TIVA LOCK

MEMORIA

Pablo León Barriga
Javier Moreno Prieto
GIERM 2020

1 ÍNDICE

2 IDEA DEL PROYECTO

Como proyecto para la asignatura se ha querido realizar una implementación relacionada con la domótica, pues es un campo que se encuentra en auge. Explorando distintas posibilidades, surgió la idea de una cerradura inteligente, la cual se adecua al campo mencionado anteriormente, y, además, incorpora un elemento de seguridad, también muy destacado en los últimos años.

Dicha cerradura se plantea como un sistema capaz de almacenar distintos usuarios que deberían identificarse para poder abrir la puerta. Para ello, se pretende utilizar la tecnología de identificación por radiofrecuencia (RFID), de forma que cada usuario poseerá una tarjeta o etiqueta RFID con una ID única que le diferenciará. Además, con el objetivo de aumentar la seguridad, se implementará un sistema de verificación en dos pasos mediante el envío de un correo electrónico al usuario que pretende abrir la cerradura con un código generado aleatoriamente y que deberá introducir para confirmar su identidad.

El sistema será gestionado por un administrador, el cual dispondrá de una ID que le permitirá acceder a funcionalidades exclusivas, principalmente la de agregar usuarios nuevos a la memoria interna del dispositivo.

La interacción usuario-máquina se realizará a través de una pantalla táctil en la que se mostrarán distintos mensajes, menús y un teclado, ofreciendo así una interfaz intuitiva y rápida.

Por tanto, los objetivos a realizar son:

- Uso del ethernet para enviar código por correo por SMTP.
- Verificación en dos pasos: lectura de tarjeta RFID y envío de código generado aleatoriamente al correo electrónico para inicio de sesión.
- Menú administrador para administración de usuarios.
- Teclado en pantalla: para añadir el correo de nuevos usuarios y meter el código de verificación.
- Guardar y leer de memoria flash para no perder los usuarios en caso de reinicio.
- Modo bajo consumo.
- Sonidos: respuesta sonora ante interacciones (lectura correcta/incorrecta de tarjeta, pulsación del teclado, etc).
- Cámara para capturar imagen del usuario que accede.

3 MICROCONTROLADOR TM4C1294

Dentro de los objetivos principales de este proyecto se encuentran la capacidad de poder enviar y recibir información a través de internet, la capacidad de poder controlar una pantalla mediante un boosterpack y de poder conectar un lector RFID y un servomotor para controlar la apertura y cierre de una puerta. El microcontrolador TM4C1294, junto con el LaunchPad EK-TM4C1294XL se ajusta a las necesidades puesto que posee un puerto Ethernet, lo que permite realizar la conexión con los servidores de Gmail para mandar el correo electrónico, y luego posee dos boosterpacks, siendo uno de ellos utilizado en su totalidad por la pantalla VM800, y el otro parcialmente por el servomotor y el lector de tarjetas RFID. Observando la gama de LaunchPads que ofrece Texas Instruments, se considera que este micro se ajusta a las necesidades antes descritas, especialmente a la hora de tener disponibles los pines suficientes para poder realizar la conexión con el lector RFID, componente esencial del sistema automático desarrollado.

Utilizando la herramienta “Memory Allocation” del programa Code Composer Studio, se observa que se utiliza un 38% de la memoria RAM, lo que equivale a 101kB utilizados. La memoria Flash utilizada será de un 10% aproximadamente una vez se introduzca el archivo binario con la información de los bitmaps. En cuanto a los periféricos utilizados, se emplea el periférico Ethernet MAC/PHY para la comunicación, el SSI (synchronous serial interface) para la comunicación con el lector de tarjetas RFID y el control de la pantalla VM800, el PWM para el servomotor, aunque a modo de demostración, y algunos GPIOs para la conexión con el servo y otros pines de los distintos bloques utilizados.

4 DESARROLLO DEL PROYECTO

El proyecto se ha dividido en varios apartados que se han desarrollado independientemente para luego unirlos cuando cada uno de ellos se hubiera completado de forma individual:

- Lectura de tarjetas RFID
- Envío de código de verificación por correo electrónico
- Escritura y lectura de memoria flash
- Interfaz de usuario

A continuación, se desarrollará cada apartado individualmente.

4.1 LECTURA DE TARJETAS RFID

Atendiendo a la definición de la Wikipedia: "RFID o identificación por radiofrecuencia (del inglés Radio Frequency Identification) es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transpondedores RFID". Esta tecnología se ha elegido pues es una forma de identificación fácil y cómoda, donde el usuario no necesita recordar ningún log-in para acceder, simplemente con una etiqueta, tarjeta o el teléfono móvil puede registrarse.

Para la lectura del RFID se utilizará el lector MFRC-522, el cual está basado en el uso de un módulo RC522 RFID. Este permite la lectura de la ID y la transmisión de los datos por SPI, UART e I2C. El dispositivo es popular en la comunidad de Arduino debido a su reducido precio y las distintas librerías existentes que permiten un uso sencillo del mismo. Para el microcontrolador usado escasea la documentación sobre la implementación del lector, únicamente se ha encontrado un proyecto para el launchpad TM4C123G que incorpora una adaptación de una librería de Arduino.

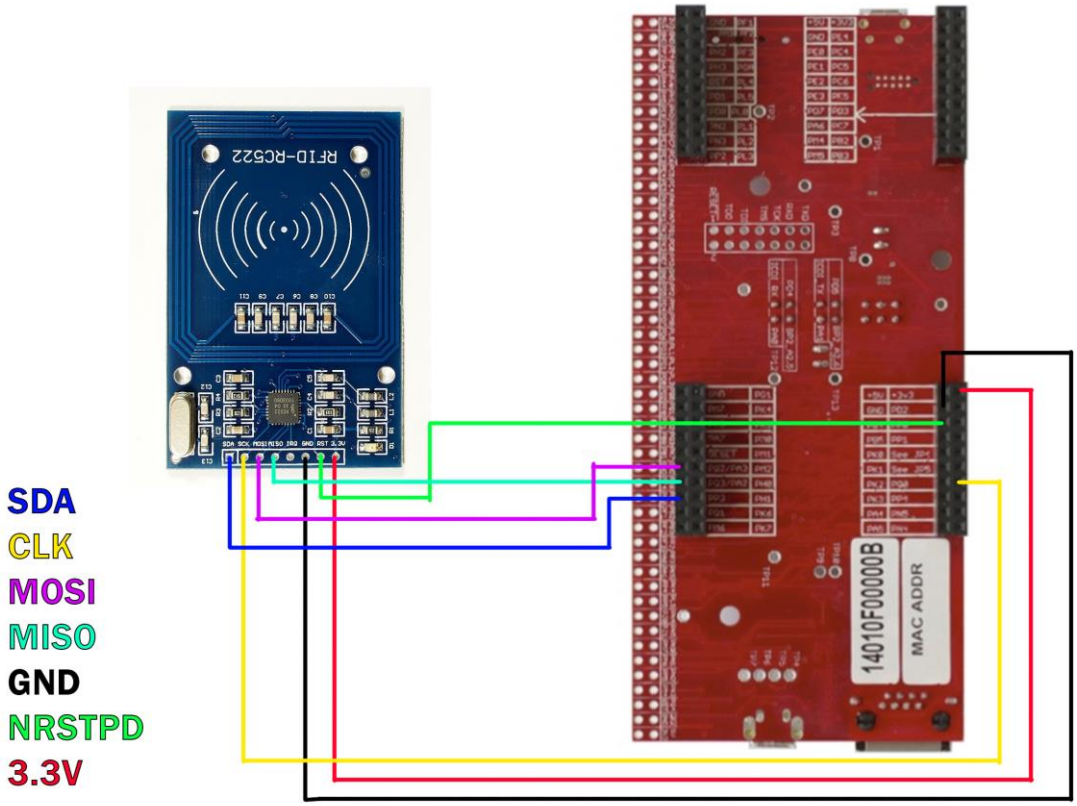
Se partirá de este proyecto para probar el correcto funcionamiento del lector.

El primer paso consistirá en adecuar los pines correspondientes, pues no coinciden con los usados en el microcontrolador para el que fue diseñado. El código utiliza SPI para la comunicación, por lo tanto, se necesitará conectar el MOSI, MISO, CLK y CS, además de un pin para el NRSTPD (reset) y la alimentación y tierra. Consultando el esquema de pines del microcontrolador las conexiones se establecerán como:

- CS: PQ0
- CLK: PQ1
- MOSI: PQ2

- MISO: PQ3
- NRSTPD: PB4

Se han elegido estos pines para el SPI ya que pertenecen todos a la base 3 de SSI, al puerto Q y se encuentran todos en el BoosterPack 2, pues el 1 quedará inutilizado por la pantalla. Una vez elegidos, se deberá cambiar la inicialización de pines y periféricos en el código.



El proyecto del que se ha partido implementaba clases, las cuales no están implementadas en C, únicamente en C++, por lo tanto, se ha tenido que eliminar la clase y adaptarlo para que el funcionamiento fuera el mismo:

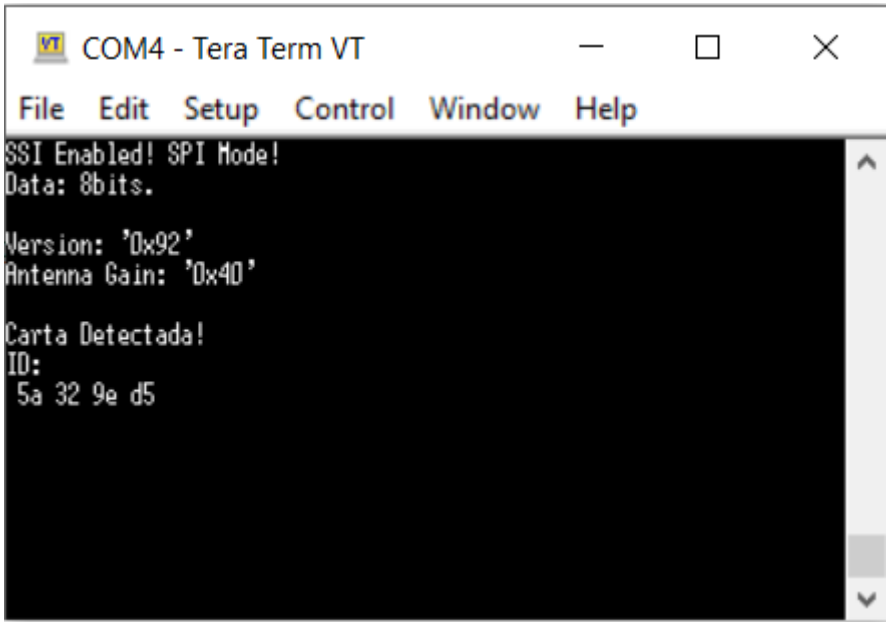
```
class Mfrc522
{
        public:
                Mfrc522(int chipSelectPin, int NRSTPD);
                void WriteReg(unsigned char addr, unsigned char val);
                unsigned char ReadReg(unsigned char addr);
                void SetBitMask(unsigned char reg, unsigned char mask);
                void ClearBitMask(unsigned char reg, unsigned char mask);
                void AntennaOn(void);
                void AntennaOff(void);
                void Reset(void);

                void Init(void);
                unsigned char Request(unsigned char reqMode, unsigned char *TagType);
                unsigned char ToCard(unsigned char command, unsigned char *sendData, unsigned char sendLen, unsigned char *backData, unsigned int *backLen);
                unsigned char Anticoll(unsigned char *serNum);

                void CalulateCRC(unsigned char *pIndata, unsigned char len, unsigned char *pOutData);
                unsigned char SelectTag(unsigned char *serNum);
                unsigned char Auth(unsigned char authMode, unsigned char BlockAddr, unsigned char *Sectorkey, unsigned char *serNum);
                unsigned char ReadBlock(unsigned char blockAddr, unsigned char *recvData);
                unsigned char WriteBlock(unsigned char blockAddr, unsigned char *writeData);
                void Halt(void);
        private:
                int _chipSelectPin;
                int _NRSTPD;
};

#endif
```

```
void Mfrc522(int chipSelectPin, int NRSTPD);
void WriteReg(unsigned char addr, unsigned char val);
unsigned char ReadReg(unsigned char addr);
void SetBitMask(unsigned char reg, unsigned char mask);
void ClearBitMask(unsigned char reg, unsigned char mask);
void AntennaOn(void);
void AntennaOff(void);
void Reset(void);
void RFID_init(void);
unsigned char Request(unsigned char reqMode, unsigned char *TagType);
unsigned char ToCard(unsigned char command, unsigned char *sendData, unsigned char sendLen, unsigned char *backData, unsigned int *backLen);
unsigned char Anticoll(unsigned char *serNum);
void CalulateCRC(unsigned char *pIndata, unsigned char len, unsigned char *pOutData);
unsigned char SelectTag(unsigned char *serNum);

unsigned char Auth(unsigned char authMode, unsigned char BlockAddr, unsigned char *Sectorkey, unsigned char *serNum);
unsigned char ReadBlock(unsigned char blockAddr, unsigned char *recvData);

unsigned char WriteBlock(unsigned char blockAddr, unsigned char *writeData);
void Halt(void);



int _chipSelectPin;

int _NRSTPD;


#endif
```

Como solo se pretende leer la ID de la etiqueta, las únicas funciones usadas serán:

- RFID_init(): inicia el lector y activa la antena.
- Request(reqMode, *TagType): esta función permite leer alguno de los datos de la tarjeta. Para leer la ID se usará como reqMode “PICC_REQIDL”. La función además devuelve “MI_OK” si se ha leído algo y “MI_ERR” en caso contrario.
- Anticoll(str): previene conflictos y copia la ID de la tarjeta a “str”.

De esta forma, al acerca una tarjeta al lector se devuelve una ID de 4 bytes que será almacenada en la memoria interna y permitirá distinguir a cada usuario.



4.2 Envío de código de verificación por correo electrónico

Para enviar el correo electrónico a cada usuario con el factor de autenticación se emplea el protocolo para transferencia simple de correo (Simple Mail Transfer Protocol, SMTP),

protocolo estándar de Internet recogido originalmente dentro de las RFC 821 y RFC 822. Actualmente está recogido en las RFC 5321 y RFC 5322, para la transferencia y el mensaje respectivamente. Dichas RFCs abarcan una amplia variedad de conceptos relacionados con la manera en que deben formarse los mensajes, la estructura de los mismos, los requerimientos del canal de comunicación entre servidor y cliente, entre otros. Para la aplicación deseada dentro del concepto de cerradura inteligente desarrollado en este trabajo, se implementa una versión "reducida" del protocolo, en el que únicamente se envía un cuerpo de texto plano sin adjuntos, que incluye el código de verificación que deberá introducir en la pantalla táctil.

El protocolo SMTP cuenta con una serie de comandos definidos que permite la comunicación entre el host y el cliente. En el caso de la aplicación del TM4C1294, el host se trata de uno de los servidores STMP de Gmail, mientras que el cliente es el mismo microcontrolador. En la siguiente tabla se recoge un pequeño resumen de aquellos comandos necesarios para realizar la comunicación SMTP con el servidor de Gmail.

- HELO o EHLO: una vez establecida la conexión, se le indica al servidor el dominio desde el que se realiza la conexión, a modo de identificación. En la práctica y como se mostrará en el siguiente punto, el servidor de Gmail no requiere de un dominio válido a la hora de identificarse.
- STARTTLS: comando que inicializa la conexión en modo TLS (Transport Layer Security), permitiendo la comunicación encriptada y de forma segura.
- AUTH LOGIN: comando que indica al servidor el deseo de identificar el usuario y contraseña de la cuenta de Gmail desde la que se quiere enviar el correo.
- MAIL FROM: indica la dirección de retorno, es decir, el remitente del mensaje.
- RCPT TO: indica el destinatario del mensaje.
- DATA: en esta parte del intercambio se rellena toda aquella información pertinente al mensaje del correo en sí. Esto incluye cabeceras como las de asunto, remitente, destinatario, hora, y por último, el cuerpo del mensaje. Éste debe finalizarse con un <CRLF>.<CRLF> , secuencia Enter-Punto-Enter.
- QUIT: indica la intención de finalizar la comunicación con el servidor.

Cada servidor puede adaptar este protocolo a sus necesidades, imponiendo la necesidad de realizar la conexión mediante un puerto seguro (TLS o SSL), u obligando al cliente a cumplir una serie de condiciones para poder completar correctamente el protocolo. En el caso de los servidores SMTP de Gmail, es necesario acceder mediante una conexión TLS o SSL. En esta aplicación se opta por utilizar el puerto 587, con protocolo de seguridad TLS. Utilizando la herramienta de cliente Telnet de Windows es posible

realizar el mismo intercambio que se realizará desde la placa, como se muestra a continuación.

Como puede observarse en la siguiente figura, el comienzo de los mensajes del cliente siempre debe ser uno de los indicados anteriormente (a excepción del usuario y la contraseña, codificados en base 64, y la información del mensaje, que permite una mayor flexibilidad), debiendo seguir una secuencia fija en los mismos. Por otro lado, el servidor devuelve un código numérico junto con un mensaje. El código es siempre el mismo, y viene recogido en las RFCs, mientras que el mensaje puede variar de servidor a servidor. Por tanto, es suficiente identificar cuál ha sido el código devuelto por el host para saber si la acción ha finalizado con éxito o no.

