Informe pràctica 2

Eduard Martín Graells
Joan Peracaula Prats

March 2018

1 Introducció

En aquesta pràctica hem tingut un primer contacte amb el ports d'Entrada/Sortida de propòsit general (anomenats GPIOs) del microcontrolador. A més, hem refrescat conceptes de programació del flux d'un programa com són els bucles i els salts condicionals, en el llenguatge C.

En particular, hem hagut de modificar el comportament de diferents LEDs en funció del *Joystick* y dels botons *S1* i *S2*. En un primer moment hem hagut de canviar el color que emeten els LEDs RGB situats a la placa superior i més endavant fer diferents progressions amb els LEDs de la placa inferior.

2 Recursos utilitzats

Per realitzar aquesta pràctica hem utilitzat els components d'Entrada/Sortida de la placa *Boosterpack MK II*. En particular hem utilitzat el *Joystick*, tenint en compte totes les accions possibles, és a dir, els pins P4.5 - P5.7 (dreta-esquerra), P5.4 - P5.5 (amunt-avall) i el centre (prémer) amb pin P4.1; la pantalla LCD per tal de controlar les diferents accions i imprimir en quin mode estem; els polsadors S1 i S2, en els pins P5.1 i P3.5 respectivament, i per últim els LEDs, d'una banda els RGB: P2.6 (vermell), P2.4 (verd), P5.6 (blau) situats a la placa *Boosterpack MK II*; i per l'altre banda els 8 LEDs de l'*Adaptador MSP432-Bioloid* connectats al port P7, lògicament, un en cada un dels pins del port.

3 Configuració dels recursos

La configuració i inicialització dels botons, el *Joystick* i els LEDs RGB ja estava feta, tot i així, els comentarem.

Per als botons *S1* i *S2* el que hem fet primerament es elegir la funcionalitat que volem que tinguin amb el registre PxSELi (on x és el port i 'i' el bit 1 o 0), en aquest cas I/O. Com volem que siguin entrades (és el medi exterior el que els activa), els configurem com a tals amb PxDIR. Com que volem que quan s'activin generin una interrupció, així ho indiquem, amb el PxIES (indicant que la interrupció es de *Low to High* voltatge i després activem la interrupció amb PxIE. Per últim, netegem el flag de les interrupcions *setejant* la posició del vector corresponent del pin a 0.

Pel *Joystick*, és el mateix procediment que pels botons ja que és una entrada amb interrupció per cada moviment. Simplement cal prestar atenció a configurar els pins de cada moviment del component.

Pels LEDs, és diferent ja que són components de sortida. Pels RGB (i també pels vuit LEDs del port 7), només hem hagut de indicar que són sortides (recordem que aquests són GIPOs estrictament) amb PxDIR i *setejar-los* a 0 amb PxOUT per inicialitzar-los a apagats.

Per controlar la pantalla LCD s'utilitzen funcions ja donades a la llibreria *lib_PAE2*.

4 Funcions dels recursos

El que hem fet és que al utilitzar un dels botons o el *Joystick*, el sistema entri en una rutina especifica per la interrupció del port, on la tractarem identificant el component que ha generat la interrupció i actualitzant la variable *estado* segons qui hagi interromput el procés. La variable *estado* ens serveix per escollir i tractar l'acció a realitzar després d'una interrupció.

Aleshores, un cop s'ha actualitzat el valor de *estado*, aquest deixa de ser el mateix que *estado_anterior* i això ens permet entrar a l'*if* on actualitzem la pantalla LCD i a més, mitjançant un *switch*, escollim la següent acció a realitzar pel robot en funció de *estado*. Aquestes accions tenen a veure en realitzar canvis als LEDs, ja siguin els RGB o els del port P7. Com que el tracte dels RGB és bastant senzill hem decidit fer-ho dins del *case*, en canvi pel tractament dels LEDs del port 7 hem preferit realitzar crides a funcions auxiliars ja que el seu tractament necessita de recorreguts (*for*) i condicionals (*if*) i hem cregut adient modular-ho.

