APLICACIÓN Y MÓDULOS KERNEL PARA ALTERA CYCLONE V

Nombre y apellidos del autor
Egilearen izen-abizenak
CABRERA MUGICA, ANDREA
ARIN DONOSO, JUAN JOSE

Lugar donde se realiza el trabajo
Lana egin deneko lekua
Facultad de Informática, EHU.

Fecha de inicio
Hasiera data
08/12/2020

ÍNDICE

1 Descripción general

1.1 Descripción

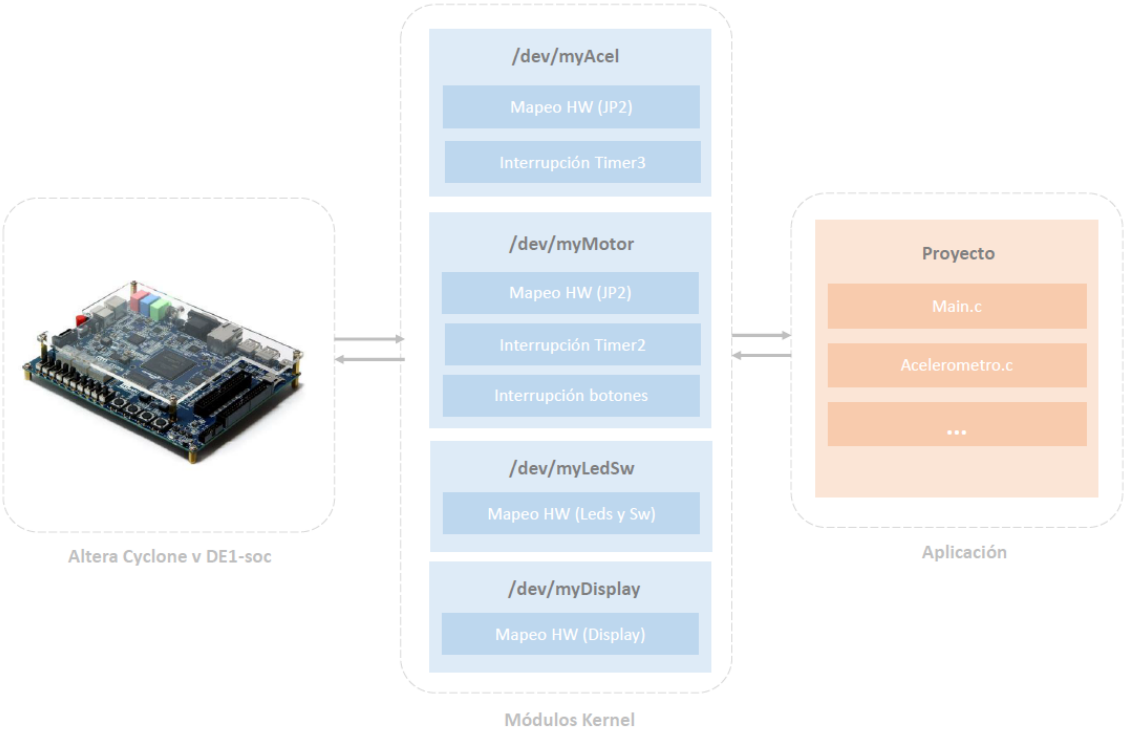
En este proyecto se ha diseñado una aplicación para el sistema operativo Linux instalado en la placa Altera. El objetivo propuesto es gestionar las interrupciones del Kernel y establecer una comunicación entre unos drivers personalizados y una aplicación que pueda acceder a ellos.

Se intentará simular una variedad de gestión de E/S. Mediante los elementos hardware que ofrece la placa Altera, el usuario podrá interactuar con el programa. Para ello, se desarrollará un programa donde se utilizarán los leds, los botones, los interruptores, los displays, dos motores y un acelerómetro.

1.2 Estructura

En la imagen inferior se muestra el diagrama de bloques general del proyecto desarrollado. Por un lado, se ha desarrollado la aplicación y por otro lado se han desarrollado los módulos de *Kernel*.

