Sistemas Electrónicos Para Automatización

PRÁCTICA 3: *Máquina de vending simple.*

03/11/2021

Francisco Javier Román Cortés 4º GIERM

Introducción

IMPORTANTE: RECORDAR QUE HAY QUE SALIR DEL MODO DEBUG ANTES DE PROBAR LOS PROGRAMAS PARA QUE NO DE FALLOS EL SYSCTLSLEEP()

En esta tercera práctica, se va a trabajar principalmente el aspecto de la creación de interfaces para interactuar con el usuario, más conocidas como HMI (Human Machine Interface). Para ello se va a hacer uso del sistema VM800 para configurar nuestras propias interfaces. De esta manera disponemos de una pantalla resistiva táctil, donde sólo podemos realizar una pulsación a la vez (no es multipunto), con la gran ventaja de que este sistema de la empresa FTDI está muy consolidado en cuanto a librerías y funciones preprogramadas que ayudan y facilitan mucho el desarrollo de aplicaciones con este sistema.

De esta manera, se va a buscar combinar las muy variadas posibilidades que ofrece el sistema de la pantalla con los elementos físicos de la placa Connected Launchpad, como forma extra de interaccionar con este sistema conjunto.

Fundamento teórico

Dados por conocidos los aspectos estudiados anteriormente respecto a la placa Connected Launchpad, básicamente necesitaremos comprender el funcionamiento del sistema VM800. Para ello, tenemos varias formas:

- Acudir a los ejemplos 1 y 2 y comprenderlos, como se comentará a continuación.
- Indagar en las librerías que se nos proporcionan “FT800_TIVA” y analizar las diferentes funciones y su comportamiento.
- Tercero y fundamental, tener presente durante el desarrollo la guía de programación diseñada por FTDI: “FT800 Programmer Guide”.

Realización de la práctica

1. Análisis del ejemplo 1 y 2.

Como se ha mencionado antes, para no empezar a “pelearse” a ciegas con la pantalla, se dedicó gran parte de la sesión principal a analizar el funcionamiento de estos ejemplos para comprender mejor el funcionamiento de la pantalla. Entonces, tras analizar la mayoría de funciones y las funcionalidades de distintas partes del código, se ejecutaron varios ejemplos y se comprobó su funcionamiento. Cabe destacar, el ejemplo de calibración el cual al ejecutarlo y obtener los parámetros de calibración e incluirlos en el resto de nuestros programas con la nueva configuración calibrada, se notó una mejora notable de la captación de las pulsaciones táctiles.

2. Diseño de una interfaz muy simple.

Se propone como problema introductorio un ejercicio que plantea un programa que interaccione con los botones y los leds de la placa. Su funcionamiento deberá ser el siguiente:

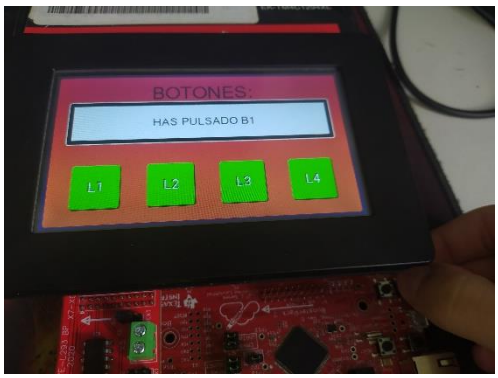
- Si se pulsa B1 deberá mostrar un mensaje tal que: “HAS PULSADO B1” y análogo para el botón B2. Destacar que el mensaje deberá permanecer mientras el botón esté pulsado.

- Por su parte, en la pantalla tendremos 4 botones (L1, L2, L3 y L4), de manera que, al pulsarlos, tendremos que tener encendidos los Leds mientras se mantengan pulsados.
- El diseño de la interfaz se propone un prototipo, pero es libre para cada alumno, en mi caso queda así:

- El sistema deberá permanecer en reposo si no se hace nada, y refrescando todo el sistema cada 50ms.

Antes de comentar el código adjuntaré el resto de imágenes que muestran el funcionamiento de este ejercicio:

-Pulsando sólo B1



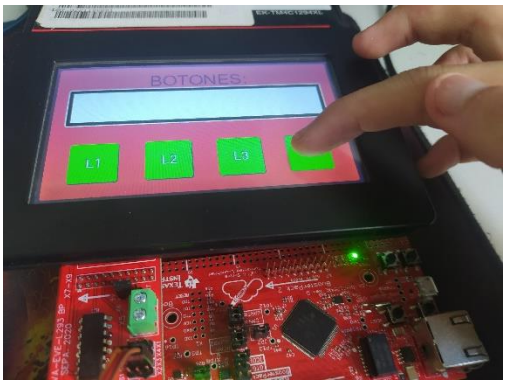
-Pulsando sólo B2

-Pulsando sólo L1

-Pulsando sólo L2

-Pulsando sólo L3

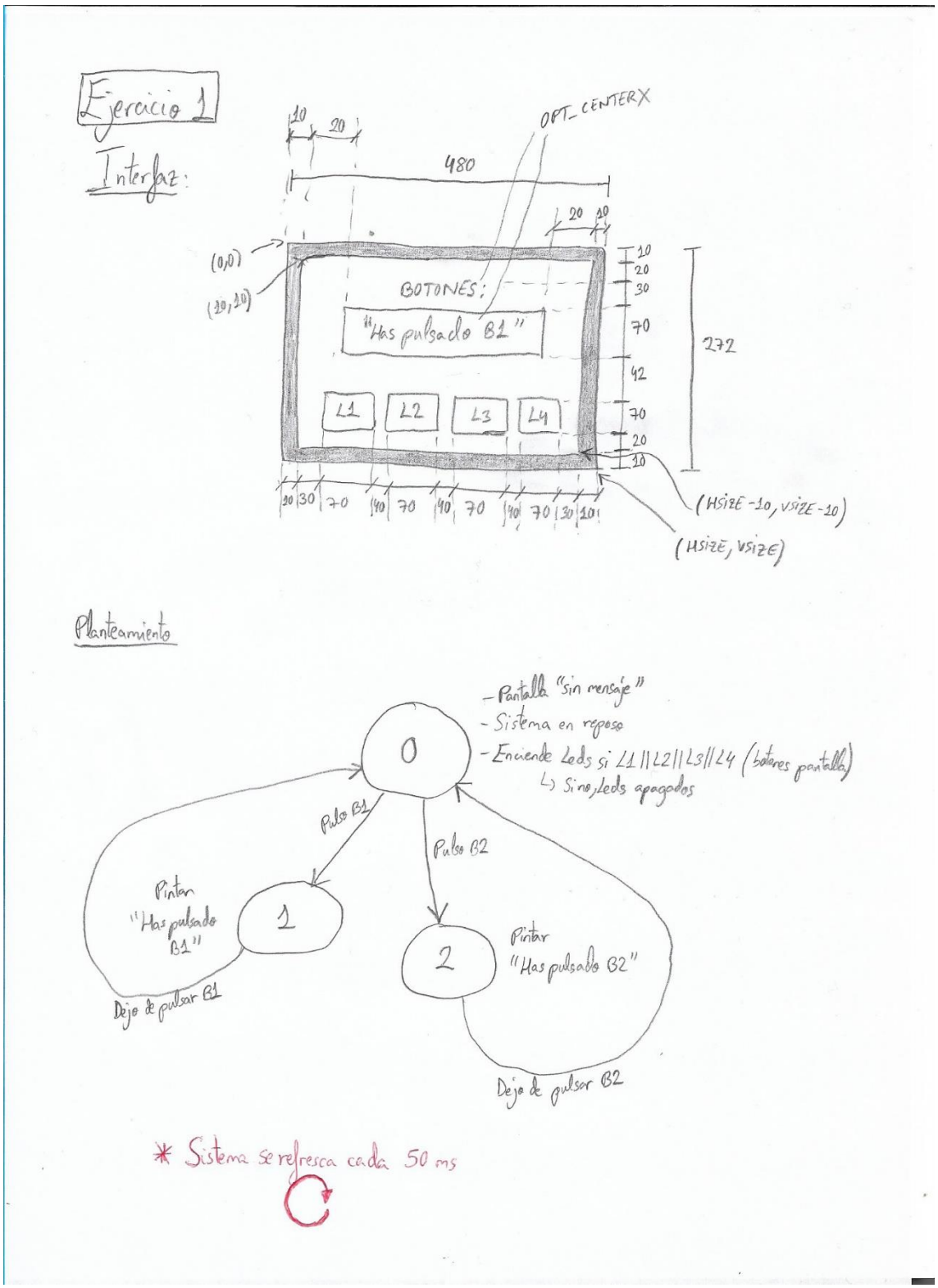
-Pulsando sólo L4



-Pulsando L4 y B1 a la vez

Recomiendo acudir al vídeo adjunto en la entrega para comprobar el funcionamiento de manera más dinámica que con estas imágenes.

