ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

PERIFÉRICOS E INTERFACES

PRÁCTICAS DE LABORATORIO

Módulo 4 – Práctica 4

Máquina de juego

Última actualización: 26 abril 2016

1 Competencias y objetivos de la práctica

En la cuarta práctica de la asignatura Periféricos e Interfaces se profundiza en el diseño de los interfaces de entrada/salida para lo que se propone utilizar el hardware diseñado en la práctica anterior pero cambiando el software para implementar una nueva funcionalidad del periférico orientada al ocio. Para ello, se propone diseñar y programar un juego que básicamente consistirá en adivinar un número de 00 a 99 en un tiempo dado.

La realización de la práctica por parte de los estudiantes se orienta a complementar la consecución de la competencia de título CII01 del Plan de Estudios de la Ingeniería Informática y que los capacita para: diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos asegurando su fiabilidad, seguridad y calidad conforme a principios éticos y a la legislación y normativa vigente. En base a ello, con la realización de esta práctica se pretende que los estudiantes alcancen satisfactoriamente las siguientes capacidades:

1. Capacidad para entender e interrelacionar los diferentes componentes hardware y software que integran un interfaz sencillo de un sistema basado en microcontrolador.
2. Capacidad para diseñar, implementar y verificar el correcto funcionamiento de dispositivos externos sencillos para ser conectados al interfaz paralelo.
3. Capacidad para el desarrollo de programas que permitan el control básico del dispositivo externo haciendo uso de un lenguaje de programación.
4. Capacidad para desarrollar programas que doten al sistema de una determinada funcionalidad.
5. Capacidad para aprender y aplicar nuevos conceptos de forma autónoma e interdisciplinar.
6. Capacidad para emplear la creatividad en la resolución de los problemas.

El logro de las citadas y pretendidas capacidades pasa por el planteamiento de un conjunto de intenciones y metas que orientan el proceso de aprendizaje de los estudiantes y que constituyen los objetivos de la práctica. De esta forma, los objetivos propuestos para esta práctica se concretan en:

1. Profundizar en el conocimiento de la estructura interna de un periférico sencillo dotándolo del software de control e interconexión con el usuario haciendo uso de la plataforma Arduino.
2. Conocer las particularidades de las diferentes señales físicas que aparecen en los conectores de la placa a efectos de poder conectar dispositivos correctamente.
3. Conocer el funcionamiento de los componentes electrónicos básicos utilizados y saber integrarlos en el sistema para lograr la funcionalidad pretendida del periférico.
4. Saber diseñar el software de control del periférico básico propuesto para su correcto funcionamiento de acuerdo a la funcionalidad exigida incluido la comunicación con el usuario y con el sistema computador si fuese necesario.

2 Documentación previa

La documentación básica a utilizar para la realización de esta práctica está disponible en la plataforma de teleenseñanza Moodle y a través del enlace correspondiente. La documentación mínima a manejar será la siguiente:

Enunciado de la práctica: este documento

- Transparencias de teoría correspondientes a los módulos 1, 2, 3 y4.
- Transparencias de presentación de la práctica 4
- Hojas de características de los componentes electrónicos (subdirectorio "Doc" de la práctica 4)
- Página web de Arduino: http://www.arduino.cc/

-

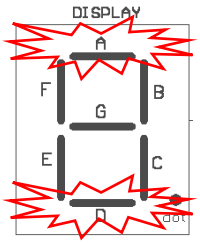
3 Descripción del dispositivo periférico (juego)

3.1.- Descripción general

La actividad práctica a realizar consistirá en el diseño, montaje y verificación del funcionamiento de un dispositivo periférico sencillo, con una funcionalidad que se orienta al ocio, en este caso, un juego. Para ello, se propone rediseñar el software de control de la práctica anterior para que el mismo hardware del "turnomatic" se comporte, ahora, como una máquina de juego capaz de proporcionar el siguiente juego:

Juego propuesto: Adivina el número
En este apartado se realiza una breve descripción del juego propuesto para esta práctica dejando para un posterior apartado (software de control) detalles más finos necesarios para implementar la práctica. En concreto, la máquina deberá generar un número aleatorio entre 00 y 99 y el usuario tendrá que adivinarlo tecleando números a través del teclado. El tiempo de cada partida es limitado y será fijado por el sistema. Los números son de uno o dos dígitos que introducirá el usuario través del teclado y terminados siempre con la tecla "#", que hará de retorno de carro o fin de la entrada de datos (ej. 9# 45# 99# ...).

Si el usuario acierta, el sistema emitirá varios pitidos que creen en el usuario la sensación de acierto o premio y mostrará en pantalla los puntos obtenidos en modo parpadeo (blinking). Si el usuario falla la máquina emitirá un pitido largo (sensación de fallo) y ayudará al usuario informando de si el número secreto es mayor o menor al introducido a través del "display". Para suministrar esta información podrá hacer uso de los segmentos "a" y "d" de cualquiera de los dígitos del display según lo siguiente: parpadeo (blinking), durante un breve periodo de tiempo, del segmento "a" o "d", dependiendo de si el número secreto es mayor o menor al dado por el usuario, respectivamente. Fíjese que el segmento "a" está en la parte más alta del display (número secreto mayor al del intento) y el segmento "d" en la parte inferior (número secreto menor al del intento).



La Figura 1muestra el diagrama de bloques del hardware a utilizar que, como ya hemos comentado, es el mismo que el utilizado en el Turnomatic y está formado por los siguientes componentes: elemento visualizador de dos dígitos (aunque usaremos un módulo de 4 dígitos del que solo usaremos dos), dos pulsadores, un teclado y un altavoz o zumbador dependiendo de la disponibilidad en el laboratorio. Todos estos componentes serán controlados por el software de control a desarrollar a través de los pines de entrada/salida del Arduino.