5 Problemes

Degut a que aquesta pràctica és com una continuació de la primera pràctica introductòria, no hem tingut problemes gaire greus. En un principi, al seguir treballant amb un fitxer antic, hem tingut problemes amb el *debug* perquè després d'una interrupció no es cridava a les funcions específiques de tractament d'aquestes. En un primer moment vam creure que la configuració dels components no era correcte o que alguna cosa malament havíem fet, però més endavant ens vam donar compte que les capçaleres dels mètodes no estaven ben escrites.

També hem tingut una certa dificultat amb l'ús del sistema hexadecimal per referir-nos als pins que ens interessaven d'un port. Creiem que és degut al poc ús pràctic d'aquest sistema, però que confiem que anirem millorant.

D'altra banda, ens hem donat compte per un error, que si no s'inicialitza un pin, no es pot utilitzar, al oblidar-nos de cridar la funció *config_P7_LEDS*.

6 Diagrama de Flux

A continuació afegim una imatge del diagrama de flux d'aquesta pràctica. A més, també l'adjuntem com a fitxer extern a l'informe.

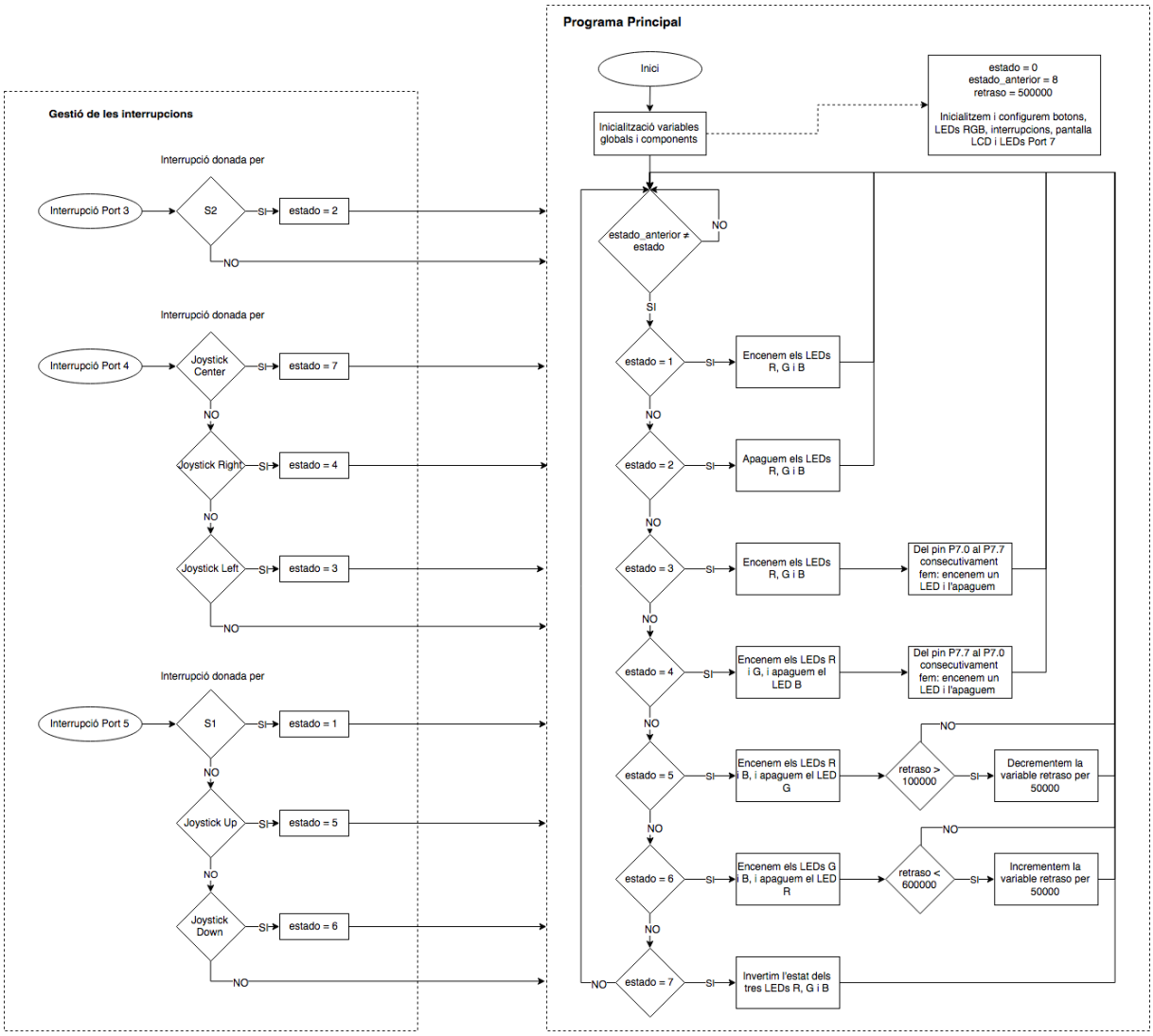

Figure 1: Diagrama de Flux

7 Conclusió

En aquesta pràctica hem estudiat i tractat amb components de tipus GPIOs. Això ens ha servit per aprendre a programar la seva configuració, inicialització i realitzar canvis del seu comportament. Els GPIOs d'entrada ens han permès treballar amb les interrupcions i ens ha ajudat a entendre el funcionament d'aquestes.