Debido a que las interrupciones son gestionadas por el sistema operativo y no se pueden gestionar desde la aplicación, es necesario crear un módulo de *Kernel*. Estos módulos son los encargados de gestionar las interrupciones y acceder al *hardware* de la placa Altera. La aplicación se ejecuta en modo usuario, por lo tanto, no puede acceder directamente al hardware de la placa. Para poder acceder a los elementos hardware se han desarrollado diferentes módulos que incluyen el mapeo de las direcciones.



1.3. Modulos de Kernel

Cada módulo de kernel creado tiene su propio descriptor dentro de la unidad de los periféricos. El funcionamiento de cada uno es el siguiente:

1. **DisplayModule:**

   Este módulo simplemente va a crear una unidad dentro de la unidad de periféricos llamada

   ```
   DEVICE_NAME "myDisplay"
   ```

   en la que se le va a conceder el acceso a escritura. En una función llamada

   ```
   display_write
   ```

   La cual se va a encargar de escribir en el dispositivo “myDisplay” los datos que se lean desde la aplicación.

2. **PioModule:**

   Este módulo se va a encargar de escribir en los LEDS superiores de la placa Altera Cyclone. Similar al funcionamiento del módulo anterior, tendremos un dispositivo llamado

   ```
   DEVICE_NAME "myPio"
   ```

   Con una función llamada

   ```
   leds_write
   ```

   en la que simplemente se limitará a escribir los valores que lea de la aplicación y los pasará al dispositivo “myPio”.
   La función asociada del MAIN a la escritura de los leds es el encargado de limitar los valores que se ván a pasar al registro asociado de los leds.

3. **AcelModule:**

   Este módulo contiene dos funciones asociadas al dispositivo en el que se incluirá las mediciones de los pines JP17 y JP19 asociadas a los valores del acelerómetro en los ejes X e Y que los añadirá a la unidad

   ```
   DEVICE_NAME "myAcel"
   ```

   Para poder leer los valores del acelerómetro y escribir en el display de 7 Segmentos su valor, hemos añadido otras dos funciones

   ```
   acel_read ();       //       Lectura
   ```

   La función de lectura es la encargada de poder facilitar a la aplicación los datos del kernel. Para poder hacer las lecturas PWM del sensor, se ha asignado el uso del **timer 3 del HPS.**

   Este va a leer continuamente por interrupción el registro de los flancos de la unidad de JP2 y mediante una máscara nos vamos a fijar en qué punto varían de valores. El valor que haya estado activo la señal del respectivo eje del acelerómetro va a ser proporcional a los valores en los que el flanco no ha variado de valor.

Por último, una variable global se encarga de guardar en el dispositivo myAcel, los resultados obtenidos por los contadores X e Y.

Debido a que las propiedades de los dispositivos de lectura y escritura en una arquitectura con SO, para poder acceder al kernel se ha creado un dispositivo en un formato en el que únicamente se permite escribir 32 bits.

Mediante el desplazamiento de bits y máscaras se ha guardado los valores del acelerómetro X en los bits menos significativos y los del eje Y en los bits más significativos. El programa aplicación se encarga del procesamiento de estos datos.

Por otro lado, dentro de este kernel se ha añadido otra función para escribir los datos obtenidos en otros periféricos, los displays de siete segmentos.

```
HEX5_HEX4_write ();  //      Escritura
```

Leemos los datos del dispositivo myAcel y los visualizamos en el display transformados a valores decimales.

4. **MotorModule:**

Este módulo es el encargado de enviar los datos del duty de los distintos motores al programa principal y mediante los periféricos KEYS, mover las posiciones de los servomotores.

Similar al módulo anterior, queremos dos funciones que nos habiliten la lectura y escritura al dispositivo denominado “myMotor” para poder interactuar con el “main”.

Para visualizar los valores del duty del motor en el dispositivo “myMotor” y crear el puente entre el programa aplicación (para su posterior procesado) y los datos de los servos se la ha llamado a la función

```
motor_read ();  //    Lectura
```

Similar a lo aplicado para el envío de los datos de los motores, se aplica desplazamientos de bits y máscaras ya que el tamaño máximo de la unidad es de 4 bytes.

Para poder mover los motores, se usa la función

```
motor_write ();  //   Escritura
```

Esta función es la encargada de escribir en los motores los valores recogidos de los KEY_SWITCH.