A continuación, tras haber mostrado las imágenes de funcionamiento, voy a adjuntar también tanto el planteamiento a mano del diseño de la interfaz como de la máquina de estados para el diseño del programa, siguiendo las recomendaciones del enunciado:



Finalmente, tras estos pasos previos de planteamiento y habiendo visto el funcionamiento de este primer ejercicio, procedo a adjuntar íntegramente el código y a explicarlo resumidamente:

```
#include <stdint.h>
#include <stdbool.h>
#include "driverlib2.h"

#include "FT800_TIVA.h"

// =============================================================================
// Function Declarations
// =============================================================================
#define dword long
#define byte char

#define MSEC 40000 //Valor para 1ms con SysCtlDelay()

#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))     //Macros para
los botones de la placa (B1,B2)
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))

#define BL1 Boton(40, VSIZE-100, 70, 70, 28, "L1")
#define BL2 Boton(150, VSIZE-100, 70, 70, 28, "L2")         //Macros para los
botones de la pantalla (L1,L2,L3,L4)
#define BL3 Boton(260, VSIZE-100, 70, 70, 28, "L3")
#define BL4 Boton(370, VSIZE-100, 70, 70, 28, "L4")

// =============================================================================
// 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
//
################################################################################
################################

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
#define MaxEst 5
```

```
int RELOJ;                                  //Variable para el reloj del
micro
int estado = 0, boton1 = 0, boton2 = 0;     //Variables para recorrer los
estados de la FSM y para actuar en función de si se pulsa B1 o B2
int e0,e1,e2,e3,e4;                         //Variables para encender los
leds de diferentes formas según se pulsen los botones de la pantalla
int flag_timer = 0;                         //Flag para el timer de 50ms
int LED[MaxEst][4]={                        //Matriz de estados de los Leds
para encenderlos de diferentes formas
                    0,0,0,0,
                    0,0,0,1,
                    0,0,1,0,
                    0,1,0,0,
                    1,0,0,0,
};


void rutina_interrupcion(void)  //Rutina de interrupción de los botones(en
este caso both_edges, interrumpe tanto por flanco de subida como de bajada)
{
    if(B1_ON)                   // Si mantiene pulsado el boton1
    {
        estado = 1;             // se activa la variable que indica que se ha
pulsado B1 y pasamos al estado 1
        boton1 = 1;
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_0);  //Se borra el flag de la
interrupción correspondiente
    }

    if(B2_ON)                   // Si se mantiene pulsado el boton2
    {
        estado = 2;             // se activa la variable que indica que se ha
pulsado B2 y pasamos al estado 2
        boton2 = 1;
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_1);  //Se borra el flag de la
interrupción correspondiente
    }
    if(B2_OFF && B1_OFF)        // Cuando se deje de pulsar alguno de los dos
que estuviese siendo pulsado, también genera interrupción, no habrá ninguno
pulsado y entonces:
    {
        estado = 0;             // se desactiva la variable que indica que se
ha pulsado B1 y B2 y se pasa al estado 0
        boton1 = 0;
        boton2 = 0;
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_1 | GPIO_PIN_0);  //Se borra
el flag de la interrupción correspondiente

    }

}

void IntTimer1(void);       //Prototipos para las rutinas de interrupción de
los Timers

void pinta_pantalla(int state){         //Se ha creado una función que recibe
el estado como parámetro y en función de dicho estado pinta unas cosas u
otras en la pantalla, así como gestiona también las actuaciones de los leds
```

```
                                        //Se ha hecho así para evitar
repeticiones muy densas de código que implicarían un código innecesariamente
grande

    ComColor(53,10,123);                    //Elige color MORADO_OSCURO
    ComLineWidth(10);                       //Ancho del marco
    ComRect(0, 0, HSIZE, VSIZE, false);     //Marco morado de la interfaz

    ComColor(242,155,78);                       //Elige color NARANJA_CLARO
    ComRect(10, 10, HSIZE-10, VSIZE-10, true);  //Pinta fondo naranja de la
interfaz

    ComColor(53,10,123);                            //Elige color
MORADO_OSCURO para el texto "BOTONES:"
    ComTXT(HSIZE/2,20, 25, OPT_CENTERX,"BOTONES:"); //texto "BOTONES:" en
morado;

    ComColor(255,255,255);                  //Elige color BLANCO
    ComRect(30, 60, HSIZE-30, 130, true);  //Pinta rectángulo relleno blanco
de "Has pulsado"

    if(state == 1 && boton1 == 1){          //Si estamos en el estado 1 y se
está pulsando el boton1:
        ComColor(0,0,0);                                        //Elige color
NEGRO para el texto
        ComTXT(HSIZE/2,85, 23, OPT_CENTERX,"HAS PULSADO B1");   //Muestra
texto "HAS PULSADO B1" en negro en la ventana de mensajes
    }

    if(state == 2 && boton2 == 1){          //Si estamos en el estado 2 y se
está pulsando el boton2:
        ComColor(0,0,0);                                       //Elige color
NEGRO para el texto
        ComTXT(HSIZE/2,85, 23, OPT_CENTERX,"HAS PULSADO B2"); //Muestra texto
"HAS PULSADO B2" en negro en la ventana de mensajes
    }

    ComColor(0,0,0);                            //Elige color NEGRO
    ComLineWidth(3);                           //Ancho del marco
    ComRect(30, 60, HSIZE-30, 130, false);     //Marco negro de "Has pulsado"

    ComColor(255,255,255);                              //Elige color blanco
para el texto de los botones
    ComFgcolor(91, 242, 78);                            //Elige color verde
para el fondo del botón
    if(BL1)                     //Si se pulsa el botón L1 de la pantalla:
    {
        e1 = 1;                 //Se enciende el Led1, el más alejado de los
botones B1 y B2 (ver imágenes)
        enciende_leds(e1);
    }

    if(BL2)                     //Si se pulsa el botón L2 de la pantalla:
    {
        e2 = 2;                 //Se enciende el Led2, a la derecha de Led1
        enciende_leds(e2);
    }

    if(BL3)                     //Si se pulsa el botón L3 de la pantalla:
```

```
    {
        e3 = 3;                 //Se enciende el Led3, a la derecha de Led2
        enciende_leds(e3);
    }

    if(BL4)                     //Si se pulsa el botón L4 de la pantalla:
    {
        e4 = 4;                 //Se enciende el Led4, el más cercano
respecto de los botones B1 y B2 (ver imágenes)
        enciende_leds(e4);
    }
    if(!BL1 && !BL2 && !BL3 && !BL4)    //Si no se pulsa ninguno de los
botones de la pantalla:
    {
        e0 = 0;                         //Se tienen todos los Leds apagados
        enciende_leds(e0);
    }
    Dibuja();                           //Se dibujan todos los elementos
anteriormente explicados

}

void enciende_leds(uint8_t Est)          //Función para encender leds más
cómodamente
{
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1*LED[Est][0]);
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0*LED[Est][1]);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4*LED[Est][2]);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0*LED[Est][3]);
}

int main(void)
{
    int i;

    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);    //Se habilita puerto J
(BOTONES)
    HAL_Init_SPI(1, RELOJ);                         //Boosterpack a usar,
Velocidad del MC
    Inicia_pantalla();                              //Arranque de la pantalla
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);    //Se habilita puerto F
(LEDS)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);    //Se habilita puerto N
(LEDS)


    /*******************************   BOTONES
*****************************************/
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);
//J0 y J1: entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO
_PIN_TYPE_STD_WPU); //Pullup en J0 y J1
    GPIOIntTypeSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_BOTH_EDGES);
// Definir tipo int: flanco bajada
    GPIOIntEnable(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);
//Habilitar pines de interrupción J0, J1
```

```
    GPIOIntRegister(GPIO_PORTJ_BASE, rutina_interrupcion);
//Registrar (definir) la rutina de interrupción del puerto J (de los botones)
    IntEnable(INT_GPIOJ);
//Habilitar interrupción del pto J
    IntMasterEnable();
//Habilitar globalmente las ints

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

    /*******************************   TIMER1
**************************************/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);           //Habilita T1
    TimerClockSourceSet(TIMER1_BASE, TIMER_CLOCK_SYSTEM);   //T1 a 120MHz
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);        //T1 periódico y
conjunto (32b)
    TimerLoadSet(TIMER1_BASE, TIMER_A,(RELOJ/20)-1);        //Se carga el
timer1 con 50ms
    TimerIntRegister(TIMER1_BASE,TIMER_A,IntTimer1);        //Cada vez que
cuente 50ms que entre en la interrupción y refresque el sistema
    IntEnable(INT_TIMER1A);                                 //Habilitar las
interrupciones globales de los timers
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);        //Habilitar las
interrupciones de timeout
    TimerEnable(TIMER1_BASE, TIMER_A);                      //Habilitar
Timer0 (no hace falta recargarlo, se activa al principio y que cuente
"infinito")

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

    /*******************************   LEDS
**************************************/
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y
F4: salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y
N1: salidas

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

    /*******************************   MODO BAJO CONSUMO
**************************************/
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOJ);           //Dejar
despierto el Pto J durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOF);           //Dejar
despierto el Pto G durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPION);           //Dejar
despierto el Pto N durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_TIMER1);          //Dejar
despierto el timer 1 durante el Sleep
    SysCtlPeripheralClockGating(true);                          //Habilitar
el apagado selectivo de periféricos

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

    // Note: Keep SPI below 11MHz here
```