Los pulsadores se utilizarán para realizar un reset del juego, para empezar una partida o para seleccionar el nivel de dificultad (optativo). Posteriormente, cuando el usuario comienza a jugar utilizará el teclado para introducir los números (siempre terminados con la tecla #) mientras que el zumbador avisará con sonidos específicos de un acierto o un fallo. El display será el encargado de mostrar el tiempo restante de la partida y estará asociado a un temporizador. Asimismo, el display también visualizará la ayuda al usuario a través de los segmentos "a" y "d" para el próximo intento.

Al igual que en el Turnomatic, el software estará organizado en dos tareas básicas: una principal asociada a loop() de Arduino y otra asociada a una rutina de servicio de interrupción que se ejecuta cada 10 ms para el refresco del display y barrido del teclado.

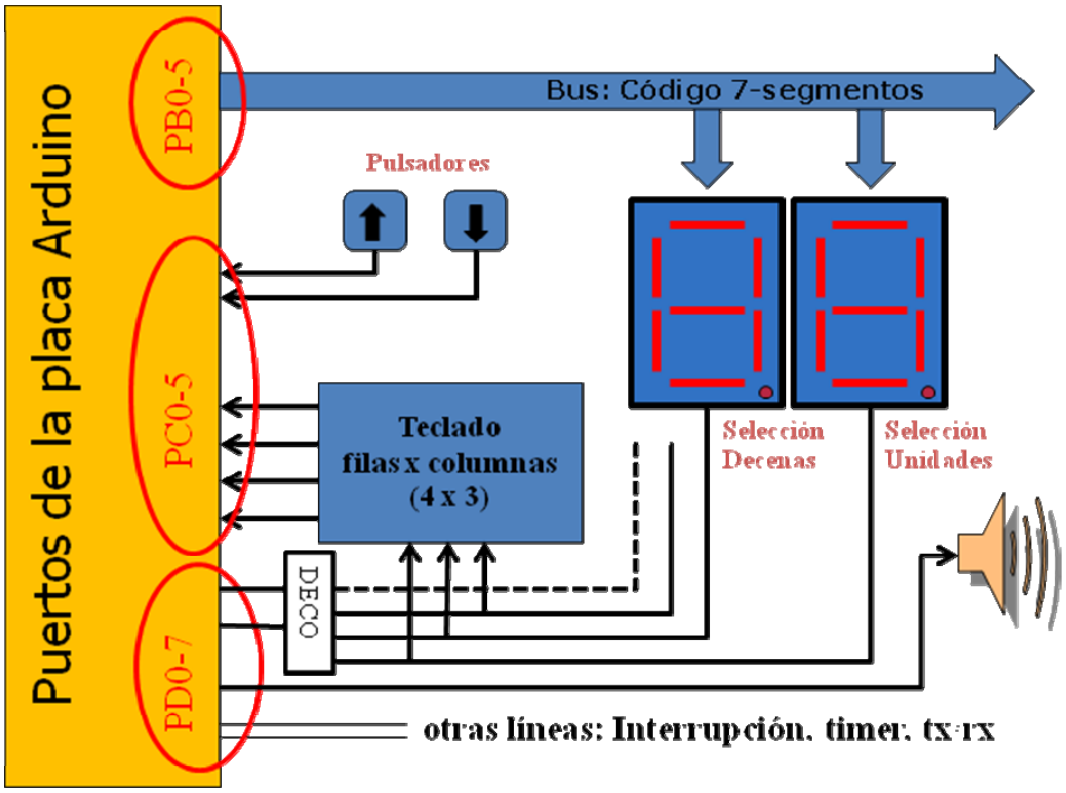


Figura 1. Diagrama de bloques de la máquina de juego

La señal para interrumpir al microcontrolador se obtendrá a partir una onda cuadrada de periodo 10 ms a generar por un pin del Arduino. El periodo seleccionado, que Ud. podrá cambiar y ver el efecto sobre el display, establece un tiempo entre exploraciones lo suficientemente pequeño para que el display no parpadee y sean detectadas todas las pulsaciones, independientemente de lo rápido que el usuario pulse las teclas.

Como se puede observar en el diagrama de bloques de la Figura 1, el código de 7 segmentos correspondiente a unidades y decenas se envía por los mismos puertos o vías de comunicación (bus) pero en diferentes instantes de tiempo (multiplexado). Así, cuando se envía la información de las unidades por el puerto B del Arduino se deberá seleccionar el display de las unidades (línea de selección a "cero") para que muestre la información. El otro dígito o display de 7 segmentos (correspondiente a las decenas) ha de estar deseleccionado (línea de selección a nivel alto o “uno” lógico) por lo que estará apagado y no mostrará información alguna. Esta operación se debe repetir transcurridos 10 ms (la señal de interrupción ya nos fija este tiempo) pero invirtiendo el orden de encendido y apagado de los displays de 7 segmentos, es decir: apagar unidades, depositar en el puerto B las decenas y activar las decenas. Si estas operaciones de visualización de unidades-decenas se realizan de forma alternativa y con la suficiente rapidez (cada 10 ms es adecuado) el display se verá estático, sin parpadeos. Esta forma de operación se puede extender a un display de “n” dígitos.

Respecto del funcionamiento del teclado vemos que es necesario realizar una exploración de las columnas una a una (colocando un cero lógico) seguido de una lectura del estado de las filas. Aprovecharemos las dos líneas de selección del display como líneas de exploración de las dos primeras columnas del teclado (la tercera columna del teclado se quedará sin explorar y, por tanto, estará inutilizada). La exploración consiste en poner un cero en una columna y leer las filas. Si se pulsa una tecla entonces aparecerá un “0” en la fila que corresponda a la tecla ya que la pulsación conectará la columna que está siendo explorada con una fila. Las dos columnas del teclado se exploran con la misma frecuencia que el barrido del display lo que es más que suficiente para no perder pulsaciones que pueda realizar el usuario. Una vez detectada la pulsación se asignará un código a la tecla pulsada.