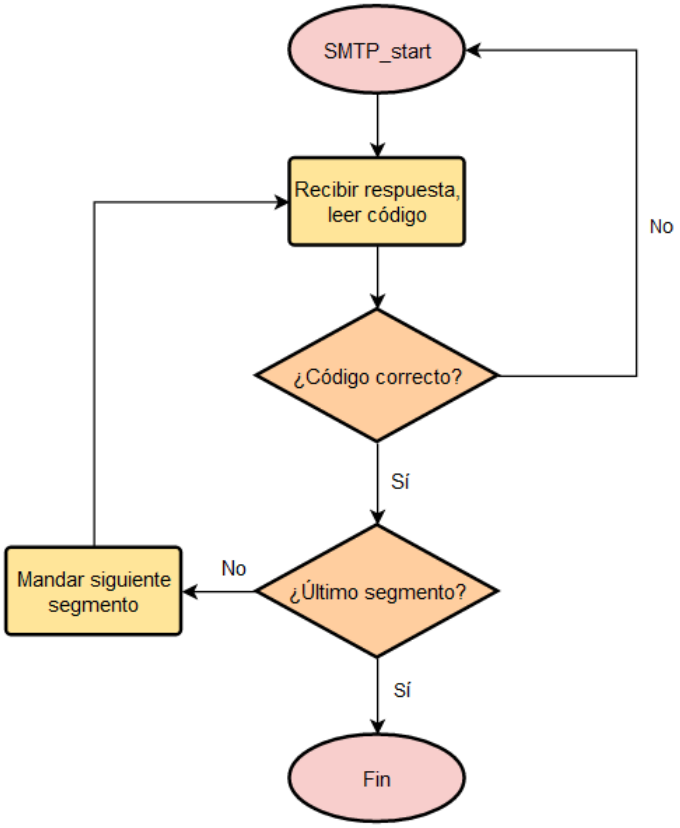
Cabe destacar que el servidor de Gmail devuelve un error de sintaxis cuando se manda el comando STARTTLS, pero esto no influye en el resultado final del intercambio, permitiendo mandar un correo electrónico mediante este procedimiento, siempre que se tenga acceso a la contraseña de una cuenta activa de Gmail.

```
220 smtp.gmail.com ESMTP y130sm33555453wmc.22 - gsmtp
HELO google
250 smtp.gmail.com at your service
STARTTLS AUTH LOGIN
555 5.5.2 Syntax error. y130sm33555453wmc.22 - gsmtp
HELO google
250 smtp.gmail.com at your service
AUTH LOGIN
334 VXNlcm5hbWU6
c21hcnQubG9jay50bTRjQGdtYWlsLmNvbQ==
334 UGFzc3dvcmQ6
dG00Yy5zbTRydC5sMGNr
235 2.7.0 Accepted
MAIL FROM:<smart.lock.tm4c@gmail.com>
250 2.1.0 OK y130sm33555453wmc.22 - gsmtp
RCPT TO:<pablo5leon5@gmail.com>
250 2.1.5 OK y130sm33555453wmc.22 - gsmtp
DATA
354  Go ahead y130sm33555453wmc.22 - gsmtp
Subject: Test email SEPA


Hola,
Esto es una prueba mandada desde el cliente Telnet de Windows.
.
250 2.0.0 OK  1607966517 y130sm33555453wmc.22 - gsmtp
QUIT
221 2.0.0 closing connection y130sm33555453wmc.22 - gsmtp
```

Para el código se ha optado por utilizar la librería orientada al stack TCP/IP “lightweight IP”, “lwIP”, desarrollada por el Instituto Sueco de Ciencia de la Computación, se programa una librería encargada de enviar correos electrónicos desde una cuenta de

Gmail creada para esta aplicación. Dentro de esta librería se incluyen todas aquellas funciones necesarias para establecer una conexión TCP/IP básica, incluyendo la función que es llamada cada vez que se recibe algún paquete desde internet.

El siguiente esquema muestra el diagrama de flujo básico del programa:



Dentro de la máquina de estados del programa principal existe un contador de reintentos, que garantiza que se intente enviar el correo al menos en tres ocasiones. Si esos tres intentos terminan fallando, se mostrará un mensaje por pantalla alertando del error a la hora de entregar el código de verificación, y se regresará al estado inicial, en el que se solicita una tarjeta.

La librería SMTP_library.h contiene las siguientes funciones, que, basándose en la librería lwIP permite enviar los correos electrónicos a los usuarios de la cerradura inteligente.

- lwIPHostTimerHandler: función requerida por la librería lwIP, se encarga de obtener la dirección IP local del microcontrolador.
- close_conn: encargada de cerrar la conexión con el servidor, ya sea porque se ha enviado el correo correctamente o porque se ha producido algún error.

- comunicacion_SMTP: función de tipo “callback”, que es llamada cada vez que se reciben datos desde internet. Dentro de la misma se encuentra el código necesario para realizar el protocolo SMTP con el servidor de Gmail. Esto se consigue mediante un switch case, que contiene cada una de las fases mostradas en la figura del ejemplo Telnet de Windows.
- connected_callback: función que es llamada una vez se establece la conexión con el servidor.
- config_SMTP: función que configura el microcontrolador para que esté preparado para realizar una conexión con un servidor.
- SMTP_start: función que realiza la conexión con el servidor SMTP de Gmail, y configura las funciones de “callback” descritas anteriormente.

Las funciones utilizadas de la librería lwIP se pueden dividir en dos bloques distintos: aquellas funciones dirigidas a configurar y establecer la conexión, y aquellas funciones encargadas de mandar la información al servidor.

Funciones encargadas de configurar el microcontrolador.

- tcp_new: crea un nuevo identificador de conexión TCP.
- tcp_bind: enlaza la placa creada con tcp_new con el puerto y dirección deseada.
- tcp_connect: realiza la conexión con el servidor deseado.
- tcp_recv: establece la función “callback” que será llamada cuando se reciben datos.

Funciones encargadas de tratar los datos recibidos y mandar la respuesta.

- tcp_recved: debe ser llamada cuando se ha procesado la información recibida.
- tcp_write: escribe información en el buffer de salida de la pila TCP/IP. Realiza un mecanismo de optimización, esperando a recibir más datos para almacenarlos antes de enviarlos al servidor. Para evitarlo, se utiliza la siguiente función.
- tcp_output: fuerza el envío de la pila TCP, permitiendo de esta manera elegir cuándo mandar el mensaje al servidor SMTP.
- pbuf_free: se encarga de eliminar la cadena una vez se ha producido el envío de información.

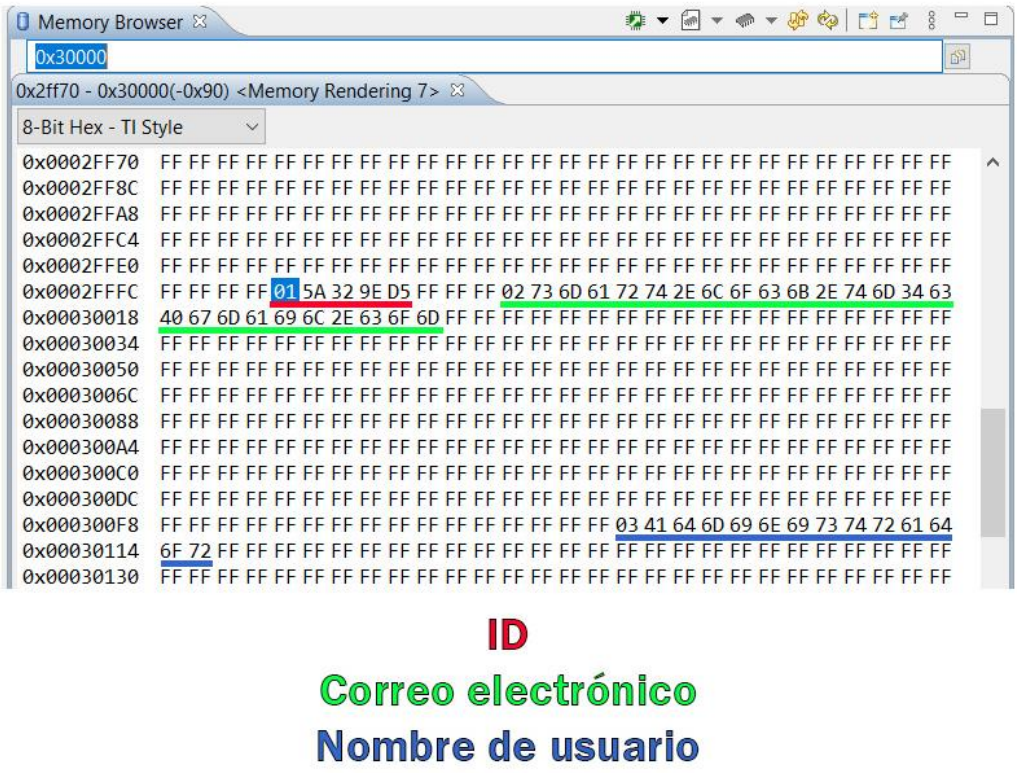
4.3 ESCRITURA Y LECTURA DE MEMORIA FLASH

4.3.1 Usuarios

Tal y como se plantea el proyecto, es necesario almacenar la ID y el correo electrónico de cada usuario. Si se guardara esta información en variables, se corre el riesgo de que el sistema pueda apagarse/reiniciarse, lo que llevaría a la pérdida de los datos. Por ese

motivo, se decide escribir estos datos en la memoria flash del microprocesador, la cual no se elimina tras un reinicio.

El microprocesador TM4C1294 tiene una memoria flash de 1024 kB, parte de la cual es ocupada por el código en sí. Se realizan distintas pruebas y usando el "Memory browser" se observa que la mayor parte de la flash resulta no ser escrita por el programa. Se decide que la escritura de los datos de los usuarios se realizará a partir de la posición de memoria 0x30000, y se seguirá la siguiente estructura:



Se tendrán tres campos guardados por cada usuario, cada uno de ellos con un indicador de "01", "02" y "03" como primer byte:

- ID: a partir del byte "01", se guardará en los cuatro siguientes cada byte de la ID leída del RFID.
- Correo electrónico: se almacenará el "02" 8 posiciones después del "01" y partir de ahí se escribirán los caracteres correspondientes a la dirección del usuario. Se reservarán 256 bytes de memoria para el correo, pues es el máximo número de caracteres permitidos según la RFC 5321.
- Nombre de usuario: comenzando por la posición 264 desde el inicio, se escrbirá el byte con "03" y le seguirán los caracteres del nombre dado.

Para cada usuario se reservarán 512 bytes de memoria, de forma que para navegar por los distintos usuarios registrados se podrán realizar saltos entre estas posiciones de manera fácil.

La escritura en la memoria flash se realiza gracias a la librería “flash.h”. Esta dispone de la función “FlashProgram”, la cual tiene como entradas el puntero de la variable a guardar, la posición en la que se quiere escribir y el tamaño de la variable. Estos parámetros deben ser de tipo “uint32_t”, lo que conlleva que cada uno de los datos que se guarden ocuparán 4 bytes de memoria, lo que puede suponer un despedicio de memoria pues si por ejemplo se quiere guardar un único carácter, el cual ocupa únicamente un byte, se estarán desaprovechando los 3 bytes de memoria sobrantes. Para una mejor optimización del espacio, se recurre al uso de operadores de desplazamiento, lo que permite que se almacenen 4 vairables de tipo “uint8_t” en la de “uint32_t” requerida:

Sin operadores de desplazamiento:

```
0x0000005A 0x00000032 0x0000009E 0x000000D5
```

Con operadores de desplazamiento:

```
0x5A329ED5 0xFFFFFFFF 0xFFFFFFFF 0xFFFFFFFF
```

Una vez resuelto el problema de escribir en la flash, es consecuente desarrollar el método de lectura de la memoria. Para ello se recurrirá a la librería “hw_types.h”, la cual dispone de una macro que será de gran utilidad: “HWREGB()”. Esta función tiene como entrada la posición de memoria que se quiere leer, y devolverá un “uint8_t” con el byte leído en la posición especificada.

Una vez conocidas las herramientas necesarias, se desarrollan distintas funciones para el manejo de la flash:

- EncontrarEspacio(): recibiendo como entrada una posición de inicio, comprobará si esa posición de memoria está ya escrita (lo que se lea sea distinto de 0xFF). En caso de que así sea, sumará 512 bytes a la posición inicial, de forma que se avance a la siguiente posición de memoria reservada para un nuevo usuario. Este proceso se repetirá hasta encontrar un espacio disponible, en cuyo caso se devolverá la posición de memoria encontrada para almacenar un nuevo usuario.
- GuardarUSUARIO(): dada una posición inicial de memoria, realizará la escritura en la flash de la ID, correo y nombre del usuario que acabe de realizar el registro.

Para ello se hará uso de los operadores de desplazamiento y saltos de memoria explicados anteriormente.

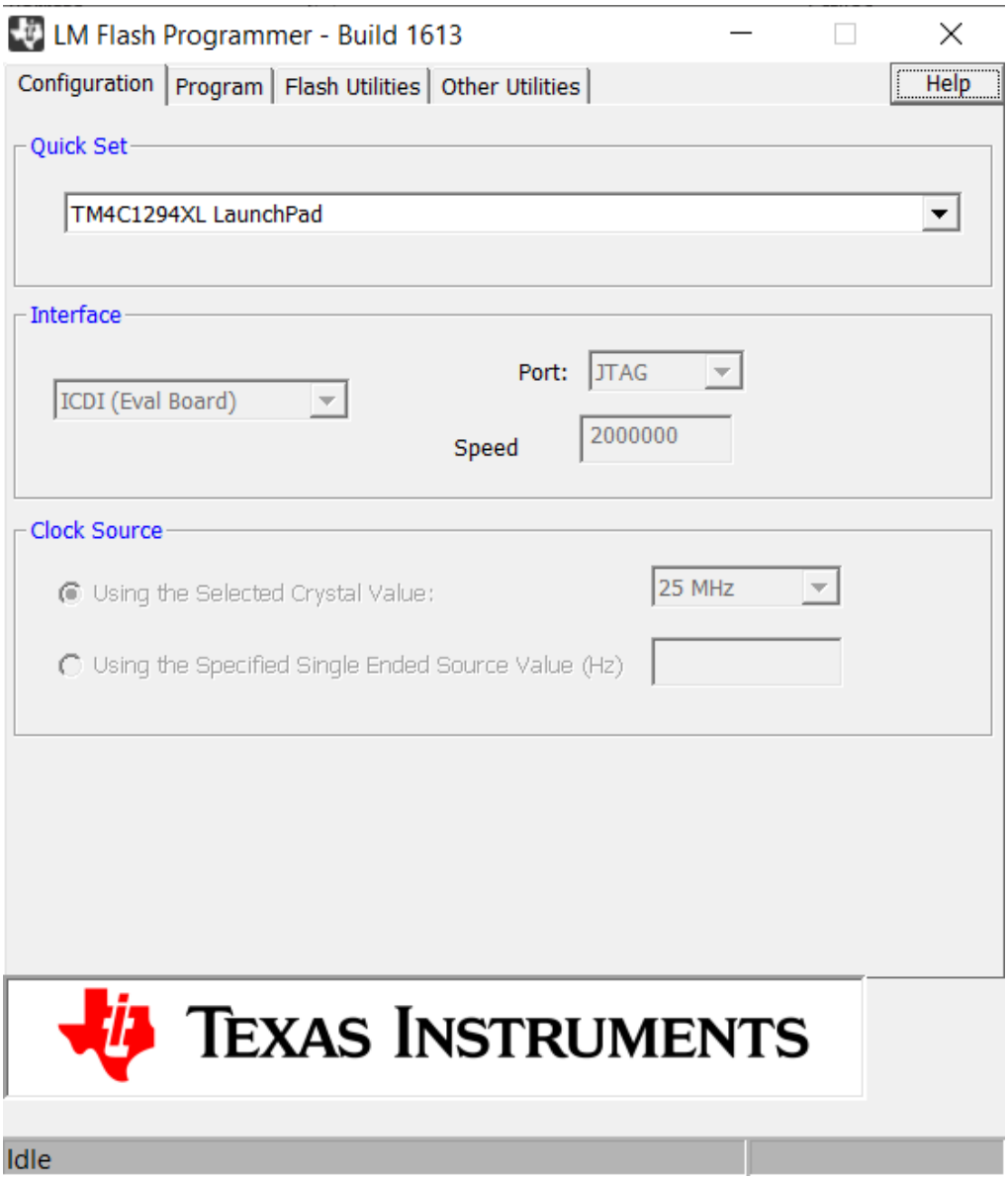
- GuardarADMIN(): su funcionamiento es similar a la función anterior con la única diferencia de que la ID, correo y nombre se especifica dentro del código, no es introducido una vez cargado el programa, pues se trata del administrador del sistema.
- ObtenerCORREO() y ObtenerNOMBRE(): estas funciones copian en un buffer el correo y el nombre respectivamente del usuario especificado recibiendo como entrada la posición inicial de este.

Combinando estas funciones, se habrá completado por tanto el almacenaje y lectura de usuarios en la flash del microprocesador.

4.3.2 Bitmaps

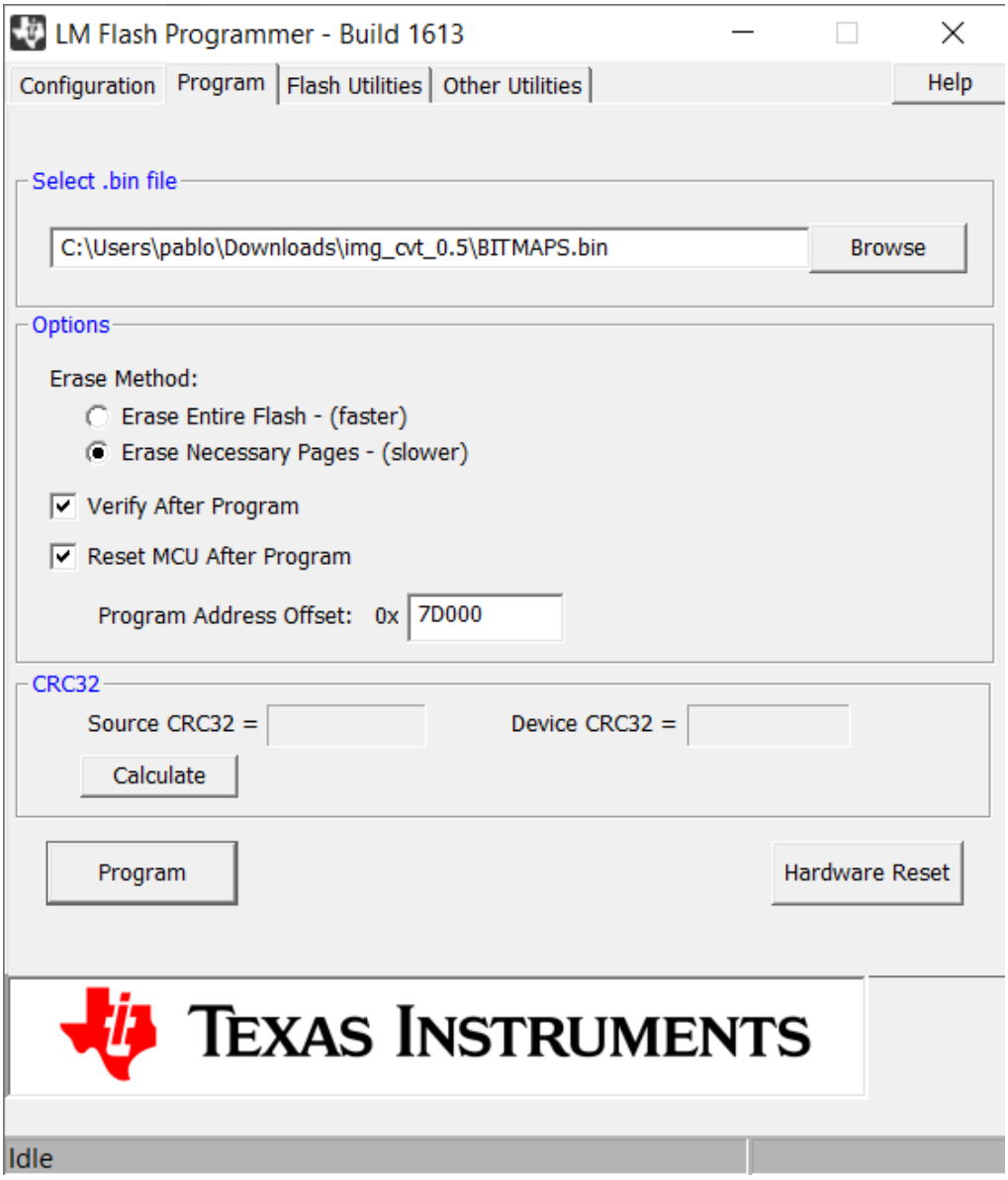
Para mejorar la interfaz de usuario, se opta por utilizar imágenes cargadas en el microcontrolador en lugar de líneas de texto plano. Puesto que tener las imágenes precargadas en el código no es una opción viable, se opta por utilizar la herramienta proporcionada por Texas Instruments LM Flash Programmer. Esta herramienta permite cargar ficheros binarios dentro de la memoria flash del microcontrolador sin necesidad de perder el programa ya incluido en el mismo, siempre que se escoja una posición de memoria adecuada, que no sobre escriba datos ya almacenados del programa.