Los KEYS superiores e inferiores mueven de izquierda a derecha los motores 1 y 2 respectivamente. Para ello, se han activado las interrupciones de los KEYS y se ha asignado una variable global para que modifiquen en su respectiva posición a cada botón.

El uso más complejo es el que se hace a la hora de mover los distintos motores con un único timer, el timer 2 del HPS.

Este timer entrará cada 100 us por interrupción. En cada entrada comparara los valores de Duty_General. Aplicando desplazamientos y máscaras accederemos a los respectivos duty de los motores.

Mediante contadores del tiempo de encendido y el tiempo de apagado controlaremos los periodos de las frecuencias a las que tienen que trabajar los servomotores (50HZ).

1.4. Conversiones

1.4.1. Conversión PWM -> Servomotores

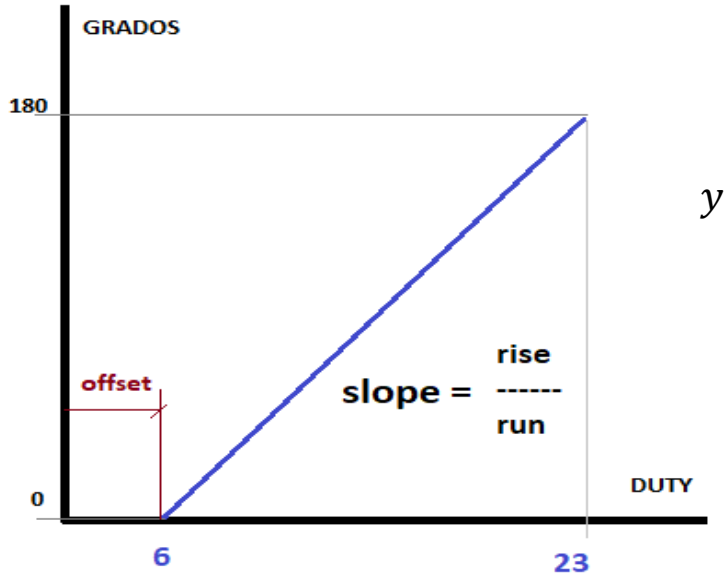
Para poder visualizar los valores recibidos de los distintos servomotores en grados, usamos una función para hacer dicha conversión.

Nuestro timer entrará en la interrupción cada 100 microsegundos y el periodo máximo del servo es de 20 milisegundos, por lo que el contador máximo se incrementará 200 tics antes de inicializarse. Los valores en los que oscilará el servo son entre 6 y 23 tics.

La función que se aplicará para calcular la aceleración va a estar en función del valor PWM que obtengamos del sensor.

Aceleracion = $F(Val_{PWM})$

Usamos una función lineal con un offset específico debido a que no tenemos una función que pueda pasar por el eje Y:



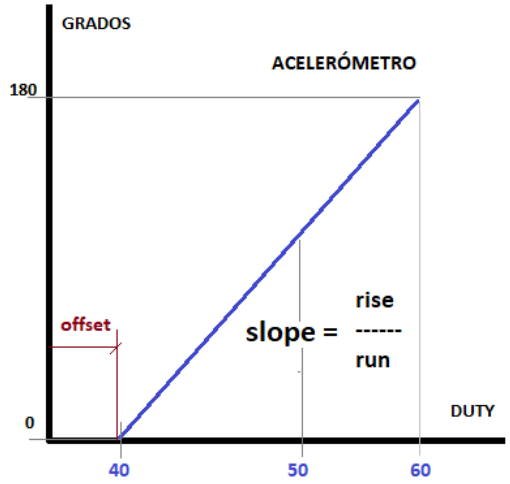
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

$$y = 10x - 50$$

1.4.2. Conversión PWM -> Acelerómetro

Para calcular la posición del acelerómetro, vamos a estar observando el valor de los pines D17 y D19 cada 100 microsegundos. De esta manera observaremos si ha habido un cambio de nivel.

Si relacionamos los grados del motor con el Ton del acelerómetro obtendremos la siguiente recta, cuya ecuación se muestra a la derecha.