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

    SysCtlDelay(RELOJ/3);

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


    Nueva_pantalla(16,16,16);   //Configura una nueva pantalla
    ComColor(21,160,6);         //Elige el color para el siguiente elemento,
VERDE OSCURO en este caso
    ComLineWidth(5);            //Elige el grosor de la línea del siguiente
elemento
    ComRect(10, 10, HSIZE-10, VSIZE-10, true);      //ComRect(int x1, int y1,
int x2, int y2, char relleno)  //esquina supizq en (10,10) y esqinfder en
(Hsize-10,Vsize-10), con TRUE o FALSE indicamos si el rectángulo esta relleno
o hueco
    ComColor(65,202,42);                            //Elige el color para el
siguiente elemento, VERDE CLARO en este caso
    ComRect(12, 12, HSIZE-12, VSIZE-12, true);      //esquina supizq en
(12,12) y esqinfder en (Hsize-12,Vsize-12), con TRUE o FALSE indicamos si el
rectángulo esta relleno o hueco
    //Este bloque pintaría un rectángulo verde oscuro y luego uno verde claro
encima, quedando un borde verde oscuro

    ComColor(255,255,255);      //Elige color blanco
    ComTXT(HSIZE/2,VSIZE/5, 22, OPT_CENTERX,"PRIMERA INTERFAZ SIMPLE");
//ComTXT(int x, int y, int fuente, int ops, char *cadena)
    ComTXT(HSIZE/2,50+VSIZE/5, 22, OPT_CENTERX," P3_1 - SEPA GIERM");
//coordenada x,y donde empieza la cadena; fuente 22; opciones (centrado en
X); "cadena de texto"
    ComTXT(HSIZE/2,100+VSIZE/5, 20, OPT_CENTERX,"F.J.R.C.");

    ComRect(40,40, HSIZE-40, VSIZE-40, false);        //Pinta un rectángulo
blanco vacío con esquinas: (40,40)      ---(HSIZE-40,40)

//     ---                  ---
    Dibuja();           //Dibuja toda la lista de comandos anteriores
    Espera_pant();      //Queda en espera hasta cambios de pantalla

#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

    while(1){                   //Bucle infinito repitiéndose
```

```
        while(flag_timer == 0 && boton1 !=1 && boton2 !=1) SysCtlSleep();
//Sistema en bajo consumo despertándose por el timer o los botones B1/B2

        switch(estado){                 //FSM

        case 0:                         //Estado 0:
            flag_timer = 0;             //Se resetea el flag del timer
            Lee_pantalla();             //Se lee la pantalla aunque no hace
falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);     //Se llama a la función que pinta
indicándole que estamos en estado 0
            break;

        case 1:                             //Estado 1:
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado 1

            break;

        case 2:                             //Estado 2:
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado
            break;

        }
    }
}


void IntTimer1(void)                                //Rutina Interrupción
Timer1:
{
    flag_timer = 1;                                  //Para que cuando el
micro se despierte cada 50ms, se vuelva a dormir al momento
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT); //Borra flag Interrupción
Timer1