De todas las opciones ofrecidas por el programa conviene utilizar la pestaña de Program, una vez se ha seleccionado el LaunchPad TM4C1294XL en la pestaña de configuración, como se muestra a continuación.



Una vez seleccionado, se pasa a la pestaña Program, eligiendo el archivo de tipo binario que se quiere volcar en la memoria. Este archivo binario se crea mediante la herramienta img_cvt proporcionada por el fabricante del FT800, que transforma una imagen en formato png o jpeg en un archivo binario que puede ser cargado posteriormente en la memoria RAM de la pantalla. También es necesario seleccionar dentro las opciones el método de borrado que se desea hacer, seleccionando en este caso la opción de eliminar únicamente las páginas necesarias, ya que de otra manera se eliminaría todo el contenido de la memoria Flash. Se seleccionan las opciones de Verificar después de programar y de Reiniciar el MCU después de programar, y por último se selecciona la

posición en memoria que se quiere guardar, en este caso se escoge la posición 0x7D000 (512 000 en decimal), posición lo suficiente alejada del almacenamiento del programa base, y de todos aquellos datos que se introduzcan para guardar nuevos usuarios.



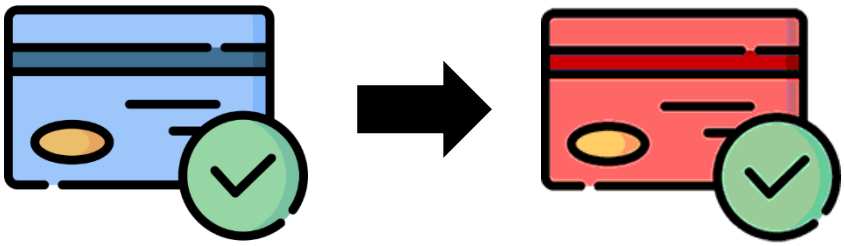
Para finalizar debe programarse sobre el microcontrolador el archivo binario. Una vez cargado, y para poder ser usados por la pantalla, las imágenes deben estar almacenadas en la RAM del FT800 para ser leídas. Con este propósito en mente se desarrollan dos funciones para almacenar en la memoria RAM la imagen deseado, leyendo desde la memoria Flash del microcontrolador, y posteriormente mostrarla por pantalla mediante la opción de Bitmaps. A continuación, se explican las dos funciones utilizadas:

- almacenar_imagen_externa: leer de la memoria flash del microcontrolador byte a byte toda la información correspondiente a cada imagen. Esta función debe ser

llamada cada vez que se quiere cargar alguna imagen nueva en la memoria RAM del chip, ya que ésta se sobreescribe con cada imagen.

- dibujar_imagen_externa: esta función llama a otras subfunciones creadas para poder dibujar un bitmap en la pantalla. Dichas funciones son:
  - cmd_bmp_source: manda la dirección RAM en la que se quiere almacenar el bitmap
  - cmd_bmp_layout: comando correspondiente a BITMAP_LAYOUT.
  - cmd_bmp_size: comando para BITMAP_SIZE. Guarda el ancho y alto de la imagen.
  - cmd_alpha_a: comando para indicar la transparencia del bitmap una vez se muestre por pantalla.

Debido a que al usar el display de 5' la información se manda en orden B→G→R, al mostrar los bitmaps por pantalla el rojo se interpretará como azul y viceversa. Para mantener los colores originales se han editado previamente con el software Photoshop. También se han ajustado el tamaño en pixel para adaptarlos a las dimensiones de la pantalla.



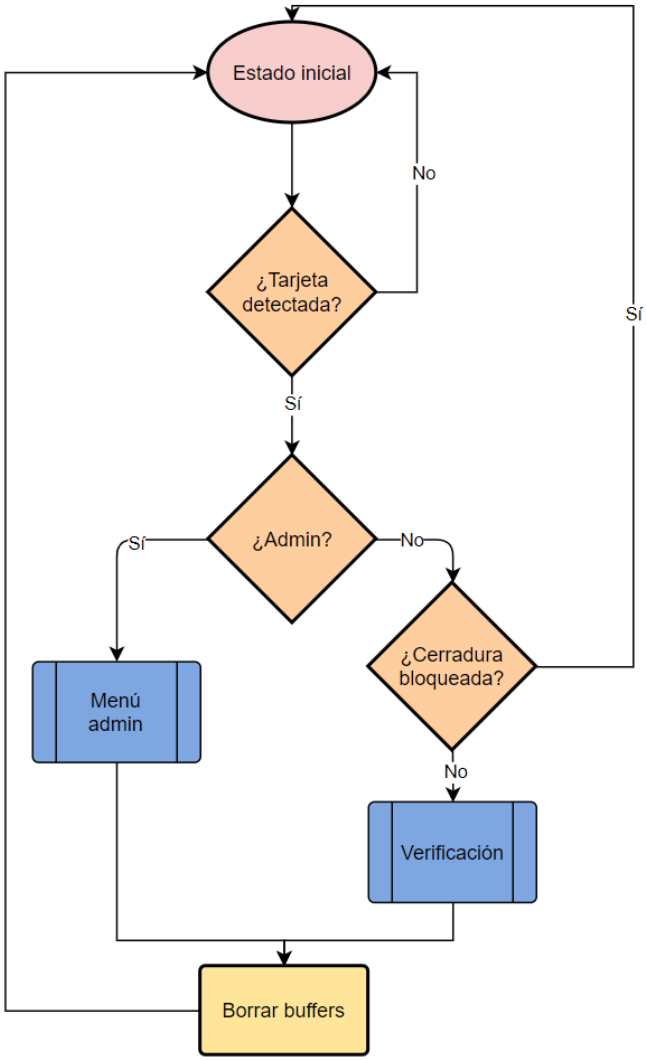
Las imágenes que se han incluido son (ya editadas):

4.4 INTERFAZ DE USUARIO

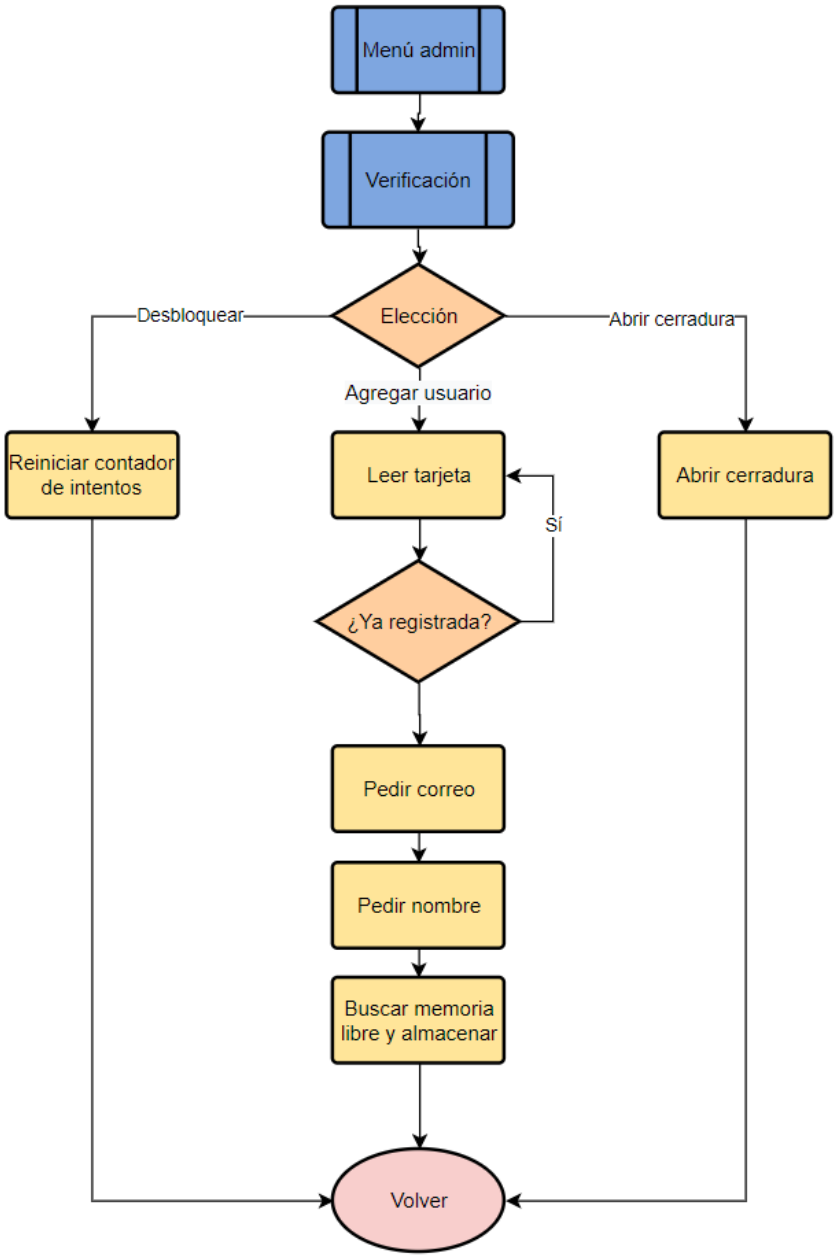
Una vez desarrollado cada bloque de forma individual, el próximo objetivo será unificarlos todos comprobando que no interfieren entre ellos y desarrollar una interfaz de usuario simple y funcional, que resulte en una cerradura inteligente segura y fácil de usar.

4.4.1 Máquina de estados

El primer paso a realizar será diseñar la máquina de estados que gobernará el funcionamiento del programa:

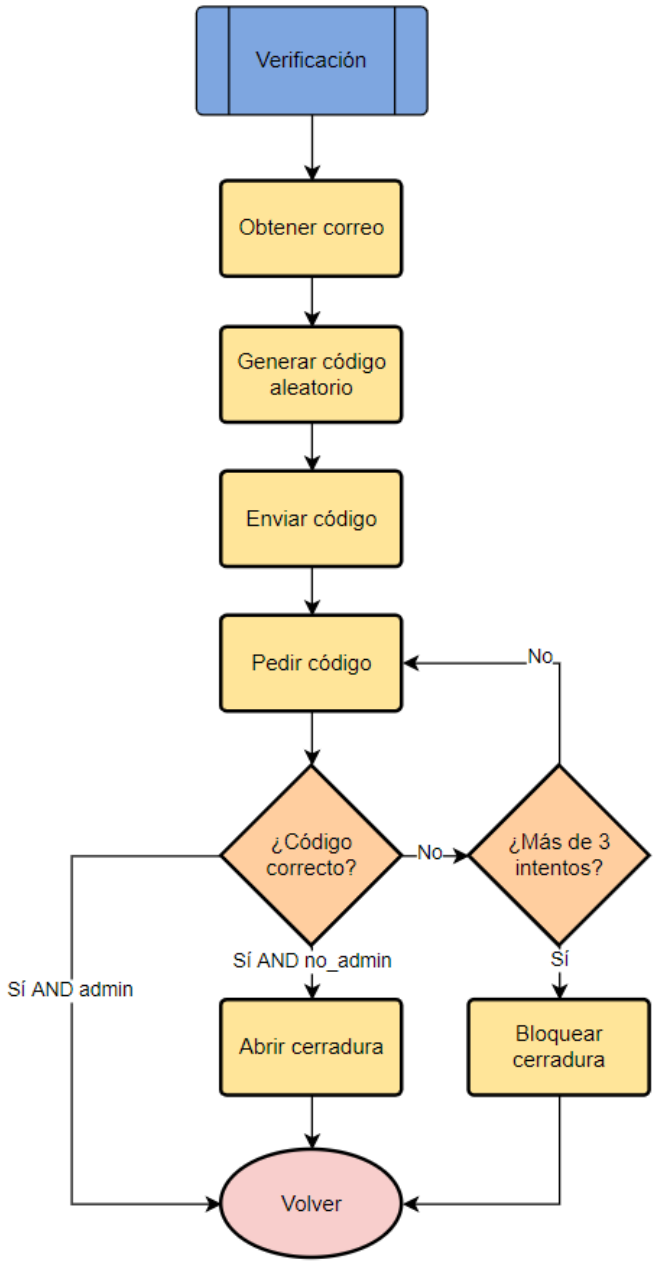


La estructura principal comenzará con la lectura de una tarjeta RFID, hasta que esto no se produzca, no se podrá interactuar con el sistema. Una vez detectada, se determinará si el administrador es quien intenta entrar o se trata de un usuario normal. En función de esto, se llegará a los procesos “menú admin” o “verificación”, que veremos a continuación.



En el menú admin habrá tres opciones:

- Agregar usuario: registra un nuevo usuario y lo almacena en la flash.
- Abrir cerradura: llama al proceso de verificación.
- Desbloquear: esta opción únicamente estará activa si la cerradura ha sido bloqueada por un exceso de intentos.



El proceso de verificación mandará el código tal y como se ha explicado en apartados anteriores y abrirá la cerradura o la bloqueará en su defecto.

Siguiendo esta filosofía de diseño, se ha implementado en el main un “switch case” como máquina de estados. La función de cada estado se ha detallado en el código del programa.

4.4.2 Pantalla táctil

La interfaz entre el usuario y la cerradura inteligente se realiza mediante una pantalla táctil que permita al usuario interaccionar con el sistema. Esto se consigue mediante el sistema embebido VM800b50, que incluye el chip FT800 EVE (Embedded Video Engine), que permite, gracias a los bitmaps creados mediante el programa img_cvt de FTDIChip

y almacenados en la memoria Flash del microcontrolador, generar una interfaz clara y concisa que permita al usuario comunicarse con el lector de tarjetas NFC y permita acceder correctamente.

El código desarrollado para estar parte del sistema se apoya en las librerías desarrolladas por el profesor de la asignatura (ft800_TIVA.h), así como del manual de programadores disponible en línea. Se han desarrollado tres funciones fundamentales, que recogen todas aquellas funcionalidades deseadas para el funcionamiento de la interfaz. Estas funciones incluyen tanto mostrar por pantalla un teclado como recoger aquello que el usuario escriba en el mismo.

- pantalla_teclado: función que llama a las subfunciones encargadas de mostrar por pantalla y recoger la información del usuario, hasta que se pulse la tecla de ENTER.
- get_teclado: se rellenan las cadenas de correo, nombre/apellidos o clave, en función del estado en el que se desee estar.
- pantalla: se encarga de mostrar por pantalla tanto el teclado como el texto ya introducido por el usuario hasta ese punto. Utiliza comandos como cmd_keys o cmd_button para mostrar por pantalla las filas de teclas, y los botones de Intro, Borrar, Bloqueo de Mayúsculas y de Caracteres Especiales.

Aprovechando las funcionalidades del display, se han añadido también cuatro efectos de sonido:

- Inicio del sistema: la animación de encendido va acompañada de un sonido.

- Tarjeta reconocida: dos notas ascendentes que indican una lectura correcta.
- Tarjeta no reconocida: dos notas descendentes que indican un fallo en la lectura.
- Respuesta táctil del teclado: para dar una sensación de pulsación en el teclado se ha introducido un sonido de click al presionar las teclas.

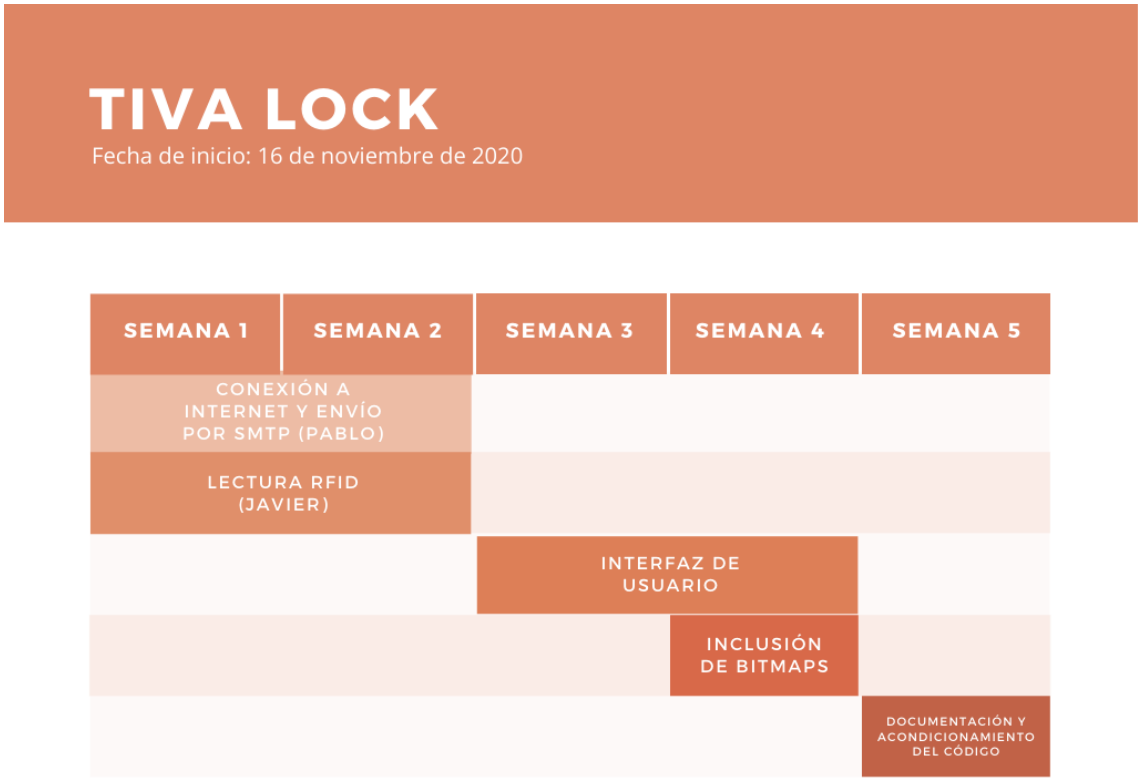
4.5 EXTRAS

Como añadido para una mejor demostración del funcionamiento de la cerradura, se ha añadido un servo y se ha construido una maqueta de una puerta que será abierta tras completar un inicio de sesión.

Para la programación del servo se ha partido del código desarrollado en las prácticas de la asignatura, conectando la señal de control a PG1.

