Taller de Proyecto 1

Trabajo Práctico N°1

CAO AGUSTIN LEONARDO | 1593/9
LIEBANA AGUSTIN | 722/6

Requerimientos del sistema

1. Requerimiento 1

1.1 Problema

Cuando el equipo se inicia deberá mostrar en la primer línea del LCD un reloj con formato HH:MM:SS y en la segunda línea el estado de la cerradura “CERRADO”.

1.2 Interpretación

Se mostrará en la pantalla principal, en la parte superior del display, un reloj con horas, minutos y segundos; el cuya máscara debe de actualizarse si el usuario no realiza ninguna operación. En la parte inferior del display indicará si la cerradura está cerrada o abierta.

2. Requerimiento 2

2.1. Problema

El sistema debe tener la clave numérica guardada por defecto 4321 de manera de poder activar o desactivar la cerradura. Si el usuario presiona la clave correcta se mostrará en la segunda línea “ABIERTO” durante 0.5 seg y luego volverá automáticamente al estado por defecto. Además se deberá activar un LED indicador en un terminal de salida.

2.2. Interpretación

Cuando se presiona la clave en el teclado matricial, se desbloquea el sistema y muestra un mensaje de ABIERTO en el display y se enciende el LED indicador durante 0.5 segundos. Luego se muestra la pantalla de inicio (con la hora y el mensaje CERRADO).

3. Requerimiento 3

3.1. Problema

Si la clave es incorrecta se mostrará el estado “DENEGADO” durante 0.2 seg y luego volverá automáticamente al estado por defecto.

3.2. Interpretación

Si se ingresa una clave incorrecta se muestra el mensaje DENEGADO por 0,2 segundos y luego se vuelve a la pantalla de inicio.

4. Requerimiento 4

4.1. Problema

Cuando se presione un tecla numérica se deberá mostrar ‘*’. para indicar que el sistema está recibiendo las entradas. El resto de las teclas no tienen ninguna función.

4.2. Interpretación

Los valores ingresados que correspondan a la contraseña deben representarse en el display con ’*’. El resto de las teclas no tienen que influir en los resultados del sistema

5. Requerimiento 5

5.1. Problema

La implementación deberá hacerse aplicando la arquitectura de planificador despachador temporizada con un tick de sistema (sEOS) y desarrollando bibliotecas de funciones básicas para utilizar el LCD y el teclado.

5.2. Interpretación

Se debe implementar un sistema operativo simple para realizar la planificación de las tareas.

Validación

Problema 1: Menú principal

Por defecto muestra el reloj y el estado de la cerradura

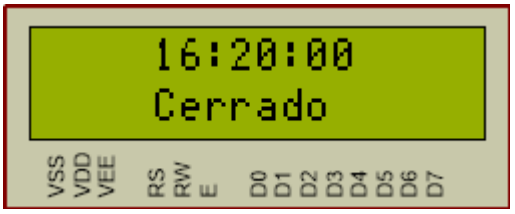


Figura 1: Pantalla por defecto

Problema 2: Abrir la cerradura

Para abrir la cerradura se debe de ingresar la clave (Figura 2A). En caso de ser correcta, se enderá el LED (Fig. 2B) mostrará la pantalla de cerradura abierta (Figura 2C).

Figura 2A: pantalla de ingreso de clave actual

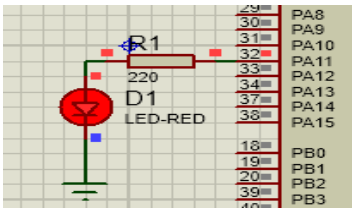


Figura 2B: LED encendido

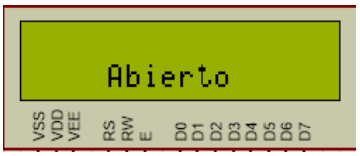


Figura 2C: pantalla indicando cerradura abierta

Problema 3: Clave incorrecta

Si se ingresa una clave incorrecta (Fig. 3A), se muestra la pantalla de acceso denegado (Fig. 3B).