}                                                   //FIN
```

Como de costumbre, voy a comentar en esta memoria también el código, aunque está comentado con bastante detalle en el propio ".c".

Siguiendo el orden cronológico del código vamos realizando lo siguiente:

En primer lugar, definimos las librerías necesarias para nuestro programa, a continuación, definimos también una serie de macros para que sea más cómoda la programación posterior, concretamente para los botones B1 y B2 de la placa y los botones L1, L2, L3 y L4 de la pantalla.

Se declaran y dan valores a ciertas variables y constantes relacionadas con el funcionamiento de la pantalla, para posteriormente, declarar el resto de variables que necesitamos para el programa, como son el reloj, el estado, unas variables de control para los botones B1 y B2, un flag para el timer de 50ms y una matriz de estados de los Leds para configurar su estado de encendido/apagado de manera más cómoda.

Pasamos ahora a la rutina de interrupción de los botones, donde esta vez se ha configurado para que la interrupción sea del tipo BOTH_EDGES, de manera que salta interrupción al pulsar y al dejar de pulsar. De esta manera, hacemos lo siguiente:

Si el botón 1 se está pulsando:

- Ponemos la variable de control boton1 a 1 y pasamos al estado 1.
- Borramos el flag de la interrupción.

Si el botón 2 se está pulsando:

- Ponemos la variable de control boton2 a 1 y pasamos al estado 2.
- Borramos el flag de la interrupción.

Si no se está pulsando ninguno de los dos (Se ha dejado de pulsar el que estuviera y salta la interrupción con ninguno pulsado):

- Se resetean las variables de control boton1 y boton2 a 0.
- Se pasa al estado 0 de nuevo.
- Se borra el flag de la interrupción.

Tras esto, ponemos el prototipo de la interrupción del timer1 y definimos la función `void pinta_pantalla(int state)`, la cual realiza las operaciones de pintado de pantalla y se hace esta función para evitar tener que repetir las mismas operaciones en cada estado. Además, como recibe el estado como parámetro, podemos actuar en consecuencia y pintar diferentes elementos en función de en qué estado estemos.

Si estamos en el estado 0, genéricamente es donde estaremos mientras no se pulse ni B1 ni B2:

Pintaremos la interfaz en sí, un marco morado en el exterior, el fondo naranja claro, un texto de "BOTONES: " en morado también y un rectángulo que hará de ventana de mensajes de color blanco.

En función de si hemos pulsado B1 o B2 (y por tanto también estemos en los estados 1 o 2), mostraremos los mensajes de "HAS PULSADO B1/B2".

Se pinta el marco negro de la ventana de mensajes, y a continuación se pintan los 4 botones de la pantalla, donde para cada uno, se indica que si se pulsan, actúen encendiendo el led correspondiente a cada botón, como se veía en las imágenes anteriores.

Tenemos además la función enciende_leds, con la que actuamos sobre el estado de los leds de manera rápida.

Ahora, en el main(), se realiza la configuración de los diferentes periféricos a usar así como el modo de bajo consumo. A continuación, antes de entrar en el while(1), pintamos una pantalla inicial, con el título del ejercicio y mi nombre, de la cual saldremos pulsando en la pantalla.

Entramos así en el while(1), donde tenemos fuera del switch las comprobaciones de "flags" para dormir el micro o no.

En cuanto a la máquina de estados, en el estado 0 reseteamos el flag del timer de 50ms y refrescamos la pantalla, para llamar a la función pinta_pantalla(estado), indicando que estamos en el estado 0. Esto se repite prácticamente en los estados 1 y 2, siendo la función pinta_pantalla(estado), la que lleva el peso de gestionar las diferentes actuaciones según en qué estado estemos, para condensar el código como ya he mencionado anteriormente.

Finalmente, tenemos la interrupción del timer1, donde cada 50ms, se activará el flag de este timer y borraremos el flag de la interrupción de Timeout.

Con esto queda explicado el primer ejercicio de la práctica y pasaremos al segundo y más complejo.

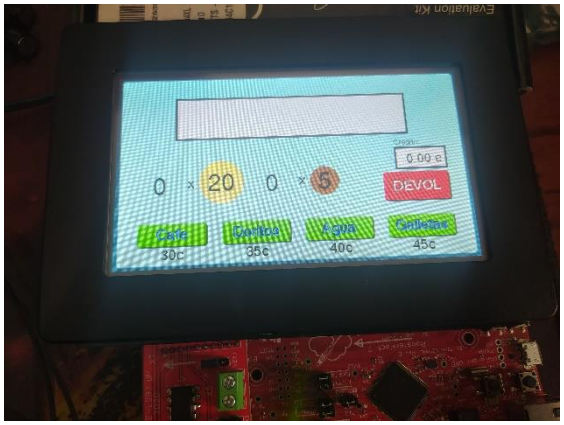
3. Máquina de Vending simple.

Sin extenderme mucho en el funcionamiento concreto de esta máquina de vending, ya viene en el enunciado su comportamiento según las actuaciones del usuario con bastante detalle, lo que se pide básicamente es diseñar y hacer funcionar una máquina de vending que dispone de 4 producto cuyo precio es de 30c, 35c, 40c y 45c. La máquina aceptará solamente monedas de 20c y de 5c.

Al igual que antes, voy a adjuntar primero una serie de imágenes de funcionamiento, aunque en este caso, al ser el sistema bastante más complejo y tener tantas funciones y pasos distintos, sólo mostraré algunos estados concretos y recomiendo encarecidamente acudir al vídeo adjunto para ver de verdad el funcionamiento del sistema. El principal interés de estas imágenes es mostrar la interfaz más que el funcionamiento.

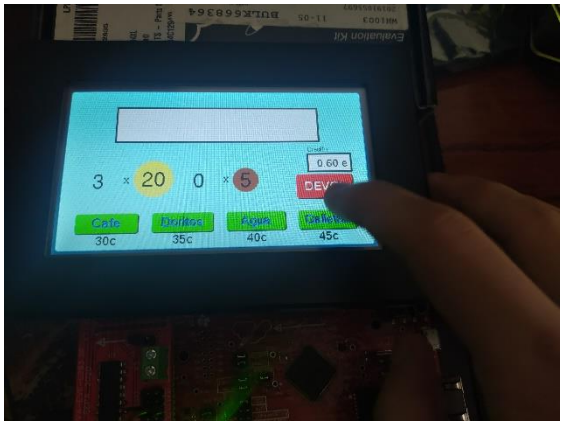
En concreto voy a mostrar imágenes de lo que sería una posible operación sobre esta máquina:

- Interfaz en "espera"

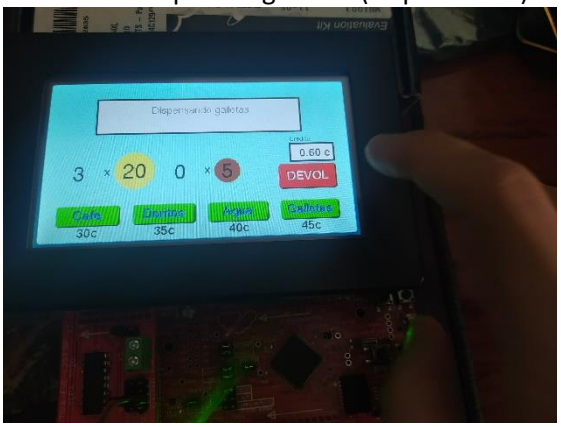


- Añadiendo monedas de 20c

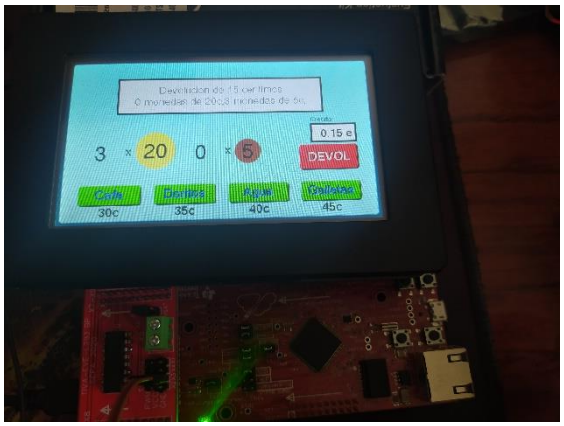
- “Yendo” a pedir galletas



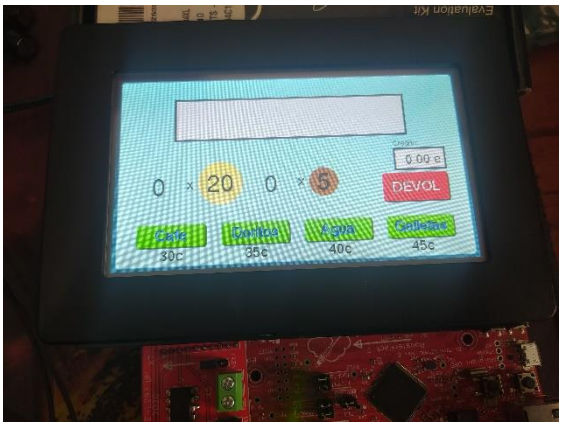
-Tras haber pedido galletas (dispensando)



- Calculando la vuelta e indicando monedas a devolver

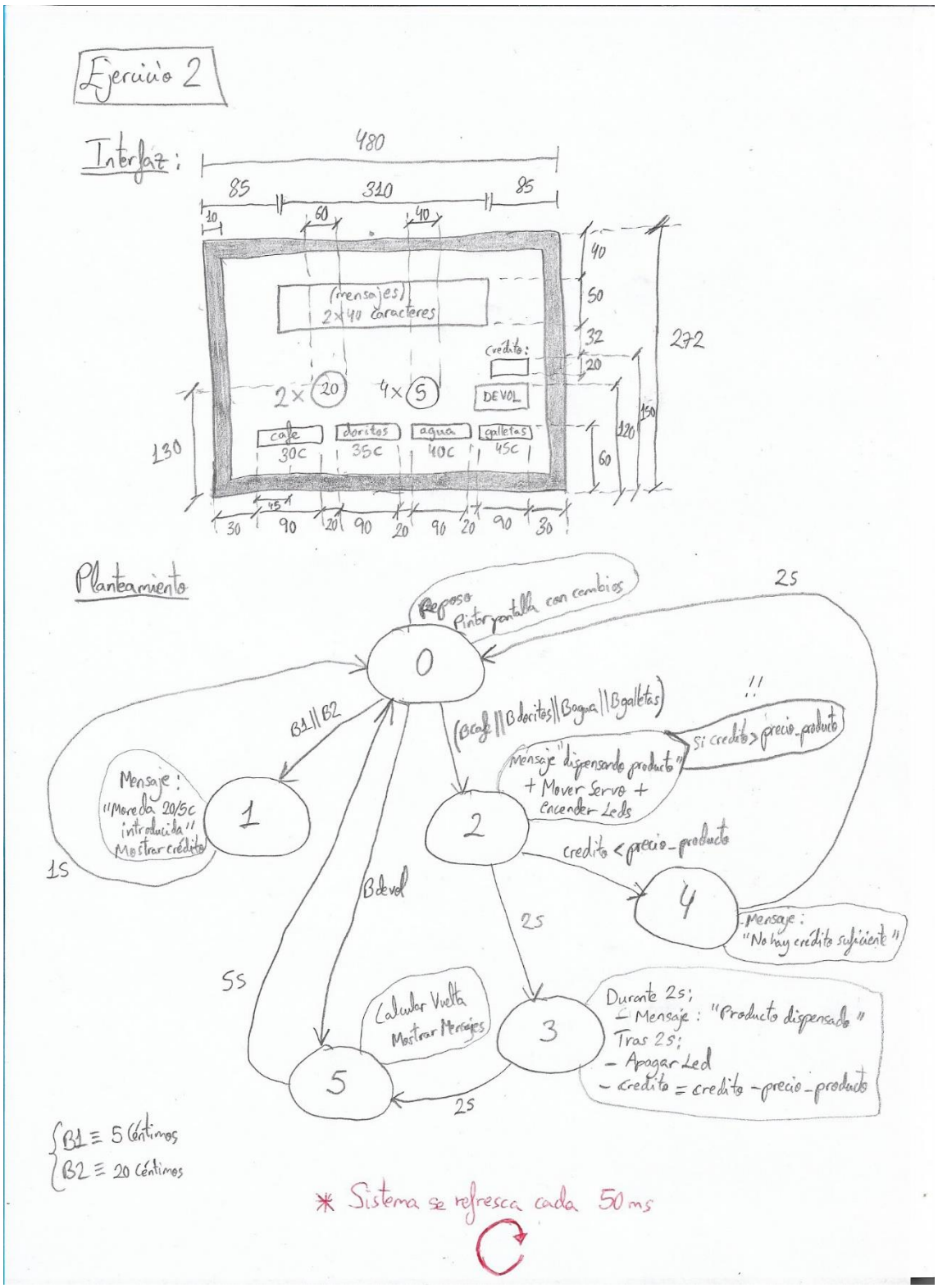


- Vuelta al reposo tras haber servido y devuelto la vuelta



Este es sólo un ejemplo, faltan muchas funcionalidades que son difícil de mostrar con imágenes, vuelvo a recalcar que sea acuda al vídeo.

De nuevo, al igual que antes, adjunto el planteamiento tanto del dimensionado de la interfaz como de la máquina de estados para este ejercicio, muy necesarios para ahorrar tiempo reposicionando elementos y tener claro los pasos a seguir dentro del programa:



Finalmente, tras estos pasos previos de planteamiento y habiendo visto el funcionamiento de este segundo ejercicio, procedo a adjuntar íntegramente el código y a explicarlo resumidamente:

```
#include <stdint.h>
#include <stdio.h>
#include <string.h>          //Declaración de librerías
#include <stdbool.h>
#include "driverlib2.h"
#include <stdlib.h>

#include "FT800_TIVA.h"


// ===========================================================================
// Function Declarations
// ===========================================================================
#define dword long
#define byte char

#define MSEC 40000 //Valor para 1ms con SysCtlDelay()

volatile int Max_pos = 4400;     // Posición máxima del servo
volatile int Min_pos = 1000;     // Posición mínima del servo
volatile int media = 2700;       // Posición central del servo

#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))     //Macros para
los botones B1 y B2 de la placa
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))

#define Bcafe     Boton(30, VSIZE-60, 90, 30, 23, "Cafe")
#define Bdoritos  Boton(140, VSIZE-60, 90, 30, 23, "Doritos")       //Macros
para los botones de la pantalla de los diferentes productos
#define Bagua     Boton(250, VSIZE-60, 90, 30, 23, "Agua")
#define Bgalletas Boton(360, VSIZE-60, 90, 30, 23, "Galletas")
#define Bdevol    Boton(360, VSIZE-120, 90, 40, 23, "DEVOL")        //Macro
para el boton de devolver

#define precio_cafe     30
#define precio_doritos  35          //Macros para guardar el precio de cada
producto
#define precio_agua     40
#define precio_galletas 45

// ===========================================================================
// 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
//
################################################################################
##############################

unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;                              //Variables y constantes
relacionadas con la configuración de la pantalla
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
#define MaxEst 5

int RELOJ,PeriodoPWM;                    //Variables para el reloj del micro y
para el periodo del PWM
int estado = 0, boton1 = 0, boton2 = 0; //Variables para recorrer los estados
de la FSM y para controlar las pulsaciones de B1 Y B2
int e0,e1,e2,e3,e4;                      //Variables para controlar el
encendido de los botones
int flag_timer1 = 0;                     //Variable para el flag del timer de
50ms
int credito = 0,creditodevolver = 0,euros = 0, cts = 0;     //Variables para
calcular credito, credito a devolver y separar credito en euros y céntimos
int contmon20 = 0, contmon5 = 0;                            //Contadores de
monedas de 20c y de 5c
int contdevol20 = 0, contdevol5 = 0;                        //Contadores de
monedas de 20c y de 5c a devolver
int cafe = 0, doritos = 0, agua = 0, galletas = 0;      //Variables para
controlar qué producto se ha elegido
int devolver = 0;                                           //Variable para
controlar la pulsación del botón de la pantalla de devolver
char buffer [sizeof(int)*8+1];
char buffer2 [sizeof(int)*8+1];
char stringcredito [sizeof(int)*8+1];                   //Arrays para pasar
strings a ComTXT con los mensajes...
char string20 [sizeof(int)*8+1];
char string5 [sizeof(int)*8+1];
char stringcreditodevolver [sizeof(int)*8+1];
char stringcts20_5devolver [sizeof(int)*8+1];
char x [5] = "x";
char punto [5] = ".";                                       //Arrays con ciertos
caracteres de interés para concatenar strings u otras operaciones
char e [5] = "e";

#define T_5 200
#define T_2 60                                              //Defines para
tiempos de espera
#define T_1 40
uint32_t t_0 = 0,t_2 = 0,t_3 = 0, t_4 = 0, t_5 = 0;     //Contadores para
esperar ciertos tiempos entre estados

int LED[MaxEst][4]={                                        //Matriz de estados
de los LEDS para actuar sobre ellos
```

```
                    0,0,0,0,
                    0,0,0,1,
                    0,0,1,0,
                    0,1,0,0,
                    1,0,0,0,
};


void rutina_interrupcion(void)                          //Rutina de
interrupción de los botones B1 y B2
{
    if(B1_ON)                   // Si se pulsa el boton1 (equivale a
introducir una moneda de 5c)
    {
        SysCtlDelay(10*MSEC);   // Debouncing B1...
        while(B1_ON);
        SysCtlDelay(10*MSEC);
        contmon5++;             // Se incrementa el contador de monedas de 5c
introducidas
        estado = 1;             // Se pasa al estado 1
        boton1 = 1;             // Se activa la variable que indica que se ha
pulsado B1
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_0);  //Se borra el flag de la
interrupción correspondiente
    }

    if(B2_ON)                   // Si se pulsa el boton2 (equivale a
introducir una moneda de 20c)
    {
        SysCtlDelay(10*MSEC);   // Debouncing B2...
        while(B2_ON);
        SysCtlDelay(10*MSEC);
        contmon20++;            // Se incrementa el contador de monedas de
20c introducidas
        estado = 1;             // Se pasa al estado 1
        boton2 = 1;             // Se activa la variable que indica que se ha
pulsado B2
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_1);  //Se borra el flag de la
interrupción correspondiente
    }

}

void IntTimer1(void);       //Prototipos para las rutinas de interrupción de
los Timers

void pinta_pantalla(int state){         //Se ha creado una función que recibe
el estado como parámetro y en función de dicho estado pinta unas cosas u
otras en la pantalla, así como gestiona también las actuaciones de los leds
                                        //Se ha hecho así para evitar
repeticiones muy densas de código que implicarían un código innecesariamente
grande

    ComColor(22,128,131);                   //Elige color celeste oscuro
    ComLineWidth(10);                       //Ancho del marco
    ComRect(0, 0, HSIZE, VSIZE, false);     //Marco morado de la interfaz

    ComColor(207,254,255);                      //Elige color celeste
```

```
    ComRect(10, 10, HSIZE-10, VSIZE-10, true);  //Pinta fondo celeste de la
interfaz

    ComColor(255,255,255);   //Elige color BLANCO
    ComRect(85, 40, HSIZE-85, 90, true);  //Pinta rectángulo relleno blanco
de la ventana de mensajes

    if(state == 1 && boton1 == 1)           //Si estamos en el estado 1 y se
está pulsando el boton1:
    {
        ComColor(0,0,0);
//Elige color NEGRO para el texto
        ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Se ha introducido una moneda de
5c");   //texto "Se ha introducido una moneda de 5c" en negro en la primera
fila de la ventana de mensajes
        ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//segunda fila de la ventana de mensajes vacía
    }

    if(state == 1 && boton2 == 1)           //Si estamos en el estado 2 y se
está pulsando el boton2:
    {
        ComColor(0,0,0);
//Elige color NEGRO para el texto
        ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Se ha introducido una moneda de
20c");  //texto "Se ha introducido una moneda de 20c" en negro en la primera
fila de la ventana de mensajes
        ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//segunda fila de la ventana de mensajes vacía
    }


    ComColor(0,0,0);                            //Elige color NEGRO
    ComLineWidth(1);                            //Ancho del marco
    ComRect(80, 35, HSIZE-80, 95, false);       //Marco negro de la ventana
de mensajes

    ComColor(0,20,255);                          //Elige Azul oscuro para
el texto de los botones de productos
    ComFgcolor(91, 242, 78);                     //Elige verde claro para
el fondo de los botones de productos

    if(Bcafe) cafe = 1;                          //Si se pulsa el botón de
café se activa su variable

    if(Bdoritos) doritos = 1;                    //Si se pulsa el botón de
doritos se activa su variable

    if(Bagua) agua = 1;                          //Si se pulsa el botón de
agua se activa su variable

    if(Bgalletas) galletas = 1;                  //Si se pulsa el botón de
galletas se activa su variable

    ComColor(255,255,255);                       //Color blanco para el
texto del botón de devolver
    ComFgcolor(255, 0, 0);                       //Color rojo para el
fondo del botón de devolver
```

```
    if(Bdevol)  devolver = 1;                       //Si se pulsa el botón de
devolver se activa su variable

    if(state == 2)                      //Si estamos en el estado 2:
    {
        if(cafe == 1){                                  //Si cafe = 1 (se ha
elegido cafe)
            if(credito < precio_cafe) estado = 4;
//Si el credito no es suficiente, vamos al estado 4

else{                                                   //Si el
credito Sí es suficiente:
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Dispensando cafe");
//Se muestra mensaje "dispensando" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
            e1 = 1;
            enciende_leds(e1);
//Se enciende el led más a la izquierda (el más lejano a B1 y B2)
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);    //Servo a
posicion extrema
            }
        }
        if(doritos == 1){                               //Si se pulsa el
botón de doritos se activa su variable
            if(credito < precio_doritos) estado = 4;
//Si el credito no es suficiente, vamos al estado 4

else{                                                   //Si el
credito Sí es suficiente:
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Dispensando doritos");
//Se muestra mensaje "dispensando" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
            e2 = 2;
            enciende_leds(e2);
//Se enciende el led a la derecha de Led1
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);    //Servo a
posicion extrema
            }
        }
        if(agua == 1){                                  //Si se pulsa el
botón de agua se activa su variable
            if(credito < precio_agua) estado = 4;
//Si el credito no es suficiente, vamos al estado 4

else{                                                   //Si el
credito Sí es suficiente:
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Dispensando agua");
//Se muestra mensaje "dispensando" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
            e3 = 3;
```