Cabe destacar que este apartado no pretende representar como sería el cerrojo de la puerta en caso de una implementación real del sistema, solo se usa como añadido visual para la demostración del proyecto, por lo que no se entrará en detalle.

5 CRONOGRAMA DEL PROYECTO



Este proyecto ha sido realizado en pareja por Pablo León Barriga y Javier Moreno Prieto. La idea del proyecto surgió el día 16 de noviembre, se realizó un esquema con los requisitos que debería cumplir y medios que harían falta para desarrollarlo, y se determinó que aquello que requería mayor investigación sería el envío de correos por SMTP y la lectura del RFID. Se decidió dividir el trabajo para avanzar a mayor velocidad, siendo Pablo quien comenzó el desarrollo de la conexión a internet y Javier el reconocimiento de las etiquetas. A pesar de la división de tareas se hicieron constantes actualizaciones y explicaciones recíprocas sobre el avance de cada miembro en su campo.

Una vez terminadas las dos partes, se pasó al desarrollo del teclado en pantalla y la escritura/lectura de la memoria flash. Para estas tareas se trabajó en común utilizando GitHub para la gestión de los archivos y Discord para la comunicación por voz, agilizando así el trabajo.

Tras completar estos dos nuevos apartados, comenzó la unificación de todas las partes en un único proyecto. Para ello, se desarrollaron distintas librerías para facilitar el manejo de las distintas partes y se diseñó la máquina de estados que coordinaría todos los procesos.

Una vez el sistema cumplió todas las funcionalidades requeridas, se centraron los esfuerzos en hacer una interfaz simple y atractiva, por lo que se comenzó la investigación del uso de imágenes en la pantalla.

Finalmente, se procedió a la limpieza y comentado del código y al desarrollo de la documentación a entregar.

6 RESULTADO FINAL

Los objetivos que se propusieron al inicio del proyecto fueron los siguientes:

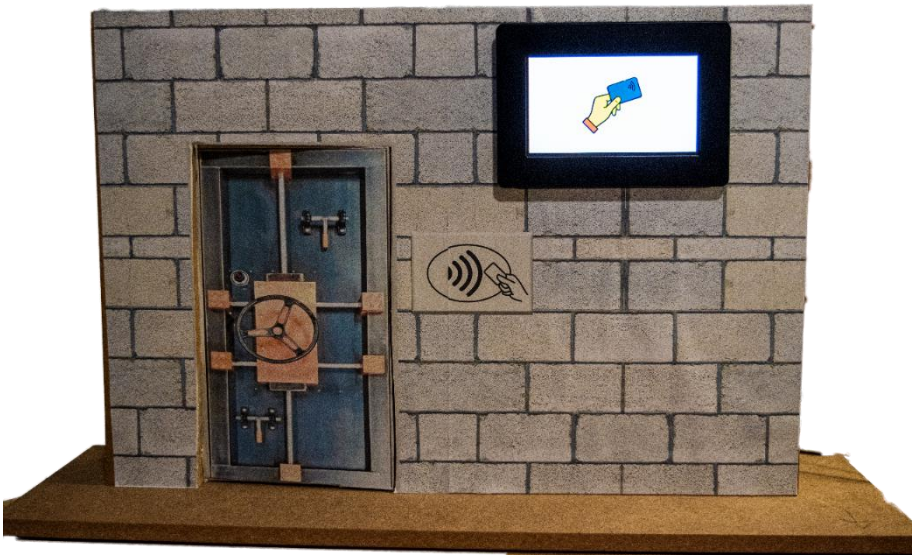
- Uso del ethernet para enviar código por correo por SMTP.
- Verificación en dos pasos: lectura de tarjeta RFID y envío de código generado aleatoriamente al correo electrónico para inicio de sesión.
- Menú administrador para administración de usuarios.
- Teclado en pantalla: para añadir el correo de nuevos usuarios y meter el código de verificación.
- Guardar y leer de memoria flash para no perder los usuarios en caso de reinicio.
- Modo bajo consumo.
- Sonidos: respuesta sonora ante interacciones (lectura correcta/incorrecta de tarjeta, pulsación del teclado, etc).
- Cámara para capturar imagen del usuario que accede.

El sistema finalmente desarrollado cumple con todos los puntos que fueron propuestos al inicio del proyecto menos el de la cámara, ya que no se ha encontrado un periférico que se adhiera a las características del proyecto.

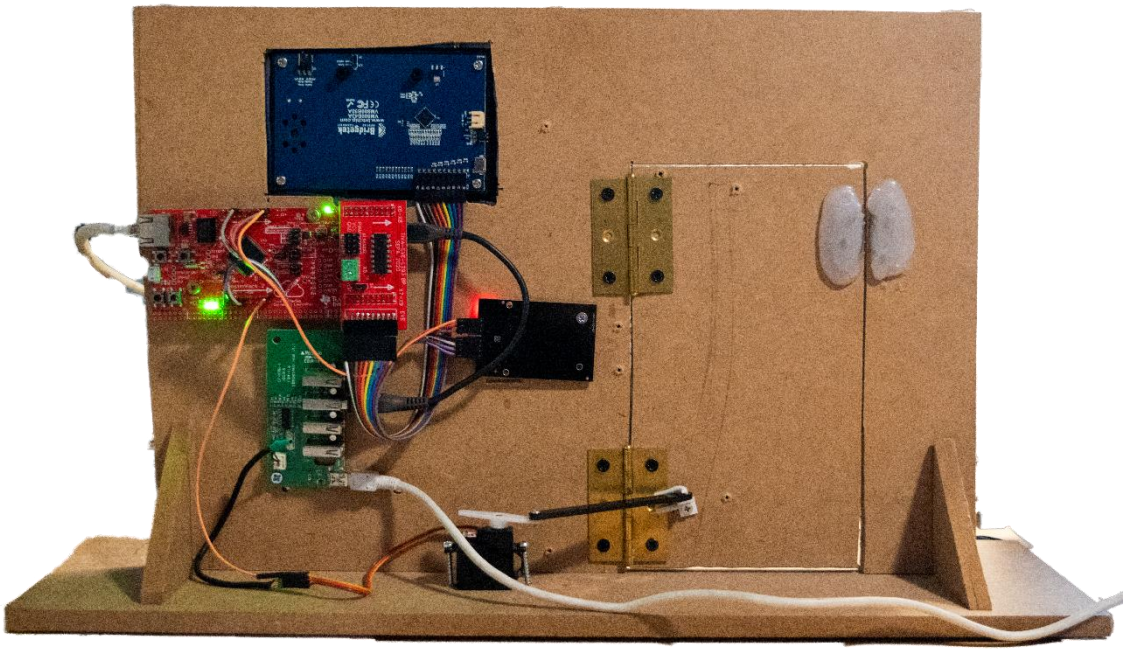
- ✓ El sistema es capaz de realizar una conexión con un servidor SMTP de Gmail, mandando a cada usuario que trata de acceder un código de verificación generado aleatoriamente utilizando como semilla de este el tiempo que ha transcurrido desde que se inició el microcontrolador, haciendo altamente improbable que se den dos códigos iguales.
- ✓ La posibilidad de mandar correos electrónicos permite implementar ese factor de verificación en dos pasos que se propuso al inicio.
- ✓ Se cuenta con un menú administrador que permite introducir nuevos usuarios o abrir la cerradura. Además, en el caso de que el sistema esté bloqueado porque se haya producido más intentos fallidos de los permitidos, saldrá la opción de desbloquear la cerradura, permitiendo volver al funcionamiento normal.
- ✓ Se ha desarrollado una interfaz gráfica con teclado en pantalla táctil, que permite a los usuarios introducir el factor de verificación que reciben mediante correo electrónico.
- ✓ Se tiene un sistema con la capacidad de almacenar en una memoria no volátil la información pertinente a los usuarios de la cerradura inteligente, de manera que, si se produjese algún tipo de pérdida de alimentación, dicha información permanece en el microcontrolador sin perderse.

- ✓ A la hora de esperar a que se acerque una nueva tarjeta RFID, el sistema entra en un modo de bajo consumo, del que despierta periódicamente para comprobar si hay alguna tarjeta que leer. En caso negativo, regresa al modo de bajo consumo.
- ✓ La misma pantalla táctil posee un pequeño altavoz que emite una serie de sonidos en función de la acción realizada.
- ✗ La cámara no se pudo implementar ya que no se encontró un periférico que se ajustase al diseño. Además, debido a que se quiere que el sistema sea independiente de un ordenador, la posibilidad de adjuntar una fotografía mediante el microcontrolador y el protocolo SMTP se tornó más complicado de lo esperado.

Por otra parte, se decide introducir la muestra de imágenes precargadas en la flash del microcontrolador, para hacer más amigable la interfaz usuario-máquina, algo que no era uno de los objetivos iniciales pero que se pudo implementar mediante LM Flash Programmer.



En la imagen puede verse el resultado final del sistema. Este se ha montado sobre una plancha de DM, con los componentes electrónicos al otro lado del tablero. En la parte delantera se muestra únicamente la pantalla táctil, y una pegatina con el logo RFID para indicar al usuario la posición del lector. Por otro lado, se tiene una puerta para ilustrar lo que sería el mecanismo completo del sistema.



En la parte trasera del sistema se tiene el LaunchPad EK-TM4C1294XL, el lector de tarjetas RC522 y la pantalla VM800B50, de 5 pulgadas. Adicionalmente se ha incluido un “hub” externo (color verde), conectado a la salida USB de un portátil, de manera que la alimentación del servomotor y del microcontrolador son independientes, lo que evita el reinicio de este último cuando se manda la señal de control al servomotor, que requiere de una corriente superior a la que el micro puede aportar.

7 BIBLIOGRAFÍA

Ethernet y SMTP

Librería ethernet energía https://energia.nu/guide/libraries/ethernet/

Ejemplo cliente SMTP: https://www.codeproject.com/Articles/28806/SMTP-Client

Video ejemplo telnet SMTP: https://www.youtube.com/watch?v=Hv3Ueqnbvd8

Github del ejemplo de lwip para cliente SMTP: https://github.com/mongoose-os-libs/lwip/blob/master/lwip/src/apps/smtp/smtp.c

RFID

Tutorial RC522 para Arduino: https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/

Librería RC522 adaptada a TM4C123: https://github.com/rafaelbrier/RC522-with-Tiva-C-tm4c123-CCS-

FT800

Manual de FT800: https://www.ftdichip.com/Support/Documents/ProgramGuides/FT800%20Programmers%20Guide.pdf