En particular hem après que quan es dóna una interrupció es para el flux del programa principal per tractar-la. I per tal que aquest esdeveniment aïllat afecti al programa principal el que fem és actualitzar el valor d'una variable global que analitzem en el bucle infinit del *main*. Un altre aspecte important és que quan es dóna una interrupció cal desactivar la resta d'interrupcions d'aquell port perquè quan estiguem tractant-la no se'n doni cap altra. Al final del tractament cal reactivar les interrupcions d'aquell port.

Pel que fa la metodologia, hem experimentat la importància d'utilitzar *breakpoints* i visualitzar els valors de variables i expressions durant el *debug*. Aquestes eines ens han permès agilitzar el procés de detecció i correcció d'errors.

Gestió de les interrupcions

Interrupció donada per

Interrupció Port 3

S2

SI → estado = 2

NO

Interrupció donada per

Interrupció Port 4

Joystick Center

SI → estado = 7

NO

Joystick Right

SI → estado = 4

NO

Joystick Left

SI → estado = 3

NO

Interrupció donada per

Interrupció Port 5

S1

SI → estado = 1

NO

Joystick Up

SI → estado = 5

NO

Joystick Down

SI → estado = 6

NO

Programa Principal

Inici

Inicialització variables globals i components

estado = 0
estado_anterior = 8
retraso = 500000

Inicialitzem i configurem botons, LEDs RGB, interrupcions, pantalla LCD i LEDs Port 7

estado_anterior ≠ estado

NO

SI

estado = 1

S1

SI → Encenem els LEDs R, G i B

NO

estado = 2

SI → Apaguem els LEDs R, G i B

s2

NO

estado = 3

SI → Encenem els LEDs R, G i B → Del pin P7.0 al P7.7 consecutivament fem: encenem un LED i l'apaguem

JL

NO

estado = 4

SI → Encenem els LEDs R i G, i apaguem el LED B → Del pin P7.7 al P7.0 consecutivament fem: encenem un LED i l'apaguem

NO

estado = 5

SI → Encenem els LEDs R i B, i apaguem el LED G → retraso > 100000

NO

SI → Decrementem la variable retraso per 50000

NO

estado = 6

SI → Encenem els LEDs G i B, i apaguem el LED R → retraso < 600000

NO

SI → Incrementem la variable retraso per 50000

NO

estado = 7

SI → Invertim l'estat dels tres LEDs R, G i B

NO

```c
/*****************************
 *
 * Practica_02_PAE Programaciï¿½ de Ports
 * i prï¿½ctica de les instruccions de control de flux:
 * "do ... while", "switch ... case", "if" i "for"
 * UB, 02/2017.
 *****************************/

#include <msp432p401r.h>
#include <stdio.h>
#include <stdint.h>
#include "lib_PAE2.h" //Libreria grafica + configuracion reloj MSP432

char saludo[16] = " PRACTICA 2 PAE";//max 15 caracteres visibles
char cadena[16];//Una linea entera con 15 caracteres visibles + uno oculto de terminacion de cadena (codigo ASCII 0)
char borrado[] = "               "; //una linea entera de 15 espacios en blanco
uint8_t linea = 1;
uint8_t estado = 0;
uint8_t estado_anterior = 8;
uint32_t retraso = 500000;

/*************************************************************************
 * INICIALIZACIï¿½N DEL CONTROLADOR DE INTERRUPCIONES (NVIC).
 *
 * Sin datos de entrada
 *
 * Sin datos de salida
 *
 *************************************************************************/
void init_interrupciones(){
    // Configuracion al estilo MSP430 "clasico":
    // Enable Port 4 interrupt on the NVIC
    // segun datasheet (Tabla "6-12. NVIC Interrupts", capitulo "6.6.2 Device-Level User Interrupts", p80-81 del documento SLAS826A-Datasheet),
    // la interrupcion del puerto 4 es la User ISR numero 38.
    // Segun documento SLAU356A-Technical Reference Manual, capitulo "2.4.3 NVIC Registers"
    // hay 2 registros de habilitacion ISER0 y ISER1, cada uno para 32 interrupciones (0..31, y 32..63, resp.),
    // accesibles mediante la estructura NVIC->ISER[x], con x = 0 o x = 1.
    // Asimismo, hay 2 registros para deshabilitarlas: ICERx, y dos registros para limpiarlas: ICPRx.

    //Int. port 3 = 37 corresponde al bit 5 del segundo registro ISER1:
    NVIC->ICPR[1] |= BIT5; //Primero, me aseguro de que no quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= BIT5; //y habilito las interrupciones del puerto
    //Int. port 4 = 38 corresponde al bit 6 del segundo registro ISERx:
    NVIC->ICPR[1] |= BIT6; //Primero, me aseguro de que no quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= BIT6; //y habilito las interrupciones del puerto
    //Int. port 5 = 39 corresponde al bit 7 del segundo registro ISERx:
    NVIC->ICPR[1] |= BIT7; //Primero, me aseguro de que no quede ninguna interrupcion residual pendiente para este puerto,
    NVIC->ISER[1] |= BIT7; //y habilito las interrupciones del puerto

    __enable_interrupt(); //Habilitamos las interrupciones a nivel global del micro.
}

/*************************************************************************
 * INICIALIZACIï¿½N DE LA PANTALLA LCD.
```

```
 55  *
 56  * Sin datos de entrada
 57  *
 58  * Sin datos de salida
 59  *
 60  ******************************************************************************/
 61 void init_LCD(void)
 62 {
 63     halLcdInit(); //Inicializar y configurar la pantallita
 64     halLcdClearScreenBkg(); //Borrar la pantalla, rellenando con el color de fondo
 65 }
 66
 67 /*******************************************************************************
 68  * BORRAR LINEA
 69  *
 70  * Datos de entrada: Linea, indica la linea a borrar
 71  *
 72  * Sin datos de salida
 73  *
 74  ******************************************************************************/
 75 void borrar(uint8_t Linea)
 76 {
 77     halLcdPrintLine(borrado, Linea, NORMAL_TEXT); //escribimos una linea en blanco
 78 }
 79
 80 /*******************************************************************************
 81  * ESCRIBIR LINEA
 82  *
 83  * Datos de entrada: Linea, indica la linea del LCD donde escribir
 84  *                   String, la cadena de caracteres que vamos a escribir
 85  *
 86  * Sin datos de salida
 87  *
 88  ******************************************************************************/
 89 void escribir(char String[], uint8_t Linea)
 90
 91 {
 92     halLcdPrintLine(String, Linea, NORMAL_TEXT); //Enviamos la String al LCD,
    sobreescribiendo la Linea indicada.
 93 }
 94
 95 /*******************************************************************************
 96  * INICIALIZACIï¿½N DE LOS BOTONES & LEDS DEL BOOSTERPACK MK II.
 97  *
 98  * Sin datos de entrada
 99  *
100  * Sin datos de salida
101  *
102  ******************************************************************************/
103 void init_botons(void)
104 {
105     //Configuramos botones y leds
106     //***************************
107
108     //Leds RGB del MK II:
109       P2DIR |= 0x50;  //Pines P2.4 (G), 2.6 (R) como salidas Led (RGB)
110       P5DIR |= 0x40;  //Pin P5.6 (B)como salida Led (RGB)
111       P2OUT &= 0xAF;  //Inicializamos Led RGB a 0 (apagados)
112       P5OUT &= ~0x40; //Inicializamos Led RGB a 0 (apagados)
113
114     //Boton S1 del MK II:
```

```
115       P5SEL0 &= ~0x02;   //Pin P5.1 como I/O digital,
116       P5SEL1 &= ~0x02;   //Pin P5.1 como I/O digital,
117       P5DIR &= ~0x02; //Pin P5.1 como entrada
118       P5IES &= ~0x02;   // con transicion L->H
119       P5IE |= 0x02;     //Interrupciones activadas en P5.1,
120       P5IFG = 0;    //Limpiamos todos los flags de las interrupciones del puerto 5
121       //P5REN: Ya hay una resistencia de pullup en la placa MK II
122
123     //Boton S2 del MK II:
124       P3SEL0 &= ~0x20;   //Pin P3.5 como I/O digital,
125       P3SEL1 &= ~0x20;   //Pin P3.5 como I/O digital,
126       P3DIR &= ~0x20; //Pin P3.5 como entrada
127       P3IES &= ~0x20;   // con transicion L->H
128       P3IE |= 0x20;   //Interrupciones activadas en P3.5
129       P3IFG = 0;  //Limpiamos todos los flags de las interrupciones del puerto 3
130       //P3REN: Ya hay una resistencia de pullup en la placa MK II
131
132     //Configuramos los GPIOs del joystick del MK II:
133       P4DIR &= ~(BIT1 + BIT5 + BIT7);   //Pines P4.1, 4.5 y 4.7 como entrades,
134       P4SEL0 &= ~(BIT1 + BIT5 + BIT7);  //Pines P4.1, 4.5 y 4.7 como I/O digitales,
135       P4SEL1 &= ~(BIT1 + BIT5 + BIT7);
136       P4REN |= BIT1 + BIT5 + BIT7;  //con resistencia activada
137       P4OUT |= BIT1 + BIT5 + BIT7;  // de pull-up
138       P4IE |= BIT1 + BIT5 + BIT7;   //Interrupciones activadas en P4.1, 4.5 y 4.7,
139       P4IES &= ~(BIT1 + BIT5 + BIT7);   //las interrupciones se generaran con transicion
    L->H
140       P4IFG = 0;    //Limpiamos todos los flags de las interrupciones del puerto 4
141
142       P5DIR &= ~(BIT4 + BIT5);  //Pines P5.4 y 5.5 como entrades,
143       P5SEL0 &= ~(BIT4 + BIT5); //Pines P5.4 y 5.5 como I/O digitales,
144       P5SEL1 &= ~(BIT4 + BIT5);
145       P5IE |= BIT4 + BIT5;  //Interrupciones activadas en 5.4 y 5.5,
146       P5IES &= ~(BIT4 + BIT5);  //las interrupciones se generaran con transicion L->H
147       P5IFG = 0;    //Limpiamos todos los flags de las interrupciones del puerto 4
148     // - Ya hay una resistencia de pullup en la placa MK II
149 }
150
151 /**********************************************************************
152  * DELAY - A CONFIGURAR POR EL ALUMNO - con bucle while
153  *
154  * Datos de entrada: Tiempo de retraso. 1 segundo equivale a un retraso de 1000000 (aprox)
155  *
156  * Sin datos de salida
157  *
158  **********************************************************************/
159 void delay_t (uint32_t temps)
160 {
161    volatile uint32_t i;
162    i = 0; //Inicialitzem comptador a zero
163
164   //Esperem mentre el comptador no ha arribat a la mateixa quantitat que temps
165   do {
166       i++;
167   } while(i < temps);
168 }
169
170 /**********************************************************************
171  * CONFIGURACIï¿½N DEL PUERTO 7. A REALIZAR POR EL ALUMNO
172  *
173  * Sin datos de entrada
174  *
```

```
175  * Sin datos de salida
176  *
177  ***************************************************************************/
178 void config_P7_LEDS (void)
179 {
180     P7DIR |= 0xFF;  //Tots els pins del port 7 configurats com a sortides
181     P7OUT &= 0x00;  //Inicialitzem els leds del port 7 a 0 (apagats)
182 }
183
184 void ledsprogressiusL2R(void){
185     int i;
186     for(i = 0x01; i <= 0x80; i = i<<1){ //encenem un led, l'apaguem i passem al seguent
    d'esquerra a dreta
187         P7OUT |= i;
188         delay_t(retraso);
189         P7OUT &= ~i;
190     }
191 }
192
193 void ledsprogressiusR2L(void){
194     int i;
195     for(i = 128; i >= 1; i = i>>1){ //encenem un led, l'apaguem i passem al seguent de
    dreta a esquerra
196         P7OUT |= i;
197         delay_t(retraso);
198         P7OUT &= ~i;
199     }
200 }
201
202 void ledsprogressius2F(void){
203     if(retraso < 600000){   //si el retard no supera 600000, augmentem per 50000
204         retraso += 50000;
205     }
206 }
207
208 void ledsprogressius2S(void){
209     if(retraso > 100000){   //si el retard no es menor de 100000, reduim per 50000
210         retraso -= 50000;
211     }
212 }
213
214 void main(void)
215 {
216
217     WDTCTL = WDTPW+WDTHOLD;         // Paramos el watchdog timer
218
219     //Inicializaciones:
220     init_ucs_16MHz();       //Ajustes del clock (Unified Clock System)
221     init_botons();         //Configuramos botones y leds
222     init_interrupciones();  //Configurar y activar las interrupciones de los botones
223     init_LCD();             // Inicializamos la pantalla
224     config_P7_LEDS(); //Inicialitzem els leds del port 7 i els apaguem
225
226     halLcdPrintLine(saludo, linea, INVERT_TEXT); //escribimos saludo en la primera linea
227     linea++;                    //Aumentamos el valor de linea y con ello pasamos a la
    linea siguiente
228
229     //Bucle principal (infinito):
230     do
231     {
232
```

```
233     if (estado_anterior != estado)          // Dependiendo del valor del estado se
    encenderï¿½ un LED u otro.
234     {
235         sprintf(cadena," estado %d", estado);    // Guardamos en cadena la siguiente frase:
    estado "valor del estado"
236         escribir(cadena,linea);          // Escribimos la cadena al LCD
237         estado_anterior = estado;          // Actualizamos el valor de estado_anterior,
    para que no estï¿½ siempre escribiendo.
238 
239         /*******************************************************+
240             A RELLENAR POR EL ALUMNO BLOQUE switch ... case
241         Para gestionar las acciones:
242         Boton S1, estado = 1
243         Boton S2, estado = 2
244         Joystick left, estado = 3