Figura 3A: Pantalla de ingreso de clave actual

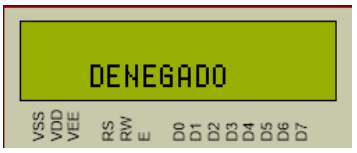


Figura 3B: Pantalla de finalización del cambio de clave

Problema 4: Cambio de hora, minutos y segundos

Cada vez que se presiona una tecla numérica, en el sistema se muestra en forma de "*" en la pantalla (Figura 4A)

Figura 4A: Pantalla de ingreso de clave actual

Problema 5: Temporización del programa

Para la temporización del programa se implementó una arquitectura de planificador despachador temporizada con un tick de sistema como sEOS (Simple Embedded Operating System). Dichos ticks de sistema se generan mediante el timer SysTick propio de las arquitecturas ARM Cortex (Fig. 5F). Los mismos ocurren cada milisegundo (1 ms), haciendo que la planificación de las tareas resulte fácil de realizar (Fig. 5A y 5B)

```
27      SysTick_Config(SystemCoreClock / 1000);
```

Fig. 5A: Programación del temporizador para generar interrupciones cada 1ms.

```
16  void SysTick_Handler(void)
17  {
18      seos_schedule_tasks();
19  }
```

Fig. 5B: Operación a realizar cada vez que ocurre una interrupción.

Cada vez que una interrupción ocurre, se controla el estado de los contadores que verifican si una tarea debe ejecutarse o no. Si una tarea ha acumulado una cantidad de ticks establecida, la misma se habilitará para ser ejecutada cuando sea oportuno una vez atendida la interrupción. Como ejemplo, podríamos decir que si quisiéramos actualizar el display cada 200ms, configuraremos el valor a alcanzar del contador del display (cont_lcd) en 200, ya que una interrupción equivale a 1ms. Eso puede verse en la figura 5C.

```
18  void seos_schedule_tasks(void)
19  {
20      if (++cont_mef >= 50)
21      {
22          flag_mef = 1;
23          cont_mef = 0;
24      }
25      if (++cont_keypad >= 60)
26      {
27          flag_keypad = 1;
28          cont_keypad = 0;
29      }
30      if (++cont_lcd >= 200)
31      {
32          flag_lcd = 1;
33          cont_lcd = 0;
34      }
35      if (++cont_clock >= 1000)
36      {
37          flag_clock = 1;
38          cont_clock = 0;
39      }
40  }
```

Fig. 5C: Programación de los contadores.

Desde el bucle infinito en el programa principal (Fig. 5D), se llama continuamente a una función que evalúa si se debe ejecutar una tarea específica (Fig. 5E).

```
29      while(1)
30      {
31          seos_dispatch_tasks();
32      }
```

\

Fig. 5D: Bucle del programa principal.

```
42  void seos_dispatch_tasks(void)
43  {
44
45      if (flag_mef)
46      {
47          mef_functions(key);
48          key = 0;
49          flag_mef = 0;
50      }
51      if (flag_keypad)
52      {
53          keypad_update(&key);
54          flag_keypad = 0;
55      }
56      if (flag_clock)
57      {
58          clock_tick();
59          flag_clock = 0;
60      }
61      if (flag_lcd)
62      {
63          lcd_refresh();
64          flag_lcd = 0;
65      }
66  }
```

Fig. 5E: Despachador de tareas.

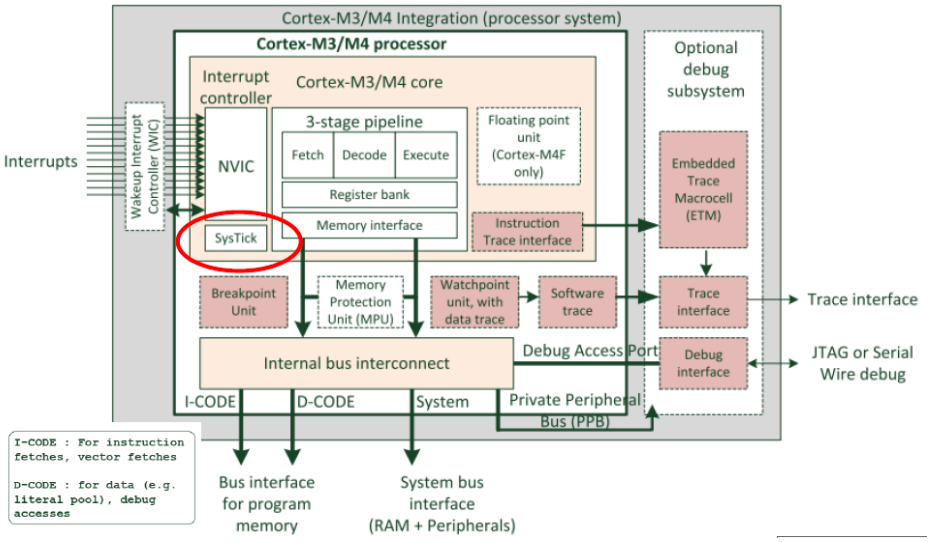


Fig. 5F: Módulo SysTick de la arquitectura ARM Cortex

(Fuente: REF:The Definitive Guide to Arm® Cortex®-M3 and Cortex®-M4 Processors", Joseph Yiu , 2014