```
            enciende_leds(e3);
//Se enciende el led a la derecha de Led2
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);    //Servo a
posicion extrema
            }
        }
        if(galletas == 1){                              //Si se pulsa el
botón de galletas se activa su variable
            if(credito < precio_galletas) estado = 4;
//Si el credito no es suficiente, vamos al estado 4

else{                                                   //Si el
credito Sí es suficiente:
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Dispensando galletas");
//Se muestra mensaje "dispensando" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
            e4 = 4;
            enciende_leds(e4);
//Se enciende el led más cercano a los botones B1 y B2
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, Max_pos);    //Servo a
posicion extrema
            }
        }
    }

    if(state == 3)                                      //Si estamos en el
estado 3:
    {
        if(cafe == 1){                                          //Si
se había elegido cafe
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Cafe dispensado");
//Se muestra mensaje "dispensado" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
        }
        if(doritos == 1){                                       //Si
se había elegido doritos
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Doritos dispensados");
//Se muestra mensaje "dispensado" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
        }
        if(agua == 1){                                          //Si
se había elegido agua
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Agua dispensada");
//Se muestra mensaje "dispensado" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
        }
```

```
        if(galletas == 1){                                          //Si
se había elegido galletas
            ComColor(0,0,0);
//Elige color NEGRO para el texto
            ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"Galletas dispensadas");
//Se muestra mensaje "dispensado" en la 1º fila
            ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
        }
    }

    if(state == 4)                                  //Si estamos en el estado
4:
    {
        ComColor(0,0,0);
//Elige color NEGRO para el texto
        ComTXT(HSIZE/2,45, 22, OPT_CENTERX,"No hay credito suficiente");
//Se muestra mensaje "No hay credito suficiente" en la 1º fila
        ComTXT(HSIZE/2,65, 22, OPT_CENTERX," ");
//2º fila vacía
        credito = contmon20 * 20 +contmon5 * 5;     //Se calcula el crédito
tras introducir monedas de 20c o 5c con los botones B1 y B2
        //Credito viene en centimos, interesa separarlo para mostrarlo en
"euros.cts", (mejor mostrar 1.20e que 120c)
        euros = credito/100;            //Se calculan los euros que hay en
credito
        cts = credito%100;              //Se calculan los centimos que hay en
credito
        }

    if(state == 5){                                 //Si estamos en el estado
5:

        creditodevolver = credito;                      //El credito a
devolver se iguala al credito actual (ya que al dispensar producto el credito
se ha actualizado restándole el precio de dicho producto)
    //Suponiendo que la máquina devuelve el menor número de monedas sin
importar que tipo de monedas se introdujeron, la vuelta se calcula tal que:
        contdevol20 = creditodevolver/20;           //Número de monedas de
20c a devolver
        contdevol5  = (creditodevolver%20)/5;       //Número de monedas de 5c
a devolver una vez ya no se puede devolver lo que falta con monedas de 20c
        ComColor(0,0,0);                                //Elige color NEGRO
para el texto

        strcpy(stringcreditodevolver,"Devolucion de ");     //Concatenación
de strings para terminar mostrando en la ventana de mensajes:
        sprintf(buffer,"%d",creditodevolver);               // "Devolución de
ZZ céntimos"          (1º fila)
        strcat( stringcreditodevolver, buffer);             // "XX monedas de
20c, YY monedas de 5c"  (2º fila)
        strcat( stringcreditodevolver, " centimos");
        sprintf(buffer,"%d",contdevol20);
        strcpy(stringcts20_5devolver, buffer);
        strcat(stringcts20_5devolver," monedas de 20c,");
        sprintf(buffer,"%d",contdevol5);
        strcat(stringcts20_5devolver,buffer);
        strcat(stringcts20_5devolver," monedas de 5c,");
        ComTXT(HSIZE/2,45, 22, OPT_CENTERX,stringcreditodevolver);
```

```
        ComTXT(HSIZE/2,65, 22, OPT_CENTERX,stringcts20_5devolver);
    }


    ComColor(0,0,0);                                    //Elige color NEGRO para
el texto
    ComTXT(75,VSIZE-30, 23, OPT_CENTERX,"30c");     //texto "30c" en negro
debajo del boton de cafe

    ComColor(0,0,0);                                    //Elige color NEGRO para
el texto
    ComTXT(185,VSIZE-30, 23, OPT_CENTERX,"35c");    //texto "35c" en negro
debajo del boton de doritos

    ComColor(0,0,0);                                    //Elige color NEGRO para
el texto
    ComTXT(295,VSIZE-30, 23, OPT_CENTERX,"40c");    //texto "40c" en negro
debajo del boton de agua

    ComColor(0,0,0);                                    //Elige color NEGRO para
el texto
    ComTXT(405,VSIZE-30, 23, OPT_CENTERX,"45c");    //texto "45c" en negro
debajo del boton de agua

    ComColor(255,255,255);                                      //Elige color
BLANCO
    ComRect(HSIZE-95, VSIZE-150, HSIZE-35, VSIZE-130, true);    //Pinta
rectángulo relleno blanco de "Crédito"

    ComColor(0,0,0);                                              //Elige
color NEGRO
    ComLineWidth(1);                                              //Ancho
del marco
    ComRect(HSIZE-100, VSIZE-155, HSIZE-30, VSIZE-125, false);    //Marco
negro de "Crédito"

    ComColor(0,0,0);                                               //Elige
color NEGRO para el texto
    ComTXT(HSIZE-100,VSIZE-170, 20, 0,"Credito:");                 //texto
"Credito: " en negro arriba del hueco donde se indica el credito

    ComColor(0,0,0);                            //Elige color NEGRO para el
texto
    sprintf(buffer,"%d",euros);
    strcpy( stringcredito, buffer );            //Concatenación de strings
para terminar mostrando en la ventana de crédito:
    strcat( stringcredito, punto );             //texto "euros.cts e" en
negro; //Ejemplo: "1.20 e", este formato de mensaje
    sprintf(buffer2,"%02d",cts);
    strcat( stringcredito, buffer2);
    strcat( stringcredito, " " );
    strcat( stringcredito, e );
    ComTXT(HSIZE-85,VSIZE-150, 23, 0,stringcredito);

    //Mostrar "%d x" al lado de la moneda de 20c, siendo %d el valor del
contador de monedas de 20c
    ComColor(0,0,0);                            //Elige color NEGRO para el
texto
    sprintf(buffer,"%d",contmon20);
```

```
    strcpy( string20, buffer );
    ComTXT(50,VSIZE-130, 25, 0,string20);
    ComTXT(95,VSIZE-125, 23, 0,x);              //texto "X" en negro;

    ComColor(247,255,166);          //Elige color amarillo "moneda 20c"
    ComCirculo(140, VSIZE-115, 30); //Dibuja la moneda de 20c

    ComColor(0,0,0);                    //Elige color NEGRO para el texto
    ComTXT(123,VSIZE-133, 25, 0,"20");  //texto "20" dentro de la moneda de
20c

    //Mostrar "%d x" al lado de la moneda de 5c, siendo %d el valor del
contador de monedas de 5c
    ComColor(0,0,0);                            //Elige color NEGRO para el
texto
    sprintf(buffer,"%d",contmon5);
    strcpy( string5, buffer );
    ComTXT(200,VSIZE-130, 25, 0,string5);
    ComTXT(245,VSIZE-125, 23, 0,x);             //texto "X" en negro;

    ComColor(209,143,21);               //Elige color "cobre" de "moneda 5c"
    ComCirculo(280, VSIZE-112, 20);     //Dibuja la moneda de 5c

    ComColor(0,0,0);                    //Elige color NEGRO para el texto
    ComTXT(272,VSIZE-130, 25, 0,"5");   //texto "5" en negro;

    Dibuja();                           //Dibuja toda la interfaz según
estados y demás

}

void enciende_leds(uint8_t Est)         //Función para encender leds más
cómodamente
{
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1*LED[Est][0]);
    GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0*LED[Est][1]);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4*LED[Est][2]);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0*LED[Est][3]);
}

int main(void)
{
    int i;