3.2.- Componentes básicos

3.2.1.- Placa Arduino: Líneas (o pines) de entrada-salida

El hardware del "turnomatic" se conectará a las líneas de entrada/salida del Arduino que será el encargado de controlar todos los componentes para darle al sistema la funcionalidad requerida. En la figura 2, se muestra una imagen de la placa Arduino y en ella podemos apreciar los conectores correspondientes a los puertos de entrada-salida.

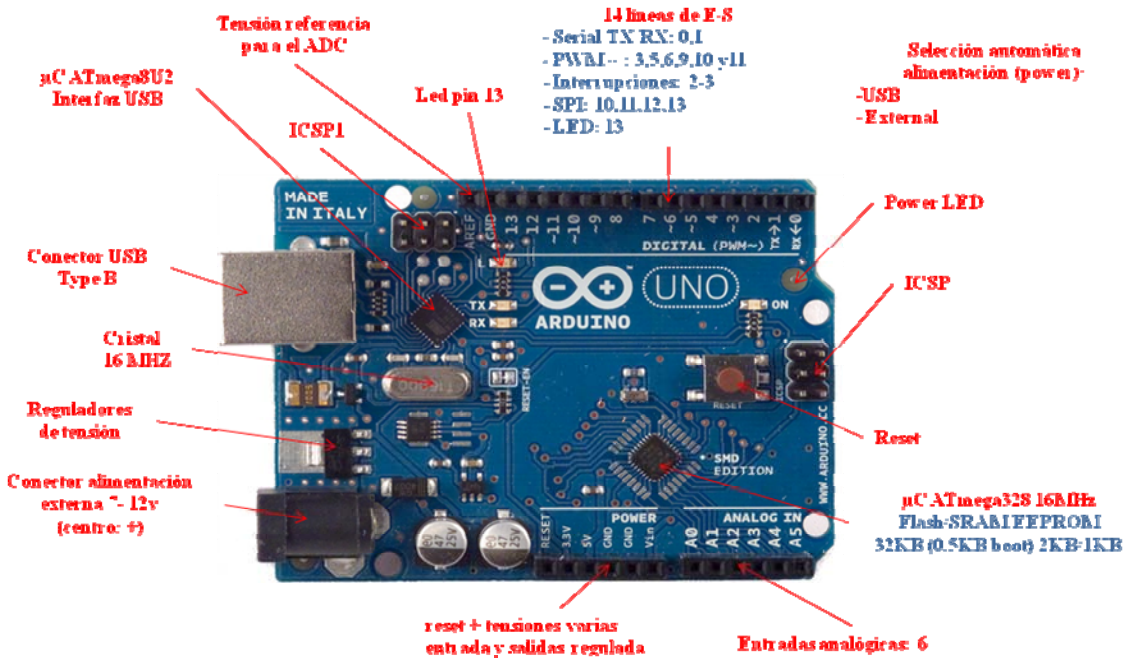


Figura 2. Placa Arduino Uno

Se distinguen dos tipos de líneas de entrada-salida: digitales y analógicas. Las líneas digitales (0-13) nos permiten enviar o recibir "unos" y "ceros" mientras que las líneas analógicas, **A0-A5**, son sólo de entrada y permiten la lectura de señales analógicas realizándose una conversión analógico-digital de 10 bits de resolución. Si, en un momento dado, se necesitasen más líneas digitales, las líneas analógicas se podrían reprogramar para que se comporten como líneas de entrada digitales o líneas de salida digitales (hacer uso del pinMode o DDRx) y entonces tendríamos un total de 20 líneas para entradas-salidas digitales referenciadas como: 0, 1, 2, 3, 4 ..… 11, 12, 13, **14, 15, 16, 17, 18, 19.**

En el microcontrolador las líneas de entrada-salida están agrupadas en puertos o registros de tamaño 6 u 8 bits con los siguientes nombres:

Puerto **PD** (PD07-PD00): Líneas 07-00
Puerto **PB** (PB05-PB00): Líneas 13-08
Puerto **PC** (PC05-PC00): Líneas A5-A0 (analógicas) o 19-14 (digitales).

Programación del modo de funcionamiento de las líneas de entrada/salida digitales
Antes de leer o escribir información por las líneas de entrada-salida es necesario programarlas para que funcionen en modo entrada o en modo salida. Esta operación se puede realizar línea a línea con la función *pinMode():*

```
pinMode (nº línea, OUTPUT);
pinMode (nºlínea, INPUT);
```

También se pueden programar a la vez (en paralelo) todas las líneas de un puerto como de entrada o como salida utilizando el registro de control interno (DDRx) asociado a cada uno de los puertos: DDRB (para el puerto PB), DDRC (para el puerto PC) y DDRD (para el puerto PD). Un "1" programa una línea como de salida y un "0" programa la línea como de entrada. Por ejemplo:

Programar todas las líneas del puerto PB de salida:

```
DDRB = B111111; o DDRB = DDRB | B111111;
```

Programar todas las líneas del puerto PB de entrada:
```
DDRB = B000000;
```

Programar todas las líneas del puerto PC(analógicas, por defecto) a entradas digitales:
```
DDRC = B000000; líneas de entrada digitales: 14-15-16-17-18-19
```

Operaciones de lectura y escritura de las líneas de entrada-salida.

Lectura línea a línea (o bit a bit):
Val = digitalRead(nº línea)

Lectura de todas las líneas asociadas a un puerto (puerto 6 u 8 bits):
Valb = PINB; // lectura de todas las líneas asociadas al puerto B (13-08)
Valc = PINC;
Vald = PIND;

Escritura línea a línea (bit a bit):
digitalWrite(nº línea, HIGH);
digitalWrite (nº línea, LOW);

Escritura de de todas las líneas asociadas a un puerto (puerto 6 u 8 bits):
PORTB = 28;
PORTC = val;
PORTD = B00110011;