Imágenes

Iconos usados: https://www.flaticon.com/

8 ANEXO

8.1 CÓDIGOS

8.1.1 MAIN_TIVA_LOCK.c

```
// ===================================================================
//                              TIVA LOCK
// ===================================================================
//                            Creado por:
//                         Pablo León Barriga
//                        Javier Moreno Prieto
// ===================================================================

#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"

#include "driverlib2.h"
#include "drivers/pinout.h"
#include "lwip/tcp.h"

#include "utils/locator.h"
#include "utils/lwiplib.h"
#include "utils/ustdlib.h"

#include "smtp_library.h"
#include "variables_globales.h"
#include "FT800_TIVA.h"
#include "Mfrc522.h"

// ===================================================================
// Variable Declarations
// ===================================================================

char chipid = 0;                        // Holds value of Chip ID read from the
FT800

unsigned long cmdBufferRd = 0x00000000;         // Store the value read from the
REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;         // Store the value read from the
REG_CMD_WRITE register
unsigned int t=0;

//
#############################################################################
############################
// User Application - Initialization of MCU / FT800 / Display
//
#############################################################################
############################

int Fin_Rx=0;
```

```
char Buffer_Rx;
unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
const int32_t REG_CAL[6]={21696,-78,-614558,498,-17021,15755638};
const int32_t REG_CAL5[6]={32146, -1428, -331110, -40, -18930, 18321010};

#define NUM_SSI_DATA            3


//
#############################################################################
############################
// Variables / funciones nuestras
//
#############################################################################
############################

// Variables para el RELOJ
int RELOJ;

// Un milisegundo con el RELOJ de 120MHz (40000*3)
#define MSEC 40000


// Funciones de configuración de periféricos, interrupciones y otros módulos
utilizados
void config_RC522();
void config_PWM();




// Funciones relacionadas con la lectura y funcionamiento del RFID
void RFID();

// Funciones relacionadas con la lectura/escritura de la memoria Flash
int EncontrarEspacio(int posicion);
void GuardarADMIN(int posicion);
void   GuardarUSUARIO(int   posicion,   unsigned   char*   ID_usuario,   char*
correo_usuario, char* nombre_usuario);
int ComprobarID(int posicion);
void ObtenerCorreo();
void ObtenerNombre();
void BorrarBufferCorreo();
void BorrarBufferNombre();
void BorrarBufferClave();
void BorrarBufferEscritura();
void BorrarBufferEscrituraCorreo();
void BorrarBufferEscrituraNombre();

// Funciones relacionadas con el temporizador
void config_timer1(int Reloj);
void interrupcion_timer1(void);
void EsperaSeg( int segundos);


// Variables PWM
volatile int Max_pos = 4400; //3750
volatile int Min_pos = 1300; //1875
int PeriodoPWM;
```

```
// Otras variables
int tarjeta_detectada=0;
char TXT_1[100];
bool clave_correcta;
unsigned long aleatoriedad=0;
int contador_intentos=0;
int resp = 0;
int dib;
int tiempo = 0;
int estado=0;
int numero_usuario = 0; // contador de que usuario se ha registrado
bool bloqueo = 0;

//--------------------------------------------------------------------------
------------------\\
//---------------DEFINES / VARIABLES PARA LA PARTE DE CORREO EN FLASH / SMTP---
-----------------\\
//--------------------------------------------------------------------------
------------------\\

// Saltos en bytes para almacenar en memoria
#define SaltoID 8
#define SaltoCorreo 256
#define SaltoUsuarios 512
#define InicioMemoria 0x30000


// https://tools.ietf.org/html/rfc5321#section-4.5.3
// Tamaño maximo que puede tener una direccion de correo electrónico según RFC
5321
#define MAX_long_correo 256
char buffer_correo[MAX_long_correo] = {0};

// Buffer para el nombre de usuario
#define MAX_long_nombre 200
char buffer_nombre[MAX_long_nombre] = {0};

// Buffer clave escrita
char rand_number[long_clave+1] = {0};
char buffer_clave[long_clave] = {0};


// Variables para distintos bucles for
int i, j;

// Variables lectura / escritura en flash
uint32_t byte_leido_flash;
unsigned char correorecuperado[MAX_long_correo] = {0}; // para comprobar la
lectura de la flash
unsigned char nombrerecuperado[MAX_long_nombre] = {0}; // para comprobar la
lectura de la flash
uint32_t byte_envio;
int long_correo = 0;
int posicion = InicioMemoria;
int posicion_libre;
int longitud_correo_rec = 0;
int ok; // para mantenernos en el while

// Utilidades RFID
```

```
int chipSelectPin = 0x2;    //PQ1
int NRSTPD = 0x10;    //PB4

#define CARD_LENGTH 5
uint8_t status;
unsigned char str[MAX_LEN];
unsigned char cardREAD[100];
unsigned char cardID[CARD_LENGTH];

//------------------------------------------------------------------------------
------------------\\
//----------------------DEFINES / VARIABLES PARA LA PARTE DE TECLADO-----------
-----------------\\
//------------------------------------------------------------------------------
------------------\\

// Dibuja una pantalla inicial, espera a que el usuario toque la pantalla
void pantalla_inicial(void);

// Funciones relacionadas con el teclado normal en pantalla
void pantalla(short int campo);
void get_teclado(short int campo);
void pantalla_teclado(short int campo);

// Indice usado en bucles dentro de las funciones del teclado
char indice_teclado = 0;

// Variables que almacenan la tecla actual y la anterior
char tecla = 0;
char tecla_p = 0;

// Esquina superior izquierda de la primera tecla, primera fila
short int fila1_inicio_x = 10;    // Coordenada x, esquina superior izquierda
short int fila1_inicio_y;// = 120;    // Coordenada y, esquina superior izquierda

// Altura y anchura de las letras
short int tecla_w;   // Anchura, se calcula más adelante
short int tecla_h = 25;   // Altura

// Anchura rectángulo filas 1, 2 y 3
short int fila1_rect_w;
short int fila2_rect_w;
short int fila3_rect_w;

// Fila 0, de los números, añadida posteriormente


// Separacion entre teclado y borde pantalla
short int tecla_borde;  // pix

// Separacion vertical entre teclas
short int tecla_separacion = 8;

// Definir cadenas de cada fila del teclado, sin la ñ
char *fila0 = "1234567890";

char *fila1_minus = "qwertyuiop";
char *fila1_mayus = "QWERTYUIOP";
char *fila1_especiales = "!$%&/()=?-";

char *fila2_minus = "asdfghjkl";
char *fila2_mayus = "ASDFGHJKL";
```

```
char *fila2_especiales = "|@#'^*,.;";

char *fila3_minus = "zxcvbnm";
char *fila3_mayus = "ZXCVBNM";
char *fila3_especiales = "[]{}<>_";

char *fila4 = "@      .";



#define n_tecla_fila_1 10
#define n_tecla_fila_2 9
#define n_tecla_fila_3 7
#define n_filas 5


// Calculo comienzo teclas siguientes filas, dependen de fila1
short int fila0_inicio_y;

short int fila2_inicio_x;
short int fila2_inicio_y;

short int fila3_inicio_x;
short int fila3_inicio_y;

short int fila4_inicio_y;

// Variables para los botones de shift, Options, Delete y Enter
short int shift_boton = 0;
short int shift_boton_p = 0;
bool SHIFT = false;

short int enter_boton = 0;
short int enter_boton_p = 0;
bool ENTER = false;

short int options_boton = 0;
short int options_boton_p = 0;
bool OPTIONS = false;

short int delete_boton = 0;
short int delete_boton_p = 0;
bool DELETE = false;

short int space_boton = 0;
short int space_boton_p = 0;
bool SPACE = false;

//-------------------------------------------------------------------------
------------------\\
//----------------------DEFINES / VARIABLES PARA LA PARTE DE BITMAPS----------
-----------------\\
//-------------------------------------------------------------------------
------------------\\

// Solo vamos a tener 2 BMP como maximo a la  vez
#define ram_addr_BMP1 0x0000
#define ram_addr_BMP2 0x6000

// Datos del tamano en bytes y de la posición de la flash en la que está
almacenado cada BMP
```

```
// LOGO
#define nbytes_logo 2512
#define flash_addr_logo 0x7D000

// Ancho y alto en pixeles de la imagen
#define ancho_pix_logo 200
#define alto_pix_logo  200

// NFC
#define nbytes_nfc 2347
#define flash_addr_nfc 0x7D9D0

// Ancho y alto en pixeles de la imagen
#define ancho_pix_nfc 200
#define alto_pix_nfc  192

// RECONOCIDO
#define nbytes_rec 3394
#define flash_addr_rec 0x7E2FB

// Ancho y alto en pixeles de la imagen
#define ancho_pix_rec 200
#define alto_pix_rec  200

// NO RECONOCIDO
#define nbytes_norec 3357
#define flash_addr_norec 0x7F03D

// Ancho y alto en pixeles de la imagen
#define ancho_pix_norec 200
#define alto_pix_norec  200

// ok
#define nbytes_ok 543
#define flash_addr_ok 0x7FD5A

// Ancho y alto en pixeles de la imagen
#define ancho_pix_ok 80
#define alto_pix_ok  80

// err
#define nbytes_err 783
#define flash_addr_err 523983

// Ancho y alto en pixeles de la imagen
#define ancho_pix_err 80
#define alto_pix_err  80

// lock
#define nbytes_lock 2850
#define flash_addr_lock 524501

// Ancho y alto en pixeles de la imagen
#define ancho_pix_lock 200
#define alto_pix_lock 200

// unlock
#define nbytes_unlock 2805
#define flash_addr_unlock 527351

// Ancho y alto en pixeles de la imagen
#define ancho_pix_unlock 200
```

```
#define alto_pix_unlock 200


// Comprueba si se ha pulsado aceptar o cancelar en la comprobacion de correo/usuario
int comp_boton(){
    // Variable a devolver en funcion de lo pulsado
    int respuesta=0;
    while(respuesta == 0)
    {
        Lee_pantalla();
        if ((POSX > (HSIZE/2-ancho_pix_ok/2 - HSIZE/4) && POSX < (HSIZE/2 + ancho_pix_ok/2 - HSIZE/4))&& (POSY > VSIZE*3/4-alto_pix_ok/2 && POSY<VSIZE*3/4+alto_pix_ok/2)){
            respuesta=1;
        }
        if ((POSX > HSIZE/2-ancho_pix_err/2 + HSIZE/4 && POSX < (HSIZE/2 + ancho_pix_err/2 + HSIZE/4))&& (POSY > VSIZE*3/4-alto_pix_err/2 && POSY<VSIZE*3/4+alto_pix_err/2)){
            respuesta=2;
        }
    }
    return respuesta;
}

// Interrupcion del timer cada 50ms
void interrupcion_timer1(){
    // Variable para contar
    tiempo++;
    // Variable para semilla de generacion del codigo aleatorio
    aleatoriedad++;
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT); // Borra la interrupción del temporizador
    SysCtlDelay(100);
}

int main(void)
{
    // Configuración del RELOJ a 120 MHz
    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    // Configuracion reloj
    config_timer1(RELOJ);

    // Configuración del PWM
    config_PWM();

    // Configuración e inicialización del lector RC522
    config_RC522();
    RFID_init();

    // Configuración del cliente SMTP
    config_SMTP(RELOJ);

    // Boosterpack 1 para FT800, pintar pantalla inicial
    HAL_Init_SPI(1, RELOJ);
    pantalla_inicial();

    // Guardamos datos del admin
    GuardarADMIN(posicion);
```

```
    // Determina los segundos a esperar en los distintos estados
    int esperar=0;

    while(1){
        // Lee posicion del dedo de la pantalla
        Lee_pantalla();

        // Pantalla en blanco y configuracion de colores
        Nueva_pantalla(0xff,0xff,0xff);
        ComColor(0,0,0);
        ComFgcolor(45, 150, 100);

        // Maquina de estados
        switch(estado){
        // Estado de reposo, espera a lectura de tarjeta RFID
        case 0:
            dib = 0;
            // Cargar imagen RFID y dibujar
            almacenar_imagen_externa(flash_addr_nfc,            ram_addr_BMP1,
nbytes_nfc);
            Nueva_pantalla(0xff,0xff,0xff);
            dibujar_imagen_externa(ram_addr_BMP1, ancho_pix_nfc, alto_pix_nfc,
HSIZE/2-ancho_pix_nfc/2, VSIZE/2-alto_pix_nfc/2, 255, 0);
            // Comprobacion de bloqueo de cerradura
            if (bloqueo == 1)
            {
                ComColor(255,0,0);
                ComTXT(HSIZE/2,VSIZE*7/8,       28,       OPT_CENTER,"Cerradura
bloqueada"); //pinta un texto
            }
            Dibuja();
            // bucle hasta que se lea tarjeta
            while(tarjeta_detectada!=1){
                SysCtlSleep();
                RFID();
            }
            estado=1;
            tarjeta_detectada=0;
            break;
            // Identificamos al usuario
        case 1:
            dib = 1;
            // Posicion de inicio de almacenamiento de usuarios
            posicion=InicioMemoria;
            // Comprobacion de usuario registrado
            ok=ComprobarID(posicion);
            // Si el usuario es reconocido
            if (ok==2)
            {
                // Cargar imagen TARJETA RECONOCIDA y dibujar
                almacenar_imagen_externa(flash_addr_rec,         ram_addr_BMP1,
nbytes_rec);
                Nueva_pantalla(0xff,0xff,0xff);
                dibujar_imagen_externa(ram_addr_BMP1,            ancho_pix_rec,
alto_pix_rec, HSIZE/2-ancho_pix_rec/2, VSIZE/2-alto_pix_rec/2, 255, 0);
                // Si la tarjeta es de admin
                if ( numero_usuario == 0 )
                    estado = 2;
                // Usuario normal
                else
                {
                    // Comprobacion de bloqueo de cerradura
```

```
                    if (bloqueo == 1)
                    {
                        ComColor(255,0,0);
                        ComTXT(HSIZE/2,VSIZE*7/8,   28,   OPT_CENTER,"Cerradura
bloqueada"); //pinta un texto
                        estado = 0;
                    }
                    else
                        estado = 14;
                }
                // Sonido de okey
                TocaNota(0x41,70-24);
                SysCtlDelay(10*MSEC);
                TocaNota(0x41,73-24);
            }
            // Si el usuario no esta registrado
            else
            {
                // Cargar imagen TARJETA NO RECONOCIDA y dibujar
                almacenar_imagen_externa(flash_addr_norec,       ram_addr_BMP1,
nbytes_norec);
                Nueva_pantalla(0xff,0xff,0xff);
                dibujar_imagen_externa(ram_addr_BMP1,          ancho_pix_norec,
alto_pix_norec, HSIZE/2-ancho_pix_norec/2, VSIZE/2-alto_pix_norec/2, 255, 0);
                estado = 0;
                // Sonido de error
                TocaNota(0x42,70-24);
                SysCtlDelay(10*MSEC);
                TocaNota(0x42,67-24);
            }
            // Espera de 1 segundo
            esperar=1;
            tarjeta_detectada = 0;
            break;
            // Pantalla bienvenida admin
        case 2:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Bienvenido admin");
            ComTXT(HSIZE/2,VSIZE*5/8, 26, OPT_CENTER,"Se esta enviando el codigo
de verificacion");
            // Se envia correo de verificacion
            estado=15;
            // Espera de 1 segundo
            esperar=1;
            break;
            // Menu admin
        case 3:
            dib = 1;
            ComColor(255,255,255);
            // Botones
            if(BotonFlat(HSIZE*1/6,   VSIZE*2/24,   HSIZE*4/6,   VSIZE*1/6,   28,
"Agregar usuario"))
                estado=4;
            if(BotonFlat(HSIZE*1/6,   VSIZE*7/24,   HSIZE*4/6,   VSIZE*1/6,   28,
"Abrir cerradura"))
                estado=19;
            if(BotonFlat(HSIZE*1/6,   VSIZE*12/24,   HSIZE*4/6,   VSIZE*1/6,   28,
"Volver"))
                estado=21;
            // Opcion de desbloquear en caso de bloqueo de cerradura
            if (bloqueo == 1)
            {
```

```
                ComFgcolor(255, 0, 0);
                if(BotonFlat(HSIZE*1/6, VSIZE*17/24, HSIZE*4/6, VSIZE*1/6, 28,
"Desbloquear"))
                {
                    bloqueo = 0;
                    estado=21;
                }
            }
            break;
            // Agregar ID usuario
        case 4:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Acerque la tarjeta del nuevo
usuario"); //pinta un texto
            // Lectura de tarjeta
            RFID();
            // Cuando se lea la  tarjeta
            if (tarjeta_detectada==1)
            {
                // Posicion de inicio de almacenamiento de usuarios
                posicion=InicioMemoria;
                // Comprobamos que esa tarjeta no se haya usado aun
                ok=ComprobarID(posicion);
                // Tarjeta todavia no registrada, la guardamos
                if (ok == 0)
                {
                    estado=5;
                    tarjeta_detectada=0;
                    TocaNota(0x41,70-24);
                    SysCtlDelay(10*MSEC);
                    TocaNota(0x41,73-24);
                }
                // Tarjeta ya usada
                else
                {
                    estado=6;
                    tarjeta_detectada=0;
                    TocaNota(0x42,70-24);
                    SysCtlDelay(10*MSEC);
                    TocaNota(0x42,67-24);
                }
            }
            break;
            // Mensaje de introduccion de correo
        case 5:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2,  28,  OPT_CENTER,"Introduzca  el  correo  del
nuevo usuario");
            // Espera de 1 segundo
            esperar = 1;
            estado = 7;
            break;
            // Mensaje de tarjeta ya usada
        case 6:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2,  28,  OPT_CENTER,"Tarjeta  ya  registrada");
//pinta un texto
            // Espera de 1 segundo
            esperar = 1;
            estado = 4;
            break;
            // Escribir correo usuario
```

```
        case 7:
            dib = 0;
            // Funcion para mostrar teclado
            pantalla_teclado(0);
            estado=8;
            break;
            // Confirmacion de correo
        case 8:
            dib = 0;
            // Cargar imagenes ACEPTAR y CANCELAR y dibujar
            almacenar_imagen_externa(flash_addr_ok, ram_addr_BMP1, nbytes_ok);
            almacenar_imagen_externa(flash_addr_err,            ram_addr_BMP2,
nbytes_err);
            Nueva_pantalla(0xff,0xff,0xff);
            dibujar_imagen_externa(ram_addr_BMP1,  ancho_pix_ok,  alto_pix_ok,
HSIZE/2-ancho_pix_ok/2 - HSIZE/4, VSIZE*3/4-alto_pix_ok/2, 255, 0);
            dibujar_imagen_externa(ram_addr_BMP2, ancho_pix_err, alto_pix_err,
HSIZE/2-ancho_pix_ok/2 + HSIZE/4, VSIZE*3/4-alto_pix_err/2, 255, 0);
            ComColor(0,0,0);
            ComTXT(HSIZE/2,VSIZE*1/3,   28,   OPT_CENTER,"Es   correcta   esta
direccion?");
            sprintf(TXT_1, "%s", buffer_correo);
            ComTXT(HSIZE/2,VSIZE/2, 26, OPT_CENTER,TXT_1);
            Dibuja();
            // Funcion para comprobar que se ha pulsado
            resp = comp_boton();
            // Ok
            if ( resp == 1)
            {
                estado=9;
                break;
            }
            // Volver a escribir
            else if (resp == 2)
            {
                // Borrar lo que se ha escrito
                BorrarBufferEscrituraCorreo();
                // Espera de 1 segundo
                esperar=1;
                estado = 7;
                break;
            }

            break;
            // Mensaje introducir nombre de usuario
        case 9:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2,  28,  OPT_CENTER,"Introduzca  el  nombre  del
nuevo usuario");
            // Espera de 1 segundo
            esperar = 1;
            estado = 10;
            break;
            // Escribir nombre usuario
        case 10:
            dib = 0;
            // Funcion para mostrar teclado
            pantalla_teclado(1);
            estado = 11;
            break;
            // Confirmacion de usuario
        case 11:
```

```
            dib = 0;
            // Cargar imagenes ACEPTAR y CANCELAR y dibujar
            almacenar_imagen_externa(flash_addr_ok, ram_addr_BMP1, nbytes_ok);
            almacenar_imagen_externa(flash_addr_err,            ram_addr_BMP2,
nbytes_err);
            Nueva_pantalla(0xff,0xff,0xff);
            dibujar_imagen_externa(ram_addr_BMP1,  ancho_pix_ok,  alto_pix_ok,
HSIZE/2-ancho_pix_ok/2 - HSIZE/4, VSIZE*3/4-alto_pix_ok/2, 255, 0);
            dibujar_imagen_externa(ram_addr_BMP2, ancho_pix_err, alto_pix_err,
HSIZE/2-ancho_pix_ok/2 + HSIZE/4, VSIZE*3/4-alto_pix_err/2, 255, 0);
            ComColor(0,0,0);
            ComTXT(HSIZE/2,VSIZE*1/3,   28,   OPT_CENTER,"Es   correcto   este
nombre?");
            sprintf(TXT_1, "%s", buffer_nombre);
            ComTXT(HSIZE/2,VSIZE/2, 26, OPT_CENTER,TXT_1);
            Dibuja();
            // Funcion para comprobar que se ha pulsado
            resp = comp_boton();
            // Ok
            if ( resp == 1)
            {
                estado = 12;
                break;
            }
            // Volver a escribir
            else if (resp == 2)
            {
                // Borrar lo que se ha escrito
                BorrarBufferEscrituraNombre();
                // Espera de 1 segundo
                esperar = 1;
                estado = 10;
                break;
            }
            break;
            // Almacenamos ID, correo y nombre
        case 12:
            dib = 0;
            posicion=InicioMemoria;
            // Busca una posicion de memoria libre para almacenar el usuario
            posicion_libre = EncontrarEspacio(posicion);
            // Funcion para guardar el usuario
            GuardarUSUARIO(posicion_libre,      cardID,      &buffer_correo[0],
&buffer_nombre[0]);
            // Se borran buffers para la proxima
            BorrarBufferEscritura();
            estado = 13;
            break;
            // Mensaje de usuario guardado
        case 13:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Usuario guardado");
            // Espera de 1 segundo
            esperar = 1;
            estado = 21;
            break;
            // Pantalla de bienvenida de usuario
        case 14:
            dib = 1;
            // Funcion para obtener el nombre del usuario
            ObtenerNombre();
            // Mensaje de bienvenida
```

```
            sprintf(TXT_1, "Bienvenido %s", nombrerecuperado);
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,TXT_1);
            ComTXT(HSIZE/2,VSIZE*5/8,  26,  OPT_CENTER,  "Se  esta  enviando  el
codigo de verificacion");
            // Espera de 1 segundo
            esperar=1;
            estado=15;
            reintentos = 0;
            break;
            // Envio de codigo de confirmacion al correo
        case 15:
            dib = 0;
            // Leemos el correo del usuario
            ObtenerCorreo();
            // Reiniciamos el contador de intentos (max 3)
            contador_intentos = 0;
            // Obtenemos la longitud del correo ( necesario para el envio )
            longitud_correo_rec = sizeof(correorecuperado);
            // Funcion encargada del envio
            correo_exito = 2;
            SMTP_start();
            while(correo_exito == 2);
            if (correo_exito == 1)
                estado = 16;
            else if (correo_exito == 0 && reintentos < 2)
                estado = 15;
            else
                estado = 22;
            break;
            // Mensaje de email
        case 16:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Introduzca la clave enviada
a su correo");
            // Espera de 1 segundo
            esperar = 1;
            estado = 17;
            break;
            // Introduccion de correo recibido
        case 17:
            dib = 0;
            // Mostrar teclado mientras no se supere el numero de intentos maximo
(3)
            if (contador_intentos < 3)
            {
                pantalla_teclado(3);
                estado=18;
            }
            // Si se ha bloqueado la cerradura
            else
                estado=20;
            break;
            // Comprobacion de clave
        case 18:
            dib = 1;
            // Variable para comprobar la coincidencia del codigo
            clave_correcta=1;
            // Bucle de comprobacion de caracteres del codigo
            for(i=0;i<long_clave;i++)
            {
                // Si no coincide alguno de los caracteres, se pone a 0
"clave_correcta" y se sale del bucle
```

```
                if (buffer_clave[i] != rand_number[i])
                {
                    clave_correcta=0;
                    break;
                }
            }
            // Si la clave es correcta
            if (clave_correcta == 1)
            {
                ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Clave correcta");
                // Si es el admin
                if (numero_usuario == 0)
                {
                    // Al menu de admin
                    estado = 3;
                    BorrarBufferClave();
                }
                else
                    estado = 19;
                // Espera de 1 segundo
                esperar = 1;
            }
            // Si no es correcta
            else
            {
                ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"Clave incorrecta");
                estado = 17;
                // Espera de 1 segundo
                esperar = 1;
                // Se borra buffer de clave para la nueva introduccion
                BorrarBufferClave();
                // Se aumenta contador de intentos
                contador_intentos++;
            }
            break;
            // Animacion de desbloqueo de puerta y apertura de puerta
        case 19:
            dib = 1;
            // Cargar imagen LOCK y dibujar
            ComColor(255,255,255);
            almacenar_imagen_externa(flash_addr_lock,           ram_addr_BMP1,
nbytes_lock);
            Nueva_pantalla(255,255,255);
            dibujar_imagen_externa(ram_addr_BMP1,              ancho_pix_lock,
alto_pix_lock, HSIZE/2-ancho_pix_lock/2, VSIZE/2-alto_pix_lock/2, 255, 0);
            Dibuja();
            // Espera de 1 segundo
            EsperaSeg(1);
            // Cargar imagene DESBLOQUEO y dibujar
            almacenar_imagen_externa(flash_addr_unlock,         ram_addr_BMP1,
nbytes_unlock);
            Nueva_pantalla(255,255,255);
            dibujar_imagen_externa(ram_addr_BMP1,            ancho_pix_unlock,
alto_pix_unlock,  HSIZE/2-ancho_pix_unlock/2,  VSIZE/2-alto_pix_unlock/2,  255,
0);
            Dibuja();
            // Servo abre la puerta
            PWMOutputState(PWM0_BASE, PWM_OUT_5_BIT, true);
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_5, Min_pos);
            // Espera de 3 segundos
            EsperaSeg(3);
            // Vuelta al inicio
```

```
            estado=0;
            // Se borran todos los buffers
            BorrarBufferCorreo();
            BorrarBufferNombre();
            BorrarBufferClave();
            // Servo cierra la puerta
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_5, Max_pos);
            EsperaSeg(1);
            PWMOutputState(PWM0_BASE, PWM_OUT_5_BIT, false);
            break;
            // Mensaje de puerta bloqueada por exceso de intentos
        case 20:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2,   28,   OPT_CENTER,"Numero   de   intentos
superado"); //pinta un texto
            ComTXT(HSIZE/2,VSIZE*2/3, 21, OPT_CENTER,"Contacte con admin para
desbloquear"); //pinta un texto
            // Se bloquea cerradura
            bloqueo = 1;
            // Espera de 2 segundos
            esperar=2;
            // Vuelta al inicio
            estado=21;
            break;
            // Vuelta a inicio y borrado de buffers
        case 21:
            dib = 0;
            // Vuelta al inicio
            estado=0;
            // Se borran todos los buffers
            BorrarBufferCorreo();
            BorrarBufferNombre();
            BorrarBufferClave();
            break;
        case 22:
            dib = 1;
            ComTXT(HSIZE/2,VSIZE/2, 28, OPT_CENTER,"No se ha podido enviar el
correo, contacte con el administrador");
            // Espera de 1 segundo
            esperar = 3;
            estado = 21;
            break;
        }
        // Algunos estados no tienen "Dibujas()" dentro, por lo que se dibujaran
al llegar aqui
        if (dib == 1)
            Dibuja();
        // Llamada a timer de X segundos si es necesario
        if (esperar > 0)
