DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
UNIVERSIDAD POLITÉCNICA DE MADRID

Universidad Politécnica de Madrid
Departamento de Ingeniería Electrónica
E.T.S.I. de Telecomunicación

Memoria del proyecto desarrollado en Sistemas Digitales II (SDG2)

Curso 2016/2017

ArkanoPi

Autores (orden alfabético): Sergio Pérez Morillo

Jaime Pérez Sánchez

Código de la pareja: MT-12

ÍNDICE GENERAL

1 Introducción

Nuestro principal **objetivo** es adquirir los conocimientos necesarios para saber manejar un microcontrolador por medio del desarrollo de un prototipo totalmente funcional, éste se basará en un videojuego muy popular, el **Akanoid**, con la diferencia de que será representado en una matiz de leds. Además se ha añadido la mejora del videojuego **Pong**. Se dividirá en dos subsistemas principales, uno hardware y otro software.

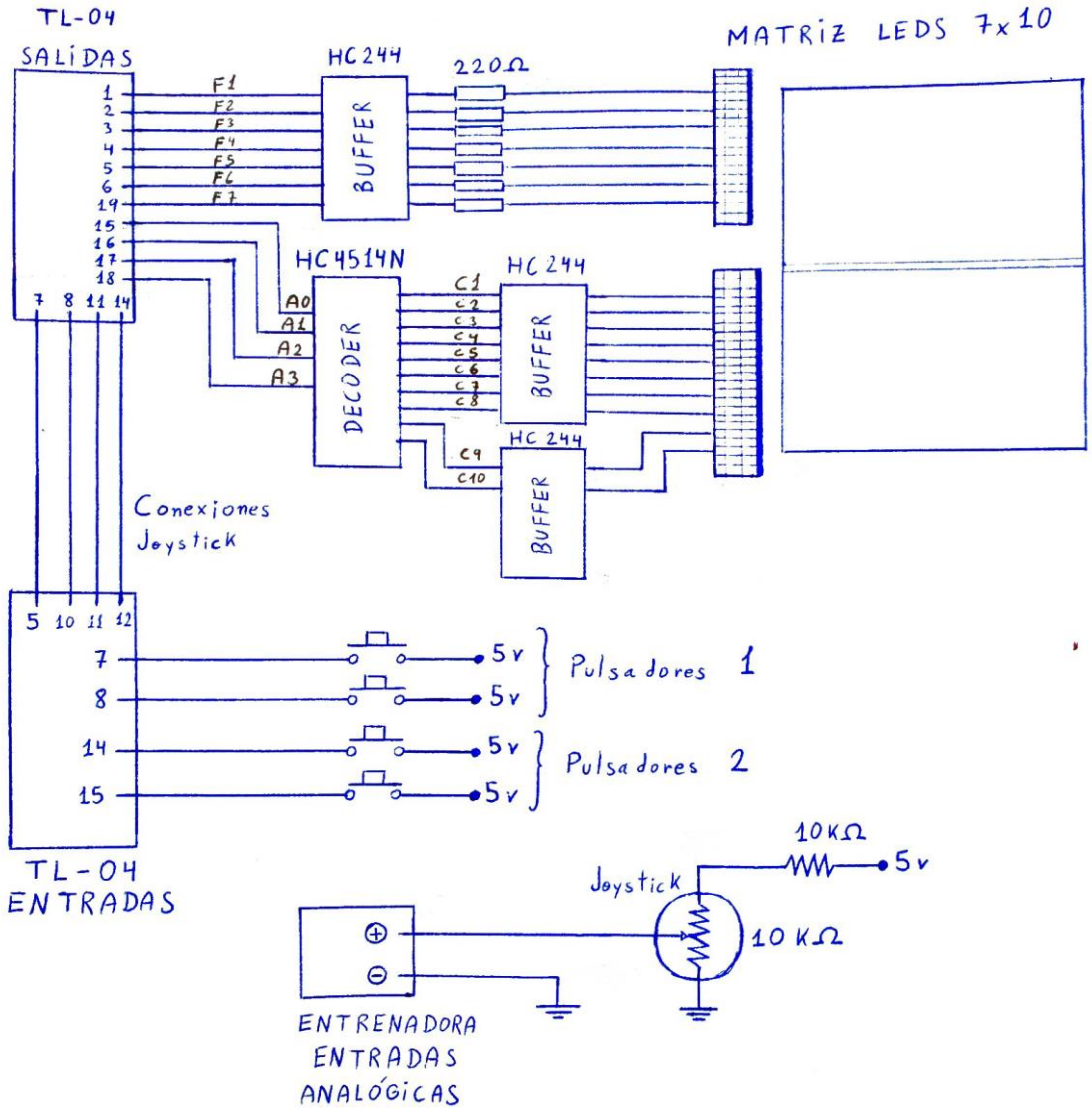
El subsistema **hardware** se encarga de presentar de manera fidedigna la información del juego en la matriz de leds y transferir los datos de manejo del usuario. La parte de visualización consta de dos partes, una para la excitación de las columnas y otra para la excitación de las filas. La parte de manejo consta de 4 pulsadores y 1 joystick.

Para el subsistema **software** utilizaremos la versión de eclipse "Mars" usada mayoritariamente en este tipo de proyectos, aquí escribiremos nuestro código para compilarlo y ejecutarlo en la RaspberryPi, nuestro microcontrolador, gracias a la compilación cruzada.

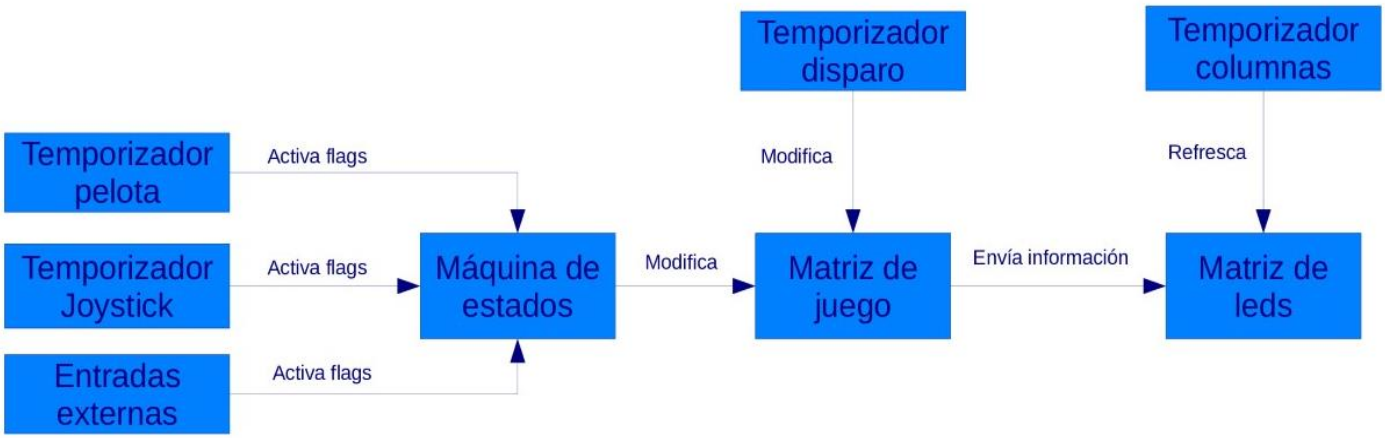
Utilizaremos diversos recursos para desarrollar nuestro código tales como librerías por ejemplo Wiringpi que facilita el uso de la Raspberrypi, máquinas de estado e interrupciones para gestionar de una manera óptima los recursos de los que disponemos o temporizadores que por el momento nos permite ejecutar rutinas periódicas

2 Diagrama de subsistemas

- **Hardware**



- **Software**



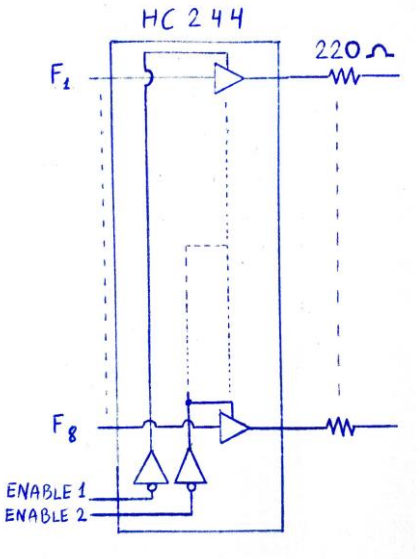
3 Descripción del subsistema Hardware

3.1 Descripción del módulo de las filas

Tiene el objetivo de llevar la información desde los pines de la Raspberry a las filas de la matriz de leds, de forma correcta y óptima para mostrar la información.

Utilizamos 7 pines de la Raspberry que se encuentran conectados a la placa TL-04 y los conectamos a un buffer (HC244) y a resistencias de 220 Ohmios, para controlar la corriente inyectada a la matriz de leds y no sobreexcitarla.

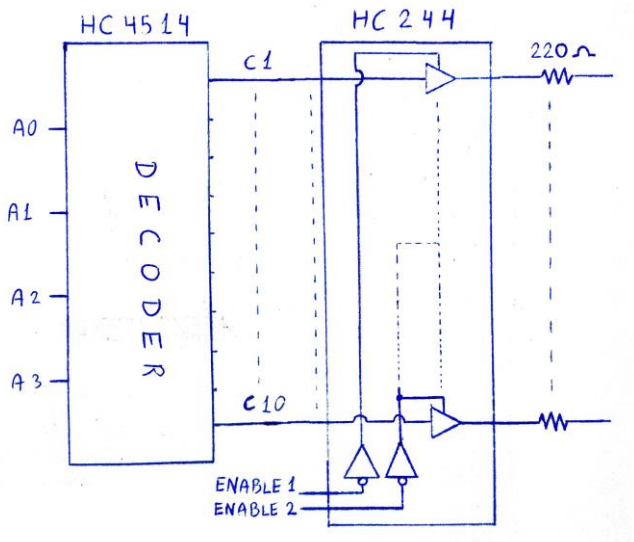
Utilizamos resistencias de 220 Ohmios ya que según la hoja de características de la matriz de leds, la corriente máxima son 30 mA y nosotros aplicamos 5 V. Teniendo en cuenta un margen de seguridad la resistencia necesaria son 200 Ohmios, así que escogemos el valor comercial más cercano: 220 Ohmios.



3.2 Descripción del módulo de las columnas

Tiene el objetivo de llevar la información desde los pines de la Raspberry a las columnas de la matriz de leds, de forma correcta y óptima para mostrar la información.

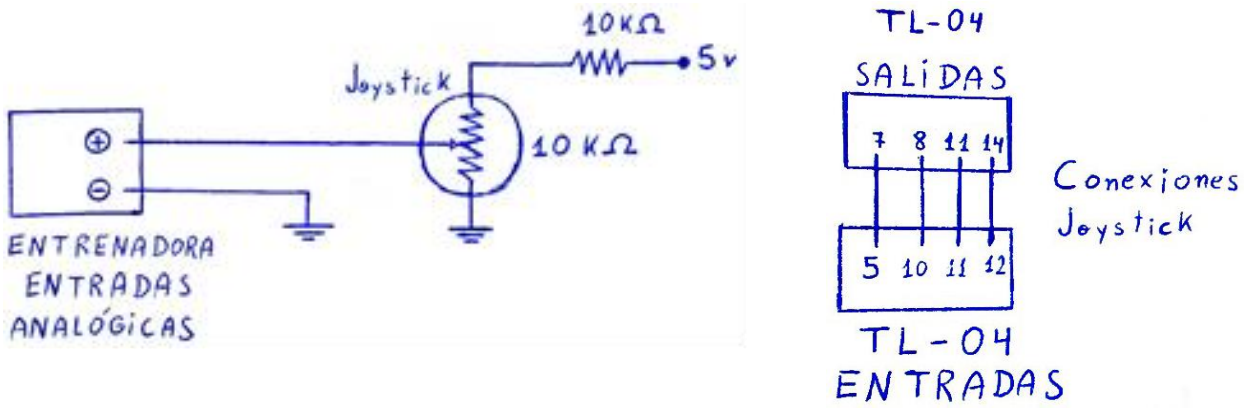
Utilizamos 4 pines de la Raspberry que se encuentran conectados a la placa TL-04 y como necesitamos utilizar 10 columnas, decidimos utilizar un decodificador de 4 a 16 (HC4514), seguido de un buffer (HC244) para limitar la corriente inyectada a la matriz de leds.



3.3 Descripción del módulo del joystick

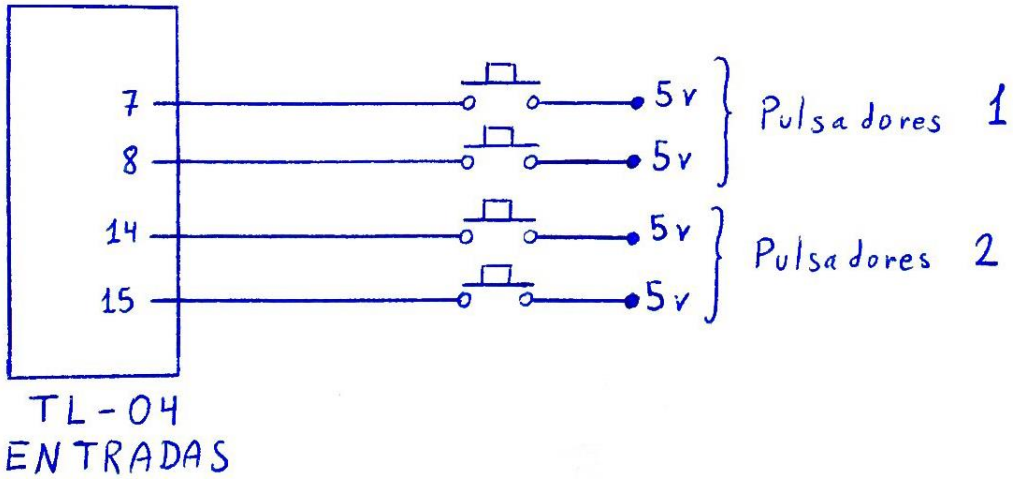
Tiene el objetivo de introducir una tensión (variable) al módulo ADC para controlar la raqueta del juego. Esta tensión es inyectada en la entrada analógica de la entrenadora a través de una pata de un potenciómetro (máx: 10KΩ) que está en serie con una resistencia de 10 KΩ. Se han seleccionado estos valores para que la tensión introducida varíe entre 0 y 2,5 V.

Además se deben realizar una serie de conexiones en las entradas y salidas de la placa TL-04, para sincronizar el CLK, activar el CS, etc…



3.4 Descripción del módulo de pulsadores

Tiene el objetivo de introducir, o no una tensión (1s ó 0s lógicos) a la Raspberry para controlar distintas opciones del juego (movimiento de raqueta, disparo, etc…). Como las entradas digitales ya están conectadas a una resistencia interna de la entrenadora, no es necesario que utilicemos resistencias que limiten la corriente. Únicamente utilizamos botones que cierran el circuito cuando son pulsados.



4 Descripción del subsistema Software

4.1 Flujo de ejecución del programa principal

El programa funciona usando una máquina de estados, interrupciones y temporizadores que cambian la información de la matriz de juego que luego será la representada en la matriz de leds.

En un principio el main ejecuta las funciones de inicialización necesarias y entra en bucle infinito para ejecutar periódicamente la función que cambia de estado la máquina de estados, está cambiara si una flag asociada al estado actual se levanta (interrupción).