    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);    //Se habilita puerto J
(BOTONES)
    HAL_Init_SPI(1, RELOJ);                         //Boosterpack a usar,
Velocidad del MC
    Inicia_pantalla();                              //Arranque de la pantalla
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);    //Se habilita puerto F
(LEDS)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);    //Se habilita puerto N
(LEDS)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);    //Se habilita puerto G
(PIN PWM)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);     //Se habilita generador
PWM 0
```

```
    /*****************************   CONFIGURACION PWM
*****************************/
    PWMClockSet(PWM0_BASE,PWM_SYSCLK_DIV_64);       // al PWM le llega un
reloj de 1.875MHz
    GPIOPinConfigure(GPIO_PG0_M0PWM4);              //Configurar el pin0 del
puerto G (PG0) a PWM (tiene 2 pasos esta config)
    GPIOPinTypePWM(GPIO_PORTG_BASE, GPIO_PIN_0);
    PWMGenConfigure(PWM0_BASE, PWM_GEN_2, PWM_GEN_MODE_DOWN |
PWM_GEN_MODE_NO_SYNC);    //Configurar el pwm0, contador descendente y sin
sincronización (actualización automática)
    PeriodoPWM=37499;                               // 50Hz  a 1.875MHz; 50
Hz es lo que usan de pwm los servos "baratillos";; Se obtiene como:
(1875000/50)-1 = 37500-1 = 37499
    PWMGenPeriodSet(PWM0_BASE, PWM_GEN_2, PeriodoPWM);      //frec:50Hz
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, media);          //Inicialmente,
1ms // Para que el servo se inicialice en la posición central
    PWMGenEnable(PWM0_BASE, PWM_GEN_2);                     //Habilita el
generador 2 (donde está PWM4)
    PWMOutputState(PWM0_BASE, PWM_OUT_4_BIT , true);        //Habilita la
salida 4

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

    /*****************************   BOTONES
*************************************/
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);
//J0 y J1: entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO
_PIN_TYPE_STD_WPU); //Pullup en J0 y J1
    GPIOIntTypeSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_FALLING_EDGE);
// Definir tipo int: flanco bajada
    GPIOIntEnable(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);
//Habilitar pines de interrupción J0, J1
    GPIOIntRegister(GPIO_PORTJ_BASE, rutina_interrupcion);
//Registrar (definir) la rutina de interrupción del puerto J (de los botones)
    IntEnable(INT_GPIOJ);
//Habilitar interrupción del pto J
    IntMasterEnable();
//Habilitar globalmente las ints

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

    /*****************************   TIMER1
*************************************/
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);           //Habilita T1
    TimerClockSourceSet(TIMER1_BASE, TIMER_CLOCK_SYSTEM);   //T1 a 120MHz
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);        //T1 periódico y
conjunto (32b)
    TimerLoadSet(TIMER1_BASE, TIMER_A,(RELOJ/20)-1);            //Se carga
el timer1 con 50ms
    TimerIntRegister(TIMER1_BASE,TIMER_A,IntTimer1);        //Cada vez que
cuente 50ms que entre en la interrupción y refresque el sistema
    IntEnable(INT_TIMER1A);                                 //Habilitar las
interrupciones globales de los timers
```

```
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);        //Habilitar las
interrupciones de timeout
    TimerEnable(TIMER1_BASE, TIMER_A);                      //Habilitar
Timer0 (no hace falta recargarlo, se activa al principio y que cuente
"infinito")

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

    /*****************************   LEDS
*****************************************/
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y
F4: salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y
N1: salidas

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

    /*****************************   MODO BAJO CONSUMO
*****************************************/
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOJ);           //Dejar
despierto el Pto J durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOF);           //Dejar
despierto el Pto G durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPION);           //Dejar
despierto el Pto N durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_TIMER1);          //Dejar
despierto el timer 1 durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_GPIOG);           //Dejar
despierto el Pto G durante el Sleep
    SysCtlPeripheralSleepEnable(SYSCTL_PERIPH_PWM0);            //Dejar
despierto el PWM durante el Sleep
    SysCtlPeripheralClockGating(true);                          //Habilitar
el apagado selectivo de periféricos

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

    // Note: Keep SPI below 11MHz here

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

    SysCtlDelay(RELOJ/3);

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


    Nueva_pantalla(16,16,16);   //Configura una nueva pantalla
```

```
    ComColor(21,160,6);         //Elige el color para el siguiente elemento,
VERDE OSCURO en este caso
    ComLineWidth(5);            //Elige el grosor de la línea del siguiente
elemento
    ComRect(10, 10, HSIZE-10, VSIZE-10, true);      //ComRect(int x1, int y1,
int x2, int y2, char relleno)  //esquina supizq en (10,10) y esqinfder en
(Hsize-10,Vsize-10), con TRUE o FALSE indicamos si el rectángulo esta relleno
o hueco
    ComColor(65,202,42);                            //Elige el color para el
siguiente elemento, VERDE CLARO en este caso
    ComRect(12, 12, HSIZE-12, VSIZE-12, true);      //esquina supizq en
(12,12) y esqinfder en (Hsize-12,Vsize-12), con TRUE o FALSE indicamos si el
rectángulo esta relleno o hueco
    //Este bloque pintaría un rectángulo verde oscuro y luego uno verde claro
encima, quedando un borde verde oscuro

    ComColor(255,255,255);      //Elige color blanco
    ComTXT(HSIZE/2,VSIZE/5, 22, OPT_CENTERX,"MAQUINA DE VENDING");
//ComTXT(int x, int y, int fuente, int ops, char *cadena)
    ComTXT(HSIZE/2,50+VSIZE/5, 22, OPT_CENTERX," P3_2 - SEPA GIERM");
//coordenada x,y donde empieza la cadena; fuente 22; opciones (centrado en
X); "cadena de texto"
    ComTXT(HSIZE/2,100+VSIZE/5, 20, OPT_CENTERX,"F.J.R.C.");

    ComRect(40,40, HSIZE-40, VSIZE-40, false);        //Pinta un rectángulo
blanco vacío con esquinas: (40,40)      ---(HSIZE-40,40)

//     ---                 ---

//   (40,VSIZE-40)---(HSIZE-40,VSIZE-40)
    Dibuja();           //Dibuja toda la lista de comandos anteriores
    Espera_pant();      //Queda en espera hasta cambios de pantalla
(pulsación)

#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

    while(1){                   //Bucle infinito repitiéndose

        while(flag_timer1 == 0 && boton1 !=1 && boton2 !=1) SysCtlSleep();
//Sistema en bajo consumo despertándose por el timer o los botones B1/B2

        switch(estado){         //FSM (ver planteamiento memoria para ver la
utilidad de cada estado más "gráficamente")

        case 0:                 //Estado 0:
            boton1 = 0;         //Se resetean las variables de control de los
botones B1 y B2
            boton2 = 0;
            flag_timer1 = 0;    //Se resetea el flag del timer de 50ms
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado 0
```

```
            if(Bcafe || Bdoritos || Bagua || Bgalletas) estado = 2;     //Si
se pulsa alguno de los botones de la pantalla de alguno de los productos
pasamos al estado 2
            if(devolver == 1) estado = 5;                               //Si
se pulsa el botón de devolver pasamos al estado 5
            break;

        case 1:                 //Estado 1:
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            credito = contmon20 * 20 +contmon5 * 5;     //Se calcula el
crédito tras introducir monedas de 20c o 5c con los botones B1 y B2
            //Credito viene en centimos, interesa separarlo para mostrarlo en
"euros.cts", (mejor mostrar 1.20e que 120c)
            euros = credito/100;            //Se calculan los euros que hay
en credito
            cts = credito%100;              //Se calculan los centimos que
hay en credito
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado 1
            t_0++;
             if(t_0>=T_1)                   //Esperamos 1s, tras introducir
una moneda y se vuelve al estado 0
             {
             t_0=0;
             estado=0;
             }

            break;

        case 2:                 //Estado 2:
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado 2
            t_2++;
            if(t_2>=T_2)                                            //Esperar 2s
y:
            {
                PWMPulseWidthSet(PWM0_BASE, PWM_OUT_4, media);  //Volver
servo al centro
                t_2=0;
                estado = 3;                                         //Pasar al
estado 3
            }

            break;

        case 3:                 //Estado 3:
            Lee_pantalla();                 //Se lee la pantalla aunque no
hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta
indicándole que estamos en estado 3
            t_3++;
            if(t_3>=T_2)                                            //Esperar 2s
y:
```