$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

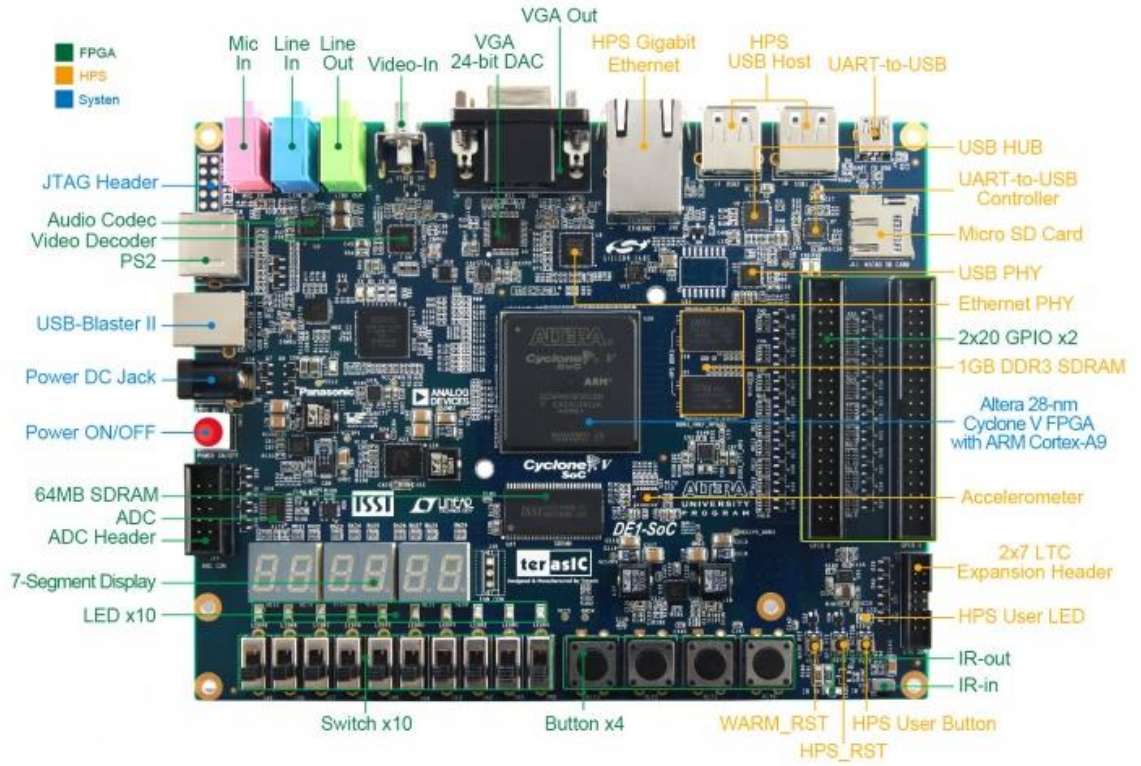
$$y = 9x - 360$$

2 Descripción detallada del HW

2.1 Placa Altera Cyclone V

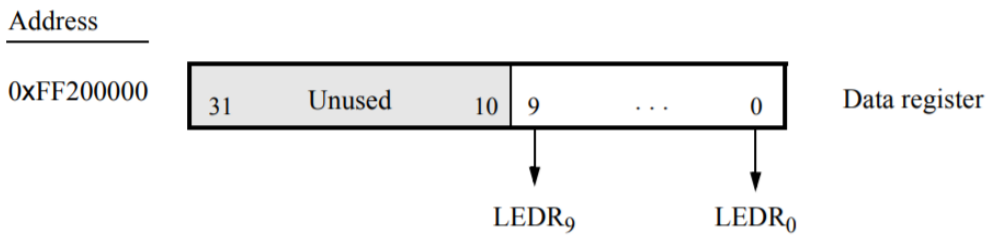
La placa Altera está compuesta por un Sistema de Procesador Duro (HPS) y una FPGA configurable. El HPS se compone de un procesador de doble núcleo ARM Cortex A9, una memoria DDR3 y un conjunto de dispositivos periféricos. La FPGA se compone de dos procesadores Nios II y varios periféricos, entre ellos, interruptores, leds, display de 7 segmentos…

En este proyecto se desarrollará una aplicación para el HPS y se utilizarán los elementos hardware de la parte de la FPGA que se describen a continuación.