Paralelamente en un thread se manipulan las activaciones de las flags (al cumplirse ciertas condiciones), también se pueden activar a través de temporizadores o funciones asociadas a la activación de los pulsadores.

Todo ello será representado en la matriz al declarar las conexiones necesarias en la inicialización del programa y utilizando un temporizador para refrescar las columnas. Para actualizar y pintar en la consola la información del juego se utilizan funciones especialmente diseñadas para ello además de las funciones de salida de la máquina de estados(explicadas más adelante en las interrupciones).

Detallamos a continuación las subrutinas y funcionalidades con más detalle enunciadas anteriormente:

4.1.1 Main

- **Pseudocódigo**:

  wiringPiSetupGpio()
  pinMode(columnas y filas para salidas)
  pinMode(puladores de entrada)
  wiringPiISR (función asociada a cada pulsador);
  temporizador = tmr_new (funcion asociada a cada temporizador);
  tmr_startms(temporizador, retardo);
  transiones = {estado inicial, función entrada, estado siguiente, función salida};
  inicio maquina de estado= fsm_new(estado inicial, transicones);
  systemSetup() y fsm_setup()
  PintaMensaje inical(pantalla de la matriz)
  While(infinito){fsm_fire(inicio maquina de estado)}

- **Funcionalidad**: Inicializar el sistema, máquina de estados, temporizadores necesarios desde el principio del juego y conexiones de entrada y salida.
- **Variables globales**: La pantalla, estado del juego y asociadas a entradas y salidas.
- **Justificación**: Usamos las funciones de inicialización dadas en la asignatura necesarias. Las transiciones son las que diseñamos para nuestra máquina de estados.

4.1.2 Thread cambia estado

- **Pseudocódigo**:

  En Bucle infinito:
  Si estado=START y (Boton1=alto o Boton2=alto) => activa flag TECLA
  Si estado=PONG y (Boton1=alto o Boton2=alto) => flag TECLA
  Si estado=END y (Boton1=alto o Boton2=alto) => flag TECLA
  Si estado=START y Pulso=bajo y (Boton3y4=alto) => flag PONG pulso=alto
  Si estado=PONG y Pulso= bajo y (Boton3y4=alto) => flag PONG pulso=alto
  Si estado=PUSH y Pulso= bajo y (Boton1y2=alto) => flag STOP pulso=alto
  Si estado=STOP y Pulso= bajo y (Boton1y2=alto) => flag STOP pulso=alto
  Si ladrillos=0 o final=verdadero => flag FINAL

Si Boton1=bajo o Boton2=bajo => pulso=bajo
Si Boton3=bajo o Boton4=bajo => pulso=bajo

- **Funcionalidad**: Activa flags si se cumplen las condiciones necesarias.
- **Variables de entrada**: thread_cambia_estado
- **Variables globales**: final, flags, pulsoPong y pulso.
- **Justificación**: Se utilizó un thread para activar flags porque nos resultaba más fácil de administrarlo así. El uso de "final" es necesario para poder pasar del estado PUSH al END. Utilizamos las variables pulsos porque está diseñado para que solo haga una activación por pulsación, sin éstas se activaría repetidamente en una sola pulsación.

4.1.3 Funciones asociadas a pulsadores

- **Pseudocódigo**: (4 funciones: boton_izq, boton_der, boton_izq_pong, boton_der_pong)
  Si Boton1=alto=> activa FLAG_RAQUETA_IZQUIERDA
  Si Boton2=alto=> activa FLAG_RAQUETA_DERECHA
  Si Boton3=alto y pong=0=> activa FLAG_RAQUETA_IZQUIERDA_PONG
  Si Boton3=alto y pong=1 estado=PUSH => Inicia el disparo
  Si Boton4=alto y pong=0=> activa FLAG_RAQUETA_DERECHA_PONG
  Si Boton4=alto y pong=1 y estado=PUSH=> final=verdadero
- **Funcionalidad**: Activa flags, reinicia el juego o inicia el disparo si se pulsa botones.
- **Variables globales**: flags, final y pong.
- **Justificación**: El uso de pong es para poder diferenciar entre juegos de manera fácil y pueda cumplir con nuestro diseño. Final se activa para reiniciar el juego.

4.1.4 Temporizador columnas

- **Pseudocódigo**: (Se inicia en main, se comenta su función asociada timer_isr)
  Incrementamos count en uno
  Apagamos filas y columnas
  Dependiendo del valor de count se enciende la columna correspondiente
  Se enciende las filas con unos correspondientes a la columna
  Se reinicia count si llega al límite de columnas
  tmr_startms(temporizador de columnas, retardo asociado)
- **Funcionalidad**: Temporizador que cada 1ms refresca la información de la matriz columna a columna, al hacerlo rápidamente engaña al ojo pareciendo que se refresca entera la matriz.
- **Variables globales**: count
- **Justificación**: Al ser una interrupción independiente a la máquina de estados no tiene ninguna transición (flag) asociada.

4.1.5 Temporizador pelota

- **Pseudocódigo**: (Se inicia en main, se comenta su función asociada timer_isr_2)
  Activa flag TIMEOUT
- **Funcionalidad**: Sirve para mover periódicamente la pelota, cada segundo.
- **Justificación**: Se implementó así al ser el movimiento de la pelota una interrupción de la máquina de estados

4.1.6 Temporizador joy

- **Pseudocódigo**: (Se inicia en main, se comenta su función asociada timer_isr_3)
  Activa flag JOY

- **Funcionalidad**: Sirve para mover la raqueta, cada 20ms aunque luego se represente cada 100ms (explicado en su interrupción).
- **Justificación**: Se implementó así al ser el movimiento de la pelota una interrupción de la máquina de estados.

4.1.7 Temporizador disparo

- **Pseudocódigo**: (Se inicia en botón3, se comenta su función asociada timer_isr_4)
  X= posición x del disparo
  Y= posición y del disparo
  Si Y no llega a posición de ladrillos => posicionY=-1 (avanza)
  Si Y llega a posición de ladrillos y hay uno=> Elimina ladrillo y disparo
  Si Y llega a posición de ladrillos y no hay uno=> posicionY=-1 (avanza)
  Si Y llega al límite de la matriz => Se elimina disparo
  Si final = falso => actualiza disparo en la matriz y reinicia el contador
- **Funcionalidad**: Mueve el disparo que se originó al pulsar el boton3 cada 200ms y elimina un ladrillo si impacta con éste.
- **Variables globales**: final
- **Justificación**: Se decidió implementar así para ahorrar código con otra flag ya que al probarlo no ralentizaba la ejecución con lo que esta interrupción queda independiente de la máquina de estados.

4.1.8 Actualizar matriz

- **Pseudocódigo**: Once funciones: ActualizaPantalla, ActualizaPantallaPong, Pintadisparo, PintaPelota, PintaDisparo, PintaLadrillos, PintaMarcador, PintaAoP, PintaOK, PintaKO, PintaMensajeInicial.
  - Los que inician su nombre con PintaXXX actualizan la matriz directamente por medio de un for posición a posición conla información que se desea.
  - Los que inician su nombre con ActualizaXXX actualizan la matriz indirectamente llamando a varios PintaXXX a la vez.
- **Funcionalidad**: Modificar la información de la matriz.
- **Variables de entrada**: Pantalla, ladrillos, raqueta… toda la información visual del juego
- **Variables globales**: Pantalla, ladrillos, raqueta… toda la información visual del juego.
- **Justificación**: Se utilizó una matriz para representa la información porque era la manera más fácil de hacerlo dadas las características del proyecto.

4.1.9 Pintar matriz

- **Pseudocódigo**: PintaPantallaPorTerminal
  Bucle for con un printf de una posición de la matriz por cada iteración.
- **Funcionalidad**: Pintar en la consola la matriz del juego
- **Variables de entrada**: Pantalla del juego
- **Justificación**: Implementado de la manera más sencilla posible.

4.1.10 Resetear matriz

- **Pseudocódigo**: Seis funciones: ReseteaMatriz, ReseteaLadrillos, ReseteaPelota, ReseteaDisparo, ReseteaRaqueta, ReseteaRaquetaPong.
  - ReseteaMatriz: Bucle for que recorre la matriz poniendo ceros (borrando información)
  - Los restantes: Bucle for que recorre la matriz situando los objetos de ésta en su lugares originales)

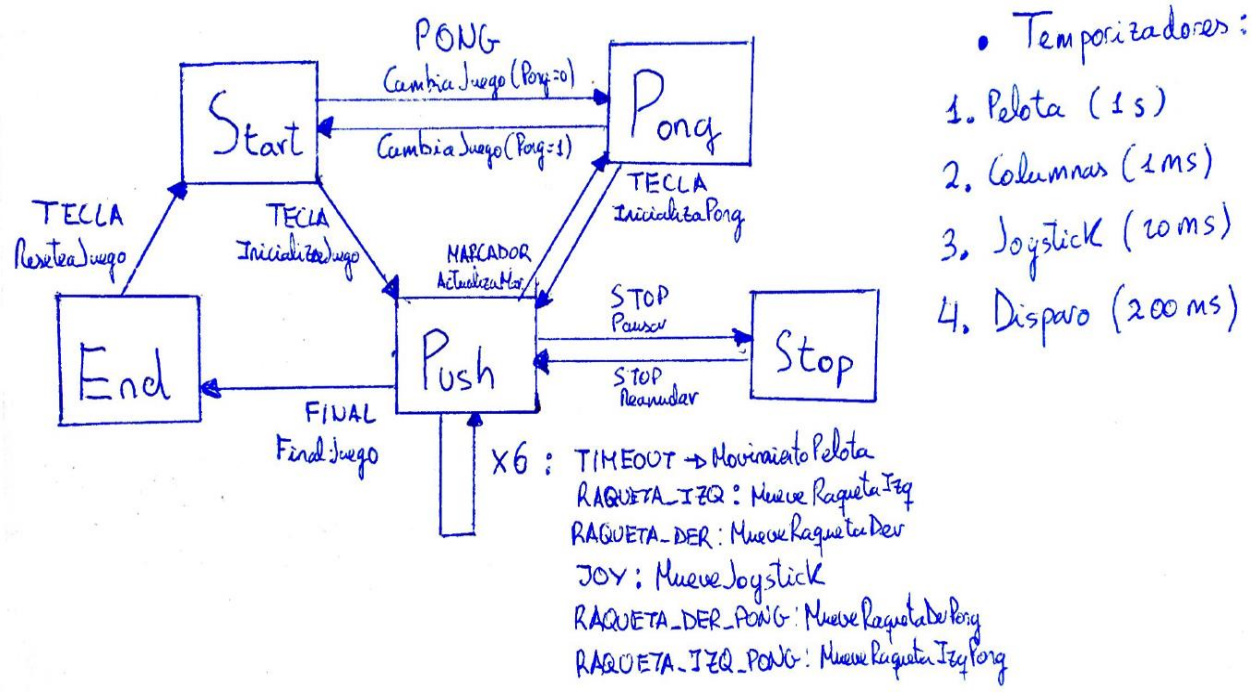
- **Funcionalidad**: Volver a los valores iniciales de la matriz.
- **Variables de entrada**: Pantalla, ladrillos, raqueta… toda la información visual del juego
- **Variables globales**: Pantalla, ladrillos, raqueta… toda la información visual del juego
- **Justificación**: Implementado de la manera más sencilla posible.

4.2 Interrupciones asociadas a la máquina de estados

Decidimos implementar una máquina de estados en nuestro proyecto para gestionar la mayoría de la información de la matriz del juego. Tiene cinco estados: START para el inicio del arkanopi, PONG para el del juego pong, PUSH para cuando se está jugando a ambos juegos, END para cuando termina y STOP para cuando se pausa.

Las transiciones entre estados funcionan de la siguiente manera: la función fsm_fire se ejecuta periódicamente lo cual corre las funciones de entrada del estado actual, estas funciones lo que hacen es comprobar si alguna flag correspondiente a una interrupción se ha levantado, si es así se cambia de estado y se ejecuta la función de salida que baja la flag y modifica la información del juego.

Dado que las funciones de entrada son iguales para todas las transiciones cambiando la flag que esta activada solo se comentará a continuación como se activa la interrupción y su función de saldida.



4.2.1 Interrupciones asociadas al estado START

- **Evento TECLA**: (Función de salida: InicializaJuego) Al pulsar los botones 1 o 2 utiliza las funciones antes descritas de reseteo necesarias para inicializar el juego de arkanopi pasando al estado PUSH para empezar a jugar. Adicionalmente resetea los contadores de los jugadores A y B del pong e inicia el temporizador de la pelota y joystick.
- **Evento PONG**: (Función de salida: CambiaJuego) Al pulsar los botones 3 y 4 a la vez se cambia al estado PONG que en la matriz se refleja en el menú de selección de juego con una P.

4.2.2 Interrupciones asociadas al estado PONG

- **Evento TECLA**: (Función de salida: InicializaPong) Al pulsar los botones 1 o 2 utiliza las funciones antes descritas de reseteo necesarias para inicializar el juego de Pong

pasando al estado PUSH para empezar a jugar. Adicionalmente inicia el temporizador de la pelota y joystick.

- **Evento PONG**: (Función de salida: CambiaJuego) Al pulsar los botones 3 y 4 a la vez se cambia al estado START que en la matriz se refleja en el menú de selección de juego con una A.

4.2.3 Interrupciones asociadas al estado PUSH

- **Evento TIMEOUT**: (Función de salida: MovimientoPelota) Periódicamente se mueve la pelota con las físicas que podéis encontrar en el código bien comentadas. Es de interés comentar que si se pierde (variable final=-1) se para la ejecución del temporizador y que hay unas física para el Arkanopi (pong=0) y para el pong (pong=1).
- **Evento JOY**: (Función de salida: MueveJoystick) Aunque el temporizador haga que se ejecute cada 20ms hay un contador interno para que solo actualice cada 100ms. Cuando llega a 100ms hace una media con los valores que ha recogido del ADC y decide la posición absoluta de la raqueta dependiendo de ésta. Adicionalmente se puso una variable global, joystickAnterior, para que solo actualice si se ha cambiado posición respecto a la anterior. El temporizador solo se sigue ejecutando mientras siga en PUSH.
- **Eventos de pulsadores para la raqueta(4)**: (Función de salida: MueveRaquetaXXX) Los cuatro eventos funcionan de igual manera, pulsas el botón correspondiente y mueve la raqueta a la dirección deseada. Se añadió código para posibles antirrebotes que harían que se moviera más posiciones de las deseadas.
- **Evento STOP**: (Función de salida: Pausar) Al pulsar los botones 1 y 2 a la vez ambos juegos se pararán, se guardará el valor de las flags en ese momento en una variable global y se para los temporizadores de disparo, pelota y joystick. Se pasa al estado STOP.
- **Evento MARCADOR**: (Función de salida: ActualizaMarcador) Al conseguir un punto cualquier jugador en el pong se visualizará el marcador de la partida en la matriz. Si se llega a 7 o se cambia de juego se reiniciará esté. Se cambia al estado PONG para poder reiniciar la posición de las raquetas y la pelota.
- **Evento FINAL**: (Función de salida: FinalJuego) Si está jugando a arkanopi mostrará en la matriz un OK si ha ganado y un KO si ha perdido. Se pasa al estado END.