```
            {
                if (cafe == 1) credito = credito-precio_cafe;           //Si se había elegido cafe, actualizar el crédito disponible restándole el precio del café
                if (doritos == 1) credito = credito-precio_doritos;     //Si se había elegido doritos, actualizar el crédito disponible restándole el precio de los doritos
                if (agua == 1) credito = credito-precio_agua;           //Si se había elegido agua, actualizar el crédito disponible restándole el precio del agua
                if (galletas == 1) credito = credito-precio_galletas;   //Si se había elegido galletas, actualizar el crédito disponible restándole el precio de las galletas
                e0 = 0;
                enciende_leds(e0);        //Apaga los leds
                t_3=0;
                cafe = 0;                 //Resetea todas las variables que gestionan la elección de productos
                doritos = 0;
                agua = 0;
                galletas = 0;
                estado = 5;               //Pasamos al estado 5
            }
            break;

        case 4:                 //Estado 4:
            e0 = 0;
            enciende_leds(e0);              //Apaga los leds
            Lee_pantalla();                 //Se lee la pantalla aunque no hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            pinta_pantalla(estado);         //Se llama a la función que pinta indicándole que estamos en estado 4
            t_4++;
            if(t_4>=T_2)                    //Se espera 2s y:
            {
                t_4=0;
                cafe = 0;                 //Resetea todas las variables que gestionan la elección de productos
                doritos = 0;
                agua = 0;
                galletas = 0;
                estado=0;                   //Volvemos al estado 0
            }
            break;

        case 5:                 //Estado 5:
            devolver = 0;
            Lee_pantalla();                 //Se lee la pantalla aunque no hace falta realmente ya que no estamos usando los valores de POSX y POSY
            Nueva_pantalla(16,16,16);       //Prepara una nueva pantalla
            euros = credito/100;            //Cálcula el crédito restante actualizado tras haber pedido un producto y haberse cobrado
            cts = credito%100;
            pinta_pantalla(estado);         //Se llama a la función que pinta indicándole que estamos en estado 5
            t_5++;
            if(t_5>=T_5)                    //Se espera 5s y se supone que la máquina ya ha devuelto la vuelta que hubiese
```

```
            {
                t_5=0;
                credito = 0;                //Se actualiza el crédito a 0, ya
que se ha devuelto lo que faltase
                euros = 0;                  //Se resetean euros
                cts = 0;                    //Se resetean cts
                contmon20 = 0;              //Se resetea contador de monedas
de 20c
                contmon5 = 0;               //Se resetea contador de monedas
de 5c
                estado=0;                   //Se vuelve de nuevo al estado 0,
estando lista la máquina para volver a funcionar
            }

            break;


        }
    }
}


void IntTimer1(void)                                //Rutina Interrupción
Timer1:
{
    flag_timer1 = 1;                                  //Para que cuando el
micro se despierte cada 50ms, se vuelva a dormir al momento
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);   //Borra flag
Interrupción Timer1

}


                                                    //FIN
```

Como de costumbre, voy a comentar en esta memoria también el código, aunque está comentado con bastante detalle en el propio “.c”.

Siguiendo el orden cronológico del código vamos realizando lo siguiente:

Primeramente declaramos las librerías necesarias para nuestro programa, para a continuación, definir macros útiles como: las posiciones del servo que interesan, macros para los botones de la placa B1 y B2, y macros para los botones de la pantalla de los productos y el botón de devolución del dinero, así como macros para guardar el precio de los diferentes productos. Se definen variables y constantes para la configuración de la pantalla, así como una cantidad relativamente grande de variables y strings a usar en el programa, que no considero necesario explicar una a una ya que está comentada en el código.

Si pasamos a la rutina de interrupción de los botones, en este ejercicio volvemos a querer detectar pulsaciones y no el hecho de que se mantenga pulsado, por tanto volvemos a interrupción tipo FALLING_EDGE, con sus debouncing… Si pulsamos el botón 1 (Introducir moneda de 5c):

- Incrementamos el contador de monedas de 5c
- Pasamos al estado 1
- Activamos la variable que detecta pulsaciones de B1

- Se borra el flag de la interrupción

Si pulsamos el botón 2 (Introducir moneda de 20c):

- Incrementamos el contador de monedas de 20c
- Pasamos al estado 1
- Activamos la variable que detecta pulsaciones de B2
- Se borra el flag de la interrupción

Ponemos el prototipo de la función de interrupción del timer1 y entramos a la función `void pinta_pantalla(int state).`

Sobre esta función recae la mayor parte del funcionamiento del sistema, ya que según el estado que reciba controla las actuaciones de del sistema y lo que tiene que pintar. Se ha hecho así, como ya se ha repetido varias veces, para evitar repetir bloques extensos de código en los diferentes estados de la FSM, quedando el código aún más largo de lo que ya es.

En esta función, se pinta la interfaz por defecto (no voy a comentar elemento a elemento como antes, carece de interés) y según ciertas condiciones varía su funcionamiento.

Si estamos en el estado 1 y se ha pulsado el boton1:

- Se mostrará un mensaje indicando que se ha introducido una moneda de 5c

Si estamos en el estado 1 y se ha pulsado el boton2:

- Se mostrará un mensaje indicando que se ha introducido una moneda de 20c

Si se pulsa alguno de los botones de la pantalla de los productos, o el de devolver, se activa la variable que controla su pulsación.

Si estamos en el estado 2 y:

- Se había elegido café:
  - Si el crédito es insuficiente, vamos al estado 4.
  - Si el crédito es suficiente, se muestra un mensaje de “dispensando producto” y se enciende el led correspondiente, a la vez que se mueve el servo.

Esta operación es análoga para el resto de los 3 productos disponibles.

Si estamos en el estado 3 (ya se ha dispensado el producto) y:

- Se había elegido café:
  - Se muestra un mensaje con “Producto dispensado”

Esta operación es análoga para el resto de los 3 productos disponibles.

Si estamos en el estado 4:

- Se mostrará un mensaje de “No hay crédito disponible”

Si estamos en el estado 5:

- Asigna el crédito actual a la variable creditodevolver, ya que en este punto el crédito actual se ha actualizado restándole el precio del producto elegido.
- Se calcula el número de monedas de cada tipo a devolver
- Se muestra por la ventana de mensajes:

- En la primera fila, los céntimos totales a devolver
- En la segunda fila, el número de monedas de 20c y 5c a devolver

Lo que queda de función vuelve a ser genérico y termina de pintar el resto de elementos de la pantalla presente en todos los estados.

Llegamos ahora al main, donde como de costumbre, configuramos los periféricos a usar, sobre esto ya no merece la pena extenderse. Al igual que en el ejercicio anterior, pintamos una pantalla inicial con el nombre del ejercicio y mi propio nombre.

Con esto hemos llegado al while(1), donde tenemos fuera del switch las comprobaciones de "flags" para dormir el micro o no.

En cuanto a la máquina de estados, en el estado 0 reseteamos el flag del timer de 50ms y las variables de boton1 y boton2 y refrescamos la pantalla, para llamar a la función pinta_pantalla(estado), indicando que estamos en el estado 0, siendo la función pinta_pantalla(estado), la que lleva el peso de gestionar las diferentes actuaciones según en qué estado estemos, para condensar el código como ya he mencionado anteriormente.
De esta manera, si se pulsa alguno de los botones de la pantalla asociados a los productos, pasaremos al estado 2, mientras que, si se pulsa el botón de devolver, se pasa al estado 5.

En el estado 1, se calcula el crédito en función de la cantidad de monedas de cada tipo que tengamos y se actualiza el crédito para mostrarlo en la pantalla. Se muestra durante 1s el mensaje de qué tipo de moneda se ha introducido.

En el estado 2, tras haber movido el servo,encendido el led,etc., tras pasar 2s, devolvemos el servo a la posición central y pasamos al estado 3.

En el estado 3, se actualiza el crédito restándole al crédito disponible el precio del producto elegido, además se apagan los leds y se resetean las variables de elección de productos antes de pasar al estado 5.

En el estado 4, también apagamos los leds, así como mostramos durante 2s un mensaje de "No hay crédito suficiente" para volver al estado 0.

En el estado 5, se resetea la variable "devolver", en caso de que se hubiese llegado a este estado mediante la pulsación del botón de devolver. Sea que hayamos llegado por eso, o por que ya se nos ha cobrado tras haber elegido el producto, en este estado se muestra la cantidad a devolver durante 5s para resetear todas las variables relacionadas con el crédito y regresar al estado 0, de manera que la máquina queda disponible de nuevo para funcionar.

Finalmente, tenemos la interrupción del timer1, donde cada 50ms, se activará el flag de este timer y borraremos el flag de la interrupción de Timeout.

Con esto queda explicado el segundo ejercicio de la práctica.

Antes de concluir la memoria, es importante destacar, aunque creo que ha quedado claro en la explicación, que combino las diferentes funcionalidades de los estados de la máquina de estados entre el código explícito en cada estado y la función "grande" pinta_pantalla(estado), llamada en cada estado, para condensar el código.