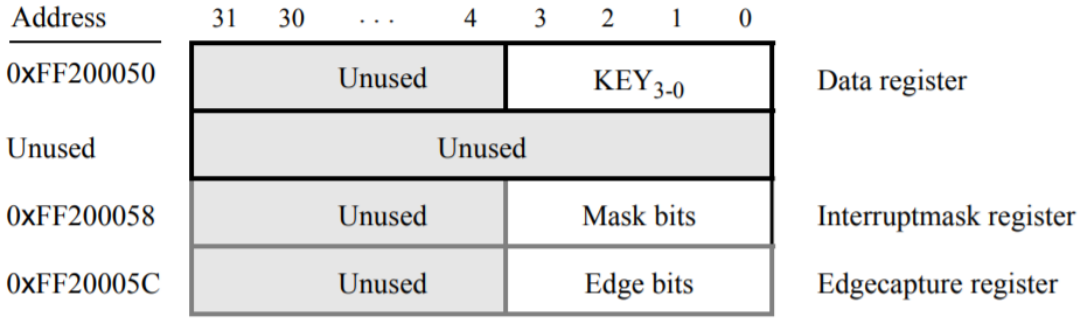
2.2 Leds

La placa altera dispone de 10 leds rojos. Para encender estos leds se debe escribir un 1 en el registro correspondiente de cada led.



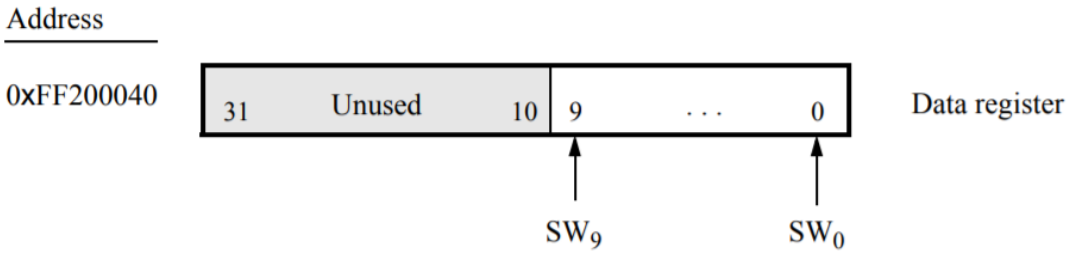
2.3 Botones

La placa dispone de cuatro botones. Cuando el registro correspondiente de cada botón muestra un 0 significa que el botón está pulsado. En este proyecto se configurarán los botones para que trabajen por interrupción. Mediante los 4 botones el usuario podrá controlar el movimiento de los motores. Se utilizarán 2 botones para cada motor. Accionando el botón de la izquierda el motor se moverá hacia la izquierda y accionando el botón de la derecha se moverá hacia la derecha.



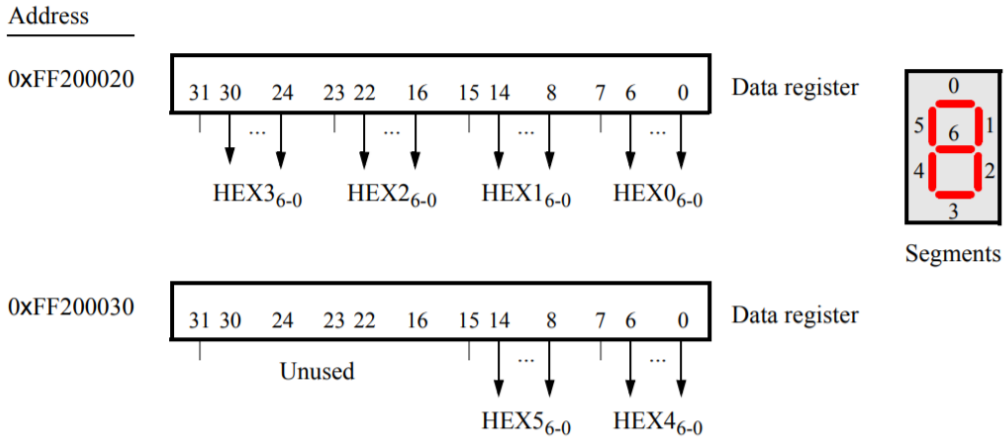
2.4 Interruptores

La placa dispone de 10 interruptores. Cuando se acciona alguno de los interruptores, muestra un 1 en el registro correspondiente.