4.2.4 Interrupciones asociadas al estado PAUSA

- **Evento STOP**: (Función de salida: Reanudar) Al pulsar los botones 1 y 2 a la vez ambos juegos se reanudarán, se le dará a flags el valor de una variable global que guardaba el estado de éstas y reinicia los temporizadores de disparo, pelota y joystick. Se pasa al estado PUSH.

4.2.5 Interrupciones asociadas al estado FINAL

- **Evento TECLA**: (Función de salida: ReseteaJuego) Resetea los contadores de los jugadores del pong y pasa al menú principal pintando una A. Cambia al estado START.

4.3 Interrupciones independientes a la máquina de estados

Hay dos interrupciones que funcionan de manera independiente a la máquina de estados, el disparo cuando se juega al Arkanopi y el refresco de las columnas, éstas ya han sido explicadas anteriormente en sus temporizadores correspondientes.

5 Descripción de las mejoras

5.1 Mejora de videojuego Pong

- **Objetivo:** Implementar un nuevo juego (para dos jugadores) en el sistema.
- **Descripción hardware:**
  Se han añadido dos pulsadores adicionales para el segundo jugador. (Más información en el módulo hardware de pulsadores)
- **Descripción software:**
  Se añade un nuevo estado (PONG) y varios métodos para la visualización de la raqueta del segundo jugador. Para una implementación más eficiente se reutilizan métodos del ArkanoPi. Para que los métodos diferencien entre los juegos se utiliza una variable global (pong), además de otras dos para la puntación de los jugadores.
  Cuando uno de los dos jugadores llega a 7 puntos marcados, se finaliza el juego.

5.2 Mejora de pausa

- **Objetivo:** Pausar el juego mientras estás jugando.

- **Descripción software:**
  Nuevo estado STOP con dos transiciones (pausar y reanudar) en el que se paran los temporizadores y se guarda el valor de las flags para que al reanudar se vuelva al mismo estado donde se dejó la partida.
  - » Pausar: Función que para los temporizadores y guarda el valor de las flags.
  - » Reanudar: Función que pone en funcionamiento los temporizadores y devuelve el valor guardado a las flags.

5.3 Mejora del joystick

- **Objetivo:** Mover la raqueta con un joystick (potenciómetro).
- **Descripción hardware:**
  Como se ha descrito en el subsistema hardware, consistirá en una resistencia y un potenciómetro que al variar éste proporcionará una tensión entre 0 y 2,5 voltios al ADC.
  Se han configurado las conexiones entre entradas y salidas necesarias.
- **Descripción software:**
  Se añade una nueva transición del estado PUSH a él mismo, que su función de salida consiste en que cada 100 ms cambiará la posición de la raqueta dependiendo de la tensión proporcionada al ADC. (Más información en el subsistema software)

5.4 Mejora del menú principal y mensajes por pantalla

- **Objetivo:** Hacer el sistema más intuitivo y fácil de utilizar.
- **Descripción software:**
  Se utiliza el mensaje inicial y las transiciones de estado entre START y PONG para representar el menú principal que muestra una “A” o una “P” dependiendo del juego seleccionado. Dentro del juego ArkanoPi se mostrará un “OK” o “KO” en función de si se gana o se pierde. En el juego Pong se ha introducido un marcador para saber la puntuación, que se muestra en la transición de PUSH a PONG.

5.5 Mejora de disparo y reinicio

- **Objetivo:** Dar funcionalidad en el ArkanoPi a los botones implementados para PONG.

- **Descripción hardware:**
  Se utilizan los pulsadores añadidos para el juego PONG. (Más información en el módulo hardware de pulsadores)
- **Descripción software:**
  - » Reinicio: Cuando se pulsa el botón 4 da la partida por perdida (muestra "KO" por pantalla) y puedes volver al menú principal.
  - » Disparo: Cuando se pulsa el botón 3 se inicia un temporizador que mueve una nueva pelota (interpretada como un disparo) de manera vertical, que o bien elimina un ladrillo si choca con él, o desaparece al llegar al final de la pantalla.

6 Principales problemas encontrados

- **VERSIÓN 1.0**
  No se encontraron problemas al implementar esta versión.

- **VERSIÓN 2.0**
  Problemas irrelevantes a la hora de bajar las flags, provocaba que se ejecutara de manera diferente en consola y eclipse. Se solucionó con facilidad.

- **VERSIÓN 3.0**
  Problemas en la representación en la matriz, había leds que se iluminaban de manera permanente, causado por una salida del decodificador rota, se solucionó usando otra salida en buen estado.

- **VERSIÓN 4.0**
  No se encontraron problemas al implementar esta versión.

- **VERSIÓN 5.0**
  No se encontraron problemas al implementar esta versión.

- **VERSIÓN 6.0**
  Problemas en la mejora del joystick, debidos al desconocimiento de las entradas y salidas que se debían conectar en la placa TL-04 para realizar una lectura correcta.

7 Manual de usuario

- ## Menú principal

  Podrás seleccionar entre los dos juegos: ArkanoPi (visualizará una "A") y Pong (visualizará una "P"). Para cambiar de juego debe pulsar los botones 3 y 4 a la vez, y para empezar pulsar el botón 1 o 2.

- ## ArkanoPi

  - **Joystick:** Mover raqueta hacia izquierda o derecha.
  - **Botón 1:** Mover raqueta hacia la izquierda.
  - **Botón 2:** Mover raqueta hacia la derecha.
  - **Botón 3:** Realizar disparo para eliminar ladrillos.
  - **Botón 4:** Reiniciar juego.
  - **Botón 1 y 2 simultáneamente:** Pausar el juego.

  Una vez que haya perdido o ganado, se mostrará un mensaje en pantalla "KO" o "OK" respectivamente y volverá al menú principal pulsando el botón 1 o 2.

- ## Pong

  - **Joystick:** Mover raqueta hacia izquierda o derecha. (JUGADOR 1)
  - **Botón 1:** Mover raqueta hacia la izquierda. (JUGADOR 1)
  - **Botón 2:** Mover raqueta hacia la derecha. (JUGADOR 1)
  - **Botón 3:** Mover raqueta hacia la izquierda. (JUGADOR 2)
  - **Botón 4:** Mover raqueta hacia la derecha. (JUGADOR 2)
  - **Botón 1 y 2 simultáneamente:** Pausar el juego.

  La partida terminará cuando un jugador consiga 7 puntos. Cada vez que se marque un punto se mostrará el marcador, para seguir jugando pulsar botón 1 o 2.

8 Bibliografía

Vía Moodle:

- Enunciado del proyecto: "Arkanopi: un videojuego arcade para la Raspberry Pi"
- Guion para la primera sesión práctica
- Tutorial 1: Introducción al entorno de desarrollo en C para Raspberry Pi
- Tutorial 2: Prácticas de lenguaje C para Raspberry Pi
- Tutorial 3: Introducción a las máquinas de estados en C para Raspberry Pi
- Tutorial 4: Iniciación al Manejo de las Entradas/Salidas del BCM 2835
- Tutorial 5: Manejo de temporizadores, interrupciones y procesos con la Raspberry Pi
- Tutorial 6: Manejo de periféricos mediante SPI con las Raspberry Pi
- Traspas introductorias para la segunda sesión práctica
- Traspas introductorias para la cuarta sesión práctica
- Traspas introductorias para la quinta y sexta sesiones prácticas
- Traspas introductorias para la séptima sesión práctica
- Hoja de características de la matriz de leds (5x7 DOT MATRIX DISPLAY)

Vía internet:

- Hoja de características del buffer HC244
- Hoja de características del decodificador HC4514