        {
            EsperaSeg(esperar);
            esperar = 0;
        }
    }
}

/* -------------------------------------------------
 * -------------------------------------------------
 * -------------BLOQUE DE CONFIGURACIONES-----------
 * -------------------------------------------------
 * ------------------------------------------------- */
```

```
// Configuracion timer
void config_timer1(int Reloj){
    uint32_t periodo1;
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);           //Habilita T0
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM);   //T0 a 120MHz
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);          //T0 periodico y
conjunto (32b)
    periodo1 = 6000000;                                    //Periodo de 50 ms
    TimerLoadSet(TIMER0_BASE, TIMER_A, periodo1 -1);
    TimerIntRegister(TIMER0_BASE,TIMER_A,interrupcion_timer1);
    IntEnable(INT_TIMER0A);                                     //Habilitar las
interrupciones globales de los timers
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);          // Habilitar las
interrupciones de timeout
    IntMasterEnable();
    TimerEnable(TIMER0_BASE, TIMER_A);
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_TIMER0);          //Habilitar el
timer 0 durante el Sleep
    SysCtlPeripheralClockGating(true);

}

// Configuración del PWM, utilizado para SERVO
void config_PWM(){
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
    PWMClockSet(PWM0_BASE,PWM_SYSCLK_DIV_64);   // al PWM le llega un reloj de
1.875MHz
    GPIOPinConfigure(GPIO_PG1_M0PWM5);          //Configurar el pin a PWM
    GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_1);
    //Configurar  el  pwm0,  contador  descendente  y  sin  sincronización
(actualización automática)
    PWMGenConfigure(PWM0_BASE,       PWM_GEN_2,       PWM_GEN_MODE_DOWN       |
PWM_GEN_MODE_NO_SYNC);
    PeriodoPWM=37499; // 50Hz  a 1.875MHz
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, PeriodoPWM); //frec:50Hz
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_5, Max_pos);   //Inicialmente, 1ms
    PWMGenEnable(PWM0_BASE, PWM_GEN_2);     //Habilita el generador 2
    //      PWMOutputState(PWM0_BASE, PWM_OUT_5_BIT , true);     //Habilita la
salida 5
}

// Configuración de periféricos para el lector de tarjetas NFC RC522
void config_RC522(){
    uint32_t junkAuxVar;

    // Periférico 3 del SSI (entero en el boosterpack 2)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI3);

    // Reset
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

    // SDA, SCK, MOSI, MISO
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOQ);


    GPIOPinConfigure(GPIO_PQ0_SSI3CLK);     // SCK
    GPIOPinConfigure(GPIO_PQ2_SSI3XDAT0);   // MOSI
    GPIOPinConfigure(GPIO_PQ3_SSI3XDAT1);   // MISO

    // SCK, MOSI, MISO
```

```
    GPIOPinTypeSSI(GPIO_PORTQ_BASE, GPIO_PIN_0|GPIO_PIN_2|GPIO_PIN_3);

    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4); //NRSTPD
    GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_1); //chipSelectPin

    // Configurar el reloj
    SSIConfigSetExpClk(SSI3_BASE,        160000000,        SSI_FRF_MOTO_MODE_0,
SSI_MODE_MASTER, 4000000, 8);

    // Activar el módulo SSI 3
    SSIEnable(SSI3_BASE);

    // Eliminar posibles "residuos"
    while(SSIDataGetNonBlocking(SSI3_BASE, &junkAuxVar)){}


    // Activar pin de CS y NRSTPD
    GPIOPinWrite(GPIO_PORTQ_BASE, chipSelectPin, chipSelectPin);
    GPIOPinWrite(GPIO_PORTB_BASE, NRSTPD, NRSTPD);

    // Inicializar el MFRC522 con los pines correspondientes
    Mfrc522(chipSelectPin,NRSTPD);


}



void pantalla_inicial(){
    Inicia_pantalla();
    // Note: Keep SPI below 11MHz here

    // ================================================================================
    // Delay before we begin to display anything
    // ================================================================================

    SysCtlDelay(RELOJ/3);

    //
================================================================================
==================================
    // PANTALLA INICIAL
    //
================================================================================
==================================

    // Fade-in de la pantalla
    VolNota(100);
    TocaNota(0x45,48);
    for(i=0;i<255;i+=5){
        Nueva_pantalla(i,i,i);
        Dibuja();
    }

    // Mostrar el logo inicial de TIVA LOCK
    almacenar_imagen_externa(flash_addr_logo, ram_addr_BMP1, nbytes_logo);
    for(i=0;i<255;i+=5)
    {
        Nueva_pantalla(255,255,255);
        dibujar_imagen_externa(ram_addr_BMP1,  ancho_pix_logo,  alto_pix_logo,
HSIZE/2-ancho_pix_logo/2, VSIZE/2-alto_pix_logo/2, i, 0);
        Dibuja();
```

```
    }
    Espera_pant();

    // Calibrar la pantalla según se haya elegido la opción de 3.5" o 5"
#ifdef VM800B35
    for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL[i]);
#endif
#ifdef VM800B50
    for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL5[i]);
#endif
}

// Función encargada de hacer poll al lector RFID, almacena tarjeta leída
void RFID()
{
    // Ver si se detecta tarjeta
    status = Request(PICC_REQIDL, str);

    // En caso de conflictos
    status = Anticoll(str);

    // Copiar la cadena que tiene la ID de la tarjeta
    memcpy(cardID, str, 10);

    // Tarjeta detectada
    if(status == MI_OK){
        tarjeta_detectada=1;
    }
}

// Función para encontrar posiciones de memoria disponibles a la hora de
almacenar nuevos usuarios
int EncontrarEspacio(int posicion)
{
    int espacio_libre = posicion;

    // Mientras la primera posición de cada bloque sea distinto de 0xFF,
continuar buscando
    while( HWREGB(espacio_libre) != 0xFF)
        espacio_libre+=SaltoUsuarios;

    // Devuelve la posición encontrada
    return espacio_libre;
}

// Almacena la información relativa con el usuario según los parámetros recibidos
void GuardarUSUARIO(int posicion, unsigned char* ID_usuario, char*
correo_usuario, char* nombre_usuario)
{
    int posicion_ini = posicion;
    int posicion_act = posicion;

    int long_nombre_usuario=0;
    int long_correo_usuario=0;

    char buff_correo[MAX_long_correo];
    char buff_nombre[MAX_long_nombre];

    // Copiar en el buffer interno las cadenas de correo y nombre de usuario
    for(j=0;j<MAX_long_correo;j++)
    {
        buff_correo[j] = *(correo_usuario + j);
```

```
    }
    for(j=0;j<MAX_long_nombre;j++)
    {
        buff_nombre[j] = *(nombre_usuario + j);
    }

    // Indicadores (PONER COMO DEFINES????)
    uint32_t indID=0x1;
    uint32_t indCorreo=0x2;
    uint32_t indNombre=0x3;

    // Para guardar id
    uint32_t numero;


    numero=0; // Poner a 0 ya que hacemos OR

    // Guardar indicador de ID y 3 primero digitos
    numero |= (uint8_t)indID;
    numero |= (uint8_t)ID_usuario[0]<<8*1;
    numero |= (uint8_t)ID_usuario[1]<<8*2;
    numero |= (uint8_t)ID_usuario[2]<<8*3;

    // Programar en la flash del microcontrolador
    FlashProgram(&numero, posicion_act, 4);

    // Repetir el proceso con los dos últimos byte, rellenar con 0xFF hasta
formar una palabra (32 bits)
    numero = 0;
    // 4o y 5o digito
    numero |= (uint8_t)ID_usuario[3]<<8*0;
    numero |= 0xFFu<<8*1;
    numero |= 0xFFu<<8*2;
    numero |= 0xFFu<<8*3;
    FlashProgram(&numero, posicion_act+4, 4); // en el siguiente hueco

    // Calcular posición para almacenar correo
    posicion_act = posicion_ini + SaltoID;

    // Guardar correo
    byte_envio = 0;
    j=0;

    // Almacenar en buffer el correo
    while(buff_correo[long_correo_usuario] != 0)
        long_correo_usuario++;

    /*
     * Guardar en la flash. Procedimiento:
     *  - Mandar el indicador de correo
     *  - Guardar el correo
     *  Se programa cada 4 bytes, y al
     *  salir del bucle se comprueba si
     *  el último no se envió, por no
     *  llegar a 4B, y se rellena con 0xFF
     *  y se programa en la flash.
     */
    for(i=0;i<long_correo_usuario+2;i++)
    {
        // Preparar byte_envio para ser enviado
        if(i == 0)
            byte_envio |= indCorreo<<8*j;          // Indicador
```

```
        else if (i<long_correo_usuario+1)
            byte_envio |= buff_correo[i-1]<<8*j;   // Correo
        else
            byte_envio |= 128<<8*j;                 // Byte de finalización de
correo

        // Comprobación para mandar 4 bytes
        if (j<3)
            // Si es menor que 4 bytes, aumentar
            j++;
        else {
            // Si se llega a 4, programar, actualizar la posición y reiniciar
las variables
            FlashProgram(&byte_envio, posicion_act, 4);
            j=0;
            posicion_act+=4;
            byte_envio = 0;
        }
    }
    // Comprobación del último byte
    for (i=0;i<4-j;i++){
        byte_envio |= 0xFF<<8*(j+i);
    }
    FlashProgram(&byte_envio, posicion_act, 4); //guardamos en la flash

    // Guardars indicador de nombre
    posicion_act = posicion_ini + SaltoCorreo + SaltoID;

    // Guardar nombre en buffer
    byte_envio = 0;
    while(buff_nombre[long_nombre_usuario] != 0)
        long_nombre_usuario++;

    // Reiniciar indice de 4 bytes
    j=0;

    // Realizar mismo procedimiento que con el buffer del correo electrónico
    for(i=0;i<long_nombre_usuario+2;i++)
    {
        if(i == 0)
            byte_envio |= indNombre<<8*j;
        else if (i<long_correo_usuario+1)
            byte_envio |= buff_nombre[i-1]<<8*j;
        else
            byte_envio |= 128<<8*j;

        if (j<3)
            j++;
        else {
            FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash
            j=0;
            posicion_act+=4;
            byte_envio = 0;
        }
    }
    for (i=0;i<4-j;i++){
        byte_envio |= 0xFF<<8*(j+i);
    }
    j=0;
    FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash
}
```

```
// Procedimiento idéntico al de guardar usuario, con la única diferencia de que
la información viene pre-almacenada
void GuardarADMIN(int posicion)
{
    int posicion_ini = posicion;
    int posicion_act = posicion;

    unsigned char correo_admin[] = "smart.lock.tm4c@gmail.com";
    int long_correo_admin;

    unsigned char nombre_admin[] = "Administrador";
    int long_nombre_admin;

    uint32_t indID=0x1;
    uint32_t indCorreo=0x2;
    uint32_t indNombre=0x3;
//    unsigned char IDadmin[]={ 0x5a, 0x32, 0x9e, 0xd5 };
    unsigned char IDadmin[]={ 0x2b, 0xc, 0xcf, 0x22 };
    uint32_t numero; // para guardar id

    // Guardamos ID e indicador de ID
    numero=0; //ponemos a 0 ya que hacemos OR
    // id y 3 primero digitos
    numero|=(uint8_t)indID;
    numero|=(uint8_t)IDadmin[0]<<8*1; //pasamos de char a uint32
    numero|=(uint8_t)IDadmin[1]<<8*2;
    numero|=(uint8_t)IDadmin[2]<<8*3;

    FlashProgram(&numero,posicion_act,4);
    numero = 0;
    // 4o y 5o digito
    numero|=(uint8_t)IDadmin[3]<<8*0;
    numero|=0xFF<<8*1;
    numero|=0xFF<<8*2;
    numero|=0xFFu<<8*3;
    FlashProgram(&numero,posicion_act+4,4); // en el siguiente hueco

    // Guardamos indicador de correo
    posicion_act = posicion_ini + SaltoID;
    // Guardamos correo
    byte_envio = 0;
    j=0;
    long_correo_admin = sizeof(correo_admin)-1;
    for(i=0;i<long_correo_admin+1;i++)
    {
        if(i == 0)
            byte_envio |= indCorreo<<8*j;
        else if (i<long_correo_admin+1)
            byte_envio |= correo_admin[i-1]<<8*j;
        else
            byte_envio |= 128<<8*j;
        if (j<3)
            j++;
        else {
            FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash
            j=0;
            posicion_act+=4;
            byte_envio = 0;
        }
    }
    for (i=0;i<4-j;i++){
        byte_envio |= 0xFF<<8*(j+i);
```

```
    }
    j=0;
    FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash

    // Guardamos indicador de nombre
    posicion_act = posicion_ini + SaltoCorreo + SaltoID;

    // Guardamos nombre
    byte_envio = 0;
    long_nombre_admin = sizeof(nombre_admin)-1;
    for(i=0;i<long_nombre_admin+1;i++)
    {
        if(i == 0)
            byte_envio |= indNombre<<8*j;
        else if (i<long_nombre_admin+1)
            byte_envio |= nombre_admin[i-1]<<8*j;
        else
            byte_envio |= 128<<8*j;
        if (j<3)
            j++;
        else {
            FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash
            j=0;
            posicion_act+=4;
            byte_envio = 0;
        }
    }
    for (i=0;i<4-j;i++){
        byte_envio |= 0xFF<<8*(j+i);
    }
    j=0;
    FlashProgram(&byte_envio,posicion_act,4); //guardamos en la flash
}

// Función que obtiene la dirección de correo almacenada en flash
void ObtenerCorreo()
{
    posicion= InicioMemoria + numero_usuario * SaltoUsuarios + SaltoID + 1;
    i=0;
    while(HWREGB(posicion) != 0xFF)
    {
        byte_leido_flash=HWREGB(posicion); //obtiene el valor de la direccion
especificada
        posicion++; //pasa al siguiente registro de memoria
        if (byte_leido_flash!=128)
            correorecuperado[i]=byte_leido_flash;
        else
            break;
        i++;
    }
}

// Mismo procedimiento que con void ObtenerCorreo();
void ObtenerNombre()
{
    posicion= InicioMemoria + numero_usuario * SaltoUsuarios + SaltoID +
SaltoCorreo + 1;
    i=0;
    while(HWREGB(posicion) != 0xFF)
    {
        byte_leido_flash=HWREGB(posicion); //obtiene el valor de la direccion
especificada
```

```
        posicion++; //pasa al siguiente registro de memoria
        if (byte_leido_flash!=128)
            nombrerecuperado[i]=byte_leido_flash;
        else
            break;
        i++;
    }
}

// BLOQUE DE ELIMINACIÓN DE BUFFERS DE CORREO, NOMBRE, NÚMERO ALEATORIO...
void BorrarBufferCorreo()
{
    for(i=0;i<MAX_long_correo;i++)
    {
        correorecuperado[i]=0;
    }
}

void BorrarBufferNombre()
{
    for(i=0;i<MAX_long_nombre;i++)
    {
        nombrerecuperado[i]=0;
    }
}

void BorrarBufferClave()
{
    for(i=0;i<long_clave;i++)
    {
        buffer_clave[i]=0;
    }
}
void BorrarBufferEscritura()
{
    for(i=0;i<MAX_long_correo;i++)
    {
        buffer_correo[i]=0;
        if(i<MAX_long_nombre)
            buffer_nombre[i]=0;
    }
}
void BorrarBufferEscrituraCorreo()
{
    for(i=0;i<MAX_long_correo;i++)
    {
        buffer_correo[i]=0;
    }
}
void BorrarBufferEscrituraNombre()
{
    for(i=0;i<MAX_long_nombre;i++)
    {
        buffer_nombre[i]=0;
    }
}



// Función para comprobar la ID, devuelve ok=x en función del resultado
int ComprobarID(int posicion)
{
```

```
    int posicion_id = posicion+1;
    int j;
    int ok=1;
    int IDflash;

    numero_usuario = 0;
    while(ok == 1)
    {
        // comparamos cada digito de la ID con la flash
        for (j=0;j<4;j++)
        {
            IDflash = HWREGB(posicion_id);
            // si coincide avanzamos a la siguiente posicion
            if (IDflash == cardID[j])
            {
                posicion_id++;
                // si han coincidido todos los numeros, lectura correcta , ok=2
                if(j==3)
                    ok = 2;
            }
            else
            {
                // si no ha coincidido el digito, pasamos a la siguiente ID
                posicion_id=posicion_id+SaltoUsuarios-j;
                // comprobamos si hay algo escrito, si no, no hay mas ID,
                // tarjeta no reconocida
                if(HWREGB(posicion_id) == 0xFF){
                    ok = 0;
                }
                else
                    numero_usuario++;
                //si hay otra ID posible volvemos al bucle y comprobamos con
esta nueva ID
                break;
            }
        }
    }
    return ok;
}



// TECLADO
void pantalla_teclado(short int campo){
    //    Lee_pantalla();
    indice_teclado = 0;
    SHIFT = false;
    while(ENTER == false)
    {
        Nueva_pantalla(150, 150, 150);

        get_teclado(campo);

        pantalla(campo);

        Dibuja();
    }
    ENTER = false;
}

void get_teclado(short int campo){
```

```
    // Leemos el registro, almacenar en tecla
    tecla = Lee_Reg(REG_TOUCH_TAG);


    // Si se puede seguir escribiendo en el vector
    switch (campo){
    case 0:{
        // Si se ha pulsado la pantalla
        if(POSX!=0x8000){
            // Si la tecla entra dentro de la tabla ASCII básica
            if (tecla > 0 && tecla<128){
                if (indice_teclado<MAX_long_correo){
                    // Evita que mantener pulsada la tecla escriba más de una
vez el valor en el vector
                    if (tecla!=tecla_p && tecla != 0){

                        TocaNota(0x50,60);
                        buffer_correo[indice_teclado]=tecla;
                        indice_teclado++;
                    }
                }
                else
                    indice_teclado=0;
            }
        }
        // Si se pulsa el botón de borrar
        if (DELETE == true){
            DELETE = false;
            // Si el indice_teclado es mayor que 0, disminuir
            if (indice_teclado>0)
            {

                TocaNota(0x50,60);
                indice_teclado--;
            }
            // Si es 0, saturar, evita que indice_teclado vaya a números
negativos
            else
                indice_teclado =0;
            // Borrar (poner a 0) valor del buffer de correo
            buffer_correo[indice_teclado] = 0;
        }
        if(SPACE == true){
            SPACE = false;
            buffer_correo[indice_teclado]=' ';
            indice_teclado++;
        }
        break;
    }
    case 1:{
        // Si se ha pulsado la pantalla
        if(POSX!=0x8000){
            // Si la tecla entra dentro de la tabla ASCII básica
            if (tecla > 0 && tecla<128){
                if (indice_teclado<MAX_long_nombre){
                    // Evita que mantener pulsada la tecla escriba más de una
vez el valor en el vector
                    if (tecla!=tecla_p && tecla != 0){

                        TocaNota(0x50,60);
                        buffer_nombre[indice_teclado]=tecla;
```

```
                        indice_teclado++;
                    }
                }
                else
                    indice_teclado=0;
            }
        }
        // Si se pulsa el botón de borrar
        if (DELETE == true){
            DELETE = false;
            // Si el indice_teclado es mayor que 0, disminuir
            if (indice_teclado>0)
            {

                TocaNota(0x50,60);
                indice_teclado--;
            }
            // Si es 0, saturar, evita que indice_teclado vaya a números
negativos
            else
                indice_teclado =0;
            // Borrar (poner a 0) valor del buffer de correo
            buffer_nombre[indice_teclado] = 0;
        }
        if(SPACE == true){
            SPACE = false;
            buffer_nombre[indice_teclado]=' ';
            indice_teclado++;
        }
        break;
    }
    case 3:{
        // Si se ha pulsado la pantalla
        if(POSX!=0x8000){
            // Si la tecla entra dentro de la tabla ASCII básica
            if (tecla > 0 && tecla<128){
                if (indice_teclado<long_clave){
                    // Evita que mantener pulsada la tecla escriba más de una
vez el valor en el vector
                    if (tecla!=tecla_p && tecla != 0){
                        TocaNota(0x50,60);
                        buffer_clave[indice_teclado]=tecla;
                        indice_teclado++;
                    }
                }
            }
        }
        // Si se pulsa el botón de borrar
        if (DELETE == true){
            DELETE = false;
            // Si el indice_teclado es mayor que 0, disminuir
            if (indice_teclado>0)
            {

                TocaNota(0x50,60);
                indice_teclado--;
            }
            // Si es 0, saturar, evita que indice_teclado vaya a números
negativos
            else
                indice_teclado =0;
            // Borrar (poner a 0) valor del buffer de correo
```

```
            buffer_clave[indice_teclado] = 0;
        }
        if(SPACE == true){
            SPACE = false;
            buffer_clave[indice_teclado]=' ';
            indice_teclado++;
        }
        break;
    }
    }

    // Actualizar valor de tecla_p
    tecla_p = tecla;
    if (ENTER == true){
        //        ENTER = false;
        indice_teclado = 0;
        // Meter llamada a escribir en flash el correo
    }
}


short int rect_h, rect_w, rect_x, rect_y;

void pantalla(short int campo){
    /*
     * BLOQUE DE CALCULOS
     */

    // Separacion entre teclado y borde pantalla
    tecla_borde = 2*fila1_inicio_x;  // pix

    /*
     * Calculo de tamaño de tecla "estándar". Teniendo en cuenta que no se manda
OPT_x.
     * Se calcula con la fila superior, y se ajustan las demás.
     * (Ancho pantalla - espacio entre teclado y borde - 3 * n letras)/(n letras)
     */
    tecla_w = (HSIZE - tecla_borde - 3 * (n_tecla_fila_1 - 1)) / n_tecla_fila_1;
// Ancho teclas segun fila 1

    // Calculo inicio filas 0, 2 y 3
    fila0_inicio_y = VSIZE - n_filas * (tecla_h + tecla_separacion);

    fila1_inicio_y = fila0_inicio_y + tecla_separacion + tecla_h;

    fila2_inicio_x = HSIZE / 2 - tecla_w * n_tecla_fila_2 / 2 - tecla_borde /
2;  // Inicio fila 2
    fila2_inicio_y = fila1_inicio_y + tecla_separacion + tecla_h;

    fila3_inicio_x = HSIZE / 2 - tecla_w * n_tecla_fila_3 / 2 - tecla_borde /
2;  // Inicio fila 3
    fila3_inicio_y = fila1_inicio_y + 2*tecla_separacion + 2*tecla_h;

    fila4_inicio_y = fila1_inicio_y + 3*tecla_separacion + 3*tecla_h;

    // Calculo anchura rectángulos
    fila1_rect_w = n_tecla_fila_1 * (3 + tecla_w) - 3;  // El 3 sale del tamaño
entre botones (VER DATASHEET, PAG 187 FT800 PROGRAMMER GUIDE)
    fila2_rect_w = n_tecla_fila_2 * (3 + tecla_w) - 3;  // El 3 sale del tamaño
entre botones (VER DATASHEET, PAG 187 FT800 PROGRAMMER GUIDE)
    fila3_rect_w = n_tecla_fila_3 * (3 + tecla_w) - 3;  // El 3 sale del tamaño
entre botones (VER DATASHEET, PAG 187 FT800 PROGRAMMER GUIDE)
```

```
    /*
     * Pintar la cadena hasta el momento.
     * Añadir un rectángulo sobre el texto.
     */
    rect_w = fila1_rect_w;
    rect_h = tecla_h;
    rect_x = fila1_inicio_x;
    rect_y = fila0_inicio_y - rect_h - 5;


    // Rectángulo
    Comando(CMD_BEGIN_RECTS);

    ComVertex2ff(rect_x, rect_y);
    ComVertex2ff(rect_x + rect_w, rect_y + rect_h);

    ComColor(255, 210, 201);
    //    ComVertex2ff(rect_x + rect_w, rect_y + rect_h);
    //    ComVertex2ff(rect_x + rect_w, rect_y);
    Comando(CMD_END);

    /*
     * Comprobar si se ha escrito algo. En caso
     * afirmativo, mostrar por pantalla. En caso
     * negativo, mostrar un mensaje invitando a hacerlo.
     */


    switch (campo){
    case 0:{
        if (indice_teclado != 0){
            ComColor(0,0,0);
            ComTXT(HSIZE/2, fila0_inicio_y - 5 - rect_h / 2, 21, OPT_CENTER,
buffer_correo);
        }
        else{
            ComColor(201,201,201);
            ComTXT(fila1_inicio_x + 5, fila0_inicio_y - 5 - rect_h / 2, 21,
OPT_CENTERY, "Introduzca su correo");
        }
        break;
    }
    case 1:{
        if (indice_teclado != 0){
            ComColor(0,0,0);
            ComTXT(HSIZE/2, fila0_inicio_y - 5 - rect_h / 2, 21, OPT_CENTER,
buffer_nombre);
        }
        else{
            ComColor(201,201,201);
            ComTXT(fila1_inicio_x + 5, fila0_inicio_y - 5 - rect_h / 2, 21,
OPT_CENTERY, "Introduzca nombre y apellidos");
        }
        break;
    }
    case 3:{
        if (indice_teclado != 0){
            ComColor(0,0,0);
            ComTXT(HSIZE/2, fila0_inicio_y - 5 - rect_h / 2, 21, OPT_CENTER,
buffer_clave);
```

```
        }
        else{
            ComColor(201,201,201);
            ComTXT(fila1_inicio_x + 5, fila0_inicio_y - 5 - rect_h / 2, 21,
OPT_CENTERY, "Introduzca clave recibida");
        }
        break;
    }
    }

    ComColor(0xff,0xff,0xff);

    /*
     * Para incluir las teclas se utiliza la función Boton, a la que le entran
     * las posiciones x,y; la altura y anchura, y la cadena a mostrar. La función
     * devuelve si se está pulsando el botón, y comparando con el estado
anterior,
     * se decide si situar las variables correspondiente a true o false.
     */

    // Incluir tecla shift
    shift_boton = Boton(fila1_inicio_x, fila3_inicio_y, (HSIZE - fila3_rect_w -
tecla_borde) / 2 - 3, tecla_h, 21, "Blq May");
    if (shift_boton == 1 && shift_boton_p == 0 && SHIFT == false){

        TocaNota(0x50,50);
        SHIFT = true;
    }
    else if (shift_boton == 1 && shift_boton_p == 0 && SHIFT == true){

        TocaNota(0x50,50);
        SHIFT = false;
    }
    shift_boton_p = shift_boton;

    // Incluir tecla options
    options_boton = Boton(fila1_inicio_x, fila4_inicio_y, (HSIZE - fila3_rect_w
- tecla_borde) / 2 - 3, tecla_h, 21, "?123");
    if (options_boton == 1 && options_boton_p == 0 && OPTIONS == false){

        TocaNota(0x50,50);
        OPTIONS = true;
    }
    else if (options_boton == 1 && options_boton_p == 0 && OPTIONS == true){

        TocaNota(0x50,50);
        OPTIONS = false;
    }
    options_boton_p = options_boton;

    // Incluir tecla de enter
    enter_boton = Boton(fila3_inicio_x + fila3_rect_w + 3, fila4_inicio_y,
(HSIZE - fila3_rect_w - tecla_borde) / 2 - 3, tecla_h, 21, "Enter");
    if (enter_boton == 1 && enter_boton_p == 0){

        TocaNota(0x50,50);
        ENTER = true;
    }
    enter_boton_p = enter_boton;

    // Incluir tecla de borrar
```

```
    delete_boton = Boton(fila3_inicio_x + fila3_rect_w + 3, fila3_inicio_y,
(HSIZE - fila3_rect_w - tecla_borde) / 2 - 3, tecla_h, 21, "DEL");
    if (delete_boton == 1 && delete_boton_p == 0){
        DELETE = true;
    }
    delete_boton_p = delete_boton;



    // Si se ha pulsado una tecla válida, almacenarla en la variable para que
se muestre sin efecto 3D
    int letra_pulsada = 0;
    if (tecla > 0 && tecla<128)
        letra_pulsada = (int)tecla;
    else
        letra_pulsada = 0;

    /*
     * Mostrar por pantalla cada una de las filas del teclado,
     * teniendo en cuenta el estado de SHIFT, OPTIONS Y ENTER.
     */
    if (OPTIONS == true){
        ComTeclas(fila1_inicio_x, fila0_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila0);
        ComTeclas(fila1_inicio_x, fila1_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila1_especiales);
        ComTeclas(fila2_inicio_x, fila2_inicio_y, fila2_rect_w, tecla_h, 21,
letra_pulsada, fila2_especiales);
        ComTeclas(fila3_inicio_x, fila3_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila3_especiales);
        ComTeclas(fila3_inicio_x, fila4_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila4);
    }
    else if (SHIFT == true || campo == 3){
        ComTeclas(fila1_inicio_x, fila0_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila0);
        ComTeclas(fila1_inicio_x, fila1_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila1_mayus);
        ComTeclas(fila2_inicio_x, fila2_inicio_y, fila2_rect_w, tecla_h, 21,
letra_pulsada, fila2_mayus);
        ComTeclas(fila3_inicio_x, fila3_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila3_mayus);
        ComTeclas(fila3_inicio_x, fila4_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila4);
    }
    else{
        ComTeclas(fila1_inicio_x, fila0_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila0);
        ComTeclas(fila1_inicio_x, fila1_inicio_y, fila1_rect_w, tecla_h, 21,
letra_pulsada, fila1_minus);
        ComTeclas(fila2_inicio_x, fila2_inicio_y, fila2_rect_w, tecla_h, 21,
letra_pulsada, fila2_minus);
        ComTeclas(fila3_inicio_x, fila3_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila3_minus);
        ComTeclas(fila3_inicio_x, fila4_inicio_y, fila3_rect_w, tecla_h, 21,
letra_pulsada, fila4);
    }
    // Incluir tecla de espacio_
    space_boton = Boton(fila3_inicio_x + fila3_rect_w / n_tecla_fila_3,
fila4_inicio_y-1, fila3_rect_w * 5/7, tecla_h+2,21," ");
    if (space_boton == 1 && space_boton_p == 0){
        TocaNota(0x50,60);
```

```
        SPACE = true;
    }
    space_boton_p = space_boton;
}

// Funcion para contar segundos
void EsperaSeg( int segundos )
{
    // Reiniciar variable
    tiempo = 0;
    // segundos a esperar = 20 * segundos pues la interrupion es cada 50ms
    while(tiempo < segundos*20);
}
```

8.1.2 FT800_TIVA.h

De estos archivos sólo se incluirá el código estrictamente nuevo.

```
//-------------------------------------------------------------------------
//----FUNCIONES NUEVAS PARA EL PROYECTO DE CURSO, PARTE DE BITMAPS------
//-------------------------------------------------------------------------

// Valores correspondientes a los bits fijos de los comandos cmd_xyz
#define BMP_SIZE_HANDLE         0x08000000
#define BMP_SIZE_WX_WY_WRAPY    0x00000000

#define BMP_LAYOUT_HANDLE       0x07000000
#define BMP_LAYOUT_FMRT_RGB565  0x00380000
#define BMP_LAYOUT_FMRT_RGB332  0x00200000

#define BMP_SOURCE_HANDLE       0x01000000

#define COLOR_A_HANDLE          0x10000000

void cmd_bmp_layout(int type, int lnstrd,int hght);
void cmd_bmp_size(int wdth,int hght);
void cmd_bmp_source(int addr);
void cmd_alpha_a(int alpha);
void dibujar_imagen_externa(int ram_addr, int ancho, long int alto, int x, int
y ,int param, int tipo_RGB);
void    almacenar_imagen_externa(int    flash_addr,    int    ram_addr,    int
nbytes_imagen);
```

8.1.3 ft800_TIVA.c

De estos archivos sólo se incluirá el código estrictamente nuevo.

```
//-------------------------------------------------------------------------
//-------------------FUNCIONES NUEVAS PARA EL PROYECTO DE CURSO--------
//-------------------------------------------------------------------------

// Función para mandar por pantalla el comando correspondiente a BITMAP_LAYOUT
void cmd_bmp_layout(int type, int lnstrd,int hght){
    // Variables internas utilizadas en la función
    unsigned long total = 0;
    unsigned long linestride = 0;
    unsigned long height = 0;
```

```
    if (type == 0)  // RGB565
    // Mover hacia la izquierda el valor de linestride
        linestride = 2*lnstrd << 9;
    else            // RGB332
        linestride = lnstrd << 9;

    // Almacenar la altura
    height = hght;

    if (type == 0)
    // Calcular el total como la suma de todos los valores
        total = BMP_LAYOUT_HANDLE + BMP_LAYOUT_FMRT_RGB565 + linestride +
height;
    else
        total = BMP_LAYOUT_HANDLE + BMP_LAYOUT_FMRT_RGB332 + linestride +
height;
    // Mandar el comando (32 bits)
    Comando(total);
}

// Función para mandar por pantalla el comando correspondiente a BITMAP_SIZE
void cmd_bmp_size(int wdth,int hght){
    // Variables internas utilizadas en la función
    unsigned long total = 0;
    unsigned long width = 0;
    unsigned long height = 0;

    // Mover hacia la izquierda el valor de la anchura
    width = wdth << (9);

    // Almacenar la altura
    height = hght;

    // Calcular el total como la suma de todos los valores
    total = BMP_SIZE_HANDLE + BMP_SIZE_WX_WY_WRAPY + width + height;

    // Mandar el comando (32 bits)
    Comando(total);
}

// Función para mandar por pantalla el comando correspondiente a BITMAP_SOURCE
void cmd_bmp_source(int addr){
    // Variables internas utilizadas en la función
    unsigned long total = 0;

    // Calcular el total como la suma de todos los valores
    total = BMP_SOURCE_HANDLE + (unsigned long)addr;

    // Mandar el comando (32 bits)
    Comando(total);
}

// Función para mandar por pantalla el comando correspondiente a ALPHA_A.
Modifica la transparencia del bitmap
```

```
void cmd_alpha_a(int alpha){
    // Variables internas utilizadas en la función
    unsigned long total = 0;

    // Calcular el total como la suma de todos los valores
    total = COLOR_A_HANDLE + (unsigned long)alpha;

    // Mandar el comando (32 bits)
    Comando(total);
}


// Muestra la imagen almacenada en memoria
void dibujar_imagen_externa(int ram_addr, int ancho, long int alto, int x, int
y ,int param, int tipo_RGB){
    // Preparar nueva pantalla
    //    Nueva_pantalla(255, 255, 255);

    // Dibujar BITMAP
    Comando(CMD_BEGIN_BMP);

    // Comando para BITMAP_SOURCE   (ft800 prog guide - pg. 96)
    cmd_bmp_source(ram_addr);

    // Comando para BITMAP_LAYOUT    (ft800 prog guide - pg. 88)       ||
https://www.ftdichip.com/Support/Documents/AppNotes/AN_303%20FT800%20Image%20
File%20Conversion.pdf , pg.11
    cmd_bmp_layout(tipo_RGB, ancho, alto);

    // Comando para BITMAP_SIZE       (ft800 prog guide - pg. 93)       ||
https://www.ftdichip.com/Support/Documents/AppNotes/AN_303%20FT800%20Image%20
File%20Conversion.pdf , pg.11
    cmd_bmp_size(ancho, alto);

    // Comando para ALPHA_A            (ft800 prog guide - pg. 117)     ||
https://www.ftdichip.com/Support/Documents/AppNotes/AN_303%20FT800%20Image%20
File%20Conversion.pdf , pg.11
    cmd_alpha_a(param);

    // Posicion en la pantalla, esquina superior izquierda
    ComVertex2ff(x, y);

    // Finalizar el bitmap
    Comando(CMD_END);

    // Dibujar lo almacenado en la pila de 4k
    //    Dibuja();
}

// Almacena en la memoria ram del FT800 la imagen almacenada en la flash del
TM4C mediante LM Flash Programmer
void    almacenar_imagen_externa(int    flash_addr,    int    ram_addr,    int
nbytes_imagen){
```

```
    // Preparar nueva pila
    Nueva_pantalla(255, 255, 255);

    // Iniciar el comando cmd_inflate
    Comando(CMD_INFLATE);

    // Posicion de memoria ram de FT800
    Comando(ram_addr);

    int img_ext_indice = 0;

    // Guardar en memoria la información
    for (img_ext_indice = 0; img_ext_indice < nbytes_imagen; img_ext_indice
++){
        EscribeRam8(HWREGB(flash_addr+img_ext_indice));
        //       UARTprintf("%d\r\n",(int)HWREGB(flash_addr+img_ext_indice));
    }
    // Completar hasta llegar a 4 bytes
    PadFIFO();

    // Mandar fin de cmd_inflate
    Comando(CMD_END);

    // Dibuja la pila
    Dibuja();
}
```