3.2.2.- Display de 7 segmentos

El dispositivo a desarrollar nos permitirá establecer turnos desde 00 hasta 99 utilizando como dispositivos de visualización dos unidades de "display de 7-segmentos" y dos pulsadores que serán utilizados para incrementar, decrementar o poner a cero el contador cuando se pulsan ambos simultáneamente. La figura 3 muestra la estructura interna de un display 7-segmentos formada básicamente por 7 segmentos (a, b, c, d, e, f y g) y un punto (dot) conectados en una configuración de ánodo común o cátodo común. El utilizado en esta práctica (HP 5082-7740) será de cátodo común por lo que para encender los segmentos será necesario aplicarles una tensión positiva y conectar a tierra (0 voltios) la pata o pin que corresponda el cátodo común, en nuestro caso, pines 1 y 6, según se aprecia en la figura. En las hojas de características del display podrá consultar todos los detalles relativos a las tensiones y corrientes necesarias para el correcto funcionamiento. Según dichas hojas, será suficiente una corriente de IF= 20 mA para que un segmento brille fuertemente. En nuestro caso y para no cargar demasiado los puertos del Arduino, limitaremos esta corriente a 8-10 mA que será suficiente para una visualización satisfactoria.

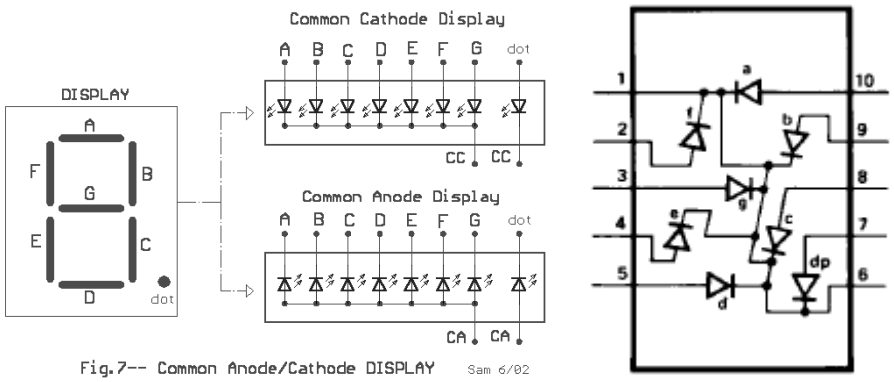


Figura 3. a) Display de 7 segmentos; b) Pines del HP 5082-7740)

También se podrá hacer uso de un display de 4 dígitos, como el mostrado en figura 3c, con el mismo principio de funcionamiento: los pines 12, 9, 8, 6 el cátodo común de los dígitos 1, 2, 3 y 4, respectivamente. En la práctica será suficiente el uso de los dígitos 3 y 4 (los dos menos significativos) ya que la aplicación sólo necesita dos dígitos.

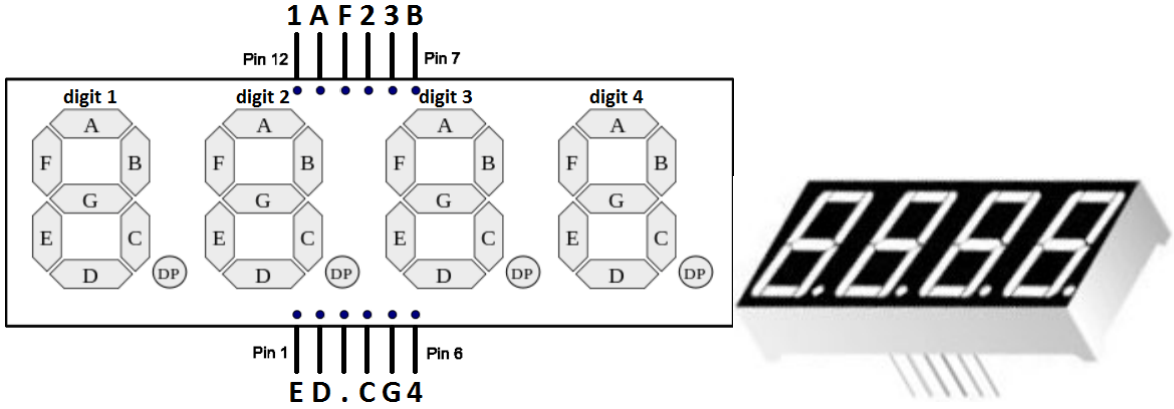


Figura 3. c) Display de 4 dígitos SMA420564

3.2.3.- Teclado y pulsadores

Otros de los componentes a utilizar será el teclado que se puede entender como una matriz de filas y columnas con una tecla en cada uno de los elementos de la matriz. Cuando se pulsa una tecla se pone en contacto una fila con una columna. En el esquema de la figura se muestra el teclado que utilizaremos en la práctica así como las señales que podemos encontrar en su conector. Por otra parte, los pulsadores son elementos muy sencillos que únicamente establecen una conexión entre dos terminales cuando se pulsan.