Resolución del problema

El problema se pensó resolver como muestra la Figura 1

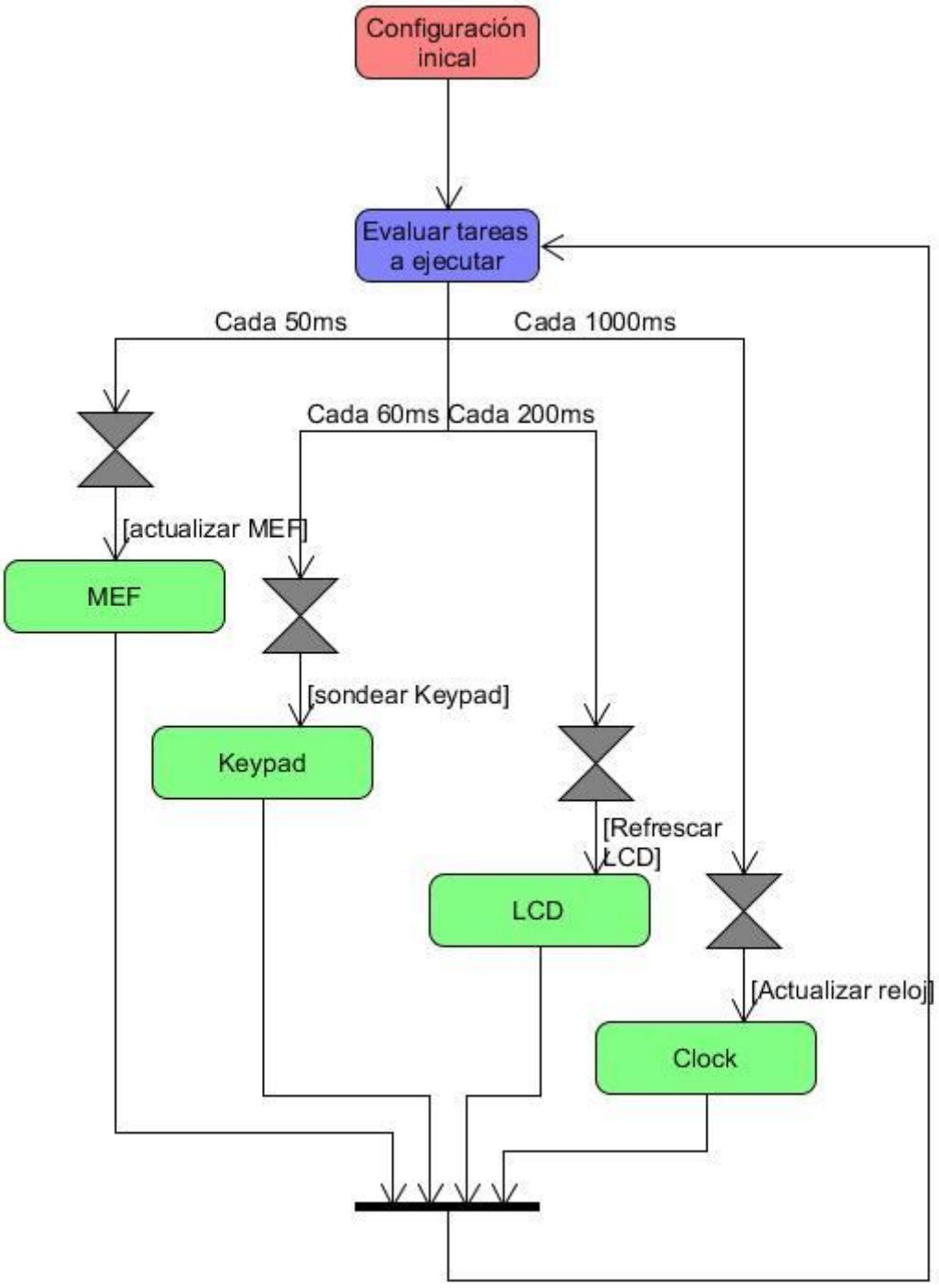


Figura 1: diagrama de flujo del programa

La figura 1 muestra el funcionamiento pensado para la solución del problema, este consiste en realizar la configuración de inicio de los componentes del programa. Luego, se pasa a evaluar las tareas a ejecutar en orden de prioridad. La tarea que más tiempo requiere al momento de ejecutarse es la del LCD, la cual requiere 18ms para actualizar todo el display. Para determinar esto, se realizaron pruebas de envío de información al módulo. Si ocurre un solapamiento de la información enviada, el simulador genera una advertencia. Se fueron probando distintos valores de temporización, siendo el mínimo necesario de 250us para mostrar un carácter común, y 8ms para un comando (Fig. 2). A 16 caracteres por línea, por dos líneas, y dos comandos para realizar el cursor, la tarea demora un total de 28 ms. Eso significa que ninguna tarea se solapa, y que tendremos tiempo de sobra para realizar una planificación segura de las tareas.

