botón "*atrás*". En la Figura V.6 se observa el diagrama de flujo de la aplicación.

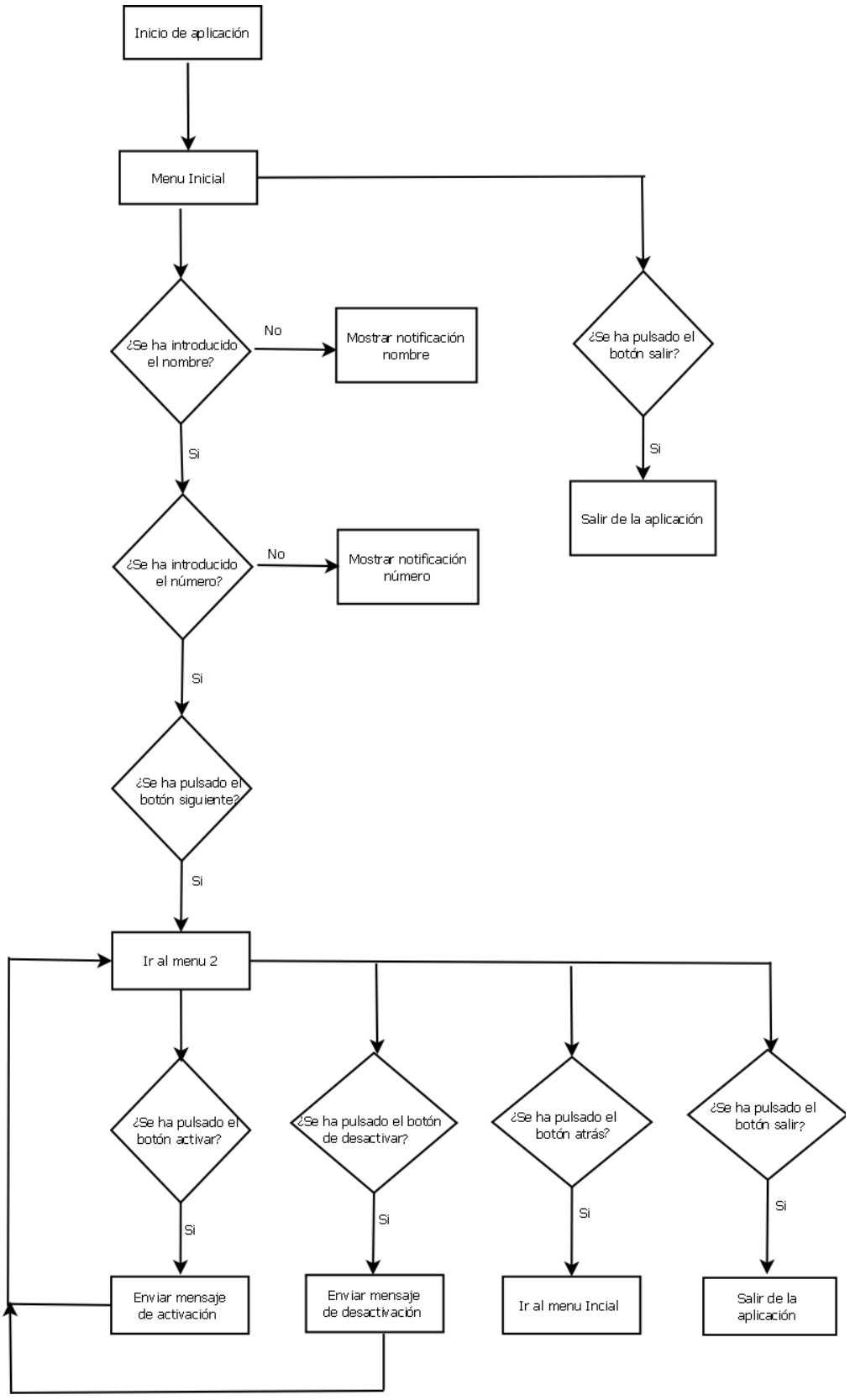


Figura V.6 Diagrama de flujo. Aplicación.

Capítulo VI: Diseño y fabricación, mediante impresión 3D, de la carcasa exterior de la alarma

Capítulo VI: Diseño y fabricación, mediante impresión 3D, de la carcasa exterior de la alarma.

VI.1.-Introducción.

El presente capítulo expone el procedimiento empleado para el diseño de una caja cuyo fin es de albergar todo la circuitería electrónica del proyecto. En el mismo, se detallan las diferentes fases del proceso, ensamblaje y resultados finales obtenidos.

El diseño de la carcasa del dispositivo de alarma que alojará en su interior el hardware, necesario para el funcionamiento del sistema, se ha realizado mediante un diseño preliminar a medida usando el software de construcción y modelado en 3D Solid Works, donde se ha hecho uso de las diferentes herramientas que ofrece el entorno para llevar a cabo pieza a pieza el diseño de la carcasa. El proceso de impresión se ha realizado usado una impresora 3D modelo i3 Prusa Hephestos donde el material empleado ha sido bobina de plástico ecológico color negro.

VI.2.- Diseño de las piezas.

Las diferentes partes que componen el diseño de la caja que recubrirá la alarma, están dimensionadas de modo que permite albergar en su justa medida todos los dispositivos de hardware usados en el proyecto. Para realizar el diseño de las piezas que componen, el prototipo en 3D, se han insertado los componentes de en un modelo de ensamblaje en Solid Works, lo que ha permitido dimensionar de manera adecuada el espacio que albergaran los dispositivos ubicados en el interior de la caja (Figura VI.1).

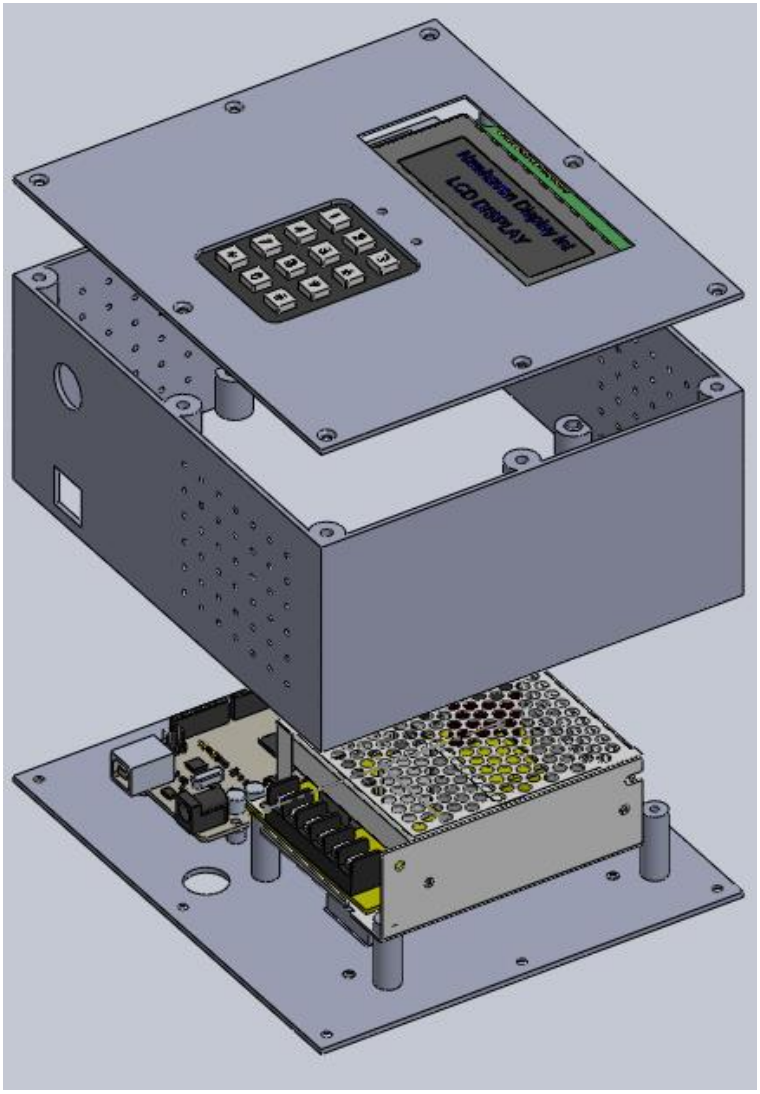


Figura VI.1. Ensamblaje.

Las piezas en las que se ha dividido el diseño han sido cuatro. La principal es la tapa inferior, encargada de albergar la placa Arduino Mega, el modem GSM, la batería y la fuente de alimentación (Figura VI.2). Las caras laterales de la caja se han dividido en dos secciones con objeto de reducir los problemas de impresión, y en cuyos extremos sendas matrices de perforación permitirán la ventilación tanto natural como forzada de los dispositivos electrónicos albergados en su interior (Figura VI.3). Ambas se han unido con objeto de conformar una única pieza.

Por último, en la tapa superior se han dispuesto los cortes necesarios para situar el display y el teclado matricial, los cuales se han atornillado desde la cara posterior de la pieza como medida pasiva de seguridad para los dispositivos (Figura VI.4).

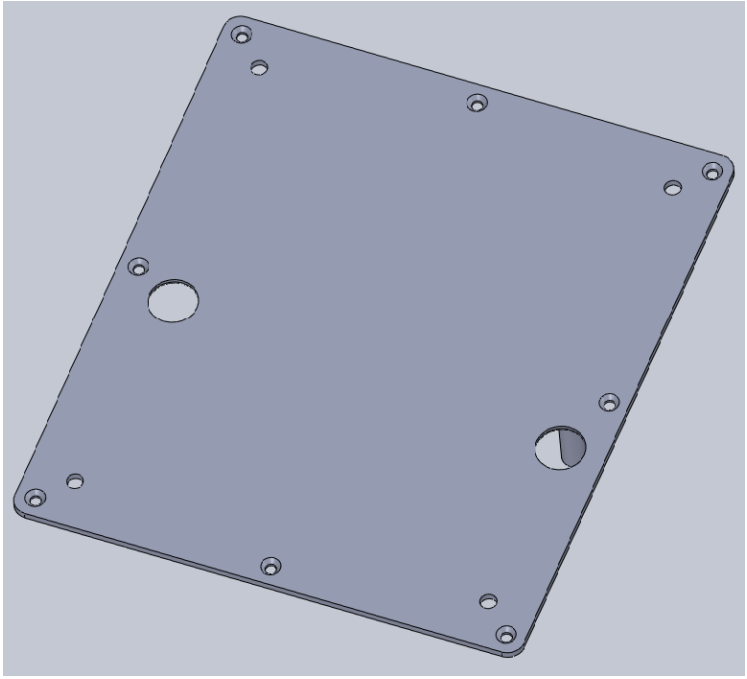


Figura VI.2. Tapa inferior.

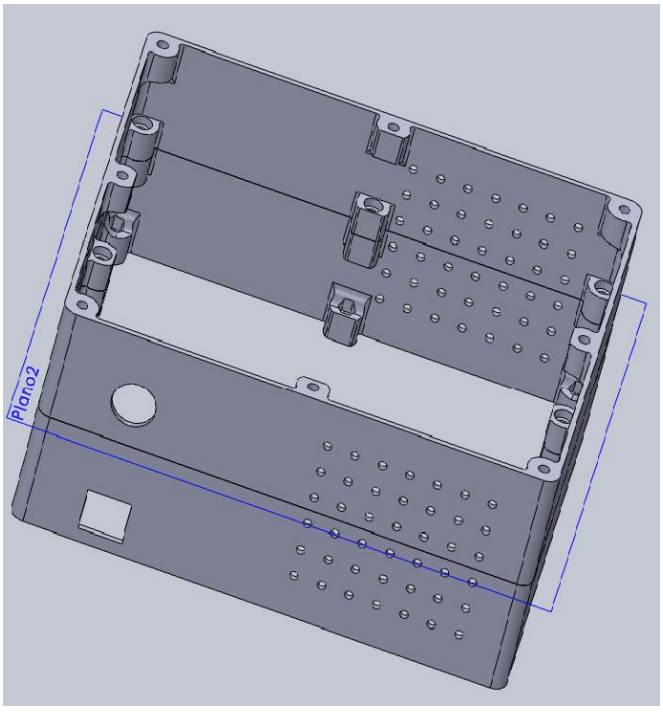


Figura VI.3. Paredes

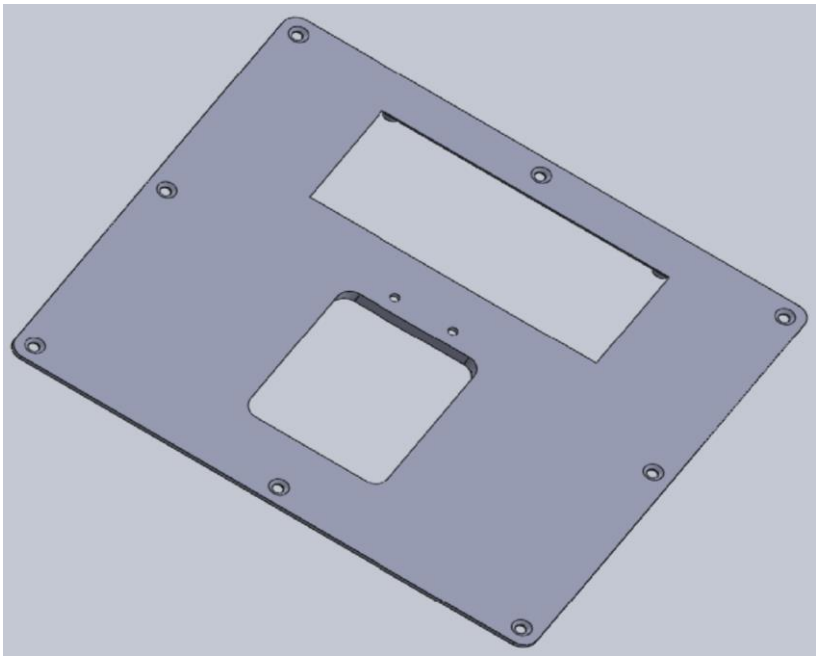


Figura VI.4. Tapa Superior

VI.3.- Fabricación de las piezas.