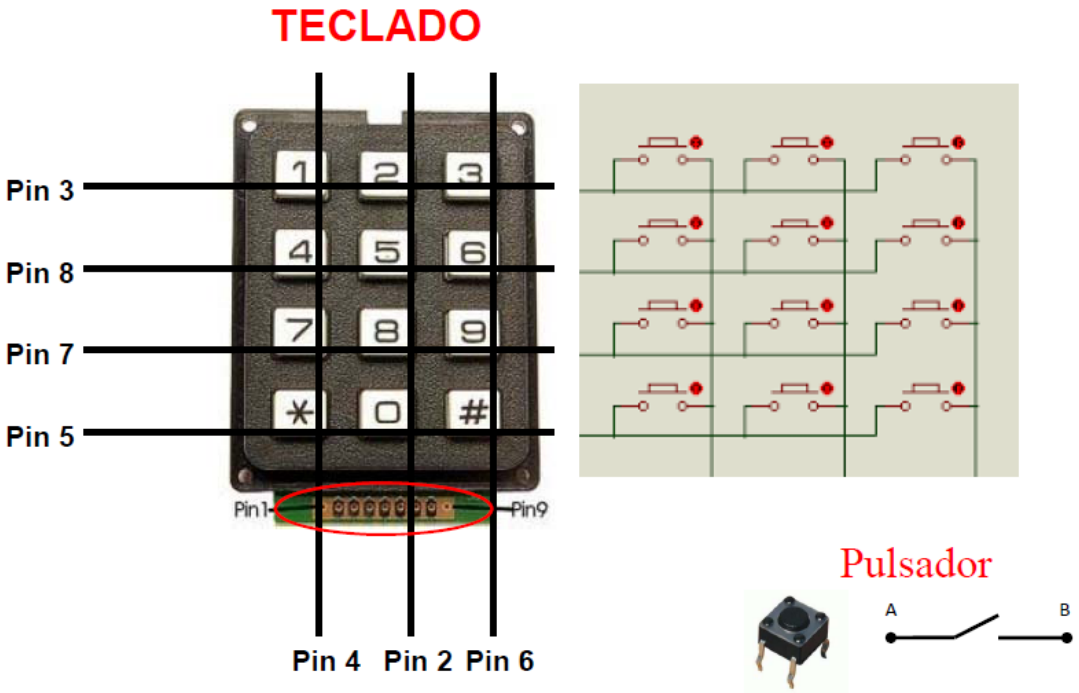
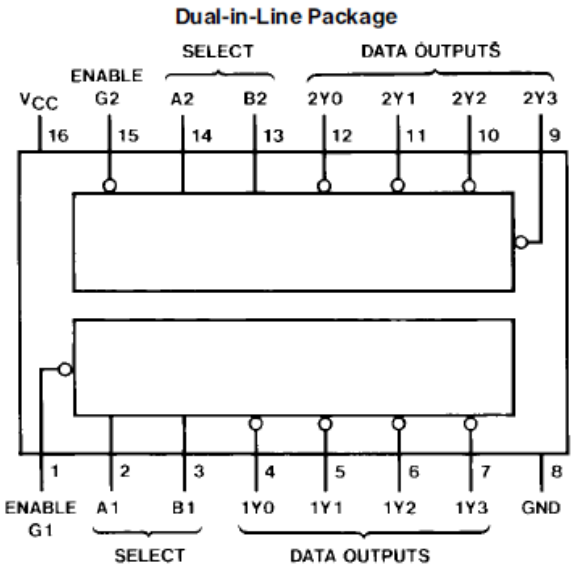


Figura 4. Teclado y pulsador sencillo

3.2.4.- Decodificador

Chip 74LS139 correspondiente a un decodificador de 2 a 4. Este chip tiene dos decodificadores internos de 2 a 4 y del que solo utilizaríamos uno. A continuación se muestra el patillaje así como la tabla de verdad:



LS139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = High Level, L = Low Level, X = Don't Care

Figura 5. Decodificador 74LS139

3.3.- Esquema general del circuito de la máquina de juego

Una vez vistos los componentes básicos a utilizar en la práctica vamos a continuar con una propuesta de circuito que el estudiante habrá de estudiar, analizar e implementar para posteriormente, y una vez entendido el funcionamiento básico, desarrollar el software de control para que el hardware diseñado tenga la funcionalidad de la máquina de juego propuesta, verificando su correcto funcionamiento. La figura 6 muestra un esquema de todas las interconexiones entre los diferentes dispositivos de la máquina de juego que será montado en una placa para el desarrollo de prototipos (breadboard) y conectado a los puertos del Arduino.