```
36  void lcd_send_cmd(uint8_t cmd)
37  {
38      GPIOB->BRR = (1 << LCD_RS); /* RS = 0 forcommand*/
39      lcd_put_value(cmd);
40      delay_us(10000);/* wait*/
41  }
42
43  void lcd_send_data(uint8_t data)
44  {
45      GPIOB->BSRR = (1 << LCD_RS); /* RS = 1 fordata */
46      lcd_put_value(data);
47      delay_us(250);/* wait*/
48  }
```

Fig. 2: Tiempos de delay mínimos y necesarios para el funcionamiento del display.

La figura 3 muestra cómo el usuario interactúa con el sistema y que operaciones puede realizar el usuario.

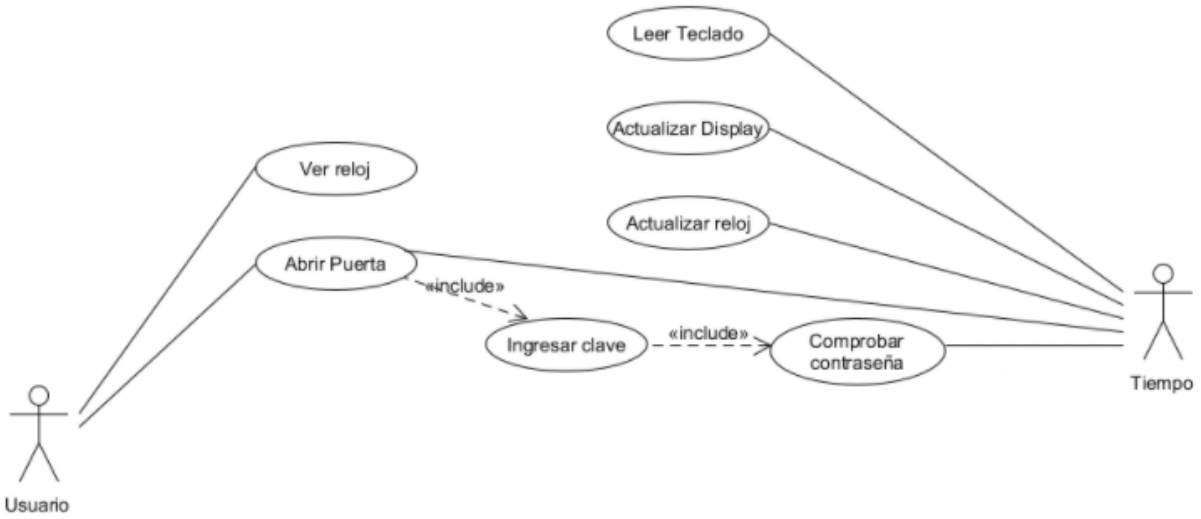


Figura 3: diagrama de casos de uso del sistema

La interfaz de usuario consiste en un display LCD 16x2 que muestra mensajes según el estado en el que se encuentre el sistema, y permite al usuario ingresar valores por el teclado matricial, que se ven reflejados en la pantalla del display. (Fig. 4)