2.5 Display 7 segmentos

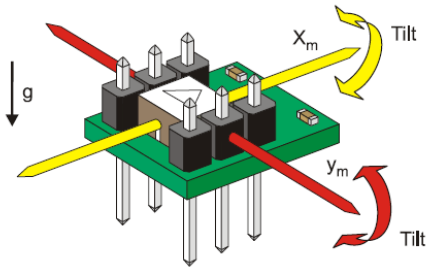
La placa dispone de 6 displays de 7 segmentos. Cada display está compuesto por 7 leds que se encienden cuando se escribe un 1 en el registro correspondiente. Se codificarán los dígitos del 0 al 9 para facilitar la implementación de los resultados mediante display.

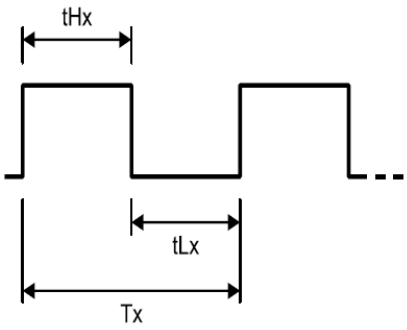


2.6 Acelerómetro

Se ha utilizado el acelerómetro Memsic 2125. Este dispositivo es un acelerómetro térmico de bajo coste capaz de medir la inclinación con un rango de ±3 g en dos ejes.

Internamente, el Memsic 2125 tiene un pequeño radiador que calienta una "burbuja" de aire dentro del dispositivo. Cuando las fuerzas gravitacionales actúan sobre esta burbuja, ésta se mueve. Este movimiento es detectado por sensores de temperatura y mediante electrónica interna convierte la posición de la burbuja [relativa a las fuerzas g] en pulsos de salida para los ejes X e Y.





Cada eje X e Y tiene un pin de salida que produce un pulso PWM de 100 KHz. La aceleración se puede calcular a partir de este pulso, donde la aceleración (g) es proporcional al tiempo de encendido dividido por el periodo (tHx/Tx). Cuando el Duty Cycle es del 50%, significa que no está inclinado y por lo tanto la aceleración es 0g.

A continuación, se muestra la fórmula proporcionada por el fabricante para calcular la aceleración. El resultado A(g) representa la fuerza g en mili-g's (1/1000 g).

$$\mathbf{A(g) = ((T1 / T2) - 0.5) / 12.5\%}$$

Para calcular la inclinación en grados se debe utilizar la siguiente fórmula:

$$\mathbf{Grad\_x = ArcSin\ (g)}$$
$$\mathbf{Grad\_y = ArcCos\ (g)}$$

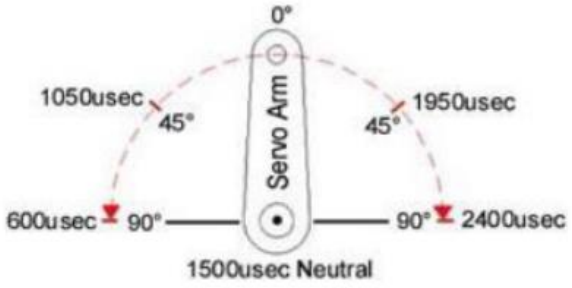
Los pines de salida X e Y del acelerómetro se han conectado a los pines D17 y D19 respectivamente. Estos pines se encuentran en el conector JP2 de la placa Altera.

Se habilitarán las interrupciones de estos dos pines y se utilizará un Timer. Para poder calcular el tiempo de encendido, primero se configurará la interrupción para que se genere en el flanco de subida y a su vez se inicializará el Timer. Después se configurará la interrupción para que se genere en el flanco de bajada y se guardará el valor del Timer en una variable. De esta mera, el Timer nos indicará el tiempo de encendido del pulso. Este proceso se repetirá continuamente.