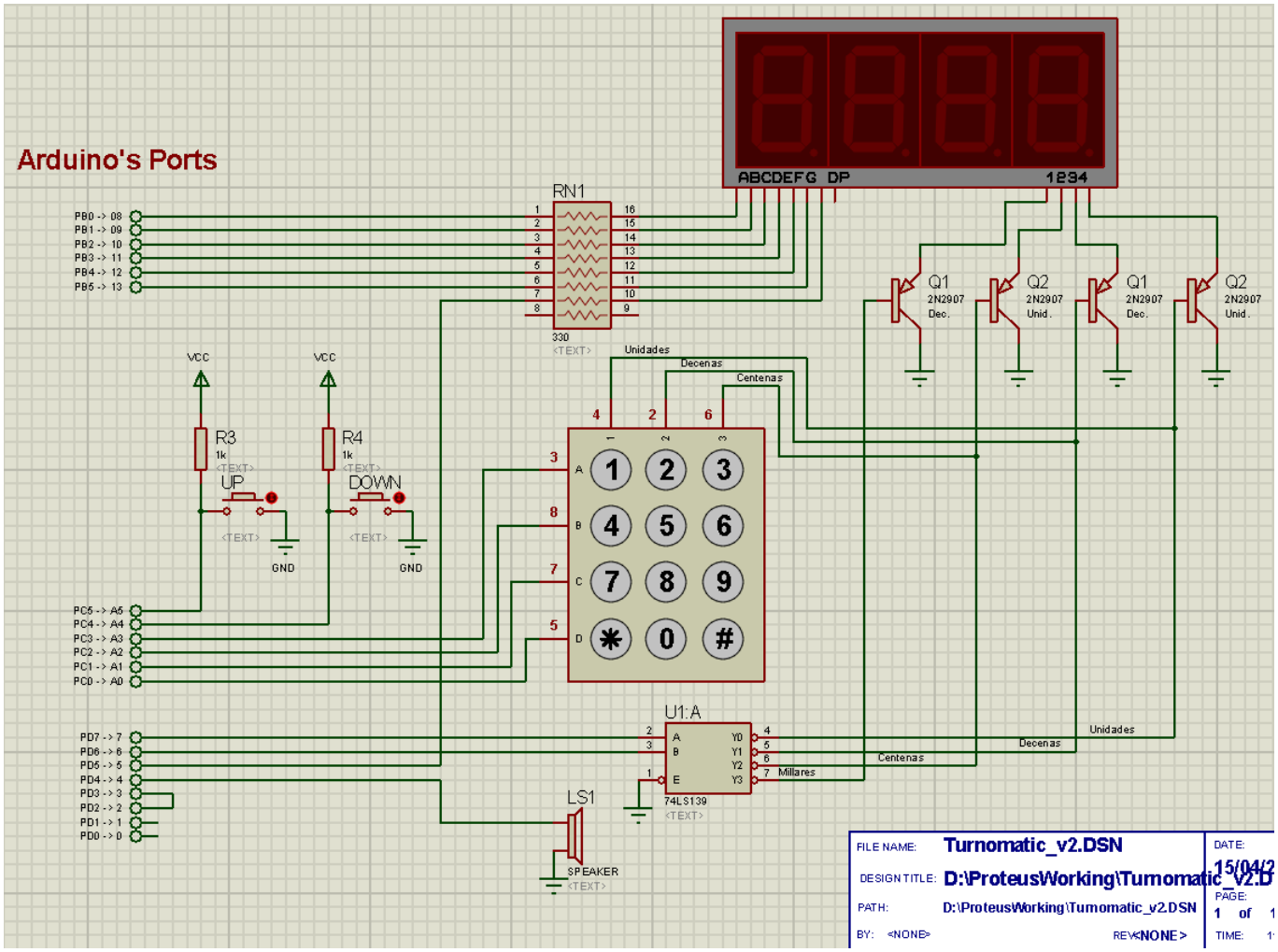


Figura 6.- Esquema del circuito de la máquina de juego (mismo que el Turnomatic)

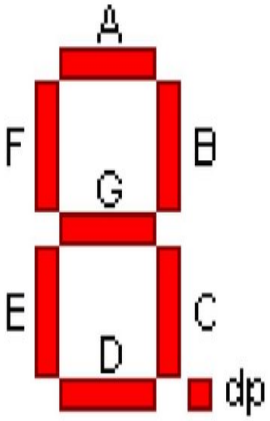
Display 7-segmentos

El diseño consta de dos dígitos implementados con displays de 7 segmentos conectados al puerto PB0-5 y PD5 del Arduino (puertos de salida). Véase como las conexiones de los segmentos al puerto de datos se hace a través de una resistencia de 330 ohmios para limitar la corriente que circula por los segmentos a efectos de protegerlos (que no se dañen por exceso de corriente). Los dos dígitos de la máquina correspondientes a las decenas y unidades se han conectado al mismo puerto de datos. Sin embargo, las conexiones asociadas al cátodo común de los displays no son comunes como se puede apreciar en la figura 6. El cátodo común del dígito más a la derecha (unidades) está conectado al emisor de un transistor que actúa como un interruptor y se gobierna con la señal *Y0* del decodificador. Cuando dicha señal está a cero el transistor se satura o conduce conectando el cátodo común a tierra y, por tanto, todos aquellos segmentos que tenga un "1" o 5 voltios a la entrada se encenderán. De igual forma, se ha conectado el dígito de la izquierda (decenas) pero en este caso es la señal Y1 del decodificador la encargada de activar el display. A continuación se muestra la tabla de verdad del decodificador que permite con dos señales del Arduino (PD6-PD7) controlar hasta 4 dígitos y/o un teclado de 4 hasta columnas:

B PD6	A PD7	Salida del Decodificador	Descripción	Expl. Teclado
0	0	Y0	Unidades	Columna 0
0	1	Y1	Decenas	Columna 1
1	0	Y2	Centenas	Columna 2
1	1	Y3	Millares	-

Para visualizar los dígitos del 0 al 9 en cada uno de los displays será necesario depositar en el puerto B la combinación de "unos y "ceros" adecuada (código de 7 segmentos) para formar el dígito deseado. La siguiente

tabla muestra el orden en el que se ha conectado los diferentes segmentos de los dígitos al puerto B así como los valores que se han de enviar para visualizar algunos de los dígitos decimales:



	Puertos Arduino								**Valor**
bit →		**PD5**	**PB5**	**PB4**	**PB3**	**PB2**	**PB1**	**PB0**	
Segmento→	***dot***	***g***	***f***	***e***	***d***	***c***	***b***	***a***	
0	-	0	1	1	1	1	1	1	0x3F=63
1	-	0	0	0	0	1	1	0	0x06=06
2	-	1	0	1	1	0	1	1	0x5B=91
3									
4									
5									
6									
7									
8									
9									
10	Otros: Todos los segmentos apagados								
11	Otros: Todos los segmentos encendidos								
12	Otros: Solo segmento "a" encendido								
13	Otros: Solo segmento "d" encendido								
...	... y los que estime necesarios								

El resto de la tabla se deja como ejercicio para el estudiante. Como se aprecia se pueden generar otros símbolos diferentes a los dígitos decimales: por ejemplo, todos los segmentos encendidos o apagados, encender sólo un segmento así como cualquier otra combinación que se necesite.

Teclado

Cada vez que se haga el barrido del display también se hará la exploración del teclado: una columna cada vez. Por ejemplo, cuando se están visualizando las unidades habrá un “cero lógico” en la primera columna del teclado por lo que se procederá a leer las filas del teclado a través de las líneas A0-A3 del Arduino para detectar si hay algún "cero" en alguna de ellas. Para ello, se habrá que programar las entradas analógicas (A0-A3) como entradas digitales (recuerde activar las resistencias de pull-up del Arduino para que estas líneas no queden al aire) y se referenciarán como las líneas: 14-15-16 y 17. Éstas se podrán leer línea a línea o en paralelo con la función “PINC”. Cuando se detecte una pulsación se codificará la tecla con el código que corresponda y se almacenará en memoria para su posterior uso.

El teclado se utilizará para introducir los números con objeto de acertar con el número secreto generado por el sistema. Los números será introducidos por secuencias similares a las mostradas a continuación recordando que el carácter "#" actúa como la tecla de "enter" o final de línea:

```
1#      --> número 1
45#     --> número 45
59#     --> número 59
00#     --> número 0
0#      --> número 0
```

Pulsadores

Los pulsadores son conectados a puertos de entrada del Arduino de forma que cuando se pulsan colocan un “0” en la línea de entrada y un "1" cuando no están pulsados. Para no dejar estas líneas al aire será necesario conectar las resistencias de pull-up internas del Arduino o bien conectar resistencias externas como las del esquema.