9 ANEXO I: Código del programa del proyecto final

A continuación se incluye el código comentado de los cuatro archivos principales: “arkanoPi_1.c”, “arkanoPi_1.h”, “akanoPiLib.c” y “arkanoPiLib.h”.

```
/*****************************************************************************************************************************************/

/*********************************************  File Name : arkanoPi_1.c  ****************************************************************/

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

#include "arkanoPi_1.h"

volatile int final = 0; // Variable para entrar al estado WAIT_END y terminar juego

enum fsm_state { // ESTADOS DE LA MAQUINA DE ESTADOS, NO DEL JUEGO (WAIT_START, WAIT_END, WAIT_PUSH)
  START=0,
  PUSH=1,
  END=2,
  STOP=3,
  PONG=4,
  };

static volatile tipo_juego juego; // Variable del juego
int jugadorA=0; //Marcadores
int jugadorB=0;
volatile int flags = 0; // Inicializar flags
volatile int flagsStop = 0; //Variable guarda valor de flags
int pulso=0; //Evita repetidas pulsaciones
int pulsoPong=0; //Evita repetidas pulsaciones
int debounceTime;
int pong=0; //Selector de métodos dependiendo del juego
///int joystickAnterior=-10; //Actualiza joystick
int count; //contador columnas
int contadorJoy=0; //contador joystick
float media=0; //media de valores del joystick

// DefiniciÃ³n de los pines de la Raspberry que utilizaremos
#define GPIO_COL_1 14
#define GPIO_COL_2 17
```

```
#define GPIO_COL_3 18
#define GPIO_COL_4 22
#define GPIO_ROW_1 0
#define GPIO_ROW_2 1
#define GPIO_ROW_3 2
#define GPIO_ROW_4 3
#define GPIO_ROW_5 04
#define GPIO_ROW_6 07
#define GPIO_ROW_7 23
#define GPIO_IZQ 16
#define GPIO_DRCHA 19
#define DEBOUNCE_TIME 100
#define REFRESCO 1
#define DISPARO 500
#define MOV_PELOTA 1000
#define MOV_JOY 20
#define SPI_ADC_CH 0
#define SPI_ADC_FREQ 1000000
#define VERBOSE 1
#define GPIO_IZQ_PONG 20
#define GPIO_DRCHA_PONG 21

tmr_t* tmr_pelota;
tmr_t* tmr_col;
tmr_t* tmr_joy;
tmr_t* tmr_disparo;

static int gpio_col[4]={GPIO_COL_1,GPIO_COL_2,GPIO_COL_3,GPIO_COL_4};  // array de columnas
static int gpio_row[7]={GPIO_ROW_1,GPIO_ROW_2,GPIO_ROW_3,GPIO_ROW_4,GPIO_ROW_5,GPIO_ROW_6,GPIO_ROW_7};   // array de filas

static void timer_isr (union sigval arg) { // Rutina de atenciÃ³n al temporizador (periodicamente)
       count++; // Siguiente columna
       int i, j;
       for (i=0; i<4; i++){ // Apagamos columnas
              digitalWrite(gpio_col[i], LOW);
       }
       for (j=0; j<7; j++){ // Apagamos filas
              digitalWrite(gpio_row[i], HIGH);
       }
       // Encendemos columnas
       if(count==0){
              digitalWrite (GPIO_COL_1,LOW);
              digitalWrite (GPIO_COL_2,HIGH);
```

```
            digitalWrite (GPIO_COL_3,LOW);
            digitalWrite (GPIO_COL_4,HIGH);
      }else if(count==1){
            digitalWrite (GPIO_COL_1,HIGH);
            digitalWrite (GPIO_COL_2,LOW);
            digitalWrite (GPIO_COL_3,LOW);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==2){
            digitalWrite (GPIO_COL_1,LOW);
            digitalWrite (GPIO_COL_2,HIGH);
            digitalWrite (GPIO_COL_3,LOW);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==3){
            digitalWrite (GPIO_COL_1,HIGH);
            digitalWrite (GPIO_COL_2,HIGH);
            digitalWrite (GPIO_COL_3,LOW);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==4){
            digitalWrite (GPIO_COL_1,LOW);
            digitalWrite (GPIO_COL_2,LOW);
            digitalWrite (GPIO_COL_3,HIGH);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==5){
            digitalWrite (GPIO_COL_1,HIGH);
            digitalWrite (GPIO_COL_2,LOW);
            digitalWrite (GPIO_COL_3,HIGH);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==6){
            digitalWrite (GPIO_COL_1,LOW);
            digitalWrite (GPIO_COL_2,HIGH);
            digitalWrite (GPIO_COL_3,HIGH);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==7){
            digitalWrite (GPIO_COL_1,HIGH);
            digitalWrite (GPIO_COL_2,HIGH);
            digitalWrite (GPIO_COL_3,HIGH);
            digitalWrite (GPIO_COL_4,LOW);
      }else if(count==8){
            digitalWrite (GPIO_COL_1,LOW);
            digitalWrite (GPIO_COL_2,LOW);
            digitalWrite (GPIO_COL_3,LOW);
            digitalWrite (GPIO_COL_4,HIGH);
      }else if(count==9){
```

```
		digitalWrite (GPIO_COL_1,HIGH);
		digitalWrite (GPIO_COL_2,LOW);
		digitalWrite (GPIO_COL_3,LOW);
		digitalWrite (GPIO_COL_4,HIGH);
	}
	int fila;
	for(fila=0; fila<7; fila++){ // Encendemos filas en funciÃ³n de la columna actual
		if(juego.arkanoPi.pantalla.matriz[count][fila]==1){
			digitalWrite (gpio_row[fila],LOW);
		}else{
			digitalWrite (gpio_row[fila],HIGH);
		}
	}
	if (count == 10) {//reiniciar contador para que no desborde
		count = -1;
	}
	tmr_startms(tmr_col, REFRESCO);
}

static void timer_isr_2 (union sigval arg) {
	flags |= FLAG_TIMEOUT;

}

static void timer_isr_3 (union sigval arg) {
	flags |= FLAG_JOY;
}
//Temporizador de diparo
static void timer_isr_4 (union sigval arg){ //Diparo
	int x = juego.arkanoPi.disparo.x; // PosiciÃ³n absoluta en x
	int y = juego.arkanoPi.disparo.y; // PosiciÃ³n absoluta en y

	if(y > 2){ //Avanza si no ha llegado a la zona de ladrillos
		juego.arkanoPi.disparo.y -= 1;
	}else if((juego.arkanoPi.ladrillos.matriz[x][y-1])==1){ //Si ha llegado a la zona de ladrillos, mira a ver si hay un ladrillo en la
siguiente posición
		juego.arkanoPi.ladrillos.matriz[x][y-1]=0;
			juego.arkanoPi.disparo.x = 12;
			juego.arkanoPi.disparo.y = 12;
			ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
			PintaDisparo((tipo_pelota*) (&(juego.arkanoPi.disparo)), (tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
			piLock (STD_IO_BUFFER_KEY);
			PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
```

```
                        piUnlock (STD_IO_BUFFER_KEY);
                        tmr_stop(tmr_disparo);
        }else if(y<=0){ //Si llega al final de la matriz
                        juego.arkanoPi.disparo.x = 12;
                        juego.arkanoPi.disparo.y = 12;
                        ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
                        PintaDisparo((tipo_pelota*) (&(juego.arkanoPi.disparo)), (tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                        piLock (STD_IO_BUFFER_KEY);
                        PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                        piUnlock (STD_IO_BUFFER_KEY);
                        tmr_stop(tmr_disparo);
        }else if((juego.arkanoPi.ladrillos.matriz[x][y-1])==0){ //Si no hay un ladrillo en la siguiente posición
                juego.arkanoPi.disparo.y -= 1;
        }

        if(final!=-1){ //Se actualiza
                ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
                PintaDisparo((tipo_pelota*) (&(juego.arkanoPi.disparo)), (tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                piLock (STD_IO_BUFFER_KEY);
                PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                piUnlock (STD_IO_BUFFER_KEY);
                tmr_startms(tmr_disparo, DISPARO);
        }
}
// espera hasta la prÃ³xima activaciÃ³n del reloj
void delay_until (unsigned int next) {
        unsigned int now = millis();

        if (next > now) {
                delay (next - now);
    }
}

//-------------------------------------------------------
// FUNCIONES DE LA MAQUINA DE ESTADOS
//-------------------------------------------------------

// FUNCIONES DE ENTRADA O COMPROBACIÃ"N DE LA MAQUINA DE ESTADOS
int CompruebaTeclaPulsada (fsm_t* this) {
        int result;

        piLock (FLAGS_KEY);
        result = (flags & FLAG_TECLA);
```

```
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTeclaPelota (fsm_t* this) {
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_PELOTA);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTeclaRaquetaDerecha (fsm_t* this) {
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_RAQUETA_DERECHA);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTeclaRaquetaIzquierda (fsm_t* this) {
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_RAQUETA_IZQUIERDA);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaJoystick (fsm_t* this){
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_JOY);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaStop (fsm_t* this){
	int result;
```

```
	piLock (FLAGS_KEY);
	result = (flags & FLAG_STOP);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaPong (fsm_t* this){
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_PONG);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTeclaRaquetaIzquierdaPong (fsm_t* this){
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_RAQUETA_IZQUIERDA_PONG);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTeclaRaquetaDerechaPong (fsm_t* this){
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_RAQUETA_DERECHA_PONG);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaTimeout (fsm_t* this){
	int result;

		piLock (FLAGS_KEY);
		result = (flags & FLAG_TIMEOUT);
		piUnlock (FLAGS_KEY);

		return result;
}
int CompruebaMarcador (fsm_t* this) {
```

```
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_MARCADOR);
	piUnlock (FLAGS_KEY);

	return result;
}
int CompruebaFinalJuego (fsm_t* this) {
	int result;

	piLock (FLAGS_KEY);
	result = (flags & FLAG_FINAL_JUEGO);
	piUnlock (FLAGS_KEY);

	return result;
}
//------------------------------------------------------
// FUNCIONES DE SALIDA O ACCION DE LA MAQUINA DE ESTADOS
//------------------------------------------------------

// void InicializaJuego (void): funcion encargada de llevar a cabo
// la oportuna inicializaciÃ³n de toda variable o estructura de datos
// que resulte necesaria para el desarrollo del juego.
void InicializaJuego (fsm_t* fsm) {
	// A completar por el alumno...

	piLock (FLAGS_KEY);
	flags &= ~FLAG_TECLA; //Bajamos flag
	piUnlock (FLAGS_KEY);

	// Reseteamos el juego
	ReseteaMatriz((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
	ReseteaLadrillos((tipo_pantalla*)(&(juego.arkanoPi.ladrillos)));
	ReseteaPelota((tipo_pelota*)(&(juego.arkanoPi.pelota)));
	ReseteaRaqueta((tipo_raqueta*)(&(juego.arkanoPi.raqueta)));
	jugadorA=0;
	jugadorB=0;
	juego.estado = WAIT_PUSH; // Cambiamos de estado de juego
	tmr_startms(tmr_pelota, MOV_PELOTA);
	tmr_startms(tmr_joy, MOV_JOY);
}
```

```
void InicializaPong (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_TECLA; //Bajamos flag
	piUnlock (FLAGS_KEY);

	// Reseteamos el juego
	ReseteaMatriz((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
	ReseteaPelota((tipo_pelota*)(&(juego.arkanoPi.pelota)));
	ReseteaRaqueta((tipo_raqueta*)(&(juego.arkanoPi.raqueta)));
	ReseteaRaquetaPong((tipo_raqueta*)(&(juego.arkanoPi.raqueta_pong)));

	juego.estado = WAIT_PUSH; // Cambiamos de estado de juego
	tmr_startms(tmr_pelota, MOV_PELOTA);
	tmr_startms(tmr_joy, MOV_JOY);
}

void CambiaJuego (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_PONG; // Bajamos flag
	piUnlock (FLAGS_KEY);

	if(pong==0){ //Cambia de Arkanopi a Pong
		pong=1;
		juego.estado = WAIT_PONG;
		PintaAoP(pong, (tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	}else if(pong==1){ //Cambia de Pong a Arkanopi
		pong=0;
		juego.estado = WAIT_START;
		PintaAoP(pong, (tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	}
}

// void MueveRaquetaIzquierda (void): funcion encargada de ejecutar
// el movimiento hacia la izquierda contemplado para la raqueta.
// Debe garantizar la viabilidad del mismo mediante la comprobaciÃ³n
// de que la nueva posiciÃ³n correspondiente a la raqueta no suponga
// que Ã©sta rebase o exceda los lÃ-mites definidos para el Ã¡rea de juego
```

```
// (i.e. al menos uno de los leds que componen la raqueta debe permanecer
// visible durante todo el transcurso de la partida).
void MueveRaquetaIzquierda (fsm_t* fsm) {
	// A completar por el alumno...
	piLock (FLAGS_KEY);
	flags &= ~FLAG_RAQUETA_IZQUIERDA; // Bajamos flag
	piUnlock (FLAGS_KEY);

	if (millis () < debounceTime) {
	debounceTime = millis () + DEBOUNCE_TIME ;
	return;
	}
	// Atención a la interrupción
		if(juego.arkanoPi.raqueta.x>-2){ // Movemos raqueta si no se sale de la pantalla
			juego.arkanoPi.raqueta.x = (juego.arkanoPi.raqueta.x)-1;
		}
		if(pong==0){
			 ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
		 }else if(pong==1){
			 ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
		 }
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	// Wait for key to be released
	while (digitalRead (GPIO_IZQ) == HIGH) {
		delay (1) ;
	}
	debounceTime = millis () + DEBOUNCE_TIME ;
}


// void MueveRaquetaDerecha (void): funciÃ³n similar a la anterior
// encargada del movimiento hacia la derecha.
void MueveRaquetaDerecha (fsm_t* fsm) {
	// A completar por el alumno...
	piLock (FLAGS_KEY);
	flags &= ~FLAG_RAQUETA_DERECHA; // Bajamos flag
	piUnlock (FLAGS_KEY);

	if (millis () < debounceTime) {
	debounceTime = millis () + DEBOUNCE_TIME ;
	return;
```

```
	}
	// Atención a la interrupción

		if(juego.arkanoPi.raqueta.x+3<12){ // Movemos raqueta si no se sale de la pantalla
			juego.arkanoPi.raqueta.x = (juego.arkanoPi.raqueta.x)+1;
		}
		if(pong==0){
			 ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
		 }else if(pong==1){
			 ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
		 }
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	// Wait for key to be released
	while (digitalRead (GPIO_DRCHA) == HIGH) {
		delay (1) ;
	}
	debounceTime = millis () + DEBOUNCE_TIME ;
}

void MueveRaquetaIzquierdaPong (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_RAQUETA_IZQUIERDA_PONG; // Bajamos flag
	piUnlock (FLAGS_KEY);

	if (millis () < debounceTime) {
	debounceTime = millis () + DEBOUNCE_TIME ;
	return;
	}
	// Atención a la interrupción
		if(juego.arkanoPi.raqueta_pong.x>-2){ // Movemos raqueta si no se sale de la pantalla
			juego.arkanoPi.raqueta_pong.x = (juego.arkanoPi.raqueta_pong.x)-1;
		}
		ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	// Wait for key to be released
	while (digitalRead (GPIO_IZQ_PONG) == HIGH) {
		delay (1) ;
	}
	debounceTime = millis () + DEBOUNCE_TIME ;
```

```
}

// void MueveRaquetaDerecha (void): funciÃ³n similar a la anterior
// encargada del movimiento hacia la derecha.
void MueveRaquetaDerechaPong (fsm_t* fsm) {
	// A completar por el alumno...
	piLock (FLAGS_KEY);
	flags &= ~FLAG_RAQUETA_DERECHA_PONG; // Bajamos flag
	piUnlock (FLAGS_KEY);

	if (millis () < debounceTime) {
	debounceTime = millis () + DEBOUNCE_TIME ;
	return;
	}
	// Atención a la interrupción

		if(juego.arkanoPi.raqueta_pong.x+3<12){ // Movemos raqueta si no se sale de la pantalla
			juego.arkanoPi.raqueta_pong.x = (juego.arkanoPi.raqueta_pong.x)+1;
		}
		ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	// Wait for key to be released
	while (digitalRead (GPIO_DRCHA_PONG) == HIGH) {
		delay (1) ;
	}
	debounceTime = millis () + DEBOUNCE_TIME ;
}

void MueveJoystick (fsm_t* fsm){
	piLock (FLAGS_KEY);
	flags &= ~FLAG_JOY;
	piUnlock (FLAGS_KEY);

	contadorJoy++;

	unsigned char ByteSPI[3]; //Buffer lectura escritura SPI
	 int resultado_SPI = 0; //Control operacion SPI
	 float voltaje_medido = 0.0; //Valor medido. A calcular a partir del buffer
	 ByteSPI[0] = 0b10011111; // Configuracion ADC (10011111 unipolar, 0-2.5v,
		//canal 0, salida 1), bipolar 0b10010111
	 ByteSPI[1] = 0b0;
```

```
	ByteSPI[2] = 0b0;
	resultado_SPI = wiringPiSPIDataRW (SPI_ADC_CH, ByteSPI, 3);//Enviamos y leemos
	       //tres bytes (8+12+4 bits)
	usleep(20);
	int salida_SPI = ((ByteSPI[1] << 5) | (ByteSPI[2] >> 3)) & 0xFFF;

	/*Caso unipolar */
	voltaje_medido = 2*2.50 * (((float) salida_SPI)/4095.0);
	media=media+voltaje_medido;
	// Pasados 100 ms selecciona posición de la raqueta haciendo la media de los últimos 5 valores cogidos (Cada 20ms)
	if(contadorJoy==5){
	       contadorJoy=0;
	       media= media/5;
	       if (media>0 && media<0.1 ){
	              juego.arkanoPi.raqueta.x = 9;
	       }else if (media>0.1 && media<0.2){
	              juego.arkanoPi.raqueta.x = 8;
	       }else if (media>0.2 && media<0.3){
	              juego.arkanoPi.raqueta.x = 7;
	       }else if (media>0.3 && media<0.4){
	              juego.arkanoPi.raqueta.x = 6;
	       }else if (media>0.4 && media<0.5){
	              juego.arkanoPi.raqueta.x = 5;
	       }else if (media>0.5 && media<0.6){
	              juego.arkanoPi.raqueta.x = 4;
	       }else if (media>0.6 && media<0.7){
	              juego.arkanoPi.raqueta.x = 3;
	       }else if (media>0.7 && media<0.8){
	              juego.arkanoPi.raqueta.x = 2;
	       }else if (media>0.8 && media<0.9){
	              juego.arkanoPi.raqueta.x = 1;
	       }else if (media>0.9 && media<1.0){
	              juego.arkanoPi.raqueta.x = 0;
	       }else if (media>1.0 && media<1.1){
	              juego.arkanoPi.raqueta.x = -1;