Una vez diseñadas las piezas, se ha procedido a la conversión de los archivos de cada una de ellas a formato .STL que permitirá que se interprete adecuadamente el contenido de los mismos por el software encargado de gestionar la impresión 3D. En este caso se ha usado el entorno del Repetiere para introducir los archivos .STL accediendo al control de los parámetros de impresión de las piezas además de disponer del Slic3r. Este nos permitirá el control de los parámetros de impresión así como la conversión de los archivos .STL en un formato legible por el microcontrolador de la impresora.

El control y configuración de los parámetros de impresión se ha realizado en el entorno del Repetire que ha permitido caracterizar el proceso de impresión modificando la temperatura del mismo, el tiempo y las capas de las piezas (Figura VI.5). La calibración de todos los parámetros se realiza de forma manual aunque dispone de opciones con las que ejecutar procesos de impresión con parámetros estipulados. Durante el tiempo de impresión se ha observado, en todo momento, el control en tiempo real de la temperatura del proceso así como las capas de material depositadas para la unificación de la pieza.

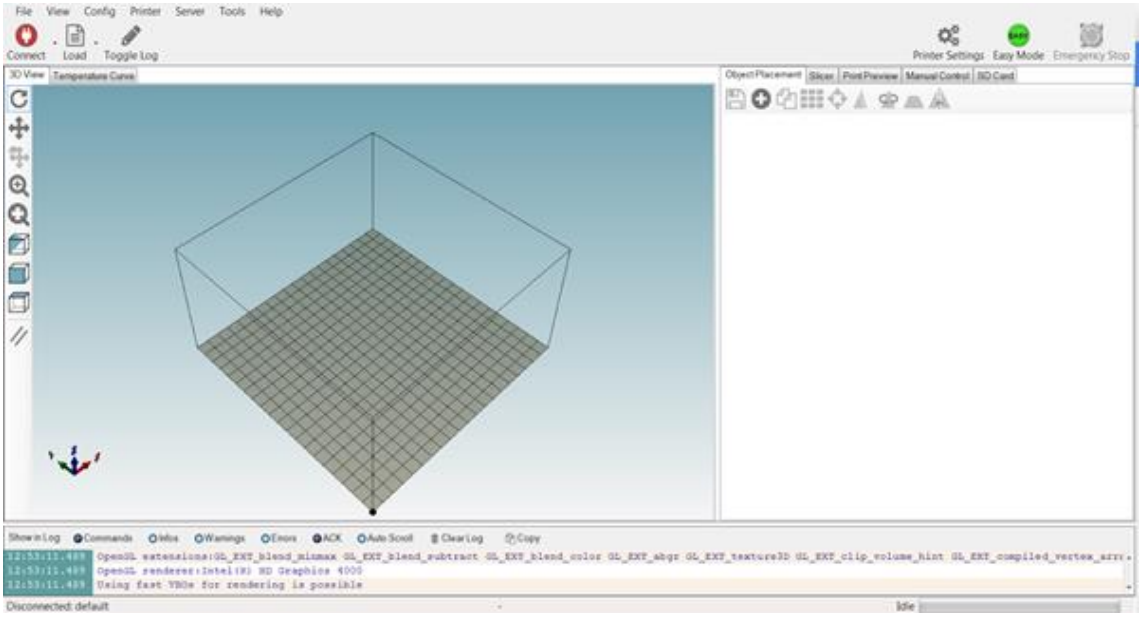


Figura VI.5. Entorno del Repetiere.

La conversión de los archivos .STL al formato de lectura del microcontrolador de la impresora se ha llevado a cabo a través del Slic3r, que es un programa de diseño CAD que permiten exportar información en este formato (Figura VI.6). Este archivo .STL se ha procesado obteniéndose un .GCODE en cuyo contenido se dispone de toda la información de impresión, de este modo se ejecuta la impresión de piezas con la información necesaria sobre la densidad y características del material utilizado con el fin de obtener diferentes acabados y resoluciones.

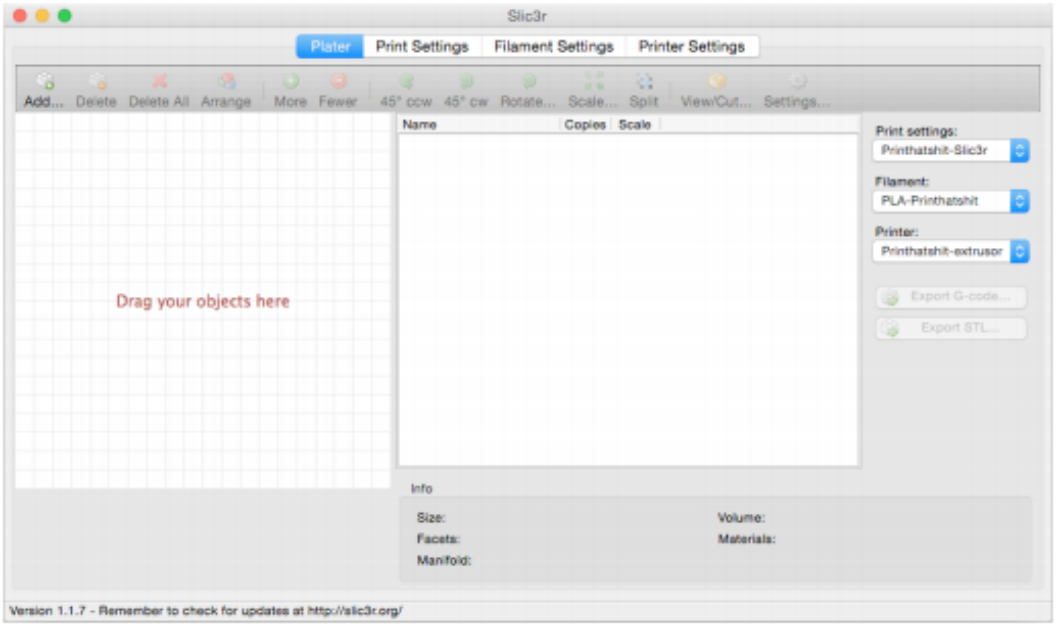


Figura VI.6. Entorno de configuración del Slic3r.

El material empleado en la impresión ha sido hilo de plástico para impresora 3D con filamento PLA biodegradable de 1,75 mm de diámetro, de superficie brillante y disponible en varios colores. En este proyecto, para la impresión de las piezas se ha elegido el color negro (Figura VI.7). Seguidamente se mostraran las especificaciones del material utilizado:

- Tipo: PLA.
- Diámetro: 1.75 mm.
- Embalaje: bobina.
- Color: negro.
- Densidad: 1,25 g/cm³. (21.5˚C).
- Temperatura de impresión: 190-225 ˚C.
- Resistencia al impacto: 5 KJ/m².
- Peso: 1 Kg.
- Longitud: 350-360 m.

Figura VI.7. Bobina de plástico PLA.

Finalmente el proceso de impresión se ha llevado a cabo con una impresora 3D Prusa i3 Hephestos, cuyas características son:

- Tecnología: FDM – FFF.
- Materiales: PLA, HIPS - Poliestireno de alto impacto.
- Tamaño máximo de impresión: 215x210x180 mm.
- Extrusor: 1.
- Tipo de extrusor: bq Witbox.

- Tamaño de filamento: 1,75 mm.
- Diámetro de boquilla: 0,40 mm.
- Espesor de capa: 60 - 300 micras.
- Base calefactable: No.
- Tipos de archivo: .STL, GCODE.
- Conectividad: USB, SD Card.
- Firmware: Marlin.
- S.O. compatibles: Windows, Mac, Linux.
- Formato de entrega: Kit.
- Peso:9,00 kg.
- Tamaño de la impresora: 460x370x510 mm.

Figura VI.8. Impresora Prusa i3 Hephestos.

Capitulo VII. Resultados experimentales

Capitulo VII: Resultados experimentales.

VII.1.- Introducción.

En el presente capítulo se exponen los resultados experimentales así como también costes estimados del proyecto indicando. En primer lugar, las verificaciones realizadas en los resultados obtenidos experimentalmente, en segundo lugar, el coste de materiales, presupuesto de la mano de obra y por último el coste total del mismo.

VII.2.-Verificaciones realizadas.

El sistema se ha testeado durante 4 días, comprobándose que tanto el "hardware" del dispositivo como aquellos aspectos más críticos de la programación han trabajado correctamente. Estos últimos son:

- Recepción y transmisión de mensajes.
- Activación de la alarma durante el tiempo predeterminado.
- Lectura de contraseñas y visualización mediante el display.
- Realización de llamadas.

Respecto a la alimentación del sistema se ha comprobado, que en caso de existir un corte de corriente, la alimentación auxiliar permite su funcionamiento ininterrumpido.
En las siguientes imágenes se muestra la verificación del funcionamiento descrito en capítulos previos.

Figura VII.1 Pantalla inicial. Alarma desactivada.

Figura VII.2. Correspondiente al mensaje mostrado en el display para activar la alarma.

Figura VII.3. Mensaje tras la instrucción de la clave correcta.

Figura VII.4. Se deberá introducir una nueva clave si la misma es incorrecta.

Figura VII.5. Mensaje mostrado tras la activación de la alarma.

Figura VII.6. Mensaje mostrado para la desactivación de la alarma.

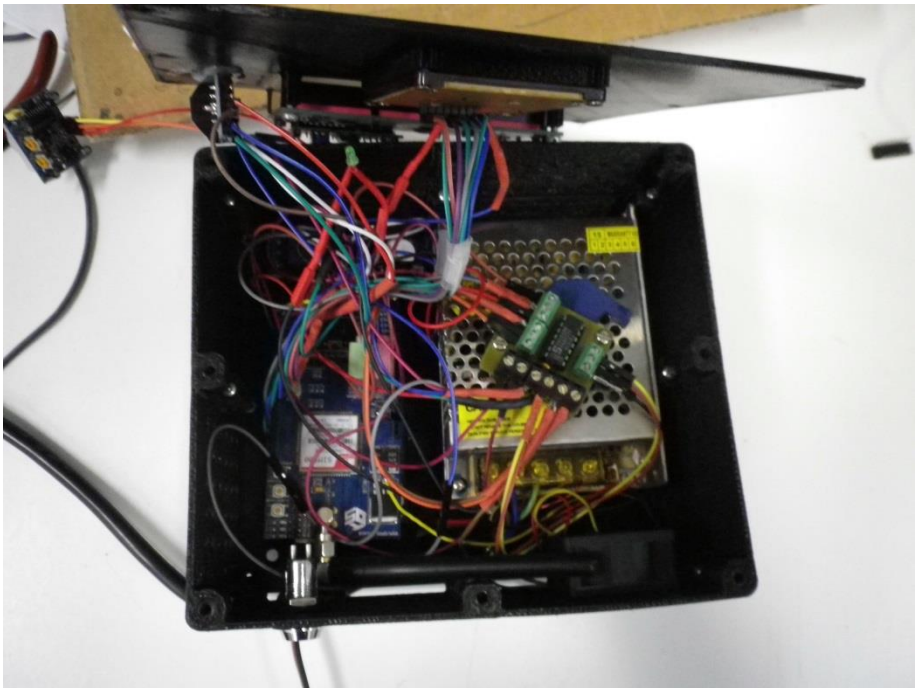


Figura VII.7. Montaje Final.

Figura VII.8.Vista panel superior

Capitulo VIII. Presupuesto.

Capitulo VIII: Presupuesto.

VIII.1.- Coste de los materiales.

Descripción	Cantidad	Coste Unitario (€/unidad)	Coste Total(€)
Arduino Mega	1	20	20
Módulo SIM900	1	46,9	46,9
Sensores de movimiento	6	5	30
Pantalla LCD	1	11	11
Reloj en tiempo real	1	5,35	5,35
Tarjeta I2C	1	2,07	2,07
Tarjeta SIM	1	5,49	5,49
Teclado matricial 4x3	1	6,79	6,79
Cables	1	8	8
Fuente de alimentación	1	23,1	23,1
Puerta OR(74LS32)	1	1,37	1,37

Filamento para impresión 3d	1	16,32	16,32
Conectores tipo clema	5	0,5	2,5
Placa de circuito impreso	2	1,5	3
Batería Ion litio	2	4,5	9
Resistencias	9	0,1	0,9
Leds indicadores	2	0,5	1
Ventilador	1	6	6
Transistor	1	1,2	1,2
Buzzer	1	1,8	1,8
NTC	1	1,4	1,4
Modulo relé	1	1,5	1,5
Diodos	3	0,6	1,8
Condensadores	4	0,2	0,8
Tornillos	20	0,1	2
Coste total materiales			**209,29**

Tabla VIII.1. Coste de los materiales

VIII.2.- Costes relacionados con la mano de obra.

Concepto	Cantidad(h)	Coste unitario(€/unidad)	Coste total(€)
Tiempo de análisis	70	30	2100
Tiempo de codificación	40	30	1200
Tiempo de implementación	150	30	4500
Tiempo de documentación	40	30	1200
Coste total mano de obra			**9000**

Tabla VIII.2. Costes relacionados con la mano de obra.

VIII.3.- Costes Generales.

Concepto	Coste total(€)
Total Coste Materiales(MT)	209,29
Total Coste Mano De Obra(MT+MO)	9000
Gastos Generales 6%(MT+MO)	552,5574
Beneficio Industrial 13%(MT+MO)	1197,2077
Coste total proyecto	**10959,0551**

Tabla VIII.3. Costes generales

Conclusiones

El presente trabajo ha permitido el diseño e implementación de un sistema de alarma para una vivienda unifamiliar. El mismo ofrece al usuario las siguientes prestaciones:

- Activación y desactivación de alarma mediante teclado matricial y vía telemática a través de una aplicación móvil.
- Aviso vía mensaje y llamada a diferentes usuarios de cualquier evento que se produzca en la misma.
- Integración del dispositivo en una caja de manera compacta.
- Posibilidad de integrar múltiples sensores tanto de activación instantánea como retardada.
- Aplicación móvil a través de la cual se controla el sistema mediante una interfaz sencilla e intuitiva.

En cuanto a los aspectos de mejora del sistema, se podrían citar los siguientes:

- Integración de un sistema de videovigilancia, permitiendo al usuario el control absoluto del mismo.
- Complementariamente, hacer uso de un detector de huellas dactilares en sustitución de la clave de acceso.
- Sustitución del cableado por comunicaciones inalámbricas.

Bibliografía

Bibliografía

[1] https://www.abus.com/es/Guia/Proteccion-antirrobo/Sistemas-de-alarma/Historia-de-los-sistemas-de-alarma

[2] http://www.microchip.com/wwwproducts/en/ATmega2560

[3] http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[4] https://es.wikipedia.org/wiki/Registro_(hardware)

[5] https://es.wikipedia.org/wiki/I%C2%B2C

[6] https://es.wikipedia.org/wiki/Serial_Peripheral_Interface

[7] https://www.arduino.cc/

[8] http://appinventor.mit.edu/explore/#

[9] https://circuits.io/eagle/

[10] http://www.avantek.es/#

[11] http://www.prometec.net/teclados-matriciales/

[12] http://www.prometec.net/sensor-pir/

[13] http://www.tuelectronica.es/tutoriales/arduino/reloj-rtc-i2c-con-arduino.html

[14] http://isa.umh.es/asignaturas/sea/practicas2C/P7/practica7.pdf

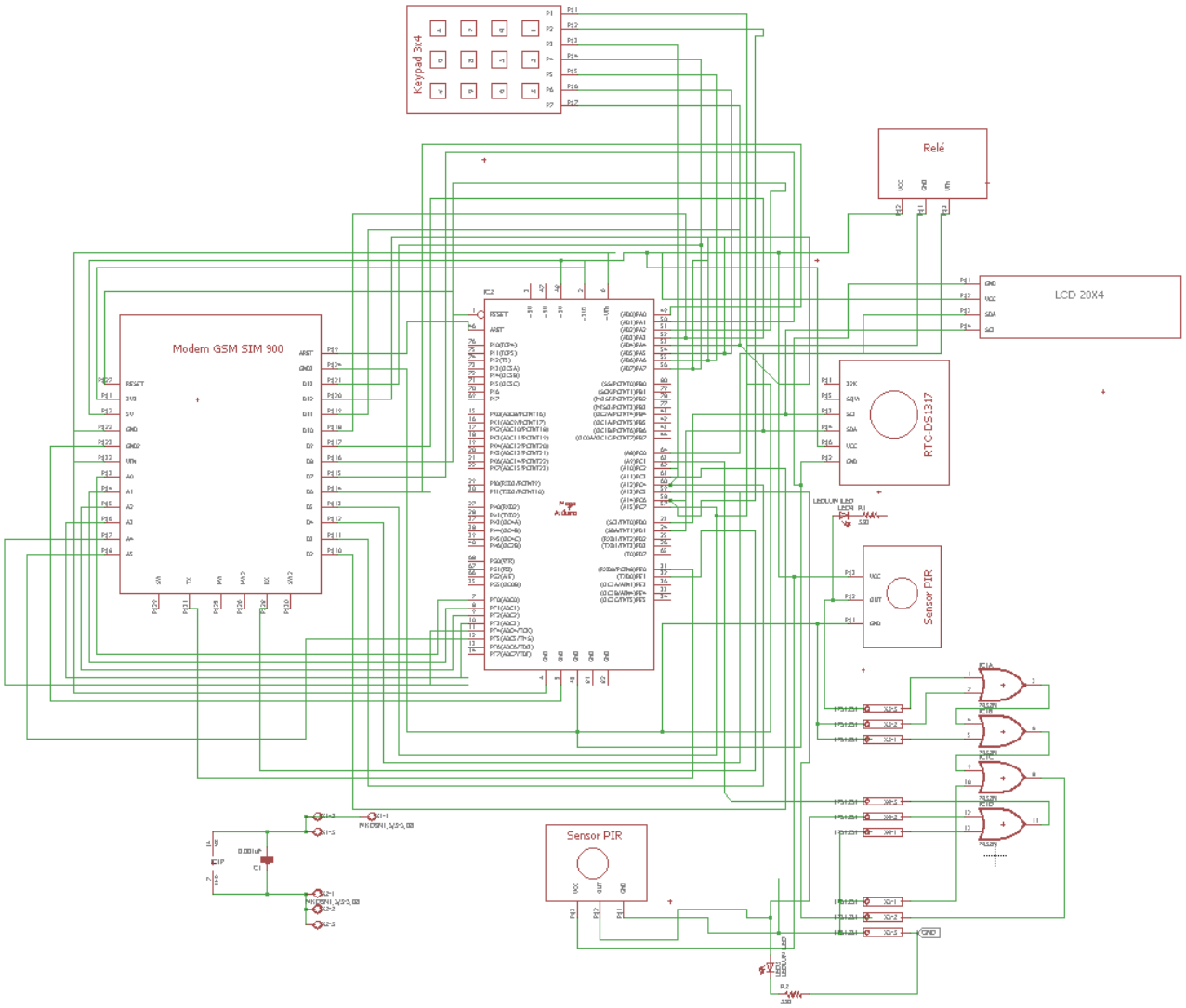
[15] http://www.instructables.com/id/M%C3%B3dulo-Serial-para-Display-LCD-con-Arduino/

[16] https://www.inventable.eu/

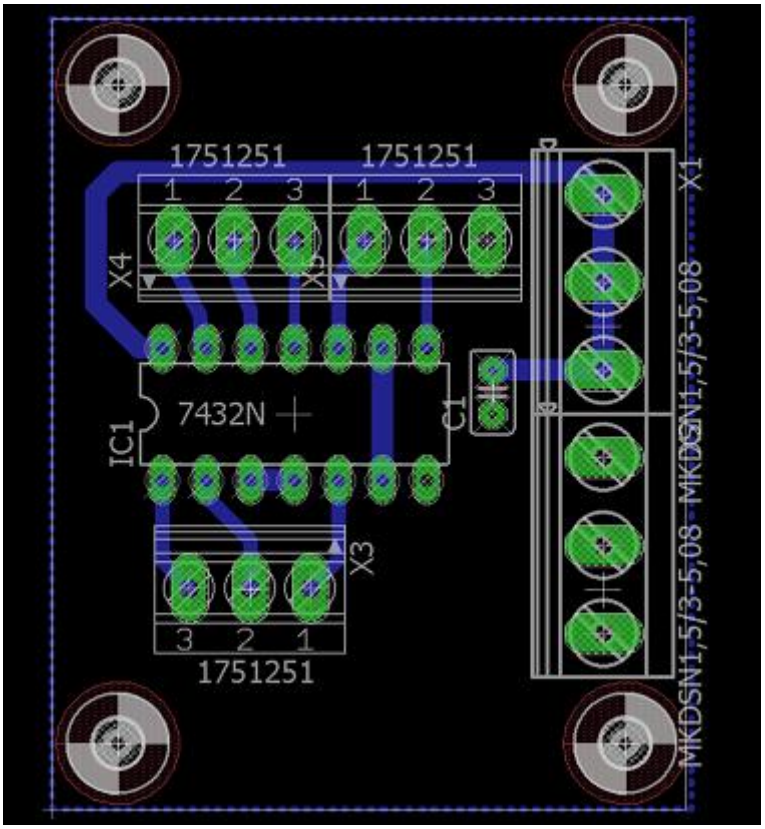
[17] http://www.tinyosshop.com

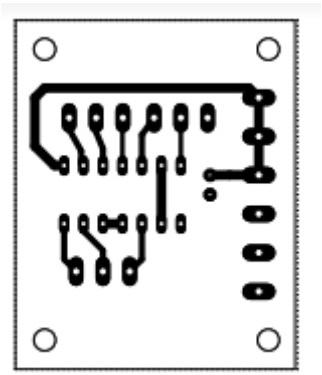
Anexos

Anexo I: Esquemas y conexiones



Anexo II: PCB y Fotolitos





<u>Anexo III</u>: Datasheets

Anexo III.1 Datasheet ATmega2560.

Atmel

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

Figure 1-2. CBGA-pinout ATmega640/1280/2560

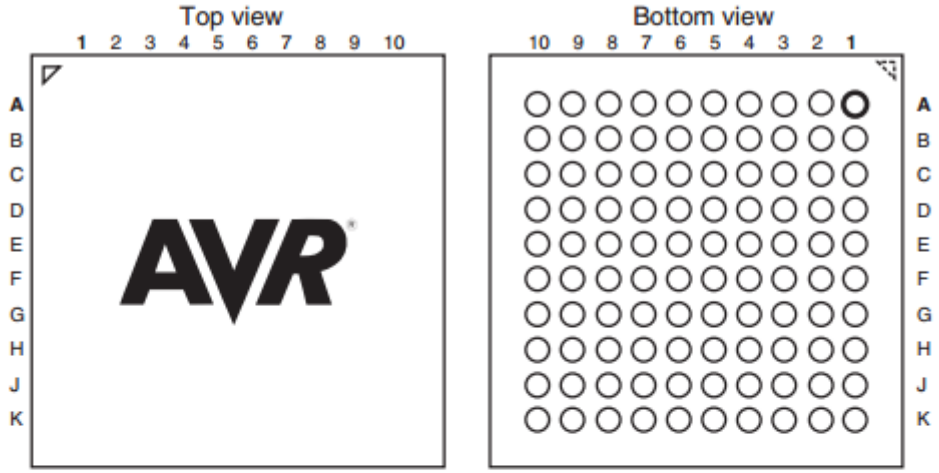


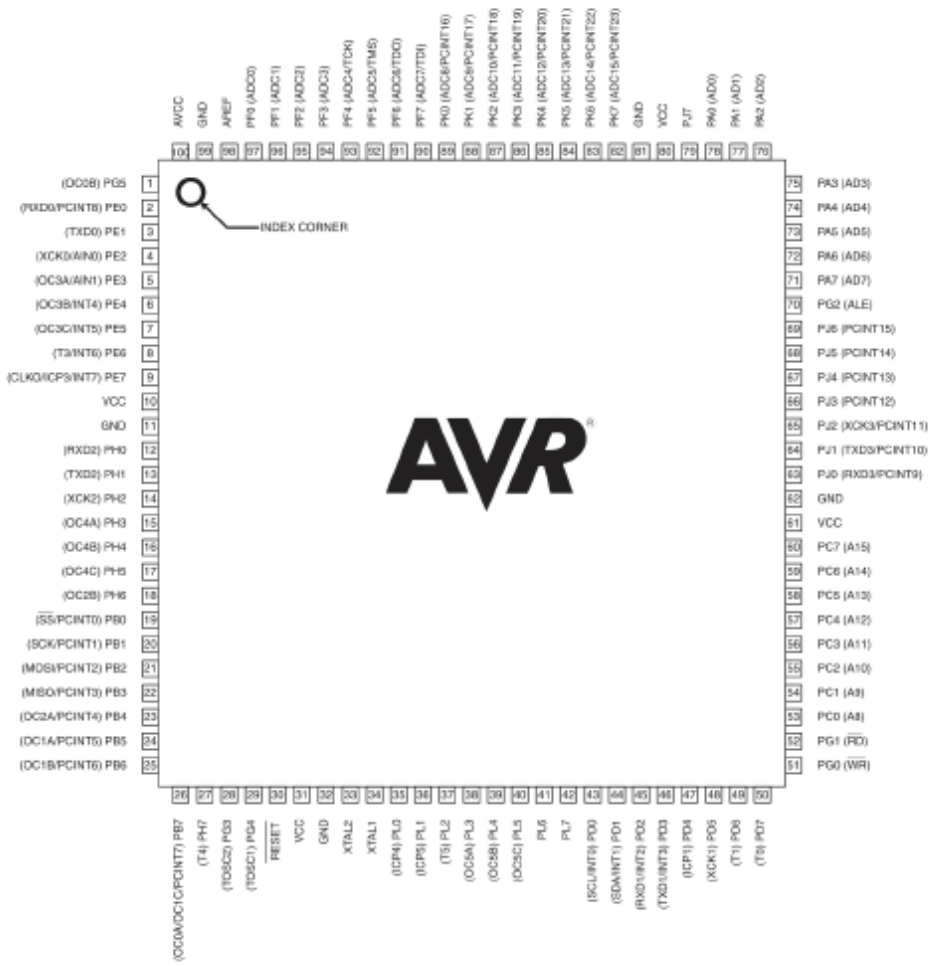
Table 1-1. CBGA-pinout ATmega640/1280/2560

	1	2	3	4	5	6	7	8	9	10
A	GND	AREF	PF0	PF2	PF5	PK0	PK3	PK6	GND	VCC
B	AVCC	PG5	PF1	PF3	PF6	PK1	PK4	PK7	PA0	PA2
C	PE2	PE0	PE1	PF4	PF7	PK2	PK5	PJ7	PA1	PA3
D	PE3	PE4	PE5	PE6	PH2	PA4	PA5	PA6	PA7	PG2
E	PE7	PH0	PH1	PH3	PH5	PJ6	PJ5	PJ4	PJ3	PJ2
F	VCC	PH4	PH6	PB0	PL4	PD1	PJ1	PJ0	PC7	GND
G	GND	PB1	PB2	PB5	PL2	PD0	PD5	PC5	PC6	VCC
H	PB3	PB4	RESET	PL1	PL3	PL7	PD4	PC4	PC3	PC2
J	PH7	PG3	PB6	PL0	XTAL2	PL6	PD3	PC1	PC0	PG1
K	PB7	PG4	VCC	GND	XTAL1	PL5	PD2	PD6	PD7	PG0

Note: The functions for each pin is the same as for the 100 pin packages shown in Figure 1-1 on page 2.

1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560



Anexo III.2 Datasheet Commandos AT SIM 900

1 Introduction

This document describes SIM900 hardware interface in great detail.
This document can help user to quickly understand SIM900 interface specifications, electrical and mechanical details. With the help of this document and other SIM900 application notes, user guide, users can use SIM900 to design various applications quickly.

2 SIM900 Overview

Designed for global market, SIM900 is a quad-band GSM/GPRS module that works on frequencies GSM 850MHz, EGSM 900MHz, DCS 1800MHz and PCS 1900MHz. SIM900 features GPRS multi-slot class 10/ class 8 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

With a tiny configuration of 24*24*3mm, SIM900 can meet almost all the space requirements in user applications, such as M2M, smart phone, PDA and other mobile devices.

SIM900 has 68 SMT pads, and provides all hardware interfaces between the module and customers' boards.

- Serial port and debug port can help user easily develop user's applications.
- Audio channel which includes a microphone input and a receiver output.
- Programmable general purpose input and output.
- The keypad and SPI display interfaces will give users the flexibility to develop customized applications.

SIM900 is designed with power saving technique so that the current consumption is as low as 1.0mA in sleep mode.

SIM900 integrates TCP/IP protocol and extended TCP/IP AT commands which are very useful for data transfer applications. For details about TCP/IP applications, please refer to *document [2]*.

2.1 SIM900 Key Features

Table 1: SIM900 key features

Feature	Implementation
Power supply	3.2V ~ 4.8V
Power saving	Typical power consumption in sleep mode is 1.0mA (BS-PA-MFRMS=9)
Frequency bands	● SIM900 Quad-band: GSM 850, EGSM 900, DCS 1800, PCS 1900. SIM900 can search the 4 frequency bands automatically. The frequency bands also can be set by AT command "AT+CBAND". For details, please refer to *document [1]*. ● Compliant to GSM Phase 2/2+
Transmitting power	● Class 4 (2W) at GSM 850 and EGSM 900 ● Class 1 (1W) at DCS 1800 and PCS 1900
GPRS connectivity	● GPRS multi-slot class 10（default）

	• GPRS multi-slot class 8 (option)
Temperature range	• Normal operation: -30°C ~ +80°C • Restricted operation: -40°C ~ -30°C and +80 °C ~ +85°C* • Storage temperature -45°C ~ +90°C
Data GPRS	• GPRS data downlink transfer: max. 85.6 kbps • GPRS data uplink transfer: max. 42.8 kbps • Coding scheme: CS-1, CS-2, CS-3 and CS-4 • Integrate the TCP/IP protocol. • Support Packet Broadcast Control Channel (PBCCH)
CSD	• Support CSD transmission
USSD	• Unstructured Supplementary Services Data (USSD) support
SMS	• MT, MO, CB, Text and PDU mode • SMS storage: SIM card
FAX	Group 3 Class 1
SIM interface	Support SIM card: 1.8V, 3V
External antenna	Antenna pad
Audio features	Speech codec modes: • Half Rate (ETS 06.20) • Full Rate (ETS 06.10) • Enhanced Full Rate (ETS 06.50 / 06.60 / 06.80) • Adaptive multi rate (AMR) • Echo Cancellation • Noise Suppression
Serial port and debug port	**Serial port:** • Full modem interface with status and control lines, unbalanced, asynchronous. • 1200bps to 115200bps. • Can be used for AT commands or data stream. • Support RTS/CTS hardware handshake and software ON/OFF flow control. • Multiplex ability according to GSM 07.10 Multiplexer Protocol. • Autobauding supports baud rate from 1200 bps to 57600bps. **Debug port:** • Null modem interface DBG_TXD and DBG_RXD. • Can be used for debugging and upgrading firmware.
Phonebook management	Support phonebook types: SM, FD, LD, RC, ON, MC.
SIM application toolkit	GSM 11.14 Release 99
Real time clock	Support RTC
Physical characteristics	Size: 24*24*3mm Weight: 3.4g
Firmware upgrade	Firmware upgradeable by debug port.

*SIM900 does work at this temperature, but some radio frequency characteristics may deviate from the GSM specification.

Anexo III.3 Datasheet sensor de movimiento HC-SR501

Product Discription

HC-SR501 is based on infrared technology, automatic control module, using Germany imported LHI778 probe design, high sensitivity, high reliability, ultra-low-voltage operating mode, widely used in various auto-sensing electrical equipment, especially for battery-powered automatic controlled products.

Specification:

- Voltage: 5V – 20V
- Power Consumption: 65mA
- TTL output: 3.3V, 0V
- **Delay time: Adjustable (.3->5min)**
- Lock time: 0.2 sec
- Trigger methods: L – disable repeat trigger, H enable repeat trigger
- Sensing range: less than 120 degree, within 7 meters
- Temperature: – 15 ~ +70
- Dimension: 32*24 mm, distance between screw 28mm, M2, Lens dimension in diameter: 23mm

Application:

Automatically sensing light for Floor, bathroom, basement, porch, warehouse, Garage, etc, ventilator, alarm, etc.

Features:

- Automatic induction: to enter the sensing range of the output is high, the person leaves the sensing range of the automatic delay off high, output low.
- Photosensitive control (optional, not factory-set) can be set photosensitive control, day or light intensity without induction.
- Temperature compensation (optional, factory reset): In the summer when the ambient temperature rises to 30 ° C to 32 ° C, the detection distance is slightly shorter, temperature compensation can be used for performance compensation.
- Triggered in two ways: (jumper selectable)
  - non-repeatable trigger: the sensor output high, the delay time is over, the output is automatically changed from high level to low level;
  - repeatable trigger: the sensor output high, the delay period, if there is human activity in its sensing range, the output will always remain high until the people left after the delay will be high level goes low (sensor module detects a time delay period will be automatically extended every human activity, and the starting point for the delay time to the last event of the time).
- With induction blocking time (the default setting: 2.5s blocked time): sensor module after each sensor output (high into low), followed by a blockade set period of time, during this time period sensor does not accept any sensor signal. This feature can be achieved sensor output time "and" blocking time "interval between the work can be applied to interval detection products; This function can inhibit a variety of interference in the process of load switching. (This time can be set at zero seconds – a few tens of seconds).
- Wide operating voltage range: default voltage DC4.5V-20V.
- Micropower consumption: static current <50 microamps, particularly suitable for battery-powered automatic control products.
- Output high signal: easy to achieve docking with the various types of circuit.

Adjustment:

- Adjust the distance potentiometer clockwise rotation, increased sensing distance (about 7 meters), on the contrary, the sensing distance decreases (about 3 meters).
- Adjust the delay potentiometer clockwise rotation sensor the delay lengthened (300S), on the contrary, shorten the induction delay (5S).

Instructions for use:

- Sensor module is powered up after a minute, in this initialization time intervals during this module will output 0-3 times, a minute later enters the standby state.
- Should try to avoid the lights and other sources of interference close direct module surface of the lens, in order to avoid the introduction of interference signal malfunction; environment should avoid the wind flow, the wind will cause interference on the sensor.
- Sensor module with dual probe, the probe window is rectangular, dual (A B) in both ends of the longitudinal direction
  - so when the human body from left to right or right to left through the infrared spectrum to reach dual time, distance difference, the greater the difference, the more sensitive the sensor,
  - when the human body from the front to the probe or from top to bottom or from bottom to top on the direction traveled, double detects changes in the distance of less than infrared spectroscopy, no difference value the sensor insensitive or does not work;
- The dual direction of sensor should be installed parallel as far as possible in inline with human movement. In order to increase the sensor angle range, the module using a circular lens also makes the probe surrounded induction, but the left and right sides still up and down in both directions sensing range, sensitivity, still need to try to install the above requirements.

1 working voltage range :DC 4.5-20V

2 Quiescent Current :50uA

3 high output level 3.3 V / Low 0V

4. Trigger L trigger can not be repeated / H repeated trigger

5. circuit board dimensions :32 * 24 mm

6. maximum 110 ° angle sensor

7. 7 m maximum sensing distance

Product Type	HC--SR501 Body Sensor Module
Operating Voltage Range	5-20VDC
Quiescent Current	<50uA
Level output	High 3.3 V /Low 0V
Trigger	L can not be repeated trigger/H can be repeated trigger(Default repeated trigger)
Delay time	5-300S(adjustable) Range (approximately .3Sec -5Min)
Block time	2.5S(default)Can be made a range(0.xx to tens of seconds
Board Dimensions	32mm*24mm
Angle Sensor	<110 ° cone angle
Operation Temp.	-15-+70 degrees
Lens size sensor	Diameter:23mm(Default)

Anexo III.4 Datasheet DS1307

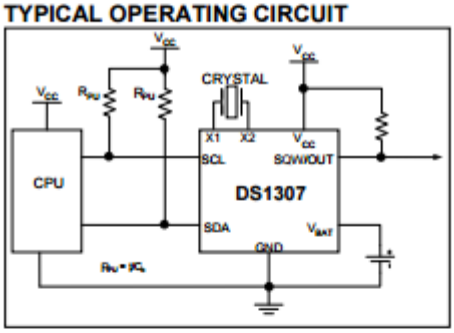
maxim integrated™

DS1307
64 x 8, Serial, I²C Real-Time Clock

GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

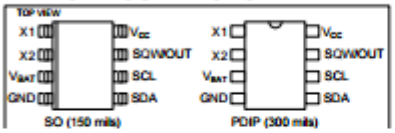
TYPICAL OPERATING CIRCUIT



BENEFITS AND FEATURES

- Completely Manages All Timekeeping Functions
  - Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year with Leap-Year Compensation Valid Up to 2100
  - 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
  - Programmable Square-Wave Output Signal
- Simple Serial Port Interfaces to Most Microcontrollers
  - I²C Serial Interface
- Low Power Operation Extends Battery Backup Run Time
  - Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
  - Automatic Power-Fail Detect and Switch Circuitry
- 8-Pin DIP and 8-Pin SO Minimizes Required Space
- Optional Industrial Temperature Range: -40°C to +85°C Supports Operation in a Wide Range of Applications
- Underwriters Laboratories® (UL) Recognized

PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*
Underwriters Laboratories, Inc. is a registered certification mark of Underwriters Laboratories, Inc.

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground .....-0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial.....0°C to +70°C
Industrial .....-40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads) .....+260°C for 10 seconds
Soldering Temperature (surface mount).....Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 0V, V_{BAT} = 3.0V; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

Anexo III.5 Datasheet SN74LS32

SDLS100

SN5432, SN54LS32, SN54S32, SN7432, SN74LS32, SN74S32
QUADRUPLE 2-INPUT POSITIVE-OR GATES
DECEMBER 1983 – REVISED MARCH 1988

- **Package Options Include Plastic "Small Outline" Packages, Ceramic Chip Carriers and Flat Packages, and Plastic and Ceramic DIPs**
- **Dependable Texas Instruments Quality and Reliability**

description

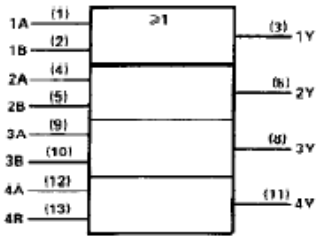
These devices contain four independent 2-input OR gates.

The SN5432, SN54LS32 and SN54S32 are characterized for operation over the full military range of −55°C to 125°C. The SN7432, SN74LS32 and SN74S32 are characterized for operation from 0°C to 70°C.

FUNCTION TABLE (each gate)

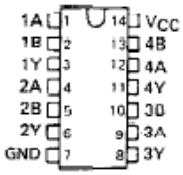
INPUTS		OUTPUT
A	B	Y
H	X	H
X	H	H
L	L	L

logic symbol†

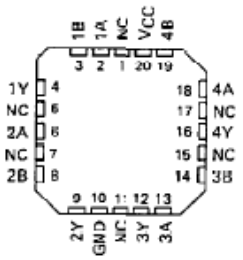


† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for D, J, N, or W packages.

SN5432, SN54LS32, SN54S32 . . . J OR W PACKAGE
SN7432 . . . N PACKAGE
SN74LS32, SN74S32 . . . D OR N PACKAGE
(TOP VIEW)

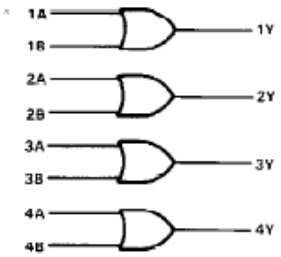


SN54LS32, SN54S32 . . . FK PACKAGE
(TOP VIEW)



NC No internal connection

logic diagram



positive logic

$Y = A + B$ or $Y = \overline{\overline{A} \cdot \overline{B}}$

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

recommended operating conditions

		SN5432			SN7432			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.8			0.8	V
I_{OH}	High-level output current			−0.8			−0.8	mA
I_{OL}	Low-level output current			16			16	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†			SN5432			SN7432			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IK}	V_{CC} = MIN,	I_I = −12 mA				−1.5			−1.5	V
V_{OH}	V_{CC} = MIN,	V_{IH} = 2 V,	I_{OH} = −0.8 mA	2.4	3.4		2.4	3.4		V
V_{OL}	V_{CC} = MIN,	V_{IL} = 0.8 V,	I_{OL} = 16 mA		0.2	0.4		0.2	0.4	V
I_I	V_{CC} = MAX,	V_I = 5.5 V				1			1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.4 V				40			40	μA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V				−1.6			−1.6	mA
I_{OS}§	V_{CC} = MAX			−20		−55	−18		−55	mA
I_{CCH}	V_{CC} = MAX,	See Note 2			16	22		16	22	mA
I_{CCL}	V_{CC} = MAX,	V_I = 0 V			23	38		23	38	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time.
NOTE 2: One input at 4.5 V, all others at GND.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see note 3)

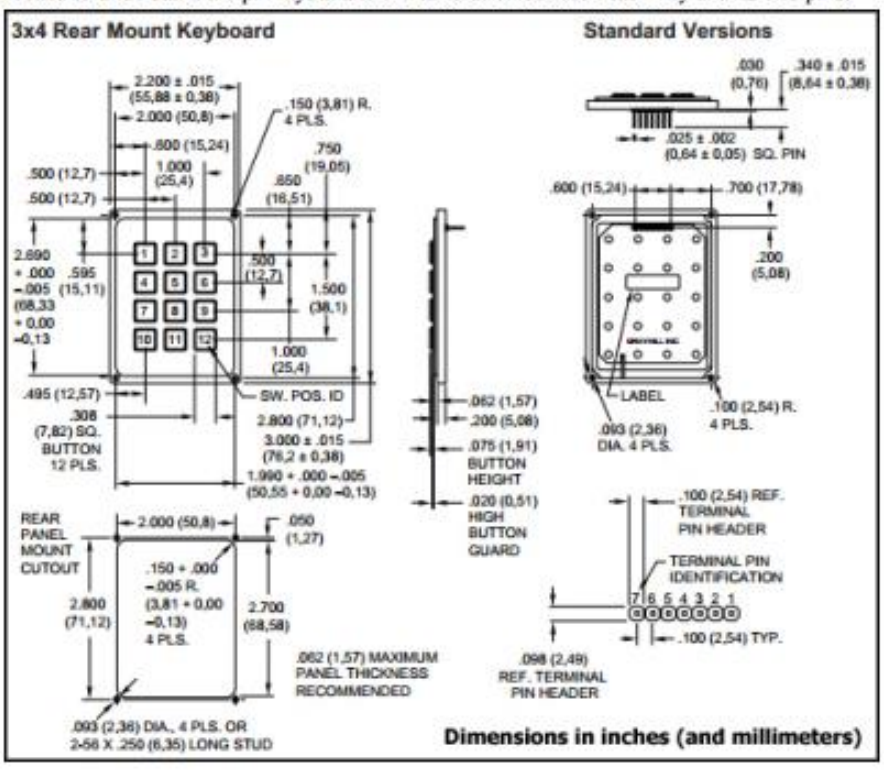
PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	A or B	Y	R_L = 400 Ω,	C_L = 15 pF		10	15	ns
t_{PHL}						14	22	ns

NOTE 3: Load circuits and voltage waveforms are shown in Section 1.

Anexo III.6 Datasheet keypad 3x4

1. Introduction:

The 12 keys keypad is widely used in many applications, some of those are: telephones and ATM machines. There are many different types of keypads and the keypad whish would be explained here would a matrix method in order to find which key is pushed. This keypad does not have pins for Vdd or Vgnd, which means it does not require a direct connection to a voltage source to perform its task. Also this keypad has 7-pins and each pin would represent a row or a column. As this keypad has 12 key, it has 3 columns and for rows. the mount of the keypad is as seen in figure 1, which shows the dimension and pins location. Moreover, the matrix of the keys related to pins is shown in figure 2. Those Figures are important in order to understand how the keypad is built, and how it would be used. Also, they shows important information that is needed to understand the pin layout and the calibration between each key and its two pins.



3. Keypad Properties:

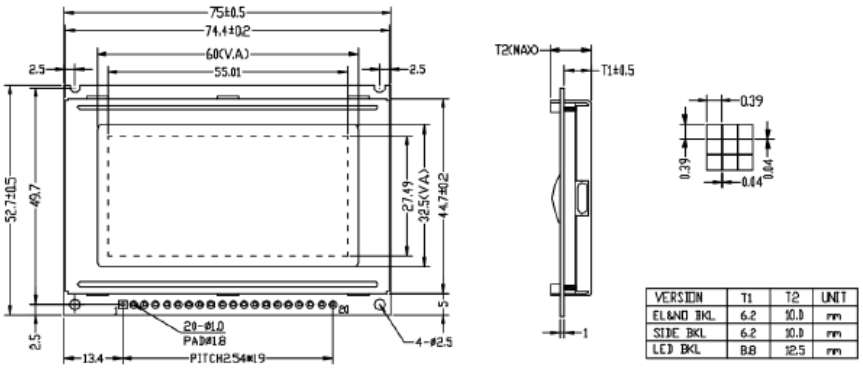
The keypad has a 3-columns and 4-rows matrix orientation as seen in figure 2. If a key is pushed then the circuit would be shorted for those specific key pins. The short circuit would always be between a row pin and a column pin. For this specific 96AB2-152-F keypad the rows 1-4 are represented by pins 1-4 and columns 1-3 are represented by pins 5-7. For example button 1 would be represented by pins 1 and 5, so if a voltage is applied to one of the pins and a voltammeter is connected to the other pin, when the button is pushed the voltammeter would read the input voltage. From Figure 2 it would be easy to construct a table that would show each button represents what character, and which pins are shorted if that button is pressed, which is provided in Table 1

Key Number	Character	Culomn Pin	Row Pin
1	'1'	5	1
2	'2'	6	1
3	'3'	7	1
4	'4'	5	2
5	'5'	6	2
6	'6'	7	2
7	'7'	5	3
8	'8'	6	3
9	'9'	7	3
10	'*'	5	4
11	'0'	6	4
12	'#'	7	4

Table 1

Anexo III.7 Datasheet GDM12864HLCM

➢ Mechanical diagram



VERSION	T1	T2	UNIT
EL&WD BKL	6.2	10.8	mm
SIDE BKL	6.2	10.8	mm
LED BKL	8.8	12.5	mm

➢ Absolute maximum ratings

Item	Symbol	Min.	Max.	Unit
Supply voltage for logic	Vdd - Vss	0	6.5	V
Input voltage	Vin	0	Vdd	
Operating temperature range	T0p	-20	70	℃
Storage temperature range	Tst	-25	75	

➢ Interface pin connections

Pin No.	Symbol	Level	Description
1	Vdd	5.0V	Supply voltage for logic and LCD (+)
2	Vss	0V	Ground
3	V0	-	Operating voltage for LCD (variable)
4~11	DB0~DB7	H/L	Data bit 0~7
12	CS2	L	Chip select signal for IC2
13	CS1	L	Chip select signal for IC1
14	/RES	L	Reset signal
15	R/W	H/L	H: read (MUP<－module),L: write (MPU－>module)
16	D/I	H/L	H: data, L: instruction code
17	E	H, H L	Chip enable signal
18	VEE	-	Operating voltage for LCD (variable)
19	A	4.2V	Backlight power supply
20	K	0V	Backlight power supply

XIAMEN OCULAR OPTICS CO., LTD. 3

SOUTH2/F, GUANGXIA BUILDING, TORCH HIGH-TECH DEVELOPMENT ARER,
XIAMEN 361006.P.R.CHINA TEL: 86-592-5650516 FAX: 86-592-5650695

Anexo IV. Código implementado.

Anexo IV.1 Código control alarma

```
#include <RTClib.h>


#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <LCD.h>
#include <Keypad.h>
#include <String.h>
#define Sensor_RE 51

#define Rele 53
#define Sensor 50
//#include <SoftwareSerial.h>
//Constructor reloj
RTC_DS1307 RTC;
//constructor LCD
LiquidCrystal_I2C          lcd(0x3F,2,1,0,4,5,6,7,3,POSITIVE);

const byte FILAS=4;
const byte COLS=3;
int lectura=0;
String leer;
```

```
//Definicion del keypad
char keys[FILAS][COLS]={
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[FILAS]={4,5,6,7};
byte colPins[COLS]={10,11,12};
//crear el keypad
Keypad matrix = Keypad(makeKeymap(keys),rowPins,colPins, FILAS, COLS);
// Variables
int aux=0;
int cursor=7;
int cursor2=7;
int rele=0;
int sensor=1;
long tiempoIncremento=0;
int pirState = LOW; // asumismos que que inicialmente el sensor PIR no ha dectectado movimiento
String  password1="121*";
String password2="121#";
//String nombre1="Manuell";
String  password3="122*";
String password4="122#";
//String nombre2="Alejandro";
String  password5="123*";
String password6="123#";
```

```
//String nombre3="Maria";
String  password7="124*";
String password8="124#";
//String nombre4="Santiago";
String nombreActiva;

String compara;
String p3="0";
int cambiar=0;
// variables para controlar el flujo de ejecución de los menus
int comprobar1=1;
int comprobar2=0;
int comprobar3=0;
int comprobar4=0;
int comprobar6=1;
String p11;
String p22;
String p1="*";// para comprobar que se pulso *
String p2= "#";// para comrpobar que se pulso #
bool se_activo=false;// me sirve para saber cuando se activa el sensor
char code[4];
int Sen;// variable donde se almacena el estado del sensor
int cont=0; // variabble para contar cuantas veces se activa la alarma
int intentos=0; // variable para contar el numero de intentos
int enciende;// variable que nos controla que la pantalla este encendida un determinado tiempo
char key ;// variable donde almacenamos el valor de la tecla pulsada
bool x=false;// variable que se utilizar para saber cuando se ha introducido la contraseña
correctamente
```

```
int controlPantalla=0;

bool AlarmaOn=false;

int controlPassword=0;

int controlPassword1=0;

long tiempoInicial, tiempoActual, tiempo_alarma,tiempoPause,tiempoSiguiente,tiempoRetardo;
// variables que se utilizan para realizar cuentas temporales

int activo1=0; // para activar tiempoEspera una unica vez

int activar=0;

int intruso=0;

int n=1; // numero de veces que detecta el sensor

int Sen_Retardo;

int retardo=0; // para controlar el retardo

bool se_activo_retardo=false;

int activar1=0;

//------- Control SMS-----------

int mensajerele=1;

int llamadarele=0;

String Comando;

//String ActivarSMS="1234";

//String desactivarSMS ="4321";

String mensaje;

String ATSMS;

String numero;

int controlSMS=0;

String numero4="+34616841445"; //Manuel

String numero3="+34659179006"; //Maria

String numero2="+34625690804"; //Santi

String numero1="+34626821262"; // Alejandro
```

```
// ------------- Definicion de Funciones ----------------

// ---------- FUNCIONES SMS-------------------

// Manda un sms cuando se activa la bocina

void mandaSmsRele(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero1);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("La alarma ha comenzado a pitar "); //mensaje que se ha enviado

delay(1500);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

// envia un sms cuando se desactiva la alarma por el usuario 1 mediante sms

void mandaSMSdesactiva(){


Serial1.print("AT+CMGS=\"");

Serial1.print(numero1);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Desactivado la alarma mediante SMS por  "+nombreActiva); //mensaje
que se ha enviado
```

```
delay(1500);
Serial1.println(( char )26) ;
delay(1000);
Serial1.println("AT+CMGD=1,4");
}
 // envia un sms cuando se desactiva la alarma por el usuario 2 mediante sms
void mandaSMSdesactiva2(){

Serial1.print("AT+CMGS=\"");
Serial1.print(numero2);
Serial1.println("\"");
delay(1000);
Serial1.println(( char )13) ;
delay(1000);
Serial1.println("Se ha Desactivado la alarma mediante SMS por "+nombreActiva); //mensaje que se ha enviado
delay(2500);
Serial1.println(( char )26) ;
delay(1000);
Serial1.println("AT+CMGD=1,4");
}
 // envia un sms cuando se activa la alarma por el usuario 1 mediante sms
void mandaSMSActiva(){

Serial1.print("AT+CMGS=\"");
Serial1.print(numero1);
Serial1.println("\"");
delay(1000);
```

```
Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante SMS por  "+nombreActiva); //mensaje que se ha enviado

delay(2500);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se activa la alarma por el usuario 2 mediante sms

void mandaSMSActiva2(){


Serial1.print("AT+CMGS=\"");

Serial1.print(numero2);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante SMS por "+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se activa la alarma por el usuario 1 mediante teclado

void ActivarTecladoSMS1(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero1);
```

```
Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante teclado por "+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se activa la alarma por el usuario 2 mediante teclado

void ActivarTecladoSMS2(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero2);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante teclado por  " +nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se activa la alarma por el usuario 3 mediante teclado

void ActivarTecladoSMS3(){

Serial1.print("AT+CMGS=\"");
```

```
Serial1.print(numero3);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante teclado por "+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se activa la alarma por el usuario 4 mediante teclado

void ActivarTecladoSMS4(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero4);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha Activado la alarma mediante teclado por "+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

 // envia un sms cuando se desactiva la alarma por el usuario 1 mediante teclado

void DesactivarTecladoSMS1(){
```

```
Serial1.print("AT+CMGS=\"");

Serial1.print(numero1);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha desactivado la alarma mediante teclado por "+nombreActiva); //mensaje
que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

envia un sms cuando se desactiva la alarma por el usuario 2 mediante teclado

void DesactivarTecladoSMS2(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero2);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha desactivado la alarma mediante teclado por "+nombreActiva); //mensaje
que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}

envia un sms cuando se desactiva la alarma por el usuario 3 mediante teclado
```

```
void DesactivarTecladoSMS3(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero3);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha desactivado la alarma mediante teclado por "+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}
```

envia un sms cuando se desactiva la alarma por el usuario 4 mediante teclado

```
void DesactivarTecladoSMS4(){

Serial1.print("AT+CMGS=\"");

Serial1.print(numero4);

Serial1.println("\"");

delay(1000);

Serial1.println(( char )13) ;

delay(1000);

Serial1.println("Se ha desactivado la alarma mediante teclado por"+nombreActiva); //mensaje que se ha enviado

delay(1000);

Serial1.println(( char )26) ;

delay(1000);

Serial1.println("AT+CMGD=1,4");

}
```

```
 // Funcion que permite realizar la lectura de SMS
void LecturaSMS(){
while(Serial1.available()){ //Si hay datos disponibles
Serial.println("Lectura");
  Comando = Serial1.readString();                    //Los almacenamos en la variable Comando
Serial.println("NUEVO SMS ENTRANTE: " + Comando);
  ATSMS= Comando.substring(1,5);
numero=Comando.substring(9,21); // poscionInicial,posicionFinal
Serial.println(numero);
mensaje=Comando.substring(Comando.length()-6,Comando.length());
mensaje.remove(4,6);
Serial.println(mensaje);
}
// se comprueba si el usario ha introducido la contraseña correcta y si el numero está dentro los numeros autorizados
if((mensaje==password1||mensaje==password3)&&(numero==numero1||numero==numero2)){
if(mensaje==password1){
mensajerele=1;
nombreActiva="Santi";
mandaSMSActiva();
        //delay(3000);
     //mandaSMSActiva()
nombreActiva="";
Serial.println("Alarma Activa");
mensaje=" ";
controlSMS=1;
lcd.clear();
    comprobar1=0;
```

```
        comprobar6=0;
        comprobar2=0;
        comprobar3=1;
        comprobar4=0;
    }
if(mensaje==password3){
mensajerele=1;
nombreActiva="Maria";
mandaSMSActiva();
      //delay(3000);
      //mandaSMSActiva()
nombreActiva="";
Serial.println("Alarma Activa");
mensaje=" ";
controlSMS=1;
lcd.clear();
     comprobar1=0;
        comprobar6=0;
        comprobar2=0;
        comprobar3=1;
        comprobar4=0;
    }
    }
if((mensaje==password5||mensaje==password7)&&(numero==numero1||numero==numero4)){
if(mensaje==password5){
mensajerele=1;
nombreActiva="Manuel";
mandaSMSActiva();
```

```
delay(3000);
mandaSMSActiva();
nombreActiva="";
Serial.println("Alarma Activa");
mensaje=" ";
controlSMS=1;
lcd.clear();
    comprobar1=0;
       comprobar6=0;
       comprobar2=0;
       comprobar3=1;
       comprobar4=0;
       }

   }
if((mensaje==password5||mensaje==password7)&&(numero==numero1||numero==numero4)){
if(mensaje==password7){
mensajerele=1;
nombreActiva="Alejandro";
mandaSMSActiva();
delay(3000);
mandaSMSActiva();
nombreActiva="";
Serial.println("Alarma Activa");
mensaje=" ";
controlSMS=1;
lcd.clear();
    comprobar1=0;
```

```
        comprobar6=0;
        comprobar2=0;
        comprobar3=1;
        comprobar4=0;
        }

    }

if((mensaje==password2||mensaje==password4)&&(numero==numero1||numero==numero2)){

if(mensaje==password2){
mensajerele=1;
nombreActiva="Santi";
mandaSMSdesactiva();
      //delay(3000);
      //mandaSMSdesactiva();
nombreActiva="";
Serial.println("Alarma Inactiva");

    // Serial.println(x);
      // Serial.println(cambiar);

mensaje=" ";

controlSMS=0;
      x=false;
digitalWrite(Rele,HIGH);
lcd.clear();
```

```
comprobar1=1;
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
      comprobar6=1;
     }
if(mensaje==password4){
mensajerele=1;
nombreActiva="Maria";
Serial.println("Alarma Inactiva");
mandaSMSdesactiva();
     //delay(3000);
     //mandaSMSdesactiva2();
nombreActiva="";
mensaje=" ";
controlSMS=0;
     x=false;
digitalWrite(Rele,HIGH);
lcd.clear();
comprobar1=1;
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
      comprobar6=1;

     }
   }
if((mensaje==password6||mensaje==password8)&&(numero==numero1||numero==numero4)){
```

```
if(mensaje==password6){
mensajerele=1;
nombreActiva="Manuel";
mandaSMSdesactiva();
delay(3000);
mandaSMSdesactiva();
nombreActiva="";
Serial.println("Alarma Inactiva");
mensaje=" ";
controlSMS=1;
lcd.clear();
    comprobar1=1;
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
      comprobar6=1;
       }

   }
if((mensaje==password6||mensaje==password8)&&(numero==numero1||numero==numero4)){
if(mensaje==password8){
mensajerele=1;
nombreActiva="Alejandro";
mandaSMSActiva();
delay(3000);
mandaSMSdesactiva();
nombreActiva="";
Serial.println("Alarma Inactiva");
```

```
mensaje=" ";
controlSMS=1;
lcd.clear();
    comprobar1=1;
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
      comprobar6=1;
       }

   }


}
                // -------- LLAMADA-------
void llamada(){

Serial1.print("ATD");  //llama
Serial1.print(numero1);
Serial1.println(";");
delay(2000);
 // Serial1.print("AT+VTD=");
 // Serial1.println(10);
 // delay(1000);
//  Serial1.print("AT+VTS=\"");
 // Serial1.print("5");
 // Serial1.println("\"");
delay(10000);
```

```
  //Serial1.println("AT+CLDTMF=2,");
  //Serial1.println("


  //Serial1.println("\"");
  //Serial1.print("ATD");  //llama
  //Serial1.print(numero1);
  //Serial1.println(";");
  //delay(8000);
  //Serial1.println("ATS0=5");
Serial1.println("ATH"); //cuelga
//delay(2000);
}
// funcion que permite mostrar la hora por el display, y donde se realizan las comprobaciones necesarias para actualizar la misma
void hora(){
    //RTC.adjust(DateTime(__DATE__,__TIME__));
DateTime now = RTC.now();// Obtiene la fecha y hora del RTC
lcd.setCursor(8,1);
 // -------     AÑO -----
if(now.month()>12){
lcd.setCursor(8,1);
lcd.print(now.year(),DEC);
}
lcd.print(now.year(),DEC);
lcd.print(" ");
  //---- MES ------
if(now.day()>28){           // condicion de actualiziacion
lcd.setCursor(5,1);
```

```
lcd.print(now.month(),DEC);
lcd.print("/");
   }
else if(now.month()<10) {
lcd.setCursor(5,1);
lcd.print("0");
lcd.print(now.month(),DEC);
lcd.print("/");

   }
else {
lcd.setCursor(5,1);
lcd.print(now.month(),DEC);
lcd.print("/");
  }

  // ------------------ DIA ----------------
if(now.hour()>24){           // condicion de actualiziacion
lcd.setCursor(2,1);
lcd.print(now.day(),DEC);
lcd.print("/");
   }
else if(now.day()<10) {
lcd.setCursor(2,1);
lcd.print("0");
lcd.print(now.day(),DEC);
lcd.print("/");
```

```
    }
else {
lcd.setCursor(2,1);
lcd.print(now.day(),DEC);
lcd.print("/");
  }


 // ------------------ HORA ---------------------

if(now.minute()>59){            // condicion de actualiziacion
lcd.setCursor(13,1);
lcd.print(now.hour(),DEC);
lcd.print(":");
    }
else if(now.hour()<10) {
lcd.setCursor(13,1);
lcd.print("0");
lcd.print(now.hour(),DEC);
lcd.print(":");
      // Serial.println("aqui");

}
else {
lcd.setCursor(13,1);
lcd.print(now.hour(),DEC);
lcd.print(":");
  }
```

```
  // ------ MINUTOS ------

if(now.second()>59){
// Serial.println("Segundo");                    // refrescar la pantalla cuando pase un minuto
lcd.setCursor(15,1);
lcd.print(":");
lcd.print(now.minute(),DEC);
}
else if(now.minute()<10){
   // Serial.println("Minuto");
lcd.setCursor(15,1);
lcd.print(":");
lcd.print("0");
lcd.print(now.minute(),DEC);
}
else{
lcd.setCursor(15,1);
lcd.print(":");
lcd.print(now.minute(),DEC);

}



}
// Mensaje mostrado el menu principal(al iniciar al sistema)

void menu1(){
```

```
if( comprobar1==1 && comprobar6==1){
lcd.clear();
      lcd.setCursor (5,0);
lcd.print("Bienvenido");
delay(10);


      lcd.setCursor (2,1);
hora();
delay(10);
      lcd.setCursor (2,2);
lcd.print("Alarma Inactiva");
      lcd.setCursor (3,3);
lcd.print("* para Activar");
// key=matrix.getKey();
      //if(key!= NO_KEY){
        // Pulse=0;
         comprobar6=0;
delay(1000);
   }
Serial.println(key);
if(comprobar1==1){
hora();
      p11=String(key); // almacenamos la teclada pulsada como un string para poder realizar las comparaciones
if(p11 == p1){
controlPassword=0;
         controlPassword1=0;
```

```
comprobar2=1;
        comprobar1=0;
        comprobar3=0;
        comprobar4=0;
mensajerele=1;
Serial.println("AQUI");
        }
    }
}
  // mensaje mostrado cuando la alarma se ha activado
void menu2(){

if( comprobar3==1){



      lcd.setCursor (3,0);
lcd.print("Alarma Vivienda");

      lcd.setCursor (2,1);
hora();
      lcd.setCursor (4,2);
lcd.print("Alarma activa");



      lcd.setCursor (2,3);
lcd.print("# para desactivar");

      comprobar6=0;
```

```
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
    }
  }

// lectura de la contraseña mediante teclado

void Read(){
if(comprobar2==1){
lcd.clear();
lcd.setCursor(2,0);
lcd.print("Introduce");
lcd.setCursor(12,0);
lcd.print("Clave");
lcd.setCursor(4,1);
lcd.print("Para activar");
while((cont<=3 && intentos<3)  ){
char key = matrix.getKey();
if ( key!= NO_KEY){
// tiempo=millis()
code[cont]= key;
Serial.println(key);
compara=String(code);
delay(10);
Serial.println(code);
lcd.setCursor(cursor,2);
lcd.print("*");
```

```
cursor++;
delay(100);
Serial.println(cont);
cont ++;

controlPassword=0;
activo1=0;
      //comprobar1=0;
      //comprobar2 = 0;
  }
  }


lcd.clear();
compara.remove(4);
Serial.println(compara);
     //   Serial.println(password+"correcto");
if(compara==password1){
controlPassword=1;
// nombre1="Manuel";
nombreActiva="Manuel";
lcd.print("Clave Correcta");
ActivarTecladoSMS1();
nombreActiva="";
     //  ActivarTecladoSMS2();
       //delay(2000);
intentos=0;
lcd.clear();
```

```
rele=1;
     x=true;
sensor=1;
cont=0;
    comprobar1=0;
       comprobar6=0;
       comprobar2=0;
       comprobar3=1;
       comprobar4=0;
cursor=7;
     }
if(compara==password3){
controlPassword=1;
      // nombre2="Alejandro"
nombreActiva="Alejandro";
lcd.print("Clave Correcta");
ActivarTecladoSMS1();
nombreActiva="";
      // ActivarTecladoSMS2();
       //delay(2000);
intentos=0;
lcd.clear();
rele=1;
     x=true;
sensor=1;
cont=0;
    comprobar1=0;
       comprobar6=0;
```

```
        comprobar2=0;
        comprobar3=1;
        comprobar4=0;
cursor=7;
     }
if(compara==password5){
controlPassword=1;
nombreActiva="Maria";
lcd.print("Clave Correcta");
ActivarTecladoSMS1();
          //nombre3="Maria";
delay(3000);
ActivarTecladoSMS1();

nombreActiva="";
intentos=0;
lcd.clear();
rele=1;
     x=true;
sensor=1;
cont=0;
    comprobar1=0;
       comprobar6=0;
       comprobar2=0;
       comprobar3=1;
       comprobar4=0;
cursor=7;
     }
```

```
if(compara==password7){
nombreActiva="Santiago";
controlPassword=1;
      // nombre4="Santiago";
lcd.print("Clave Correcta");
ActivarTecladoSMS1();
delay(3000);
ActivarTecladoSMS1();
nombreActiva="";

intentos=0;
lcd.clear();
rele=1;
     x=true;
sensor=1;
cont=0;
    comprobar1=0;
       comprobar6=0;
       comprobar2=0;
       comprobar3=1;
       comprobar4=0;
cursor=7;
     }

if(intentos == 3 ){
cursor=7;
lcd.print("No hay mas intentos");
lcd.setCursor(3,1);
```

```
lcd.print("Pulse 0 para ");
lcd.setCursor(5,2);
lcd.print("Reiniciar");
// llamada();
while(String(key) != p3) {
key=matrix.getKey();
      comprobar1=1;
      comprobar6=1;
      comprobar2=0;
      comprobar3=0;
      comprobar4=0;
intentos=0;
cont=0;
       }
      }



else if((compara!=password1||compara!=password3|| compara!=password5||
compara!=password7)&&(controlPassword==0)){
if(intentos<3){
      comprobar2=1;
      comprobar6=0;
      comprobar3=0;
      comprobar4=0;
cursor=7;
lcd.setCursor(8,0);
lcd.print("Error");
lcd.setCursor(5,1);
```

```
lcd.print("Introduce");
lcd.setCursor(5,2);
lcd.print("Nueva Clave");
delay(2000);
// lcd.print("   ");
        }
//lcd.clear();
for(int i=0;i<=3;i++){



cont--;
      }



compara.remove(0,4);
  //if(controlPassword==0){
intentos++;
}
 }
}
```

// Activa el rele cuando alguno de los sensores(instantaneo o retardados) se ha activado y se ha introducido la contraseña correcta

```
void RELE_ON (){



if((se_activo == true || retardo==1) ){ //&& x == true


Serial.print("cuenta:");
```

```
        Serial.println(n);
digitalWrite(Rele,LOW);


if(mensajerele==1){
delay(2000);
mandaSmsRele();
Serial.print("Mensaje rele");
mensajerele=0;
llamadarele=0;
        }


rele=1;
retardo=0;
if(activar==0){
tiempoInicial=millis();
      //Serial.println(tiempoInicial);
      tiempo_alarma=tiempoInicial+180000;  //  ALARMA ON  ---> 3min
     // Serial.println(tiempo_alarma);
activar=1;
     // controlSMS=0; // una vez se haya activado el sensor(se_activo=true) se cumplira el if y desactivamos la variable que nos indica que se ha recibido el mensaje de activacion
      }
      }
tiempoActual=millis();
if(tiempo_alarma-175000<=tiempoActual &&llamadarele==0){
llamada(); // se realiza una llamada a los 5 segundos de la activacion del rele
llamadarele=1;
      }
```

```
if(tiempoActual > tiempo_alarma){
Serial.println("APAGAR RELE");
digitalWrite(Rele,HIGH);
      se_activo=false;
activar=0;
      n=1;
rele=0;
mensajerele=1;
      }

else if((tiempo_alarma-tiempoActual)/1000>0){
Serial.print("Tiempo de espera: ");
Serial.println((tiempo_alarma-tiempoActual)/1000);
          }
          /* if(digitalRead(Rele)==LOW){
llamadarele=0;
mensajerele=1;
Serial.print("jojojojojoOOOOOOOO");
       // activar1=0;
      }*/

 }
  //permite la lecutra de los sensores retardados
void LecturaSensorRetardado(){
    Sen_Retardo=digitalRead(Sensor_RE);
if(Sen_Retardo==HIGH){
     se_activo_retardo=true;
```

```
if(activar1==0){
tiempoPause=millis();
tiempoRetardo=tiempoPause+60000;// tomaremos un retardo del sensor de 1min
    activar1=1;
    }
    }

    /*if(Sen_Retardo==HIGH && tiempo_alarma-tiempoActual<10000){
retardo=1;
     Sen_Retardo=LOW;
activar=0;
     */

tiempoSiguiente=millis();
if(tiempoSiguiente>tiempoRetardo &&  se_activo_retardo==true){
retardo=1;
     activar1=0;
activar=0;
n++;
      se_activo_retardo=false;
    }
else if((tiempoRetardo-tiempoSiguiente)/1000 >0){

Serial.print("Tiempo Sensor Retardado :");
Serial.println((tiempoRetardo-tiempoSiguiente)/1000);
    }
 }
```

```
 // permite la lecutra de los sensores instantaneos

void LecturaSensor(){

      Sen=digitalRead(Sensor);

if(Sen==HIGH){

/*if(rele == 1 && tiempoActual<tiempo_alarma &&  tiempo_alarma-
tiempoActual<25000+tiempoIncremento){

tiempoIncremento=30000-(tiempo_alarma-tiempoActual)+tiempoIncremento; //mientras este la
alarma activa iremos incrmentado el tiempo cada vez que se active el sensor

activar=0;

Serial.println(tiempoIncremento);*/

Serial.println("Sensor Activo");


se_activo=true;

if( pirState == LOW){

n++; // para contar cuantas veces se activa el sensor

activar=0;

pirState = HIGH;

       }

Serial.println("SENSOR ACTIVO");

      }


else{

pirState = LOW;

       se_activo=false;


Serial.println("SENSOR  NO ACTIVO");

}

      }
```

```
/* if(sensor==1){
   Sen=digitalRead(Sensor);

if(Sen==HIGH){
tiempoSiguiente=millis();
delay(50);
    se_activo=true;
Serial.println("SENSOR ACTIVO");
// digitalWrite(Sensor,LOW);
// AlarmaOn=true;
if (pirState == LOW) {
Serial.println("Movimiento detectado");
pirState=HIGH;
activar=0;
n++;

     }
   }
    //AlarmaOn=true;

    //Solamente queremos detectar el cambio de estado
else{
    // tiempo_alarma=0;
//if( pirState == HIGH){
// digitalWrite(Sensor,HIGH);
Serial.println("SENSOR  NO ACTIVO");
Serial.println("Movimiento finalizado");
//pirState = LOW;
```

```
      se_activo=false;
//AlarmaOn=false;
  }
   }
   }
     }*/
     // permite desactivar la alarma
void Read_2(){
if(comprobar4==1){
lcd.clear();
lcd.setCursor(2,0);
lcd.print("Introduce");
lcd.setCursor(12,0);
lcd.print("Clave");
lcd.setCursor(2,1);
lcd.print("para desactivar");
   //delay(1000);
while((cont<=3 && intentos<3 )  ){
//  LecturaSensorRetardado();
LecturaSensor();
RELE_ON();


char key = matrix.getKey();
if ( key!= NO_KEY){
// tiempo=millis()
code[cont]= key;
Serial.println(key);
```

```
compara=String(code);
delay(10);
Serial.println(code);
lcd.setCursor(cursor2,2);
lcd.print("*");
cursor2++;
Serial.println(cont);
cont ++;
  }
  }


lcd.clear();
compara.remove(4);
Serial.println(compara);
        //Serial.println(password1+"correcto");
if(compara==password2){
nombreActiva="Manuel";
digitalWrite(Sensor,LOW);
digitalWrite(Rele,HIGH);

       x=false;
cambiar = 0;
       //salir=0;
       //intruso=0;
     controlPassword1=1;
lcd.print("Clave correcta");
DesactivarTecladoSMS1();
```

```
nombreActiva="";
        comprobar1=1;
        comprobar6=1;
       comprobar2=0;
       comprobar3=0;
       comprobar4=0;
controlSMS=0;
aux=1;
       //DesactivarTecladoSMS();
     // DesactivarTecladoSMS2();
      // delay(2000);
       cursor2=7;
intentos=0;
lcd.clear();
cont=0;
     //comprobar1=5;

}
if(compara==password4){
nombreActiva="Alejandro";
digitalWrite(Sensor,LOW);
digitalWrite(Rele,HIGH);

       x=false;
cambiar = 0;
       //salir=0;
       //intruso=0;
     controlPassword1=1;
```

```
lcd.print("Clave correcta");
DesactivarTecladoSMS1();
        //DesactivarTecladoSMS1();
nombreActiva="";
         comprobar1=1;
         comprobar6=1;
        comprobar2=0;
        comprobar3=0;
        comprobar4=0;
controlSMS=0;
aux=1;
        //DesactivarTecladoSMS();
      // DesactivarTecladoSMS2();
       // delay(2000);
        cursor2=7;
intentos=0;
lcd.clear();
cont=0;
      //comprobar1=5;

      }
if(compara==password6){
nombreActiva="Maria";
digitalWrite(Sensor,LOW);
digitalWrite(Rele,HIGH);

        x=false;
cambiar = 0;
```

```
        //salir=0;
        //intruso=0;
      controlPassword1=1;
lcd.print("Clave correcta");
DesactivarTecladoSMS1();
delay(3000);
DesactivarTecladoSMS1();
nombreActiva="";
         comprobar1=1;
         comprobar6=1;
        comprobar2=0;
        comprobar3=0;
        comprobar4=0;
controlSMS=0;
aux=1;
        //DesactivarTecladoSMS();
      // DesactivarTecladoSMS2();
       // delay(2000);
        cursor2=7;
intentos=0;
lcd.clear();
cont=0;
      //comprobar1=5;

      }
if(compara==password8){
nombreActiva="Santiago";
digitalWrite(Sensor,LOW);
```

```
digitalWrite(Rele,HIGH);


       x=false;
cambiar = 0;
       //salir=0;
       //intruso=0;
     controlPassword1=1;
lcd.print("Clave correcta");
DesactivarTecladoSMS1();
delay(3000);
DesactivarTecladoSMS1();
nombreActiva="";
        comprobar1=1;
        comprobar6=1;
       comprobar2=0;
       comprobar3=0;
       comprobar4=0;
controlSMS=0;
aux=1;
       //DesactivarTecladoSMS();
     // DesactivarTecladoSMS2();
      // delay(2000);
       cursor2=7;
intentos=0;
lcd.clear();
cont=0;
     //comprobar1=5;
```

```
}

if(intentos == 3 ){
        //RELE_ON();

        cursor2=7;
lcd.clear();
lcd.setCursor(2,0);
lcd.print("Intruso Detectado");
Serial.println("Intruso");
llamada(); // realiza una llamada si se ha fallado 3 veces al introducir la contraseña
lcd.clear();
        x=true;
cambiar=0;
intentos=0;
cont=0;
        comprobar1=0;
        comprobar6=0;
       comprobar2=0;
       comprobar3=1;
comprobar4=0;

        }



else
if((compara!=password2||compara!=password4||compara!=password6||compara!=password8)&
&(controlPassword1==0)){
if(intentos<3){
```

```
            comprobar1=0;
        comprobar6=0;
        comprobar2=0;
        comprobar3=0;
        comprobar4=1;
            cursor2=7;
lcd.setCursor(8,0);
lcd.print("Error");
lcd.setCursor(5,1);
lcd.print("Introduce");
lcd.setCursor(5,2);
lcd.print("Nueva Clave");
delay(2000);
}

for(int i=0;i<=3;i++){



cont--;
}



compara.remove(0,4);
intentos++;
        }
   }
  }
```

```
 // permite apagar la pantalla pasado 4 seg y activarla cuando se pulsa una tecla
void ComprobarEncendido(){
  // Serial.println("Estoy en CE");
if(key!=NO_KEY){
enciende=0;
}
if(enciende<100){
lcd.setBacklight(HIGH);
delay(10);
enciende++;

      }
if(enciende==100){
lcd.setBacklight(LOW);
}

   }

void Activar_Rele(){

if((x==true && cambiar != 1)|| (controlSMS==1)){// hemos introducido la contraseña correctamente
Serial.println("rele");
menu2();
  // delay(500)
hora();
LecturaSensor();
LecturaSensorRetardado();
```

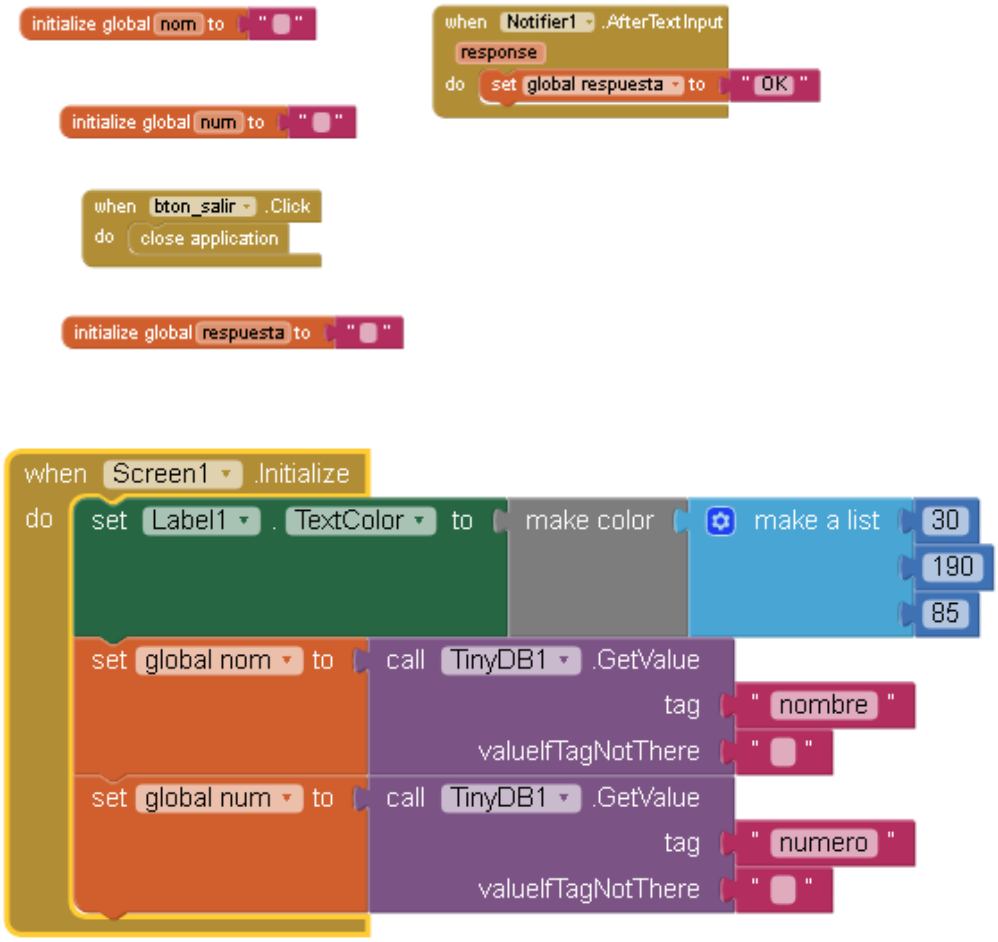
```
   // key=matrix.getKey();
if(key =='#'){
      comprobar6=0;
comprobar2=0;
       comprobar3=0;
       comprobar4=1;
ComprobarEncendido();
cambiar=1;
Read_2();

   }

if(aux==0){
    RELE_ON();
     }
aux=0;
  }
}




void setup() {

pinMode(8,INPUT);
if(digitalRead(8)==LOW){
digitalWrite(8,HIGH);
  }
```

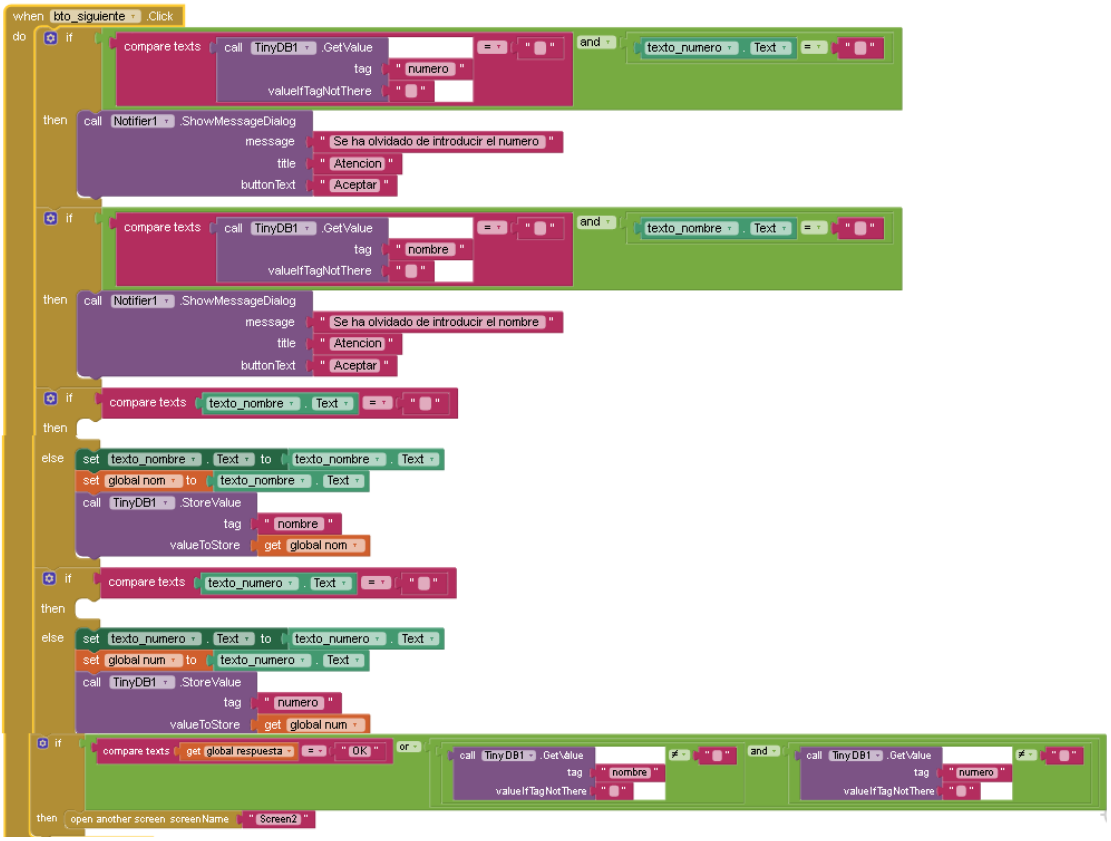
```
pinMode(Sensor,INPUT);
digitalWrite(Sensor,LOW);
pinMode(Sensor_RE,INPUT);
digitalWrite(Sensor_RE,LOW);
RTC.begin();
  //RTC.adjust(DateTime(__DATE__,__TIME__));

Serial.begin(9600);
Wire.begin(); // Inicia el puerto I2C
 lcd.begin (20,4);  // inicializar lcd (filas*columnas)
//lcd.setBacklightPin(3,POSITIVE);
lcd.setBacklight(HIGH);
delay(100);
Serial1.begin(9600);
Serial1.println("ATE0");
Serial1.println(char(13));//Iniciamos una instancia de la librería SoftwareSerial
Serial1.println("AT+CMGF=1");            //Configuramos el módulo para trabajar con los SMS en modo texto
delay(100);
Serial1.println("AT+CNMI=2,2,0,0,0");     //Configuramos el módulo para que nos muestre los SMS recibidos por comunicacion serie
Serial1.println("AT+CMGD=1,4");
 // Comando AT para eliminar todos los SMS
pinMode(Rele,OUTPUT);
digitalWrite(Rele,HIGH);
Serial1.println("AT+CSQ");
    }
```

```
void loop() {
key=matrix.getKey();
LecturaSMS();
menu1();

//cambiado = matrix.keyStateChanged();

ComprobarEncendido();
Read();
menu2();
   Activar_Rele();
if(comprobar4==1){
    Read_2();

}
}
```

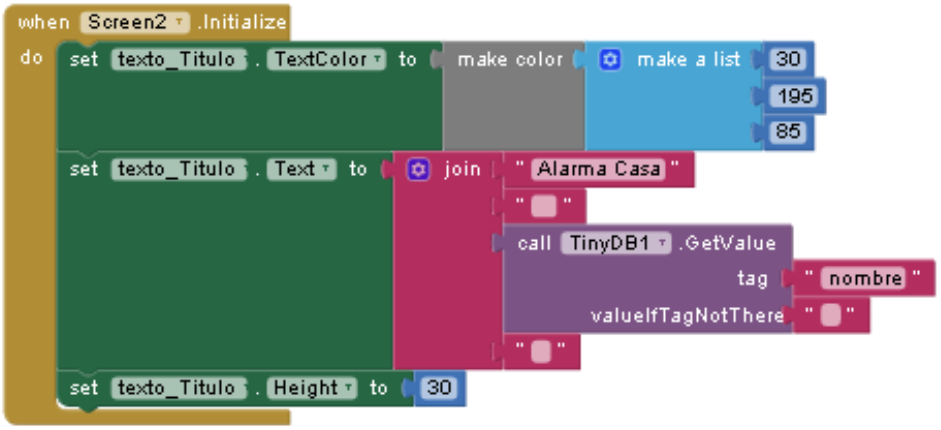
Anexo IV.2 Aplicación móvil

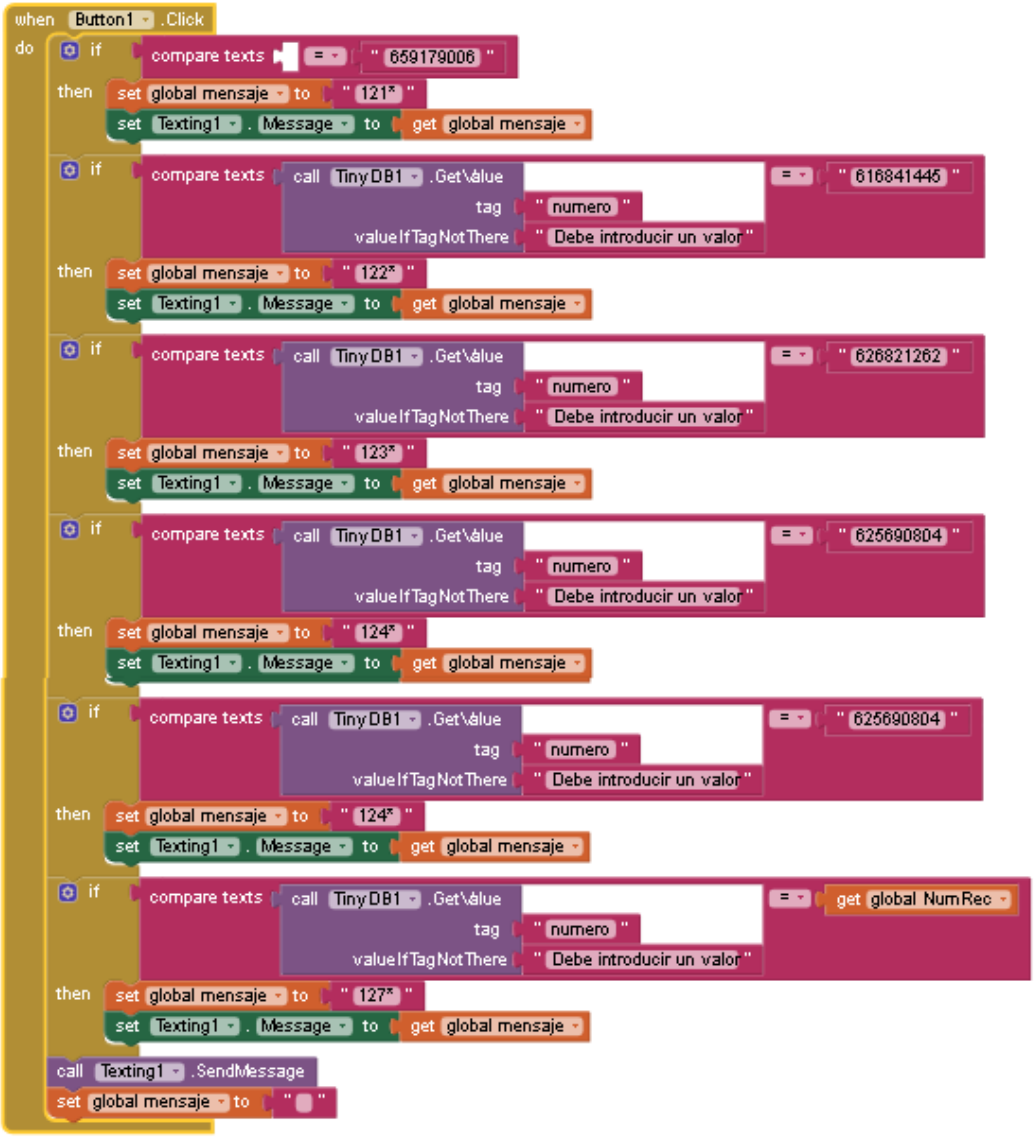
Pantalla Inicial

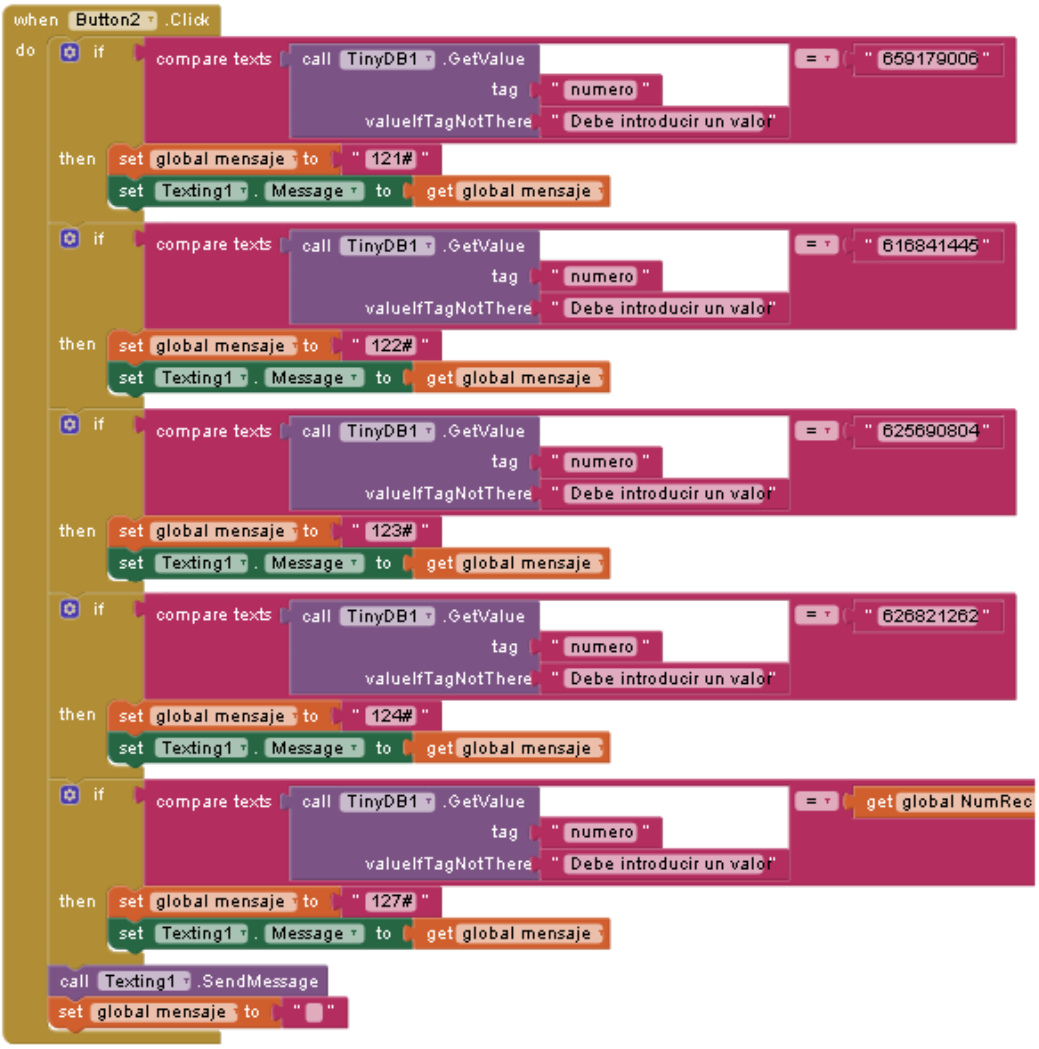



when bto_siguiente .Click
do
if
compare texts
call TinyDB1 .GetValue
tag
" numero "
valueIfTagNotThere
=
" "
and
texto_numero . Text = " "
then
call Notifier1 .ShowMessageDialog
message
" Se ha olvidado de introducir el numero "
title
" Atencion "
buttonText
" Aceptar "
if
compare texts
call TinyDB1 .GetValue
tag
" nombre "
valueIfTagNotThere
=
" "
and
texto_nombre . Text = " "
then
call Notifier1 .ShowMessageDialog
message
" Se ha olvidado de introducir el nombre "
title
" Atencion "
buttonText
" Aceptar "
if
compare texts
texto_nombre . Text
=
" "
then
else
set texto_nombre . Text to texto_nombre . Text
set global nom to texto_nombre . Text
call TinyDB1 .StoreValue
tag
" nombre "
valueToStore
get global nom
if
compare texts
texto_numero . Text
=
" "
then
else
set texto_numero . Text to texto_numero . Text
set global num to texto_numero . Text
call TinyDB1 .StoreValue
tag
" numero "
valueToStore
get global num
if
compare texts
get global respuesta
=
" OK "
or
call TinyDB1 .GetValue
tag
" nombre "
valueIfTagNotThere
≠
" "
and
call TinyDB1 .GetValue
tag
" numero "
valueIfTagNotThere
≠
" "
then
open another screen screenName
" Screen2 "

Pantalla de Activación/desactivación






when Button2 .Click
do
if
compare texts
call TinyDB1 .GetValue
tag "numero"
valueIfTagNotThere "Debe introducir un valor"
= "659179006"
then
set global mensaje to "121#"
set Texting1 . Message to get global mensaje
if
compare texts
call TinyDB1 .GetValue
tag "numero"
valueIfTagNotThere "Debe introducir un valor"
= "616841445"
then
set global mensaje to "122#"
set Texting1 . Message to get global mensaje
if
compare texts
call TinyDB1 .GetValue
tag "numero"
valueIfTagNotThere "Debe introducir un valor"
= "625690804"
then
set global mensaje to "123#"
set Texting1 . Message to get global mensaje
if
compare texts
call TinyDB1 .GetValue
tag "numero"
valueIfTagNotThere "Debe introducir un valor"
= "626821262"
then
set global mensaje to "124#"
set Texting1 . Message to get global mensaje
if
compare texts
call TinyDB1 .GetValue
tag "numero"
valueIfTagNotThere "Debe introducir un valor"
= get global NumRec
then
set global mensaje to "127#"
set Texting1 . Message to get global mensaje
call Texting1 .SendMessage
set global mensaje to ""