Para activar estas resistencias de pull-up se procede de la siguiente forma:

1.- Se programa el puerto o líneas que interesen como de entrada
2.- Se escribe un "1" en las líneas en las que queremos activar el pull-up.

Ejemplo para el puerto B:

```
DDRB = B00000000; // puerto B de entrada (o líneas 8-13)
PORTB = B00000011; activamos pull-up interno de las líneas PB0 (línea 8) y
PB1 (línea 9).
```

Es importante destacar que tanto los pulsadores como las filas del teclado se han conectado al puerto C (PORTC) del Arduino que por defecto son entradas analógicas. Para usarlas en modo digital es necesario programarlas como entradas digitales de la siguiente forma:

```
DDRC = B000000 // líneas PC5(A5)-PC0(A0) de entrada (digital)
PORTC = B111111 // Activación de las líneas de pull-up internas
```

Ahora ya podremos utilizar las líneas A5-A0 como líneas de entrada y utilizar la función:

```
Val = digitalRead(nº línea)
```

El nº de línea será correlativo al 13, es decir, 14 (para A0), 15 (para A1) … y 19 (para A5).

Ejemplo: lectura del pulsador "UP"

```
Val1 = digitalRead(18); // pulsador DOWN conectado a A4 (línea 18)
Val2 = digitalRead(19); // pulsador UP conectado a A5 (línea 19)
```

Altavoz

Su función es la de emitir sonidos. Para generar una señal de un tono determinado se podrá hacer uso de la función:

```
tone(nº línea, frecuencia, duración)
```

3.4.- Software de la máquina de juego

El software de la máquina de juego se desarrollará bajo el entorno de programación de Arduino haciendo uso de las utilidades que nos ofrece su lenguaje de programación para controlar los diversos aspectos del hardware. Como en todo programa que se desarrolle en este entorno, tendremos:

1.- Declaración de variables
2.- Código de inicialización (se ejecuta sólo una vez): void setup() { ….}
3.- Código de ejecución indefinida: void loop() { …. }

Ejemplo de programa:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  This example code is in the public domain.
*/
// Declaración de variables
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// Inicialización: setup()
// the setup routine runs once when you press reset:
void setup() {
```

```
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// Bucle indefinido: loop()
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
// posibles llamadas a funciones: function_1() …. Función_n()
}

void function_1() { … }

void function_2() { … }

void function_3() { … }

void function_n() { … }
```

El software se organizará en base a las dos tareas básicas que se describen a continuación:

Tarea 1: Tarea principal

Esta tarea constituye la tarea principal de la aplicación ubicada en la función loop() del programa y será interrumpida por el timer cada 10 ms para ejecutar la rutina de servicio encargada de la visualización y exploración del teclado.

La función principal de esta tarea es implementar la dinámica del juego que pasamos a detallar:

Fase 1: setup()

Ejecución de la función setup() donde se realizan tareas de inicialización y la programación de los modos de funcionamiento de los puertos así como cualquier otra tarea que solo sea necesario ejecutar una vez.

Fase 2: loop()

Fase 2.1: Start

El sistema entra en un estado de espera consultando los pulsadores. Si se pulsa "**UP**" entonces se empieza la partida: generará número secreto e inicializará el temporizador (variable global "count") con el valor 60 que corresponde al tiempo, en segundos, que el sistema da al usuario para acertar el número secreto. Poner a cero el flag de número nuevo (flag=0). El temporizador (variable "count") se actualizará disminuyendo su valor en una unidad por cada 100 entradas en la rutina de servicio de la interrupción. Tenga en cuenta que 100 interrupciones equivalen a 1 segundo ya que se produce una interrupción cada 10 milisegundos.

Fase 2.2: Jugando

Actualizar las variables asociadas a los dígitos decenas-unidades del display con el valor del temporizador o variable "count". De esta forma, siempre tendremos actualizado el tiempo restante que se muestra en el display.

Si el temporizador llega a cero se termina la partida, el usuario ha perdido y no gana puntos. Emitir sonido de fallo, mostrar 00 en display con parpadeo, y volver a Fase 2.1

Si hay condición de reset o final de partida (UP y DOWN pulsados simultáneamente) ir a fase 2.1 directamente. Quedamos a la espera de una nueva partida.

Comprobar si se ha introducido número nuevo.

Si hay número nuevo (flag=1) **entonces**:

Comparar con el número secreto:

Si hay acierto, poner el display en parpadeo, generar sonidos de premio y poner a cero flag de número nuevo (flag=0). Leer el valor del temporizador y actualizar puntos conseguidos con ese valor. Fin de partida. Ir a Fase 2.1 start.

Si no hay acierto, generar sonido de fallo y mostrar ayuda de la máquina en el display durante 3 segundos: segmento "a" en parpadeo si el número secreto es mayor que el introducido y segmento "d" en parpadeo si el número secreto es menor que el introducido.

Poner a cero flag de número nuevo (flag=0)

Ir a Fase 2.2

sino:

Ir a Fase 2.2:

Tarea 2.- Exploración del display y teclado (rutina de servicio de la interrupción)

Display.-

El sistema dispone de un display con dos dígitos de 7-seg. para visualizar las decenas-unidades del temporizador y otras informaciones. En cada instante sólo habrá un display activo (decenas o unidades) que será seleccionado con su correspondiente línea de selección. Cuando se selecciona un display los segmentos se encienden de acuerdo a la información que esté en el bus. Así, por ejemplo, para visualizar las unidades se habrá de inhabilitar ambos displays, colocar la información de las unidades (codificada en 7 seg.) en el bus, y luego seleccionar el display de unidades. Para mostrar el contador completo, con sus dos dígitos, bastará con seleccionar las decenas y unidades de forma alternativa y, con la suficiente rapidez, como para que el contador no parpadee.

Teclado.-

Si el flag de número nuevo está a cero (flag=0) entonces se procederá a explorar el teclado para un nuevo número para lo que será necesario detectar secuencias de teclas terminadas con el símbolo "#". El número nuevo resultante será almacenado en una variable y se activará el flag de número nuevo (flag=1).

Al igual que en la práctica anterior las líneas de selección de los displays (activas a nivel bajo o "0") también se aprovechan para explorar las columnas del teclado de modo que cuando se visualizan las unidades, simultáneamente, se explora la primera columna del teclado. De igual forma, para el resto de las columnas. Note que aunque no esté conectado el tercer dígito del display será necesario explorar la tercera columna del teclado.

Variable del temporizador (count).-

Por último, actualizará la variable utilizada como temporizador (count). Cada 100 entradas o ejecuciones de la rutina de servicio se ha de decrementar en uno la variable count. Para ello, debe tener en cuenta que la rutina de servicio se ejecuta cada 10 ms (100 x 10 = 1 segundo).

Para garantizar que esta tarea se ejecute cada 10 ms haremos uso de las interrupciones. La señal de petición de interrupción se generará por la línea 3 con la función:

```
tone(nº línea, frecuencia)
```

Esta función permite generar una señal de onda cuadrada con un periodo de 10 ms que la utilizaremos para interrumpir al microcontrolador por la línea o pin nº 2 (interrupción externa nivel 0). Para que pueda interrumpir y ejecutar la rutina de servicio asociada (tarea 2) será necesario ejecutar la función:

```
attachInterrupt(nº interrupción, rutina de servicio, modo)
```

4 Realización práctica

Las tareas a desarrollar serán las siguientes:

Implementación básica (70%):
1.- Revisar el montaje del circuito comprobando todas las interconexiones de acuerdo al esquema del circuito de la figura 6. Realice los ajustes que estime oportunos.
2.- Probar, si lo estima necesario, el funcionamiento del display enviando códigos de 7 segmentos a los puertos y activando y desactivando las unidades y decenas.
3.- Implemente un contador de dos dígitos que cuente de 00 a 99. Con un pulsador se pone a cero y arranca y, con el otro, la cuenta se detiene.
3.- Implementar la tarea 2 y comprobar su funcionamiento conjuntamente con la interrupción.
4.- Implementar la tarea 1 y comprobar el funcionamiento del juego

Mejoras (optativo) (30%):

Mejora 1.- (1 punto)
Como mejoras se propone hacer uso del otro pulsador (DOWN) para que en la fase Fase 2.1-start, el usuario pueda seleccionar el nivel de dificultad del juego: 0, 1 y 2. Esto puede consistir en dar menos tiempo y más puntos. Por ejemplo:

Nivel dificultad	Tiempo restante (seg)	Peso	Puntos
0	60	1	60
1	40	3	120
2	20	8	180

Caso práctico.- Jugador jugando con dificultad 1
Si cuando acierta le quedan aún 15 segundos para el final de la partida entonces obtendría: 15x3 = 45 puntos.

Al finalizar de la partida se enviarán al "Monitor Serial" los intentos realizados y los puntos conseguidos. No se guarda información del jugador ni de la partida.

Mejora 2.- (1 punto)
Se añade a la mejora 1 la capacidad de registrar a los jugadores. Antes de la partida se pide, a través del "Monitor Serial", el nombre del jugador (nombre sencillo de 4 a 8 caracteres, por ejemplo) y se crea un registro. Cuando finaliza una partida, se envía al "Monitor Serial" los puntos conseguidos y los intentos y se registra la mejor jugada. Diseño de un menú con las opciones adecuadas para acceder a los registros de los jugadores: lista de jugadores y sus datos asociados (puntos e intentos) a la mejor partida.

Mejora 3.- (1 punto)
A proponer por el alumno previa consulta al profesorado.

Mejoras propuestas por el estudiante:
Otras mejoras son posibles pero procurando que la complejidad sea similar (o mayor!) a las propuestas.

En el Anexo A se muestra una imagen del montaje final de la práctica. **Para realizar las conexiones** utilice, exclusivamente, **cables especiales suministrados** para tal fin como los mostrados en la figura. Otros cables pueden partirse y quedar dentro de los conectores de la placa Arduino inutilizando el pin afectado.

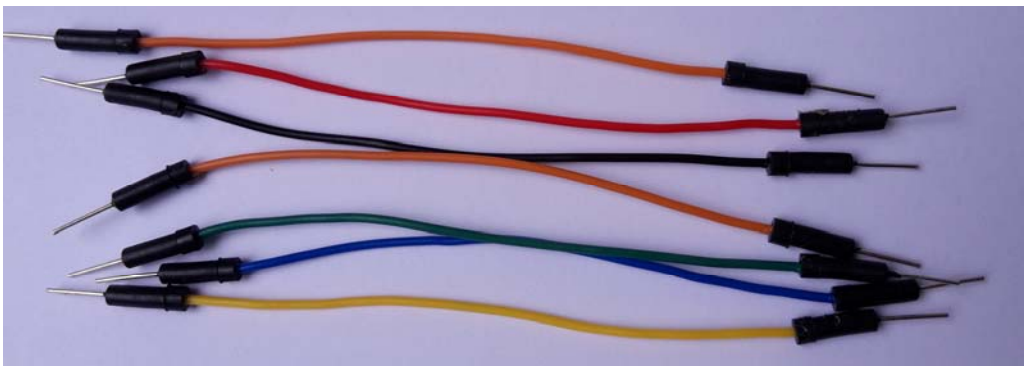


Figura 7.- Esquema del circuito del "turnomatic"

5 Defensa y entrega de la memoria

De la realización de la práctica se realizará una memoria redactada de acuerdo a la normativa de prácticas y en la que se detallaran todos los aspectos hardware y software de la práctica desarrollada. Una vez transcurrido el plazo para la realización de la práctica y en hora convenida se realizará una defensa de la misma contestando a cuantas preguntas el profesor formule.

Sesiones de laboratorios quincenales: 3
Fecha límite de entrega: última semana del curso

ANEXO A