	       }else if (media>1.1 && media<1.2){
	              juego.arkanoPi.raqueta.x = -2;
	       }else{
	             // printf("error");
	       }//Actualiza
	      /// if (joystickAnterior!=juego.arkanoPi.raqueta.x){
	              if(pong==0){
	                     ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
```

```
                        }else if(pong==1){
                              ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
                        }
                        piLock (STD_IO_BUFFER_KEY);
                        PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                        piUnlock (STD_IO_BUFFER_KEY);
                        media=0;
                  /// }
                   ///joystickAnterior=juego.arkanoPi.raqueta.x;
            }
            resultado_SPI=resultado_SPI+1;

           #ifdef VERBOSE
                  /** piLock (STD_IO_BUFFER_KEY);
                   printf("Lectura ADC MAX1246: %d\n", resultado_SPI);
                   printf("Primer byte: %02X \n", ByteSPI[0]);
                   printf("Segundo Byte: %02X \n", ByteSPI[1]);
                   printf("Tercer byte: %02X \n", ByteSPI[2]);**/
                   //printf("Valor entero: %i \n", salida_SPI);
                   //printf("Voltaje medido: %f \n",voltaje_medido);
                   //fflush(stdout);
                   //piUnlock (STD_IO_BUFFER_KEY);
            #endif
            if (juego.estado == WAIT_PUSH){ //Si seguimos jugando se reinicia el temporizador
                  tmr_startms(tmr_joy, MOV_JOY);
            }
}

// void MovimientoPelota (void): funciÃ³n encargada de actualizar la
// posiciÃ³n de la pelota conforme a la trayectoria definida para Ã©sta.
// Para ello deberÃ¡ identificar los posibles rebotes de la pelota para,
// en ese caso, modificar su correspondiente trayectoria (los rebotes
// detectados contra alguno de los ladrillos implicarÃ¡n adicionalmente
// la eliminaciÃ³n del ladrillo). Del mismo modo, deberÃ¡ tambiÃ©n
// identificar las situaciones en las que se dÃ© por finalizada la partida:
// bien porque el jugador no consiga devolver la pelota, y por tanto Ã©sta
// rebase el lÃ-mite inferior del Ã¡rea de juego, bien porque se agoten
// los ladrillos visibles en el Ã¡rea de juego.
void MovimientoPelota (fsm_t* fsm) {
      // A completar por el alumno...
      piLock (FLAGS_KEY);
      flags &= ~FLAG_PELOTA; // Bajamos flag
      flags &= ~FLAG_TIMEOUT;
```

```
    piUnlock (FLAGS_KEY);

    int x = juego.arkanoPi.pelota.x; // PosiciÃ³n absoluta en x
    int y = juego.arkanoPi.pelota.y; // PosiciÃ³n absoluta en y
    int xv = juego.arkanoPi.pelota.xv; // Trayectoria en x
    int yv = juego.arkanoPi.pelota.yv; // Trayectoria en y

if(pong==0){
          // LÃ³gica de cambio de trayectoria por choque

          // Baja drcha
          if((xv==1) && (yv==1)){
                if(x>=9){                                  // Choca con pared
                      juego.arkanoPi.pelota.xv=-1; // Izq
                }if((y)>=5){                        // Choca con raqueta o pierdes
                      if(x>=9){      // Choca con pared
                            if((x-1)==(juego.arkanoPi.raqueta.x)+2){ // Choca con raqueta drcha
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=-1; // Sube izq
                            }else if( (x-1) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=0; // Sube recto
                            }else if( (x-1) == ((juego.arkanoPi.raqueta.x)) ){ // Choca con raqueta izq
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=-1; // Sube izq
                            }else{
                                  piLock (FINAL_KEY);
                                  final = -1;
                                  piUnlock (FINAL_KEY);
                            }
                      }else{ // No choca con pared
                            if((x+(juego.arkanoPi.pelota.xv))==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=-1; // Sube izq
                            }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=0; // Sube recto
                            }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                                  juego.arkanoPi.pelota.yv=-1;
                                  juego.arkanoPi.pelota.xv=1; // Sube drcha
                            }else{
                                  piLock (FINAL_KEY);
                                  final = -1; // Termina el juego
```

```
                                piUnlock (FINAL_KEY);
                        }
                }
        }if((juego.arkanoPi.ladrillos.matriz[x+1][y+1])==1){ //Choca con ladrillo
                juego.arkanoPi.pelota.yv=1;
                juego.arkanoPi.pelota.xv=-1; // Baja izq
                juego.arkanoPi.ladrillos.matriz[x+1][y+1]=0;
        }

// Sube drcha
}else if((xv==1) && (yv==-1)){
        if((x)>=9){          // Choca con pared
                juego.arkanoPi.pelota.xv=-1; // Izq
                if( (juego.arkanoPi.ladrillos.matriz[x-1][y-1]) == 1){ // Choca con pared y ladrillo
                        (juego.arkanoPi.ladrillos.matriz[x-1][y-1]) = 0;
                        juego.arkanoPi.pelota.yv=1; // Baja
                }
        }if(y<=0){                  // Choca con techo
                juego.arkanoPi.pelota.yv=1; // Baja
                if((juego.arkanoPi.ladrillos.matriz[x+1][y+1])==1){ // Choca con techo y ladrillo
                        juego.arkanoPi.pelota.xv=-1; // Izq
                        (juego.arkanoPi.ladrillos.matriz[x+1][y+1]) = 0;
                }
        }if((juego.arkanoPi.ladrillos.matriz[x+(juego.arkanoPi.pelota.xv)][y-1])==1){ //Choca con ladrillo
                juego.arkanoPi.pelota.yv=1;
                juego.arkanoPi.pelota.xv=1; // Baja drcha
                juego.arkanoPi.ladrillos.matriz[x+1][y-1]=0;
        }

// Baja recto
}else if((xv==0) && (yv==1)){
        if((y)>=5){                        // Choca con raqueta o pierdes
                if( x == (juego.arkanoPi.raqueta.x) ){ // Choca con raqueta izq
                        if(x<=0){           // Choca con pared
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                        }else{
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                        }
                }else if( x == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                        juego.arkanoPi.pelota.yv=-1;
                        juego.arkanoPi.pelota.xv=0; // Sube recto
```

```
                }else if( x == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                        if((x)>=9){          // Choca con pared
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                        }else{
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                        }
                }else{
                        piLock (FINAL_KEY);
                        final = -1; // Termina el juego
                        piUnlock (FINAL_KEY);
                }
        }

// Sube recto
}else if((xv==0) && (yv==-1)){
        if(y<=0){            // Choca con techo
                juego.arkanoPi.pelota.yv=1; // Baja
        }if((juego.arkanoPi.ladrillos.matriz[x][y-1])==1){ //Choca con ladrillo
                juego.arkanoPi.pelota.yv=1;
                juego.arkanoPi.pelota.xv=0; // Baja recto
                juego.arkanoPi.ladrillos.matriz[x][y-1]=0;
        }

// Baja izq
}else if((xv==-1) && (yv==1)){
        if(x<=0){                                  // Choca con pared
                (juego.arkanoPi.pelota.xv)=1; // Drcha
        }if((y)>=5){ // Choca con raqueta o pierdes
                if(x<=0){      // Choca con pared
                        if((x+1)==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                        }else if( (x+1) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=0; // Sube recto
                        }else if( (x+1) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                        }else{
                                piLock (FINAL_KEY);
                                final = -1; // Termina el juego
```

```
                                piUnlock (FINAL_KEY);
                        }
                }else{ // No choca con pared
                        if((x+(juego.arkanoPi.pelota.xv))==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                        }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=0; // Sube recto
                        }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                                juego.arkanoPi.pelota.yv=-1;
                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                        }else{
                                piLock (FINAL_KEY);
                                final = -1; // Termina el juego
                                piUnlock (FINAL_KEY);
                        }
                }
        }if((juego.arkanoPi.ladrillos.matriz[x-1][y+1])==1){ //Choca con ladrillo
                juego.arkanoPi.pelota.yv=1;
                juego.arkanoPi.pelota.xv=1; // Baja drcha
                juego.arkanoPi.ladrillos.matriz[x-1][y+1]=0;
        }

// Sube izq
}else if((xv==-1) && (yv==-1)){
        if((x)<=0){          // Choca con pared
                juego.arkanoPi.pelota.xv=1; // Drcha
                if( (juego.arkanoPi.ladrillos.matriz[x+1][y-1]) == 1){ // Choca con pared y ladrillo
                        (juego.arkanoPi.ladrillos.matriz[x+1][y-1]) = 0;
                        juego.arkanoPi.pelota.yv=1; // Baja
                }
        }if(y<=0){                  // Choca con techo
                juego.arkanoPi.pelota.yv=1; // Baja
                if((juego.arkanoPi.ladrillos.matriz[x-1][y+1])==1){ // Choca con techo y ladrillo
                        juego.arkanoPi.pelota.xv=1; // Drcha
                        (juego.arkanoPi.ladrillos.matriz[x-1][y+1]) = 0;
                }
        }if((juego.arkanoPi.ladrillos.matriz[x-1][y-1])==1){ //Choca con ladrillo
                juego.arkanoPi.pelota.yv=1;
                juego.arkanoPi.pelota.xv=-1; // Baja izq
                juego.arkanoPi.ladrillos.matriz[x-1][y-1]=0;
        }
```

```
		}

		// LÃ³gica de movimiento
		(juego.arkanoPi.pelota.x)=(juego.arkanoPi.pelota.x)+(juego.arkanoPi.pelota.xv);
		(juego.arkanoPi.pelota.y)=(juego.arkanoPi.pelota.y)+(juego.arkanoPi.pelota.yv);

		ActualizaPantalla((tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
		if(final != -1){
			tmr_startms(tmr_pelota, MOV_PELOTA);
		}

	}else if (pong==1){
		// LÃ³gica de cambio de trayectoria por choque

			// Baja drcha
			if((xv==1) && (yv==1)){
				if(x>=9){                                  // Choca con pared
					juego.arkanoPi.pelota.xv=-1; // Izq
				}if((y)>=5){                        // Choca con raqueta o pierdes
					if(x>=9){      // Choca con pared
						if((x-1)==(juego.arkanoPi.raqueta.x)+2){ // Choca con raqueta drcha
							juego.arkanoPi.pelota.yv=-1;
							juego.arkanoPi.pelota.xv=-1; // Sube izq
						}else if( (x-1) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
							juego.arkanoPi.pelota.yv=-1;
							juego.arkanoPi.pelota.xv=0; // Sube recto
						}else if( (x-1) == ((juego.arkanoPi.raqueta.x)) ){ // Choca con raqueta izq
							juego.arkanoPi.pelota.yv=-1;
							juego.arkanoPi.pelota.xv=-1; // Sube izq
						}else{
							jugadorA++;
							if(jugadorA==7 || jugadorB==7){
								piLock (FINAL_KEY);
									final = -1;
									piUnlock (FINAL_KEY);
							}if (jugadorA<8 && jugadorB<8){
								piLock (FLAGS_KEY);
									flags |= FLAG_MARCADOR; // Activamos flag Tecla
									piUnlock (FLAGS_KEY);
```

```
                                        }
                                }
                        }else{ // No choca con pared
                                if((x+(juego.arkanoPi.pelota.xv))==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                                        juego.arkanoPi.pelota.yv=-1;
                                        juego.arkanoPi.pelota.xv=-1; // Sube izq
                                }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta
centro
                                        juego.arkanoPi.pelota.yv=-1;
                                        juego.arkanoPi.pelota.xv=0; // Sube recto
                                }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta
drcha
                                        juego.arkanoPi.pelota.yv=-1;
                                        juego.arkanoPi.pelota.xv=1; // Sube drcha
                                }else{
                                        jugadorA++;
                                        if(jugadorA==7 || jugadorB==7){
                                                piLock (FINAL_KEY);
                                                        final = -1;
                                                        piUnlock (FINAL_KEY);
                                        }else if (jugadorA<8 && jugadorB<8){
                                                piLock (FLAGS_KEY);
                                                        flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                        piUnlock (FLAGS_KEY);
                                        }
                                }
                        }
                }if((juego.arkanoPi.ladrillos.matriz[x+1][y+1])==1){ //Choca con ladrillo
                        juego.arkanoPi.pelota.yv=1;
                        juego.arkanoPi.pelota.xv=-1; // Baja izq
                        juego.arkanoPi.ladrillos.matriz[x+1][y+1]=0;
                }

        //Sube drcha
        }else if((xv==1) && (yv==-1)){
           if(x>=9){ //Choca con pared
                 juego.arkanoPi.pelota.xv=-1;
           }if(y<=1){ //Choca con raqueta o pierde
              if(x>=9){
                 if((x-1)==(juego.arkanoPi.raqueta_pong.x)+2){
                    juego.arkanoPi.pelota.yv=1;
                                juego.arkanoPi.pelota.xv=-1; // Baja izq
                 }else if( (x-1) == ((juego.arkanoPi.raqueta_pong.x)+1) ){ // Choca con raqueta centro
```

```
                                        juego.arkanoPi.pelota.yv=1;
                                        juego.arkanoPi.pelota.xv=0; // Baja recto
                                    }else if( (x-1) == ((juego.arkanoPi.raqueta_pong.x)) ){ // Choca con raqueta izq
                                        juego.arkanoPi.pelota.yv=1;
                                        juego.arkanoPi.pelota.xv=-1; // Baja izq
                                    }else{
                                        jugadorB++;
                                        if(jugadorA==7 || jugadorB==7){
                                            piLock (FINAL_KEY);
                                                final = -1;
                                                piUnlock (FINAL_KEY);
                                        }else if (jugadorA<8 && jugadorB<8){
                                            piLock (FLAGS_KEY);
                                                flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                piUnlock (FLAGS_KEY);
                                        }
                                    }
                            }else{ //No choca con pared
                               if((x+1)==(juego.arkanoPi.raqueta_pong.x)){ // Choca con raqueta izq
                                        juego.arkanoPi.pelota.yv=1;
                                        juego.arkanoPi.pelota.xv=-1; // Baja izq
                                    }else if( (x+1) == ((juego.arkanoPi.raqueta_pong.x)+1) ){ // Choca con raqueta centro
                                        juego.arkanoPi.pelota.yv=1;
                                        juego.arkanoPi.pelota.xv=0; // Baja recto
                                    }else if( (x+1) == ((juego.arkanoPi.raqueta_pong.x)+2) ){ // Choca con raqueta drcha
                                        juego.arkanoPi.pelota.yv=1;
                                        juego.arkanoPi.pelota.xv=1; // Baja drcha
                                    }else{
                                        jugadorB++;
                                        if(jugadorA==7 || jugadorB==7){
                                            piLock (FINAL_KEY);
                                                final = -1;
                                                piUnlock (FINAL_KEY);
                                        }else if (jugadorA<8 && jugadorB<8){
                                            piLock (FLAGS_KEY);
                                                flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                piUnlock (FLAGS_KEY);
                                        }
                                    }
                            }
                        }

                    // Baja recto
```

```
                }else if((xv==0) && (yv==1)){
                        if((y)>=5){                         // Choca con raqueta o pierdes
                                if( x == (juego.arkanoPi.raqueta.x) ){ // Choca con raqueta izq
                                        if(x<=0){           // Choca con pared
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                                        }else{
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                                        }
                                }else if( x == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                                        juego.arkanoPi.pelota.yv=-1;
                                        juego.arkanoPi.pelota.xv=0; // Sube recto
                                }else if( x == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                                        if((x)>=9){         // Choca con pared
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                                        }else{
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                                        }
                                }else{
                                        jugadorA++;
                                        if(jugadorA==7 || jugadorB==7){
                                                piLock (FINAL_KEY);
                                                        final = -1;
                                                        piUnlock (FINAL_KEY);
                                        }else if (jugadorA<8 && jugadorB<8){
                                                piLock (FLAGS_KEY);
                                                        flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                        piUnlock (FLAGS_KEY);
                                        }
                                }
                        }

                // Sube recto
                }else if((xv==0) && (yv==-1)){
                   if(y<=1){   // Choca con raqueta o pierdes
                      if( x == (juego.arkanoPi.raqueta_pong.x) ){ //Choca con raqueta izq
                         if(x<=0){
                            juego.arkanoPi.pelota.yv=1;
                                                juego.arkanoPi.pelota.xv=1; // Baja drcha
                         }else{
```

```
                  juego.arkanoPi.pelota.yv=1;
                                    juego.arkanoPi.pelota.xv=-1; // Baja izq
               }
            }else if( x == ((juego.arkanoPi.raqueta_pong.x)+1) ){ // Choca con raqueta centro
               juego.arkanoPi.pelota.yv=1;
                              juego.arkanoPi.pelota.xv=0; // Baja recto
            }else if( x == ((juego.arkanoPi.raqueta_pong.x)+2) ){ // Choca con raqueta drcha
               if((x)>=9){          // Choca con pared
                              juego.arkanoPi.pelota.yv=1;
                              juego.arkanoPi.pelota.xv=-1; // Baja izq
                           }else{
                              juego.arkanoPi.pelota.yv=1;
                              juego.arkanoPi.pelota.xv=1; // Baja drcha
                           }
            }else{
                           jugadorB++;
                           if(jugadorA==7 || jugadorB==7){
                              piLock (FINAL_KEY);
                                 final = -1;
                                 piUnlock (FINAL_KEY);
                           }else if (jugadorA<8 && jugadorB<8){
                              piLock (FLAGS_KEY);
                                 flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                 piUnlock (FLAGS_KEY);
                           }
                           }
      }

   // Baja izq
      }else if((xv==-1) && (yv==1)){
             if(x<=0){                                 // Choca con pared
                    (juego.arkanoPi.pelota.xv)=1; // Drcha
             }if((y)>=5){ // Choca con raqueta o pierdes
                    if(x<=0){      // Choca con pared
                           if((x+1)==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                              juego.arkanoPi.pelota.yv=-1;
                              juego.arkanoPi.pelota.xv=1; // Sube drcha
                           }else if( (x+1) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta centro
                              juego.arkanoPi.pelota.yv=-1;
                              juego.arkanoPi.pelota.xv=0; // Sube recto
                           }else if( (x+1) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta drcha
                              juego.arkanoPi.pelota.yv=-1;
                              juego.arkanoPi.pelota.xv=1; // Sube drcha
```

```
                                        }else{
                                                jugadorA++;
                                                if(jugadorA==7 || jugadorB==7){
                                                        piLock (FINAL_KEY);
                                                                final = -1;
                                                                piUnlock (FINAL_KEY);
                                                }else if (jugadorA<8 && jugadorB<8){
                                                        piLock (FLAGS_KEY);
                                                                flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                                piUnlock (FLAGS_KEY);
                                                }
                                        }
                                }else{ // No choca con pared
                                        if((x+(juego.arkanoPi.pelota.xv))==(juego.arkanoPi.raqueta.x)){ // Choca con raqueta izq
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=-1; // Sube izq
                                        }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+1) ){ // Choca con raqueta
centro
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=0; // Sube recto
                                        }else if( (x+(juego.arkanoPi.pelota.xv)) == ((juego.arkanoPi.raqueta.x)+2) ){ // Choca con raqueta
drcha
                                                juego.arkanoPi.pelota.yv=-1;
                                                juego.arkanoPi.pelota.xv=1; // Sube drcha
                                        }else{
                                                jugadorA++;
                                                if(jugadorA==7 || jugadorB==7){
                                                        piLock (FINAL_KEY);
                                                                final = -1;
                                                                piUnlock (FINAL_KEY);
                                                }else if (jugadorA<8 && jugadorB<8){
                                                        piLock (FLAGS_KEY);
                                                                flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                                piUnlock (FLAGS_KEY);
                                                }
                                        }
                                }
                        }if((juego.arkanoPi.ladrillos.matriz[x-1][y+1])==1){ //Choca con ladrillo
                                juego.arkanoPi.pelota.yv=1;
                                juego.arkanoPi.pelota.xv=1; // Baja drcha
                                juego.arkanoPi.ladrillos.matriz[x-1][y+1]=0;
                        }
```

```
                // Sube izq
                }else if((xv==-1) && (yv==-1)){
                   if(x<=0){ // Choca con pared
                      juego.arkanoPi.pelota.xv=1;
                   }if(y<=1){ // Choca con raqueta o pierdes
                      if(x<=0){
                         if( (x+1)==(juego.arkanoPi.raqueta_pong.x) ){ // Choca con raqueta izq
                            juego.arkanoPi.pelota.yv=1;
                                                      juego.arkanoPi.pelota.xv=1; // Baja drcha
                         }else if( (x+1) == ((juego.arkanoPi.raqueta_pong.x)+1) ){ // Choca con raqueta centro
                                                juego.arkanoPi.pelota.yv=1;
                                                juego.arkanoPi.pelota.xv=0; // Baja recto
                                             }else if( (x+1) == ((juego.arkanoPi.raqueta_pong.x)+2) ){ // Choca con raqueta drcha
                                                juego.arkanoPi.pelota.yv=1;
                                                juego.arkanoPi.pelota.xv=1; // Baja drcha
                                             }else{
                                             jugadorB++;
                                             if(jugadorA==7 || jugadorB==7){
                                                piLock (FINAL_KEY);
                                                   final = -1;
                                                   piUnlock (FINAL_KEY);
                                             }else if (jugadorA<8 && jugadorB<8){
                                                piLock (FLAGS_KEY);
                                                   flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                   piUnlock (FLAGS_KEY);
                                             }
                                             }
                      }else{
                         if( (x-1)==(juego.arkanoPi.raqueta_pong.x) ){
                            juego.arkanoPi.pelota.yv=1;
                                                   juego.arkanoPi.pelota.xv=-1; // Baja izq
                         }else if( (x-1) == ((juego.arkanoPi.raqueta_pong.x)+1) ){ // Choca con raqueta centro
                                                juego.arkanoPi.pelota.yv=1;
                                                juego.arkanoPi.pelota.xv=0; // Baja recto
                                             }else if( (x-1) == ((juego.arkanoPi.raqueta_pong.x)+2) ){ // Choca con raqueta drcha
                                                juego.arkanoPi.pelota.yv=1;
                                                juego.arkanoPi.pelota.xv=1; // Baja drcha
                                             }else{
                                             jugadorB++;
                                             if(jugadorA==7 || jugadorB==7){
                                                piLock (FINAL_KEY);
                                                   final = -1;
                                                   piUnlock (FINAL_KEY);
```

```
                                                }else if (jugadorA<8 && jugadorB<8){
                                                       piLock (FLAGS_KEY);
                                                              flags |= FLAG_MARCADOR; // Activamos flag Tecla
                                                              piUnlock (FLAGS_KEY);
                                                }
                                                }
                                    }
                                }

                            }
                            // LÃ³gica de movimiento
                               (juego.arkanoPi.pelota.x)=(juego.arkanoPi.pelota.x)+(juego.arkanoPi.pelota.xv);
                               (juego.arkanoPi.pelota.y)=(juego.arkanoPi.pelota.y)+(juego.arkanoPi.pelota.yv);

                               ActualizaPantallaPong((tipo_arkanoPi*) (&(juego.arkanoPi)));
                               piLock (STD_IO_BUFFER_KEY);
                               PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
                               piUnlock (STD_IO_BUFFER_KEY);
                               if(final != -1){
                                      tmr_startms(tmr_pelota, MOV_PELOTA);
                               }

    }
}
void Pausar (fsm_t* fsm){
       piLock (FLAGS_KEY);
       flags &= ~FLAG_STOP; // Bajamos flag
       flagsStop = flags; //Guardamos valor actual de flags
       piUnlock (FLAGS_KEY);
       juego.estado = WAIT_STOP;
       tmr_stop(tmr_pelota); //Paramos temporizadores
       tmr_stop(tmr_joy);
       tmr_stop(tmr_disparo);
}

void Reanudar (fsm_t* fsm){
       piLock (FLAGS_KEY);
       flags &= ~FLAG_STOP; // Bajamos flag
       flags = flagsStop; //Devolvemos el valor guardado a las flags
       piUnlock (FLAGS_KEY);
       juego.estado = WAIT_PUSH;
       tmr_pelota = tmr_new (timer_isr_2); //Inicializamos los temporizadores
       tmr_startms(tmr_pelota, MOV_PELOTA);
```

```
	tmr_joy = tmr_new (timer_isr_3);
	tmr_disparo = tmr_new (timer_isr_4);
	tmr_startms(tmr_disparo, DISPARO);
	tmr_startms(tmr_joy, MOV_JOY);
}

void ActualizaMarcador (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_MARCADOR; // Bajamos flag
	piUnlock (FLAGS_KEY);
	PintaMarcador(jugadorA, jugadorB, (tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_arkanoPi*) (&(juego.arkanoPi)));
	piLock (STD_IO_BUFFER_KEY);
	PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
	piUnlock (STD_IO_BUFFER_KEY);
	juego.estado = WAIT_PONG; // Cambiamos estado del juego
}

// void FinalJuego (void): funciÃ³n encargada de mostrar en la ventana de
// terminal los mensajes necesarios para informar acerca del resultado del juego.
void FinalJuego (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_FINAL_JUEGO; // Bajamos flag
	piUnlock (FLAGS_KEY);
	if(pong ==0){//Para Arkanopi, pinta OK o KO
		if(CalculaLadrillosRestantes((tipo_pantalla*) (&(juego.arkanoPi.ladrillos)))==0){
			PintaOK((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
			piLock (STD_IO_BUFFER_KEY);
			PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
			piUnlock (STD_IO_BUFFER_KEY);
		}else if(CalculaLadrillosRestantes((tipo_pantalla*) (&(juego.arkanoPi.ladrillos)))>0){
			PintaKO((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
			piLock (STD_IO_BUFFER_KEY);
			PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
			piUnlock (STD_IO_BUFFER_KEY);
		}
	}
	if(pong ==1){
		PintaMarcador(jugadorA, jugadorB, (tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_arkanoPi*) (&(juego.arkanoPi)));
		piLock (STD_IO_BUFFER_KEY);
		PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
		piUnlock (STD_IO_BUFFER_KEY);
	}
	pong=0;
```

```
	juego.estado = WAIT_END; // Cambiamos estado del juego
}

//void ReseteaJuego (void): funciÃ³n encargada de llevar a cabo la
// reinicializaciÃ³n de cuantas variables o estructuras resulten
// necesarias para dar comienzo a una nueva partida.
void ReseteaJuego (fsm_t* fsm) {
	piLock (FLAGS_KEY);
	flags &= ~FLAG_TECLA; // Bajamos flag
	piUnlock (FLAGS_KEY);
	// Reseteamos el juego
	jugadorA=0;
	jugadorB=0;
	juego.estado = WAIT_START; // Cambiamos de estado del juego
	PintaAoP(pong, (tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_arkanoPi*) (&(juego.arkanoPi)));
	piLock (STD_IO_BUFFER_KEY);
	PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
	piUnlock (STD_IO_BUFFER_KEY);

}
//------------------------------------------------------
// FUNCIONES DE INICIALIZACION
//------------------------------------------------------
// int systemSetup (void): procedimiento de configuracion del sistema.
// RealizarÃ¡, entra otras, todas las operaciones necesarias para:
// configurar el uso de posibles librerÃ-as (e.g. Wiring Pi),
// configurar las interrupciones externas asociadas a los pines GPIO,
// configurar las interrupciones periÃ³dicas y sus correspondientes temporizadores,
// crear, si fuese necesario, los threads adicionales que pueda requerir el sistema
int systemSetup (void) {
	int x = 0;
	piLock (STD_IO_BUFFER_KEY);
	// sets up the wiringPi library
	if (wiringPiSetupGpio () < 0) {
		printf ("Unable to setup wiringPi\n");
		piUnlock (STD_IO_BUFFER_KEY);
		return -1;
    }
	if (wiringPiSPISetup (SPI_ADC_CH, SPI_ADC_FREQ) < 0) { //Conexion del canal 0
		//(GPIO 08 en numeracion BCM) a 1 MHz
		printf ("No se pudo inicializar el dispositivo SPI (CH 0)") ;
		exit (1);
		return -2;
```

```
	}
	// Lanzamos thread para exploracion del teclado convencional del PC
	x = piThreadCreate (thread_cambia_estado);
	if (x != 0) {
		printf ("it didn't start!!!\n");
		piUnlock (STD_IO_BUFFER_KEY);
		return -1;
    }
	piUnlock (STD_IO_BUFFER_KEY);
	return 1;
}

// int fsm_setup (void): procedimiento de configuraciÃ³n de la mÃ¡quina de estados
void fsm_setup(fsm_t* inicial_fsm) {
	piLock (FLAGS_KEY);
	flags = 0;
	piUnlock (FLAGS_KEY);

	InicializaJuego(inicial_fsm);

	piLock (STD_IO_BUFFER_KEY);
	printf("\nBIENVENIDO A ARKANOPI\n");
	piUnlock (STD_IO_BUFFER_KEY);
}

// Thread de exploraciÃ³n del teclado del PC
PI_THREAD (thread_cambia_estado) {
	//int teclaPulsada;
	while(1) {
		delay(100); // Wiring Pi function: pauses program execution for at least 10 ms

		//Empieza el juego