8.1.4 smtp_library.h

```
#ifndef SMPT_LIBRARY_H_
#define SMPT_LIBRARY_H_

// Definiciones para la configuración del reloj del sistema (ticks)
#define SYSTICKHZ               100
#define SYSTICKMS               (1000 / SYSTICKHZ)

// Definiciones para la prioridad de las interrupciones
#define SYSTICK_INT_PRIORITY    0x80
#define ETHERNET_INT_PRIORITY   0xC0

// Dirección IP
uint32_t g_ui32IPAddress;

// Variable booleana volatil para el led. Parpadea a la frecuencia de SISTICKHZ
volatile bool g_bLED;

void lwIPHostTimerHandler(void);
void SysTickIntHandler(void);
static void close_conn (struct tcp_pcb *pcb);
static err_t echo_recv( void *arg, struct tcp_pcb *pcb, struct pbuf *p, err_t
err);
static tcp_connected_fn connected_callback(void *arg, struct tcp_pcb *tpcb,
err_t err);
void SMTP_start(void);
void config_SMTP(uint32_t RELOJ);
```

```
#endif /* SMPT_LIBRARY_H_ */
```

8.1.5 smtp_library.c

```
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"

#include "driverlib2.h"
#include "drivers/pinout.h"
#include "lwip/tcp.h"

#include "utils/locator.h"
#include "utils/lwiplib.h"
#include "utils/ustdlib.h"

#include "smtp_library.h"
#include "variables_globales.h"



// Variable buffer para recibir el mensaje devuelto por el servidor SMTP
char g_pctcpBuffer[4096] = {0};

// Variable que indica que se ha encontrado una IP
bool IP_establecida_una_vez = false;

// Cadena para almacenar la direccion IP
char cadena_IP[16];

// Estado actual de la comunicación SMTP
short int estado_actual = 0;

/*
 * Vector que almacena la secuencia de respuestas que se reciben
 * desde el servidor smtp.gmail.com. Se comprobará que el código
 * recibido después de cada intercambio de mensajes sea el adecuado,
 * y en caso contrario se cerrará la conexión y se reintentará
 * hasta un máximo de N veces mandar el correo electrónico.
 */
short int secuencia_respuestas[] = {220, 250, 555, 250, 334, 334, 235, 250,
250, 354, 250, 221};

// Almacena el código recibido para comparar
short int codigo_recibido = 0;

// Buffer en el que se almacena el código recibido
char buffer_codigo_recibido[3]={0};
```

```
// Actua de indice en bucles for
char indice;


// Funcion requerida por la librería lwIP.
void lwIPHostTimerHandler(void)
{
    uint32_t IP_obt_lwIPAddrGet;

    // Obtener IP actual
    IP_obt_lwIPAddrGet = lwIPLocalIPAddrGet();

    // Si la IP obtenita es distinta a la anterior
    if(IP_obt_lwIPAddrGet != g_ui32IPAddress)
    {
        // Si lwIPLocalIPAddrGet() devuelve 0xffff..., aún no hay un link
activo
        if(IP_obt_lwIPAddrGet == 0xffffffff)
        {
        }
        // Si lwIPLocalIPAddrGet() devuelve 0x000..., aún no hay direccion IP
        else if(IP_obt_lwIPAddrGet == 0)
        {
        }
        else
        {
            // Permite entrar en el if siguiente
            IP_establecida_una_vez = true;
        }

        // Guardar IP
        g_ui32IPAddress = IP_obt_lwIPAddrGet;
    }

    // Si se obtiene IP, mostrar
    if((IP_obt_lwIPAddrGet != 0) && (IP_obt_lwIPAddrGet != 0xffffffff) &&
IP_establecida_una_vez == true)
    {
        IP_establecida_una_vez = false;
    }
}

// Función interrupciones de los ticks del sistema
void SysTickIntHandler(void)
{
    // LLamada al temporizadaor del lwIP
    lwIPTimer(SYSTICKMS);

    // Cambiar estado del led
    g_bLED = true;
}

// Devolución de llamada para cerrar la conexión
```

```
static void close_conn (struct tcp_pcb *pcb)
{
    // A todas las funciones se les pasa un parámetro nulo, de manera que no
recibe o maneja argumentos

    tcp_arg(pcb, NULL);
    tcp_sent(pcb, NULL);
    tcp_recv(pcb, NULL);
    tcp_close(pcb);
}

/*
 * Defines con aquellos parámetros correspondiente con la
 * comunicación entre cliente (TM4C) y servidor (smtp.gmail.com)
 * del protocolo SMTP que permanecen "constantes". Estos parámetros
 * son constantes en todos los correos. Lo que debe modificarse es el
 * correo electronico del usuario al que se le quiere enviar el código
 * de verificación.
 */
#define SMTP_CMD_EHLO               "HELO GOOGLE"
#define SMTP_CMD_EHLO_LEN           11

#define SMTP_CMD_STARTTLS           "STARTTLS AUTH LOGIN"
#define SMTP_CMD_STARTTLS_LEN       19

#define SMTP_CMD_AUTHLOGIN          "AUTH LOGIN"
#define SMTP_CMD_AUTHLOGIN_LEN      10

#define SMTP_CMD_MAIL_FROM          "MAIL FROM: <"
#define SMTP_CMD_MAIL_FROM_LEN      12

#define SMTP_CMD_RCPT_TO            "RCPT TO: <"
#define SMTP_CMD_RCPT_TO_LEN        10

#define SMTP_CMD_DATA               "DATA"
#define SMTP_CMD_DATA_LEN           4

#define SMTP_CMD_FROM               "From: <"
#define SMTP_CMD_FROM_LEN           7

#define SMTP_CMD_TO                 "To: <"
#define SMTP_CMD_TO_LEN             5

#define SMTP_CMD_SUBJECT          "Subject: TIVA LOCK - Código de verificación"
#define SMTP_CMD_SUBJECT_LEN        43
#define SMTP_CMD_GENERIC_END        ">\r\n"
#define SMTP_CMD_GENERIC_END_LEN    3


#define SMTP_CMD_BODY_FINISHED      "\r\n.\r\n"
#define SMTP_CMD_BODY_FINISHED_LEN  5

#define SMTP_CMD_QUIT               "QUIT"
#define SMTP_CMD_QUIT_LEN           4
```

```
#define SMTP_USER                   "c21hcnQubG9jay50bTRjQGdtYWlsLmNvbQ=="
#define SMTP_USER_LEN               36
#define SMTP_PASS                   "dG00Yy5zbTRydC5sMGNr"
#define SMTP_PASS_LEN               20
#define SMTP_EMAIL_FROM             "smart.lock.tm4c@gmail.com"
#define SMTP_EMAIL_FROM_LEN         25
#define SMTP_EMAIL_TO               "thisisnotspamatalltrustme@gmail.com"
#define SMTP_EMAIL_TO_LEN           35

#define SMTP_TEST               "MENSAJE DESDE TIVA TM4C4912"
#define SMTP_TEST_LEN           27

#define SMTP_CRLF                "\r\n"
#define SMTP_CRLF_LEN            2

// 0 si no se envía, 1 si se envía
int correo_exito = 2;

// Lleva cuenta de reintentos
int reintentos = 0;

int long_mensaje = 0;
// Funcion que maneja la información devuelta por el host
static err_t comunicacion_SMTP( void *arg, struct tcp_pcb *pcb, struct pbuf
*p, err_t err)
{

    int ind_rand = 0;
    char *data_rcvd;    // Caracter en el que se almacena el puntero que apunta
a la posicion de memoria de datos recibidos
    char cuerpo_mensaje[] = "\n\rAqui tiene el codigo de verificacion para TIVA
LOCK \n\rSi no lo ha solicitado contacte con el administrador.";
    int long_cuerpo = 0;
    // Si no hay errores y p es no null
    if (err == ERR_OK && p != NULL)
    {

        // Llamada tcp_recved cuando se han procesado los datos
        tcp_recved(pcb, p->tot_len);

        // Almacenar el puntero a la información recibida
        data_rcvd = (char *)p->payload;



        // Se almacena en el buffer el código recibido
        for (indice=0;indice<3;indice++){
            buffer_codigo_recibido[indice] = *(data_rcvd + indice);
        }

        // Convertir cadena con código a short int
        codigo_recibido = atoi(buffer_codigo_recibido);
```

```
        /*
         * Antes de entrar en el switch-case se comprueba que el código
         * de confirmación recibido se corresponde con el que almacenado
         * en el vector de respuestas. En caso contrario, se cerrará la
         * conexión, y se aumentará el contador de intentos.
         */
        if (codigo_recibido != secuencia_respuestas[estado_actual]){
            estado_actual = 0;
            correo_exito = 0;
            reintentos++;
            close_conn(pcb);    // Cerrar la conexión
        }
        else{
            switch (estado_actual){

            // HELO GOOGLE
            case 0:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_EHLO,               SMTP_CMD_EHLO_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // STARTTLS
            case 1:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_STARTTLS,       SMTP_CMD_STARTTLS_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // HELO GOOGLE
            case 2:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_EHLO,               SMTP_CMD_EHLO_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // AUTH LOGIN
            case 3:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_AUTHLOGIN,     SMTP_CMD_AUTHLOGIN_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // USUARIO, BASE 64
            case 4:{
                // Se rellena el buffer de salida con el mensaje
```

```
                tcp_write(pcb,SMTP_USER, SMTP_USER_LEN, TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // CONTRASENA, BASE 64
            case 5:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_PASS, SMTP_PASS_LEN, TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // MAIL FROM:
            case 6:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_MAIL_FROM,      SMTP_CMD_MAIL_FROM_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_EMAIL_FROM,           SMTP_EMAIL_FROM_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CMD_GENERIC_END,  SMTP_CMD_GENERIC_END_LEN,
TCP_WRITE_FLAG_COPY);
                break;
            }

            // RCPT TO:
            case 7:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_RCPT_TO,         SMTP_CMD_RCPT_TO_LEN,
TCP_WRITE_FLAG_COPY);
                // AQUÍ SE METE LA DIRECCIÓN DE CORREO A LA QUE SE QUIERE
ENVIAR EL CÓDIGO
                // tcp_write(pcb, DIRECCION DE CORREO, LONGITUD DIRECCION DE
CORREO, 0);
                //                              tcp_write(pcb,SMTP_EMAIL_FROM,
SMTP_EMAIL_FROM_LEN, TCP_WRITE_FLAG_COPY);
                //                char hola[100] = "pablo5leon5@gmail.com";
                long_mensaje = 0;
                while (correorecuperado[long_mensaje]!= 0)
                    long_mensaje++;
                // cuerpo del mensaje
                tcp_write(pcb,&correorecuperado[0],             long_mensaje,
TCP_WRITE_FLAG_COPY);



                tcp_write(pcb,SMTP_CMD_GENERIC_END,  SMTP_CMD_GENERIC_END_LEN,
TCP_WRITE_FLAG_COPY);
                break;
            }

            // DATA
            case 8:{
                // Se rellena el buffer de salida con el mensaje
```

```
                tcp_write(pcb,SMTP_CMD_DATA,                SMTP_CMD_DATA_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // INFORMACIÓN DEL MENSAJE
            case 9:{

                // from
                tcp_write(pcb,SMTP_CMD_FROM,                SMTP_CMD_FROM_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_EMAIL_FROM,            SMTP_EMAIL_FROM_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CMD_GENERIC_END,  SMTP_CMD_GENERIC_END_LEN,
TCP_WRITE_FLAG_COPY);

                // to
                tcp_write(pcb,SMTP_CMD_TO,                    SMTP_CMD_TO_LEN,
TCP_WRITE_FLAG_COPY);
                // AQUÍ SE METE LA DIRECCIÓN DE CORREO A LA QUE SE QUIERE
ENVIAR EL CÓDIGO
                // tcp_write(pcb, DIRECCION DE CORREO, LONGITUD DIRECCION DE
CORREO, 0);
                tcp_write(pcb,SMTP_EMAIL_FROM,
SMTP_EMAIL_FROM_LEN,TCP_WRITE_FLAG_COPY);


                tcp_write(pcb,SMTP_CMD_GENERIC_END,  SMTP_CMD_GENERIC_END_LEN,
TCP_WRITE_FLAG_COPY);

                // subject
                tcp_write(pcb,SMTP_CMD_SUBJECT,          SMTP_CMD_SUBJECT_LEN,
TCP_WRITE_FLAG_COPY);
                // AQUÍ SE METE EL SUJETO DEL CORREO ELECTRONICO
                // tcp_write(pcb, SUBJECT, LONGITUD SUBJECT, 0);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);

                // enter
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);

                // cuerpo del mensaje
                for (ind_rand=0;ind_rand<long_clave+1;ind_rand++)
                    rand_number[ind_rand] = 0;
                long_mensaje = 0;
                srand(aleatoriedad);
                while (long_mensaje<long_clave+1){
                    if(long_mensaje<long_clave){
                        //
while(rand_number[long_mensaje]<48 || rand_number[long_mensaje]>90)
                        while(rand_number[long_mensaje]<'0'                ||
(rand_number[long_mensaje]>'9'    &&    rand_number[long_mensaje]<'A')    ||
(rand_number[long_mensaje]>'Z') )
                            rand_number[long_mensaje] = rand() % 256;
```

```
                    }
                    else
                        rand_number[long_mensaje] = 0;
                    long_mensaje++;
                }
                // cuerpo del mensaje
                while (cuerpo_mensaje[long_cuerpo]!= 0)
                    long_cuerpo++;
                tcp_write(pcb,&rand_number[0],         long_mensaje         ,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,&cuerpo_mensaje[0],        long_cuerpo        ,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);
                // finaliza mensaje con enter . enter
                tcp_write(pcb,SMTP_CMD_BODY_FINISHED,
SMTP_CMD_BODY_FINISHED_LEN, TCP_WRITE_FLAG_COPY);
                break;
            }

            // QUIT
            case 10:{
                // Se rellena el buffer de salida con el mensaje
                tcp_write(pcb,SMTP_CMD_QUIT,               SMTP_CMD_QUIT_LEN,
TCP_WRITE_FLAG_COPY);
                tcp_write(pcb,SMTP_CRLF, SMTP_CRLF_LEN, TCP_WRITE_FLAG_COPY);

                correo_exito = 1;
                break;
            }
            }
            // Se obliga a mandar el mensaje
            tcp_output(pcb);

            // Aumentar el estado actual
            estado_actual++;
        }


        // Liberar y "dereferenciar" aquellas cadenas que hayan quedado
residuales
        pbuf_free(p);
    }
    else
    {
        // Si se produce un error, liberar el buffer
        pbuf_free(p);
    }

    // Si el host remoto solicita cerrar la conexión
    if(err == ERR_OK && p == NULL)
    {
        close_conn(pcb);
    }
    return ERR_OK;
```

```
}


// Función llamada una vez se establece conexión, prepara el estado del intercambio
static tcp_connected_fn connected_callback(void *arg, struct tcp_pcb *tpcb, err_t err){
    // Colocar el estado actual al comienzo del intercambio de mensajes entre el host y el cliente
    estado_actual = 0;
    return ERR_OK;
}

// Funcion que inicializa la conexión para el SMTP
void SMTP_start(void)
{
    // Crear una estructura de tipo tcp_pcb
    struct tcp_pcb *tcp_pcb;

    // Nuevo identificador de conexión TCP
    tcp_pcb = tcp_new();

    // Cualquier dirección, puerto 587 (TLS gmail)
    tcp_bind(tcp_pcb, IP_ADDR_ANY, 587);

    // Variable en la que se almacena una de las muchas direcciones IP de los servidores SMTP de GMAIL
    struct ip_addr IP_smtp;
    IP4_ADDR(&IP_smtp,74,125,133,109);

    // Mientras no se tenga direccion IP
    while ((g_ui32IPAddress == 0) || (g_ui32IPAddress == 0xffffffff)){
        SysCtlDelay(40000*10);
    }

    // Conectarse al servidor SMTP de GMAIL, llamar a connected_callback
    tcp_connect(tcp_pcb,       &IP_smtp,       587,       (tcp_connected_fn) connected_callback);


    // Indicar qué función será llamada cuando se reciban datos del servidor
    tcp_recv(tcp_pcb, comunicacion_SMTP);
}

void config_SMTP(uint32_t RELOJ){
    uint32_t ui32User0, ui32User1;
    uint8_t pui8MACArray[8];


    //
    // Make sure the main oscillator is enabled because this is required by
    // the PHY.  The system must have a 25MHz crystal attached to the OSC
    // pins. The SYSCTL_MOSC_HIGHFREQ parameter is used when the crystal
    // frequency is 10MHz or higher.
```

```
    //
    SysCtlMOSCConfigSet(SYSCTL_MOSC_HIGHFREQ);


    // Configurar los pines ETHERNET del microcontrolador
    PinoutSet(true, false); // Mandando true/false se activan los pines del
ETHERNET y no lo del bus USB


    // Configurar interrupciones de ticks del sistema
    SysTickPeriodSet(RELOJ / SYSTICKHZ);
    SysTickEnable();
    SysTickIntEnable();
    //
    // Configure the hardware MAC address for Ethernet Controller filtering of
    // incoming packets.  The MAC address will be stored in the non-volatile
    // USER0 and USER1 registers.
    //
    FlashUserGet(&ui32User0, &ui32User1);
    if((ui32User0 == 0xffffffff) || (ui32User1 == 0xffffffff))
    {
        //
        // We should never get here.  This is an error if the MAC address has
        // not been programmed into the device.  Exit the program.
        // Let the user know there is no MAC address.
        //
        while(1)
        {
        }
    }

    //
    // Convert the 24/24 split MAC address from NV ram into a 32/16 split MAC
    // address needed to program the hardware registers, then program the MAC
    // address into the Ethernet Controller registers.
    //
    pui8MACArray[0] = ((ui32User0 >>  0) & 0xff);
    pui8MACArray[1] = ((ui32User0 >>  8) & 0xff);
    pui8MACArray[2] = ((ui32User0 >> 16) & 0xff);
    pui8MACArray[3] = ((ui32User1 >>  0) & 0xff);
    pui8MACArray[4] = ((ui32User1 >>  8) & 0xff);
    pui8MACArray[5] = ((ui32User1 >> 16) & 0xff);

    //
    // Initialize the lwIP library, using DHCP.
    //
    lwIPInit(RELOJ, pui8MACArray, 0, 0, 0, IPADDR_USE_DHCP);

    //    SMTP_start();

    // Establecer prioridad de interrupciones. TICKS>EMAC
    IntPrioritySet(INT_EMAC0, ETHERNET_INT_PRIORITY);
    IntPrioritySet(FAULT_SYSTICK, SYSTICK_INT_PRIORITY);
```

```
}
```

8.1.6 variables_globales.h

```
#ifndef VARIABLES_GLOBALES_H_
#define VARIABLES_GLOBALES_H_


#define long_clave 5


// Buffer para almacenar lo recibido
extern char g_pctcpBuffer[4096];

// Variable que indica que se ha establecido la IP
extern bool IP_establecida_una_vez;

// Almacena la cadena IP
extern char cadena_IP[16];

// Estado actual del switch-case SMTP
extern short int estado_actual;

// Variables para almacenar correo, clave y otras variables
extern unsigned char correo[];
extern uint32_t correonum[100];
extern unsigned char correorecuperado[256];
extern char rand_number[long_clave+1];
extern unsigned long aleatoriedad;
extern int correo_exito;
extern int reintentos;


#endif /* VARIABLES_GLOBALES_H_ */
```