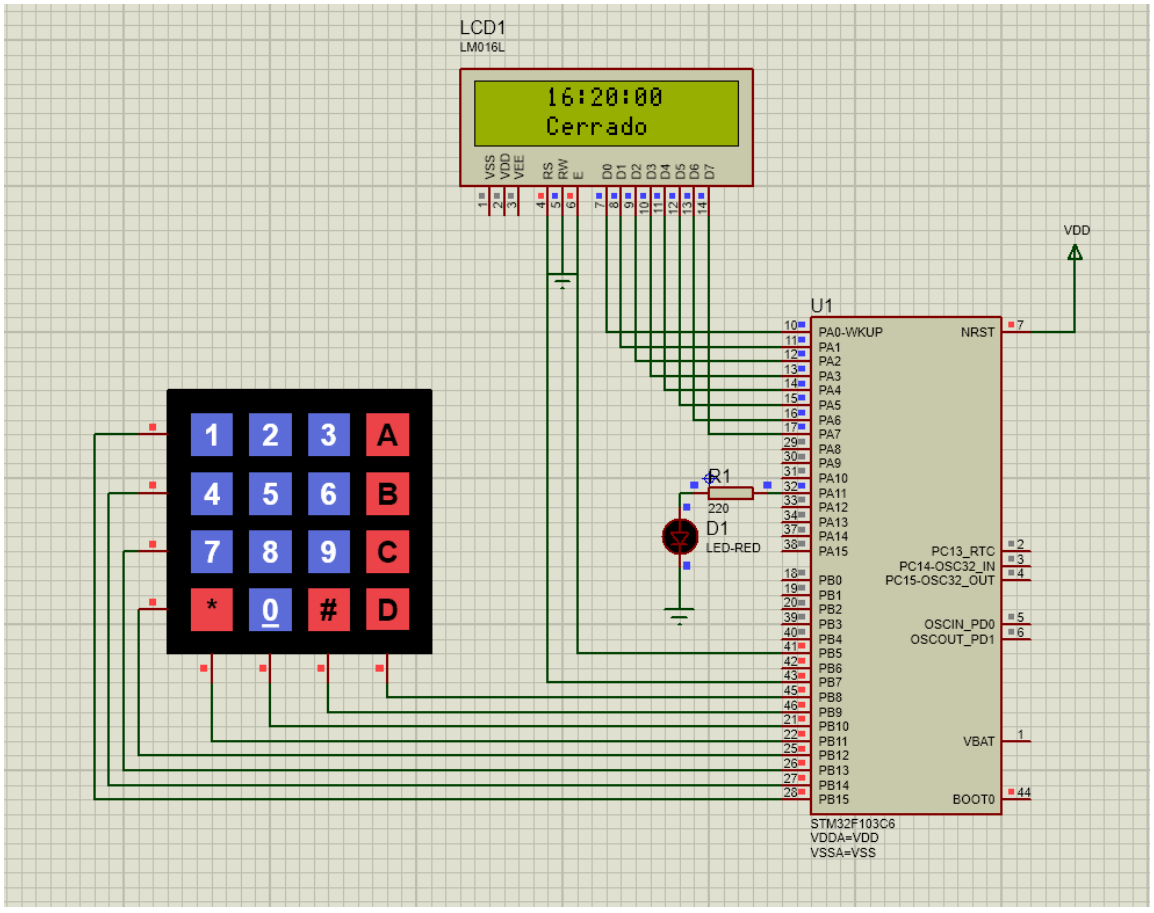


Fig. 4: Circuito del sistema.

Para realizar los requerimientos se tienen en cuenta los siguientes componentes y módulos que permiten el funcionamiento del sistema:

1. Display LCD

Se trata de un display LCD de 2x16 conectado en modo de 4 bits al microcontrolador, se utilizaron las librerías lcd.c y lcd.h

Se utilizó la configuración de puertos del display por defecto como se puede ver el la Figura 5, el display está configurado para su conexión al puerto A y B, como se había realizado por la cátedra.

```
22  void lcd_init(void)
23  {
24      GPIOA->CRL = 0x33333333;
25      GPIOB->CRL = 0x33344444;
26      GPIOB->BRR = (1<<LCD_EN); //same as GPIOA->ODR &= ~(1<<LCD_EN);/* LCD_EN = 0 */
27      GPIOB->BRR = (1<<LCD_RW); //same as GPIOA->ODR &= ~(1<<LCD_RW);/* LCD_RW = 0 */
28      lcd_send_cmd(0x38);/* init.LCD 2 line,5´7 matrix */
29      lcd_send_cmd(0x0C);/* display on, cursor on */
30      lcd_send_cmd(0x01);/* clear LCD */
31      lcd_send_cmd(0x06);/* shift cursor right */
32  }
```

Figura 5: configuración de puerto del display LCD

Por otro lado, se implementaron distintas funciones dentro de la librería (Fig. 6) que facilitan la utilización del sistema, junto con un buffer que permite refrescar la información del display con lo que sea que tenga dentro cada vez que se invoca la función lcd_refresh().