		if(juego.estado == WAIT_START && (digitalRead (GPIO_IZQ) == HIGH || digitalRead (GPIO_DRCHA) == HIGH)){
			piLock (FLAGS_KEY);
			flags |= FLAG_TECLA; // Activamos flag Tecla
			piUnlock (FLAGS_KEY);
			printf("2");
		}
		//Cambiamos de juego a Arkanopi
		if(juego.estado == WAIT_PONG && (digitalRead (GPIO_IZQ) == HIGH || digitalRead (GPIO_DRCHA) == HIGH)){
			piLock (FLAGS_KEY);
			flags |= FLAG_TECLA; // Activamos flag Tecla
			piUnlock (FLAGS_KEY);
```

```
                        printf("3");
                }
                //Cambiamos de juego a Pong
                if(pulsoPong == 0 && juego.estado == WAIT_START && digitalRead (GPIO_IZQ_PONG) == HIGH && digitalRead (GPIO_DRCHA_PONG) ==
HIGH){
                        piLock (FLAGS_KEY);
                        flags |= FLAG_PONG; // Activamos flag Tecla
                        piUnlock (FLAGS_KEY);
                        pulsoPong=1;
                        printf("4");
                } //Cambiamos de juego a Arkanopi
                if(pulsoPong == 0 && juego.estado == WAIT_PONG  && digitalRead (GPIO_IZQ_PONG) == HIGH && digitalRead (GPIO_DRCHA_PONG) ==
HIGH){
                        piLock (FLAGS_KEY);
                        flags |= FLAG_PONG; // Activamos flag Tecla
                        piUnlock (FLAGS_KEY);
                        pulsoPong=1;
                        printf("5");
                }
                //Pausamos el juego
                if(pulso==0 && juego.estado == WAIT_PUSH && digitalRead (GPIO_IZQ) == HIGH && digitalRead (GPIO_DRCHA) == HIGH){
                        piLock (FLAGS_KEY);
                        flags |= FLAG_STOP; // Activamos flag stop
                        piUnlock (FLAGS_KEY);
                        pulso=1;
                }
                //Reanudamos el juego
                if(pulso==0 && juego.estado == WAIT_STOP && digitalRead (GPIO_IZQ) == HIGH && digitalRead (GPIO_DRCHA) == HIGH){
                        piLock (FLAGS_KEY);
                        flags |= FLAG_STOP; // Activamos flag stop
                        piUnlock (FLAGS_KEY);
                        pulso=1;

                }//Seguro para que solo haya una pulsación
                if(digitalRead (GPIO_IZQ) == LOW && digitalRead (GPIO_DRCHA) == LOW){
                        pulso=0;
                }//Seguro para que solo haya una pulsación
                if(digitalRead (GPIO_IZQ_PONG) == LOW && digitalRead (GPIO_DRCHA_PONG) == LOW){
                        pulsoPong=0;
                }

                int aux = CalculaLadrillosRestantes((tipo_pantalla*) (&(juego.arkanoPi.ladrillos)));
                if(aux == 0 || final == -1){ // Si hemos ganado o perdido, finaliza el juego
```

```
                        piLock (FLAGS_KEY);
                        flags |= FLAG_FINAL_JUEGO; // Activamos flag Final Juego
                        piUnlock (FLAGS_KEY);


                        piLock (FINAL_KEY);
                        final = 0;
                        piUnlock (FINAL_KEY);
                }
                if(juego.estado == WAIT_END && (digitalRead (GPIO_IZQ) == HIGH || digitalRead (GPIO_DRCHA) == HIGH)){
                                        piLock (FLAGS_KEY);
                                        flags |= FLAG_TECLA; // Activamos flag Tecla
                                        piUnlock (FLAGS_KEY);
                }
        }
}

void boton_izq (void){
        piLock (FLAGS_KEY);
        flags |= FLAG_RAQUETA_IZQUIERDA; // Activamos flag Raqueta Izquierda
        piUnlock (FLAGS_KEY);
}

void boton_drcha (void){
        piLock (FLAGS_KEY);
        flags |= FLAG_RAQUETA_DERECHA; // Activamos flag Raqueta Izquierda
        piUnlock (FLAGS_KEY);
}
void boton_izq_pong (void){
        if(pong==1){
                piLock (FLAGS_KEY);
                flags |= FLAG_RAQUETA_IZQUIERDA_PONG; // Activamos flag Raqueta Izquierda
                piUnlock (FLAGS_KEY);
        }
        if(pong==0 && juego.estado==WAIT_PUSH){ //Inicia el disparo
                int x=(juego.arkanoPi.raqueta.x)+1;
                ReseteaDisparo((tipo_pelota*)(&(juego.arkanoPi.disparo)), x );
                tmr_disparo = tmr_new (timer_isr_4);
                tmr_startms(tmr_disparo, DISPARO);
        }
}

void boton_drcha_pong (void){
```

```
	if(pong==1){
		piLock (FLAGS_KEY);
		flags |= FLAG_RAQUETA_DERECHA_PONG; // Activamos flag Raqueta Izquierda
		piUnlock (FLAGS_KEY);
	}
	if(pong==0 && juego.estado==WAIT_PUSH){
		final=-1; //Reinicio
	}
}

int main ()
{	// Configuracion e inicializacion del sistema
	wiringPiSetupGpio(); // ConfiguraciÃ³n de Raspberry: Activamos pines que utilizaremos como salidas
	int i, j;
	for (i=0; i<4; i++){
		pinMode(gpio_col[i], OUTPUT);
	}
	for (j=0; j<7; j++){
		pinMode(gpio_row[i], OUTPUT);
	}
	pinMode(GPIO_IZQ, INPUT);
	pinMode(GPIO_DRCHA, INPUT);
	wiringPiISR (GPIO_IZQ, INT_EDGE_RISING, boton_izq);
	wiringPiISR (GPIO_DRCHA, INT_EDGE_RISING, boton_drcha);
	pinMode(GPIO_IZQ_PONG, INPUT);
	pinMode(GPIO_DRCHA_PONG, INPUT);
	wiringPiISR (GPIO_IZQ_PONG, INT_EDGE_RISING, boton_izq_pong);
	wiringPiISR (GPIO_DRCHA_PONG, INT_EDGE_RISING, boton_drcha_pong);
	count = -1; // Inicializamos variable para refrescar columnas
	tmr_col = tmr_new (timer_isr); // Indicamos el tiempo de refresco por columna
	tmr_startms(tmr_col, REFRESCO);
	tmr_pelota = tmr_new (timer_isr_2);
	tmr_joy = tmr_new (timer_isr_3);
	unsigned int next;

		// Maquina de estados: lista de transiciones
		// {EstadoOrigen,FunciÃ³nDeEntrada,EstadoDestino,FunciÃ³nDeSalida}
		fsm_trans_t mueve_tabla[] = {
				{ START,   CompruebaTeclaPulsada,  PUSH, InicializaJuego },
				{ PUSH, CompruebaTimeout, PUSH,   MovimientoPelota },
				{ PUSH, CompruebaTeclaPelota, PUSH,   MovimientoPelota },
				{ PUSH, CompruebaTeclaRaquetaIzquierda, PUSH,   MueveRaquetaIzquierda },
				{ PUSH, CompruebaTeclaRaquetaDerecha, PUSH,   MueveRaquetaDerecha },
```

```
                        { PUSH, CompruebaJoystick, PUSH,   MueveJoystick },
                        { PUSH, CompruebaFinalJuego, END,   FinalJuego },
                        { PUSH, CompruebaStop, STOP, Pausar},
                        { START,  CompruebaPong,  PONG, CambiaJuego },
                        { PONG,   CompruebaPong,  START, CambiaJuego },
                        { PONG,   CompruebaTeclaPulsada,  PUSH, InicializaPong },
                        { PUSH, CompruebaTeclaRaquetaIzquierdaPong, PUSH,   MueveRaquetaIzquierdaPong },
                        { PUSH, CompruebaTeclaRaquetaDerechaPong, PUSH,   MueveRaquetaDerechaPong },
                        { STOP, CompruebaStop, PUSH, Reanudar},
                        { PUSH, CompruebaMarcador, PONG,   ActualizaMarcador },
                        { END, CompruebaTeclaPulsada, START,   ReseteaJuego },
                        { -1, NULL, -1, NULL },
        };
        // Declaramos estado inicial de la mÃ¡quina de estados
        fsm_t* inicial_fsm = fsm_new (START, mueve_tabla, NULL);
        // Configuracion e inicializacion del sistema
        systemSetup();
        fsm_setup(inicial_fsm);
        next = millis();
        juego.estado = WAIT_START; // Declaramos el estado inicial del juego
        PintaMensajeInicialPantalla((tipo_pantalla*)(&(juego.arkanoPi.pantalla)), (tipo_pantalla*) (&(juego.arkanoPi.pantalla)));
        piLock (STD_IO_BUFFER_KEY);
        PintaPantallaPorTerminal((tipo_pantalla*)(&(juego.arkanoPi.pantalla)));
        piUnlock (STD_IO_BUFFER_KEY);
        while (1) {
               fsm_fire (inicial_fsm); // Ejecutamos mÃ¡quina de estados
               next += CLK_MS;
               delay_until (next);
        }

        fsm_destroy (inicial_fsm);
}
```

```
/*****************************************************************************************************************************************/

/************************************************  File Name : arkanoPi_1.h  ************************************************************/

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

#ifndef _ARKANOPI_H_
#define _ARKANOPI_H_

#include <time.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/time.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
#include "kbhit.h" // para poder detectar teclas pulsadas sin bloqueo y leer las teclas pulsadas
#include "arkanoPiLib.h"

#include "fsm.h"

#include "tmr.h"

#define CLK_MS 2 // PERIODO DE ACTUALIZACION DE LA MAQUINA ESTADOS

typedef enum { // Estados del juego
       WAIT_START=0,
       WAIT_PUSH=1,
       WAIT_END=2,
       WAIT_STOP=3,
       WAIT_PONG=4
       } tipo_estados_juego;

typedef struct { // Struct del juego
       tipo_arkanoPi arkanoPi;
       tipo_estados_juego estado;
       char teclaPulsada;
} tipo_juego;
```

```
// Declaración de FLAGS
#define FLAG_TECLA                                  0x01
#define FLAG_PELOTA                             0x02 //Ya no se usa
#define FLAG_RAQUETA_DERECHA                    0x04
#define FLAG_RAQUETA_IZQUIERDA                  0x08
#define FLAG_FINAL_JUEGO                        0x10 // Flag para indicar final del juego
#define FLAG_TIMEOUT                            0x20 // Mover pelota
#define FLAG_JOY                                    0x40 //Joystick
#define FLAG_STOP                                   0x80 //Pausa
#define FLAG_PONG                                   0x100 //Cambia de juego
#define FLAG_RAQUETA_DERECHA_PONG       0x200
#define FLAG_RAQUETA_IZQUIERDA_PONG             0x400
#define FLAG_MARCADOR                               0x800
// A 'key' which we can lock and unlock - values are 0 through 3
//     This is interpreted internally as a pthread_mutex by wiringPi
//     which is hiding some of that to make life simple.
#define       FLAGS_KEY     1
#define       STD_IO_BUFFER_KEY    2
#define       FINAL_KEY     0

//-------------------------------------------------------
// FUNCIONES DE TEMPORIZADORES, INICIO Y PULSADORES
//-------------------------------------------------------

static void timer_isr (union sigval arg);
static void timer_isr_2 (union sigval arg);
static void timer_isr_3 (union sigval arg);
static void timer_isr_4 (union sigval arg);
void delay_until (unsigned int next);
int systemSetup (void);
void fsm_setup(fsm_t* inicial_fsm);
void boton_izq (void);
void boton_drcha (void);
void boton_izq_pong (void);
void boton_drcha_pong (void);

//-------------------------------------------------------
// FUNCIONES DE ENTRADA O DE TRANSICION DE LA MAQUINA DE ESTADOS
//-------------------------------------------------------
// Prototipos de funciones de entrada
int CompruebaTeclaPulsada (fsm_t* this);
int CompruebaTeclaPelota (fsm_t* this);
int CompruebaTeclaRaquetaDerecha (fsm_t* this);
```

```
int CompruebaTeclaRaquetaIzquierda (fsm_t* this);
int CompruebaFinalJuego (fsm_t* this);
int CompruebaJoystick (fsm_t* this);
int CompruebaStop (fsm_t* this);
int CompruebaPong (fsm_t* this);
int CompruebaTeclaRaquetaIzquierdaPong (fsm_t* this);
int CompruebaTeclaRaquetaDerechaPong (fsm_t* this);
int CompruebaTimeout (fsm_t* this);
int CompruebaMarcador (fsm_t* this);

//------------------------------------------------------
// FUNCIONES DE SALIDA O DE ACCION DE LA MAQUINA DE ESTADOS
//------------------------------------------------------

void InicializaJuego (fsm_t*);
void MueveRaquetaIzquierda (fsm_t*);
void MueveRaquetaDerecha (fsm_t*);
void MovimientoPelota (fsm_t*);
void FinalJuego (fsm_t*);
void ReseteaJuego (fsm_t*);
void InicializaPong (fsm_t* fsm);
void CambiaJuego (fsm_t* fsm);
void MueveRaquetaIzquierdaPong (fsm_t* fsm);
void MueveRaquetaDerechaPong (fsm_t* fsm);
void MueveJoystick (fsm_t* fsm);
void Pausar (fsm_t* fsm);
void Reanudar (fsm_t* fsm);
void ActualizaMarcador (fsm_t* fsm);

//------------------------------------------------------
// FUNCIONES DE INICIALIZACION
//------------------------------------------------------
//int systemSetup (void);
//void fsm_setup (fsm_t* inicial_fsm);
PI_THREAD (thread_cambia_estado);
#endif /* ARKANOPI_H_ */
```

```
/*****************************************************************************************************************************************/

/************************************************  File Name : arkanoPiLib.c  ************************************************************/

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

#include "arkanoPiLib.h"

int ladrillos_basico[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
              {1,1,0,0,0,0,0},
};
int cero[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {0,1,1,1,1,1,0},
              {1,0,0,0,0,0,1},
              {1,0,0,0,0,0,1},
              {1,0,0,0,0,0,1},
              {0,1,1,1,1,1,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
};
int uno[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {0,0,1,0,0,0,0},
              {0,1,0,0,0,0,0},
              {1,1,1,1,1,1,1},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
```

```
};
int dos[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {0,1,1,0,0,0,1},
              {1,0,0,0,0,1,1},
              {1,0,0,0,1,0,1},
              {1,0,0,1,0,0,1},
              {0,1,1,0,0,0,1},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
};
int tres[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {0,1,0,0,0,1,0},
              {1,0,0,0,0,0,1},
              {1,0,0,1,0,0,1},
              {1,0,0,1,0,0,1},
              {0,1,1,0,1,1,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
};
int cuatro[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {0,0,0,1,1,0,0},
              {0,0,1,0,1,0,0},
              {0,1,0,0,1,0,0},
              {1,1,1,1,1,1,1},
              {0,0,0,0,1,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0},
};
int cinco[MATRIZ_ANCHO][MATRIZ_ALTO] = {
              {1,1,1,1,0,1,0},
              {1,0,0,1,0,0,1},
              {1,0,0,1,0,0,1},
              {1,0,0,1,0,0,1},
              {1,0,0,0,1,1,0},
```

```
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
};
int seis[MATRIZ_ANCHO][MATRIZ_ALTO] = {
		{0,1,1,1,1,1,0},
		{1,0,0,1,0,0,1},
		{1,0,0,1,0,0,1},
		{1,0,0,1,0,0,1},
		{0,1,0,0,1,1,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
};
int siete[MATRIZ_ANCHO][MATRIZ_ALTO] = {
		{1,0,0,0,0,0,0},
		{1,0,0,0,1,1,1},
		{1,0,0,1,0,0,0},
		{1,0,1,0,0,0,0},
		{1,1,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
		{0,0,0,0,0,0,0},
};


//-------------------------------------------------------
// FUNCIONES DE INICIALIZACION / RESET
//-------------------------------------------------------

void ReseteaMatriz(tipo_pantalla *p_pantalla) {
	int i, j = 0;

	for(i=0;i<MATRIZ_ANCHO;i++) {
		for(j=0;j<MATRIZ_ALTO;j++) {
			p_pantalla->matriz[i][j] = 0;
		}
```

```
	}
}

void ReseteaLadrillos(tipo_pantalla *p_ladrillos) {
	int i, j = 0;

	for(i=0;i<MATRIZ_ANCHO;i++) {
		for(j=0;j<MATRIZ_ALTO;j++) {
			p_ladrillos->matriz[i][j] = ladrillos_basico[i][j];
		}
	}
}

void ReseteaPelota(tipo_pelota *p_pelota) {
	// Pelota inicialmente en el centro de la pantalla
	p_pelota->x = MATRIZ_ANCHO/2 - 1;
	p_pelota->y = MATRIZ_ALTO/2 - 1;

	// Trayectoria inicial
	p_pelota->yv = 1;
	p_pelota->xv = 0;
}

void ReseteaDisparo(tipo_pelota *p_pelota, int posicionX) {
	// Pelota inicialmente en el centro de la pantalla
	p_pelota->x = posicionX;
	p_pelota->y = 6;

	// Trayectoria inicial
	p_pelota->yv = -1;
	p_pelota->xv = 0;
}

void ReseteaRaqueta(tipo_raqueta *p_raqueta) {
	// Raqueta inicialmente en el centro de la pantalla
	p_raqueta->x = MATRIZ_ANCHO/2 - p_raqueta->ancho/2;
	p_raqueta->y = MATRIZ_ALTO - 1;
	p_raqueta->ancho = RAQUETA_ANCHO;
	p_raqueta->alto = RAQUETA_ALTO;
}

void ReseteaRaquetaPong(tipo_raqueta *p_raqueta) {
	// Raqueta inicialmente en el centro de la pantalla
```

```
	p_raqueta->x = MATRIZ_ANCHO/2 - p_raqueta->ancho/2;
	p_raqueta->y = 0;
	p_raqueta->ancho = RAQUETA_ANCHO;
	p_raqueta->alto = RAQUETA_ALTO;
}

//------------------------------------------------------
// FUNCIONES DE VISUALIZACION (ACTUALIZACION DEL OBJETO PANTALLA QUE LUEGO USARA EL DISPLAY)
//------------------------------------------------------

// void PintaMensajeInicialPantalla (...): metodo encargado de aprovechar
// el display para presentar un mensaje de bienvenida al usuario
void PintaMensajeInicialPantalla (tipo_pantalla *p_pantalla, tipo_pantalla *p_pantalla_inicial) {
	// A completar por el alumno...
	int mensaje_inicial[MATRIZ_ANCHO][MATRIZ_ALTO] = { // Declaramos mensaje inicial
			{0,1,1,1,1,1,1},
			{1,0,0,1,0,0,0},
			{1,0,0,1,0,0,0},
			{1,0,0,1,0,0,0},
			{0,1,1,1,1,1,1},
			{0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0},
	};
	int i, j = 0;

	for(i=0;i<MATRIZ_ANCHO;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
		for(j=0;j<MATRIZ_ALTO;j++) {
			p_pantalla->matriz[i][j] = mensaje_inicial [i][j];
		}
		printf("\n");
	}
}

void PintaAoP(int pong, tipo_pantalla *p_pantalla, tipo_arkanoPi *p_arkanoPi){
	int mensaje_inicial[MATRIZ_ANCHO][MATRIZ_ALTO] = { // Declaramos mensaje inicial
				{0,1,1,1,1,1,1},
				{1,0,0,1,0,0,0},
				{1,0,0,1,0,0,0},
				{1,0,0,1,0,0,0},
				{0,1,1,1,1,1,1},
```

```
                        {1,1,1,1,1,1,1},
                        {1,0,0,1,0,0,0},
                        {1,0,0,1,0,0,0},
                        {1,0,0,1,0,0,0},
                        {0,1,1,0,0,0,0},
                };
                ReseteaMatriz((tipo_pantalla*)(&(p_arkanoPi->pantalla)));
                int i, j = 0;
                if(pong==0){ //Pinta A
                        for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                for(j=0;j<MATRIZ_ALTO;j++) {
                                        p_pantalla->matriz[i][j] = mensaje_inicial [i][j];
                                }
                                printf("\n");
                        }
                }
                if(pong==1){ //Pinta P
                        for(i=5;i<10;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                for(j=0;j<MATRIZ_ALTO;j++) {
                                        p_pantalla->matriz[i][j] = mensaje_inicial [i][j];
                                }
                                printf("\n");
                                }
                        }
}
void PintaOK (tipo_pantalla *p_pantalla) {
        int mensaje_inicial[MATRIZ_ANCHO][MATRIZ_ALTO] = { // Declaramos mensaje inicial
                        {1,1,1,1,1,1,1},
                        {1,0,0,0,0,0,1},
                        {1,0,0,0,0,0,1},
                        {1,0,0,0,0,0,1},
                        {1,1,1,1,1,1,1},
                        {1,1,1,1,1,1,1},
                        {0,0,0,1,0,0,0},
                        {0,0,1,0,1,0,0},
                        {0,1,0,0,0,1,0},
                        {1,0,0,0,0,0,1},
        };
        int i, j = 0;

        for(i=0;i<MATRIZ_ANCHO;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                for(j=0;j<MATRIZ_ALTO;j++) {
                        p_pantalla->matriz[i][j] = mensaje_inicial [i][j];
```

```
		}
		printf("\n");
	}
}
void PintaKO (tipo_pantalla *p_pantalla) {
	int mensaje_inicial[MATRIZ_ANCHO][MATRIZ_ALTO] = { // Declaramos mensaje inicial
			{1,1,1,1,1,1,1},
			{0,0,0,1,0,0,0},
			{0,0,1,0,1,0,0},
			{0,1,0,0,0,1,0},
			{1,0,0,0,0,0,1},
			{1,1,1,1,1,1,1},
			{1,0,0,0,0,0,1},
			{1,0,0,0,0,0,1},
			{1,0,0,0,0,0,1},
			{1,1,1,1,1,1,1},
	};
	int i, j = 0;

	for(i=0;i<MATRIZ_ANCHO;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
		for(j=0;j<MATRIZ_ALTO;j++) {
			p_pantalla->matriz[i][j] = mensaje_inicial [i][j];
		}
		printf("\n");
	}
}

void PintaMarcador(int jugadorA, int jugadorB, tipo_pantalla *p_pantalla, tipo_arkanoPi *p_arkanoPi){
		ReseteaMatriz((tipo_pantalla*)(&(p_arkanoPi->pantalla)));
		int i, j = 0;
		//Pinta para jugador A y B sus marcadores dependiendo de los valores que se le pasen
		if(jugadorA==0){
			for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
					for(j=0;j<MATRIZ_ALTO;j++) {
						p_pantalla->matriz[i][j] = cero[i][j];
					}
					printf("\n");
			}
		}
		if(jugadorA==1){
					for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
							for(j=0;j<MATRIZ_ALTO;j++) {
								p_pantalla->matriz[i][j] = uno[i][j];
```

```
                                        }
                                        printf("\n");
                                }
                        }
        if(jugadorA==2){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i][j] = dos[i][j];
                                        }
                                        printf("\n");
                                }
                        }
        if(jugadorA==3){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i][j] = tres[i][j];
                                        }
                                        printf("\n");
                                }
                        }
        if(jugadorA==4){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i][j] = cuatro[i][j];
                                        }
                                        printf("\n");
                                }
                        }
        if(jugadorA==5){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i][j] = cinco[i][j];
                                        }
                                        printf("\n");
                                }
                        }
        if(jugadorA==6){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i][j] = seis[i][j];
                                        }
                                        printf("\n");
                                }
```

```
                }
        if(jugadorA==7){
                        for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                for(j=0;j<MATRIZ_ALTO;j++) {
                                        p_pantalla->matriz[i][j] = siete[i][j];
                                }
                                printf("\n");
                        }
                }
        if(jugadorB==0){
                        for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                for(j=0;j<MATRIZ_ALTO;j++) {
                                        p_pantalla->matriz[i+5][j] = cero[i][j];
                                }
                                printf("\n");
                        }
                }
                if(jugadorB==1){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i+5][j] = uno[i][j];
                                        }
                                        printf("\n");
                                }
                        }
                if(jugadorB==2){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i+5][j] = dos[i][j];
                                        }
                                        printf("\n");
                                }
                        }
                if(jugadorB==3){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                        for(j=0;j<MATRIZ_ALTO;j++) {
                                                p_pantalla->matriz[i+5][j] = tres[i][j];
                                        }
                                        printf("\n");
                                }
                        }
                if(jugadorB==4){
                                for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
```

```
                                                for(j=0;j<MATRIZ_ALTO;j++) {
                                                      p_pantalla->matriz[i+5][j] = cuatro[i][j];
                                                }
                                                printf("\n");
                                          }
                                    }
                              if(jugadorB==5){
                                          for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                                for(j=0;j<MATRIZ_ALTO;j++) {
                                                      p_pantalla->matriz[i+5][j] = cinco[i][j];
                                                }
                                                printf("\n");
                                          }
                                    }
                              if(jugadorB==6){
                                          for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                                for(j=0;j<MATRIZ_ALTO;j++) {
                                                      p_pantalla->matriz[i+5][j] = seis[i][j];
                                                }
                                                printf("\n");
                                          }
                                    }
                              if(jugadorB==7){
                                          for(i=0;i<5;i++) { // Pintamos mensaje inicial por pantalla (PENDIENTE DE CAMBIAR)
                                                for(j=0;j<MATRIZ_ALTO;j++) {
                                                      p_pantalla->matriz[i+5][j] = siete[i][j];
                                                }
                                                printf("\n");
                                          }
                                    }


}


// void PintaPantallaPorTerminal (...): metodo encargado de mostrar
// el contenido o la ocupaciÃ³n de la matriz de leds en la ventana de
// terminal o consola. Este mÃ©todo sera fundamental para facilitar
// la labor de depuraciÃ³n de errores (por ejemplo, en la programaciÃ³n
// de los diferentes movimientos tanto de la raqueta como de la pelota).
void PintaPantallaPorTerminal  (tipo_pantalla *p_pantalla) {
      // A completar por el alumno...
      /*int i, j = 0;
```

```
	printf("\n");
	printf("\n");
	printf("[PANTALLA]");
	printf("\n");
	for(i=0;i<MATRIZ_ALTO;i++) { // Pintamos pantalla
		for(j=0;j<MATRIZ_ANCHO;j++) {
			printf("%d", p_pantalla->matriz[j][i]);
		}
		printf("\n");
	}*/
}



// void PintaLadrillos(...): funcion encargada de â€œpintarâ€ los ladrillos
// en sus correspondientes posiciones dentro del Ã¡rea de juego
void PintaLadrillos(tipo_pantalla *p_ladrillos, tipo_pantalla *p_pantalla) {
	int i, j = 0;

	for(i=0;i<MATRIZ_ANCHO;i++) {
		for(j=0;j<MATRIZ_ALTO;j++) {
			p_pantalla->matriz[i][j] = p_ladrillos->matriz[i][j];
		}
    }
}

// void PintaRaqueta(...): funcion encargada de â€œpintarâ€ la raqueta
// en su posicion correspondiente dentro del Ã¡rea de juego
void PintaRaqueta(tipo_raqueta *p_raqueta, tipo_pantalla *p_pantalla) {
	int i, j = 0;

	for(i=0;i<RAQUETA_ANCHO;i++) {
		for(j=0;j<RAQUETA_ALTO;j++) {
			if (( (p_raqueta->x+i >= 0) && (p_raqueta->x+i < MATRIZ_ANCHO) ) &&
				( (p_raqueta->y+j >= 0) && (p_raqueta->y+j < MATRIZ_ALTO) ))
				p_pantalla->matriz[p_raqueta->x+i][p_raqueta->y+j] = 1;
		}
	}
}



// void PintaPelota(...): funcion encargada de â€œpintarâ€ la pelota
```

```
// en su posicion correspondiente dentro del Ã¡rea de juego

void PintaPelota(tipo_pelota *p_pelota, tipo_pantalla *p_pantalla) {
	if( (p_pelota->x >= 0) && (p_pelota->x < MATRIZ_ANCHO) ) {
		if( (p_pelota->y >= 0) && (p_pelota->y < MATRIZ_ALTO) ) {
			p_pantalla->matriz[p_pelota->x][p_pelota->y] = 1;
		}
		else {
			printf("\n\nPROBLEMAS!!!! posicion y=%d de la pelota INVALIDA!!!\n\n", p_pelota->y);
			fflush(stdout);
		}
	}
	else {
		printf("\n\nPROBLEMAS!!!! posicion x=%d de la pelota INVALIDA!!!\n\n", p_pelota->x);
		fflush(stdout);
	}
}
// void PintaDisparo(...): funcion encargada de â€œpintarâ€ el disparo
// en su posicion correspondiente dentro del Ã¡rea de juego

void PintaDisparo(tipo_pelota *p_pelota, tipo_pantalla *p_pantalla) {
	if( (p_pelota->x >= 0) && (p_pelota->x < MATRIZ_ANCHO) ) {
		if( (p_pelota->y >= 0) && (p_pelota->y < MATRIZ_ALTO) ) {
			p_pantalla->matriz[p_pelota->x][p_pelota->y] = 1;
		}
		else {
			printf("\n\nPROBLEMAS!!!! posicion y=%d de la pelota INVALIDA!!!\n\n", p_pelota->y);
			fflush(stdout);
		}
	}
	else {
		//printf("\n\nPROBLEMAS!!!! posicion x=%d de la pelota INVALIDA!!!\n\n", p_pelota->x);
		fflush(stdout);
	}
}

// void ActualizaPantalla(...): metodo cuya ejecucion estara ligada a
// cualquiera de los movimientos de la raqueta o de la pelota y que
// sera el encargado de actualizar convenientemente la estructura de datos
// en memoria que representa el Ã¡rea de juego y su correspondiente estado.

void ActualizaPantalla(tipo_arkanoPi* p_arkanoPi) {
    // Borro toda la pantalla
```

```
       ReseteaMatriz((tipo_pantalla*)(&(p_arkanoPi->pantalla)));

       // Actualizamos los objetos de la pantalla
       PintaLadrillos((tipo_pantalla*)(&(p_arkanoPi->ladrillos)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       PintaPelota((tipo_pelota*)(&(p_arkanoPi->pelota)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       PintaRaqueta((tipo_raqueta*)(&(p_arkanoPi->raqueta)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       PintaDisparo((tipo_pelota*)(&(p_arkanoPi->disparo)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
}

// void ActualizaPantallaPong(...): metodo cuya ejecucion estara ligada a
// cualquiera de los movimientos de la raqueta o de la pelota y que
// sera el encargado de actualizar convenientemente la estructura de datos
// en memoria que representa el Ã¡rea de juego y su correspondiente estado.

void ActualizaPantallaPong(tipo_arkanoPi* p_arkanoPi) {
    // Borro toda la pantalla
       ReseteaMatriz((tipo_pantalla*)(&(p_arkanoPi->pantalla)));

       // Actualizamos los objetos de la pantalla
       PintaPelota((tipo_pelota*)(&(p_arkanoPi->pelota)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       PintaRaqueta((tipo_raqueta*)(&(p_arkanoPi->raqueta_pong)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       PintaRaqueta((tipo_raqueta*)(&(p_arkanoPi->raqueta)), (tipo_pantalla*)(&(p_arkanoPi->pantalla)));
}

// void InicializaArkanoPi(...): metodo encargado de la inicializaciÃ³n
// de toda variable o estructura de datos especificamente ligada al
// desarrollo del juego y su visualizacion.

void InicializaArkanoPi(tipo_arkanoPi *p_arkanoPi) {

       ReseteaMatriz((tipo_pantalla*)(&(p_arkanoPi->pantalla)));
       ReseteaLadrillos((tipo_pantalla*)(&(p_arkanoPi->ladrillos)));
       ReseteaPelota((tipo_pelota*)(&(p_arkanoPi->pelota)));
       ReseteaRaqueta((tipo_raqueta*)(&(p_arkanoPi->raqueta)));

}

// int CalculaLadrillosRestantes(...): funciÃ³n encargada de evaluar
// el estado de ocupacion del area de juego por los ladrillos y
// devolver el numero de estos

int CalculaLadrillosRestantes(tipo_pantalla *p_ladrillos) {
       int num_ladrillos_restantes = 0;
```

```
	int i,j;
	for(i=0;i<LADRILLOS_ANCHO;i++) { // Contamos los ladrillos restantes
			for(j=0;j<LADRILLOS_ALTO;j++) {
				if(p_ladrillos->matriz[i][j]==1){
					num_ladrillos_restantes++;
				}
			}
	    }


	return num_ladrillos_restantes;
}
```

```
/*****************************************************************************************************************************************/

/************************************************  File Name : arkanoPi_1.c  ************************************************************/

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

#ifndef _ARKANOPILIB_H_
#define _ARKANOPILIB_H_

#include <stdio.h>

// CONSTANTES DEL JUEGO
#define MATRIZ_ANCHO       10
#define MATRIZ_ALTO  7
#define LADRILLOS_ANCHO 10
#define LADRILLOS_ALTO     2
#define RAQUETA_ANCHO              3
#define RAQUETA_ALTO               1

typedef struct {
       // Posicion
       int x;
       int y;
       // Forma
       int ancho;
       int alto;
} tipo_raqueta;

typedef struct {
       // Posicion
       int x;
       int y;
       // Trayectoria
       int xv;
       int yv;
} tipo_pelota;

typedef struct {
       // Matriz de ocupación de las distintas posiciones que conforman el display
       // (correspondiente al estado encendido/apagado de cada uno de los leds)
       int matriz[MATRIZ_ANCHO][MATRIZ_ALTO];
} tipo_pantalla;
```

```
typedef struct {
  tipo_pantalla ladrillos; // Notese que, por simplicidad, los ladrillos comparten tipo con la pantalla
  tipo_pantalla pantalla;
  tipo_raqueta raqueta;
  tipo_raqueta raqueta_pong;
  tipo_pelota pelota;
  tipo_pelota disparo;
} tipo_arkanoPi;

extern tipo_pantalla pantalla_inicial;

//------------------------------------------------------
// FUNCIONES DE INICIALIZACION / RESET
//------------------------------------------------------
void ReseteaMatriz(tipo_pantalla *p_pantalla);
void ReseteaLadrillos(tipo_pantalla *p_ladrillos);
void ReseteaPelota(tipo_pelota *p_pelota);
void ReseteaDisparo(tipo_pelota *p_pelota, int posicionX);
void ReseteaRaqueta(tipo_raqueta *p_raqueta);
void ReseteaRaquetaPong(tipo_raqueta *p_raqueta);

//------------------------------------------------------
// FUNCIONES DE VISUALIZACION (ACTUALIZACION DEL OBJETO PANTALLA QUE LUEGO USARA EL DISPLAY)
//------------------------------------------------------
void PintaMensajeInicialPantalla (tipo_pantalla *p_pantalla, tipo_pantalla *p_pantalla_inicial);
void PintaPantallaPorTerminal (tipo_pantalla *p_pantalla);
void PintaLadrillos(tipo_pantalla *p_ladrillos, tipo_pantalla *p_pantalla);
void PintaRaqueta(tipo_raqueta *p_raqueta, tipo_pantalla *p_pantalla);
void PintaPelota(tipo_pelota *p_pelota, tipo_pantalla *p_pantalla);
void PintaDisparo(tipo_pelota *p_pelota, tipo_pantalla *p_pantalla);
void ActualizaPantalla(tipo_arkanoPi* p_arkanoPi);
void ActualizaPantallaPong(tipo_arkanoPi* p_arkanoPi);
void PintaAoP(int pong, tipo_pantalla *p_pantalla, tipo_arkanoPi *p_arkanoPi);
void PintaMarcador(int jugadorA, int jugadorB, tipo_pantalla *p_pantalla, tipo_arkanoPi *p_arkanoPi);
void InicializaArkanoPi(tipo_arkanoPi *p_arkanoPi);
int CalculaLadrillosRestantes(tipo_pantalla *p_ladrillos);
void PintaOK (tipo_pantalla *p_pantalla);
void PintaKO (tipo_pantalla *p_pantalla);

#endif /* _ARKANOPILIB_H_ */
```