2.7 Motor

En esta práctica se van a utilizar dos motores Hitec HS-422. El servo Hitec HS-422 permiten una rotación de 180º (-90º a +90º), para ello se debe proporcionar una señal PWM con una frecuencia de 50Hz. El ancho del pulso del duty cicle de la señal puede ir de 600 usec (-90º) a 2400 usec (+90º), siendo 1500 usec (0º) el punto central. Los valores TMAX y TMIN se van a reducir en un 2% para no forzar los topes.

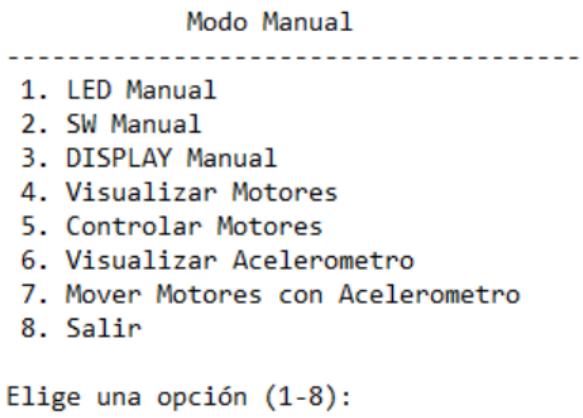
Los pines de control de los motores se han conectado a los pines D5 y D7. Estos pines se encuentran en el conector JP2 de la placa Altera.



3 Interfaz de usuario

Con el objetivo de dar opción al usuario de interactuar con la aplicación y la placa se ha desarrollado una interfaz de usuario. Esta interfaz se muestra mediante el terminal del sistema operativo. De esta manera el usuario puede seleccionar una de las 6 opciones que se le ofrecen e interactuar con los elementos de la placa por separado. En la imagen inferior se muestra el menú que visualizará el usuario.

A continuación, se describen las 6 opciones del menú:



1. LED Manual:

En esta opción el usuario introduce cuantos leds quiere encender de 0 a 10. Una vez acabado, el programa vuelve al menú principal.

2. SW Manual:

En esta opción, el usuario enciende y apaga los interruptores y se enciende el Led superior al interruptor. Para poder empezar con la interacción, el usuario debe encender el interruptor que se encuentra a la izquierda (SW9) y cuando apague este interruptor, terminará la interacción y volverá al menú.

Para iniciar el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

3. DISPLAY Manual:

En esta opción el usuario introduce un número entre 0 y 999999 en la terminal y se muestra en el display.

Para iniciar el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

4. Visualizar Motores:

En esta opción el usuario puede mover el motor con los botones de la placa y se visualiza la posición de los motores en el terminal.

Los botones superiores KEY2-KEY3 y KEY0-KEY1 están asignados para aumentar o reducir los grados del motor dos y uno respectivamente. Los grados de giro de cada servo se visualizarán en la pantalla principal.

Para iniciar el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

5. **Controlar motores:**
   Mediante la inserción de los grados, el servomotor se cambiará a la posición con el ángulo indicado. En caso de error, nos enviará una notificación para volver a intentarlo. Mediante el cambio del SWITCH 0, elegiremos que motor vamos a mover.

   Para cerrar/encender el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

6. Visualizar Acelerómetro:

En esta opción se visualiza en el terminal la posición del acelerómetro en una unidad adimensional.

Para cerrar/encender el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

7. Mover motores con acelerómetro:

Mediante esta opción, el usuario podrá cambiar las posiciones del servo moviendo las posiciones de los ejes de la placa acelerómetro. Para poder cambiar la visualización y el movimiento del servomotor cambiaremos el SWITCH 0.

Para cerrar/encender el juego, el usuario deberá de encender el SWITCH 9. Para poder volver al menú se le advertirá al usuario de que debe apagar todos los interruptores. El programa no seguirá hasta que el usuario realice dicha acción.

8. Salir:

Esta opción termina la ejecución del programa.

4 Verificación de la aplicación

A continuación, se describen los casos de prueba para verificar el funcionamiento de la aplicación. En cada caso se define el objetivo de la prueba, la acción que se debe llevar a cabo (pulsar un botón, introducir un número en el terminal…), el comportamiento previsto y el resultado de la verificación.