245         Joystick right, estado = 4
246         Joystick up, estado = 5
247         Joystick down, estado = 6
248         Joystick center, estado = 7
249         *******************************************************/
250         switch (estado) {
251             case 1 :
252                 //Si polsem S1, encenem els 3 LEDs RGB
253                 P2OUT |= 0x50;  //Leds P2.4 (G), 2.6 (R) a 1 (encesos)
254                 P5OUT |= 0x40; //Led P5.6(B) a 1 (ences)
255                 break;
256 
257             case 2 :
258                 //Si polsem S2, apaguem els 3 LEDs RGB
259                 P2OUT &= ~0x50;  //Leds P2.4 (G), 2.6 (R) a 0 (apagats)
260                 P5OUT &= ~0x40; //Led P5.6(B) a 1 (apagat)
261                 break;
262 
263             case 3 :
264                 //Si polsem el joystick  a l'esquerra, encenem els 3 LEDs RGB
265                 P2OUT |= 0x50;  //Leds P2.4 (G), 2.6 (R) a 1 (encesos)
266                 P5OUT |= 0x40; //Led P5.6(B) a 1 (ences)
267                 ledsprogressiusR2L();
268                 break;
269 
270             case 4 :
271                 //Si polsem el joystick  a la dreta, LEDs vermell (R) i verd (G)  encesos,
    blau (B) apagat
272                 P2OUT |= 0x50;  //Leds P2.4 (G), 2.6 (R) a 1 (encesos)
273                 P5OUT &= ~0x40; //Led P5.6(B) a 0 (apagat)
274                 ledsprogressiusL2R();
275                 break;
276 
277             case 5 :
278                 //Si polsem el joystick  amunt, LEDs vermell (R) i  blau (B) encesos,  verd
    (G) apagat
279                 P2OUT |= 0x40;  //Led P2.6 (R) a 1 (ences)
280                 P2OUT &= ~0x10;  //Led P2.4 (G) a 0 (apagat)
281                 P5OUT |= 0x40; //Led P5.6(B) a 1 (ences)
282                 ledsprogressius2S();
283                 break;
284 
285             case 6 :
286                 //Si polsem el joystick  avall, LEDs   verd (G)  i blau (B) encesos,
    vermell (R) apagat
287                 P2OUT &= ~0x40;  //Led P2.6 (R) a 0 (apagat)
```

```
288                 P2OUT |= 0x10;  //Led P2.4 (G) a 1 (ences)
289                 P5OUT |= 0x40; //Led P5.6(B) a 1 (ences)
290                 ledsprogressius2F();
291                 break;
292
293             case 7 :
294                 //Si polsem el joystick al centre, s'ha d'invertir l'estat dels 3 LEDs RGB
295                 P2OUT ^= 0x50;  //Invertim leds P2.4 (G), 2.6 (R)
296                 P5OUT ^= 0x40; //Invertim led P5.6(B)
297                 break;
298         }
299     }
300
301      /*P2OUT ^= 0x40;       // Conmutamos el estado del LED R (bit 6)
302      delay_t(retraso);  // periodo del parpadeo
303      P2OUT ^= 0x10;     // Conmutamos el estado del LED G (bit 4)
304      delay_t(retraso);  // periodo del parpadeo
305      P5OUT ^= 0x40;     // Conmutamos el estado del LED B (bit 6)
306      delay_t(retraso);  // periodo del parpadeo*/
307
308     }while(1); //Condicion para que el bucle sea infinito
309 }
310
311
312 /**********************************************************************
313  * RUTINAS DE GESTION DE LOS BOTONES:
314  * Mediante estas rutinas, se detectarï¿½ quï¿½ botï¿½n se ha pulsado
315  *
316  * Sin Datos de entrada
317  *
318  * Sin datos de salida
319  *
320  * Actualizar el valor de la variable global estado
321  *
322  **********************************************************************/
323
324 //ISR para las interrupciones del puerto 3:
325 void PORT3_IRQHandler(void){//interrupcion del pulsador S2
326     uint8_t flag = P3IV; //guardamos el vector de interrupciones. De paso, al acceder a
    este vector, se limpia automaticamente.
327     P3IE &= 0xDF;  //interrupciones del boton S2 en port 3 desactivadas
328     estado_anterior=0;
329
330     /*********************************************************+
331         A RELLENAR POR EL ALUMNO
332     Para gestionar los estados:
333     Boton S1, estado = 1
334     Boton S2, estado = 2
335     Joystick left, estado = 3
336     Joystick right, estado = 4
337     Joystick up, estado = 5
338     Joystick down, estado = 6
339     Joystick center, estado = 7
340     *********************************************************/
341     estado = 2;
342
343     P3IE |= 0x20;   //interrupciones S2 en port 3 reactivadas
344 }
345
346 //ISR para las interrupciones del puerto 4:
347 void PORT4_IRQHandler(void){  //interrupciï¿½n de los botones. Actualiza el valor de la
```

```
   variable global estado.
348     uint8_t flag = P4IV; //guardamos el vector de interrupciones. De paso, al acceder a
   este vector, se limpia automaticamente.
349     P4IE &= 0x5D;   //interrupciones Joystick en port 4 desactivadas
350     estado_anterior=0;
351
352     /*******************************************************+
353         A RELLENAR POR EL ALUMNO BLOQUE switch ... case
354     Para gestionar los estados:
355     Boton S1, estado = 1
356     Boton S2, estado = 2
357     Joystick left, estado = 3
358     Joystick right, estado = 4
359     Joystick up, estado = 5
360     Joystick down, estado = 6
361     Joystick center, estado = 7
362     *******************************************************/
363     switch(flag){
364         case 0x04 : //pin 1
365             estado = 7; //center
366             break;
367         case 0x0C : //pin 5
368             estado = 4; //right
369             break;
370         case 0x10 : //pin 7
371             estado = 3; //Left
372             break;
373     }
374
375
376     /*********************************************
377      * HASTA AQUI BLOQUE CASE
378      *********************************************/
379
380     P4IE |= 0xA2;   //interrupciones Joystick en port 4 reactivadas
381 }
382
383 //ISR para las interrupciones del puerto 5:
384 void PORT5_IRQHandler(void){  //interrupcii¿½n de los botones. Actualiza el valor de la
   variable global estado.
385     uint8_t flag = P5IV; //guardamos el vector de interrupciones. De paso, al acceder a
   este vector, se limpia automaticamente.
386     P5IE &= 0xCD;   //interrupciones Joystick y S1 en port 5 desactivadas
387     estado_anterior=0;
388
389     /*******************************************************+
390         A RELLENAR POR EL ALUMNO BLOQUE switch ... case
391     Para gestionar los estados:
392     Boton S1, estado = 1
393     Boton S2, estado = 2
394     Joystick left, estado = 3
395     Joystick right, estado = 4
396     Joystick up, estado = 5
397     Joystick down, estado = 6
398     Joystick center, estado = 7
399     *******************************************************/
400     switch(flag){
401         case 0x0A : //pin 4
402             estado = 5; //up
403             break;
404         case 0x0C : //pin 5
```

```
405             estado = 6; //down
406             break;
407         case 0x04 : //pin 1
408             estado = 1; //polsador S1
409             break;
410     }
411
412
413     /*********************************************
414      * HASTA AQUI BLOQUE CASE
415      *********************************************/
416
417     P5IE |= 0x32;   //interrupciones Joystick y S1 en port 5 reactivadas
418 }
419
420
```