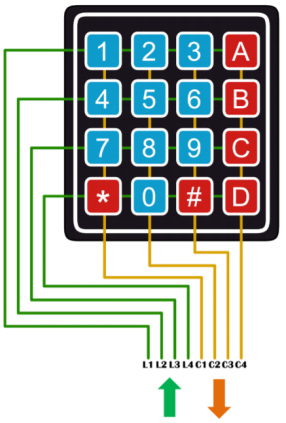
```
 7   char lcd_buffer_high[16];
 8   char lcd_buffer_low[16];
 9
10   void lcd_init(void);
11   void lcd_clrscr(void);
12   void lcd_home(void);
13   void lcd_go_to_xy(uint8_t x, uint8_t y);
14   void lcd_put_value(char val);
15   void lcd_put_string(char *str, uint8_t len);
16   void lcd_send_cmd(uint8_t cmd);
17   void lcd_send_data(uint8_t data);
18   void lcd_load_buffer_high(char *str, uint8_t len);
19   void lcd_load_buffer_low(char *str, uint8_t len);
20   void lcd_refresh(void);
```

Figura 6: Funciones y buffer implementados.

2. Teclado Matricial

El teclado matricial de 4x4, es una matriz de teclas (Figuras 7 y 8), que conecta los pines de salida del MCU a las filas del teclado y los pines de entrada del MCU a las columnas del teclado, con pull up activado.

```
44  uint8_t keypad_scan (uint8_t *pkey)
45  {
46      uint16_t i, input;
47
48      for (i = 0; i < 4; i++)
49       {
50          GPIOB -> ODR = ~(1 << (12 + i));
51          input = keypad_get_imput ();
52
53          if (input != 0x0F00)
54          {
55              *pkey = keypad_reference ((GPIOB -> IDR) >> 8);
56              return 1;
57          }
58
59          GPIOB -> ODR |= 1 << (12 + i);
60       }
61
62      return 0;
63  }
```



Figuras 7 y 8: módulo encargado de escanear el teclado
y circuito del teclado matricial 4x4

Cuando una de las teclas es presionada,este conecta uno de los pines de salida con el pin de entrada correspondiente, obteniendo así, el número columna correspondiente a la tecla presionada, quedando como incógnita el número de fila de la tecla. Para conseguir el número de la fila, se realiza un escaneo, verificando fila por fila, hasta que uno de los pines de entrada obtenga respuesta.

Cuando se detecta una entrada, se compara la dirección del pin de salida con el pin de entrada y se da con un valor ASCII establecido que coincide con la tecla presionada. (Fig. 9).

El mismo se encuentra conectado al puerto B, específicamente al puerto GPIOB alto (pines 8 a 15). Se utilizan como entrada los pines 8 a 11 y como salida los pines 12 a 15

Los archivos involucrados en el control del display son keypad.c y keypad.h

```
71  uint8_t keypad_reference (uint8_t reading)
72  {
73      uint8_t key = 1;
74
75      switch (reading)
76      {
77          case 0x77: key='1'; break;
78          case 0x7B: key='2'; break;
79          case 0x7D: key='3'; break;
80          case 0x7E: key='A'; break;
81
82          case 0xB7: key='4'; break;
83          case 0xBB: key='5'; break;
84          case 0xBD: key='6'; break;
85          case 0xBE: key='B'; break;
86
87          case 0xD7: key='7'; break;
88          case 0xDB: key='8'; break;
89          case 0xDD: key='9'; break;
90          case 0xDE: key='C'; break;
91
92          case 0xE7: key='*'; break;
93          case 0xEB: key='0'; break;
94          case 0xED: key='#'; break;
95          case 0xEE: key='D'; break;
96
97          default: key=1;
98      }
99
100     return key;
101 }
```

Fig.9: Respuestas a teclas.

3. Reloj

El reloj aprovecha el uso del módulo SysTick que genera interrupciones cada 1ms para generar la hora a mostrar. Mediante el sEOS ya explicado, se habilita la actualización del reloj cada 1000 ticks del sistema.

- Los archivos involucrados son: clock.h y clock.c

4. Máquina de estados finita de las funciones

La máquina de estados finita de las funciones sirve para controlar las acciones del programa según el valor ingresado en el teclado y sirve para conectar las funcionalidades de los distintos componentes.

La figura 10 muestra el diagrama de la máquina de estados, junto con las variables y eventos y sus respectivos valores para el cambio de estados.

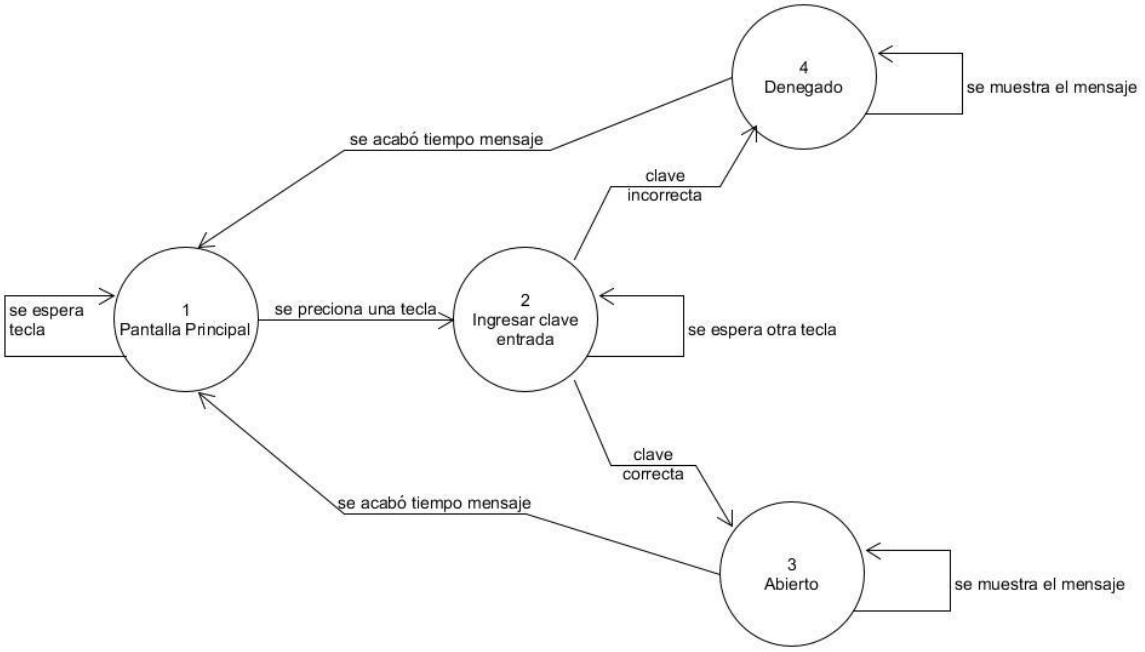


Fig. 10: Máquina de estados finitos

La transición de estados, la descripción de los mismos y los posibles estados a los que puede pasar según las variables y eventos, están especificados en la tabla 1.

ID	Nombre	Descripción	Razón	Destino
1	Pantalla principal	Mostrar Hora y "CERRADO", se espera una entrada por teclado. La hora es obtenida de una máscara comandada por el Reloj.	estado por defecto	1,2
2	Ingresar clave entrada	Mostrar "Ingresar clave" y "*" por cada caracter ingresado. Se mantiene hasta haber ingresado una clave de 4 caracteres.	se presiona un valor numérico en el estado 1	2,3,4
3	Abierto	Mostrar "ABIERTO" por 0.5 segundos y regresar a 1.	se ingresó toda la clave y es correcta	3.1
4	Denegado	Mostrar "DENEGADO" por 0.2 segundos y regresar a 1.	se ingresó toda la clave y es incorrecta	4.1

Tabla 2: funciones de la máquina de estados

La máquina de estados encargada de las funciones del sistema es de tipo Moore. Cuando la máquina transita por un estado, dentro del mismo se determina, según sea el estado por el que transita y la entrada introducida al ingresar al mismo, el próximo estado que se ejecutará.

Código y Video de Muestra

Repositorio del proyecto:

https://github.com/alcaolpg/t1-tp1

Video muestra de simulación del sistema:

https://youtu.be/PzH2DuLXrv0