Caso de prueba 1
Objetivo: *Comprobar que el módulo generado de los motores se ha instalado* **Entrada**: **insmod Mod_Motores (**`Botones, Timer2 y motores)` **Comportamiento previsto**: *Al pulsar los botones de la placa se moverán los motores para la izquierda o derecha.*
Resultado de la verificación: Correcto
Caso de prueba 2
Objetivo: *Comprobar el control de los motores* **Entrada**: `Ir pulsando el botón de la derecha e izquierda de cada motor.` **Comportamiento previsto**: *Cuando el motor llegue a su rotación máxima (90º) el tiempo de encendido del PWM debe ser Ton = 2400 us. Cuando el motor llegue a su rotación mínima (-90º) el tiempo de encendido del PWM debe ser Ton = 600 us.*
Resultado de la verificación: *Correcto*
Caso de prueba 3
Objetivo: *Comprobar el funcionamiento de los leds* **Entrada**: `5 (modo 1: en la terminal)`
Comportamiento previsto: *Se encenderán 5 leds de la placa altera, empezando desde la derecha. Utilizando el debugger, el valor de la variable de los leds debe ser 0x1F.*
Resultado de la verificación: *Correcto*

Caso de prueba 4
Objetivo: *Comprobar el funcionamiento de los leds cuando se introduce un número mayor a 10* **Entrada**: `15 (modo 1: en la terminal)` **Comportamiento previsto**: *Los leds se quedarán en el estado en el que estaban, ignorando el valor (15) introducido.*
Resultado de la verificación:

Caso de prueba 5
Objetivo: *Comprobar el funcionamiento de los interruptores cuando SW9 está encendido* **Entrada**: `Encender el interruptor SW1 (modo 2: en la terminal)` **Comportamiento previsto**: *Se encenderá el led 1. Si se lee el valor de los 9 primeros interruptores tendrá que ser el mismo valor que el de los leds.*
Resultado de la verificación:

Caso de prueba 6
Objetivo: *Comprobar el funcionamiento de los interruptores cuando SW9 está apagado* **Entrada**: `Encender el interruptor SW1 (modo 2: en la terminal)` **Comportamiento previsto**: *No se debe encender ningún led, ya que la interacción está deshabilitada.*
Resultado de la verificación:

Caso de prueba 7
Objetivo: *Comprobar el funcionamiento de los displays* **Entrada**: `1998 (modo 3: en la terminal)` **Comportamiento previsto**: *Se debe ver el número 1998 en los cuatro primeros displays.*
Resultado de la verificación:

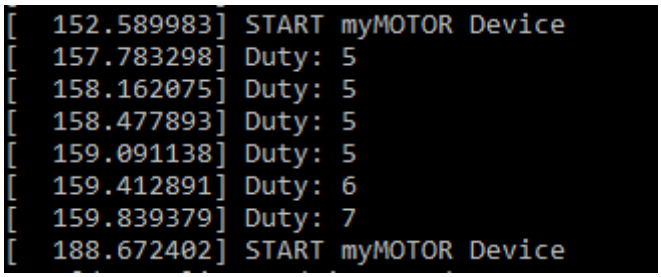
Caso de prueba 8
Objetivo: *Comprobar el funcionamiento de los displays* **Entrada**: `123456 (modo 3: en la terminal)` **Comportamiento previsto***: El display debe mostrar un mensaje de error.*
Resultado de la verificación:

Caso de prueba 9
Objetivo: *Comprobar la visualización de los motores* **Entrada**: `Mover el motor a 90° (modo 4: en la terminal)` **Comportamiento previsto***: Cuando el motor esté en 90º en la terminal deberá mostrar la rotación correspondiente.*
Resultado de la verificación:

5 ANEXOS

Los resultados obtenidos dentro del Kernel al instalar los módulos han sido los siguientes.

6.1. Instalación MotorModule



6.2.

Caso de prueba 10
Objetivo: *Comprobar la visualización del acelerómetro* **Entrada**: `Inclinar 45° el acelerómetro (modo 5: en la terminal)` **Comportamiento previsto***: Cuando el acelerómetro esté en 45º en la terminal deberá mostrar la inclinación correspondiente.*
Resultado de la verificación: