ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

PERIFÉRICOS E INTERFACES

PRÁCTICAS DE LABORATORIO

Módulo 1 – Práctica 3

Programación de interfaces paralelos

Última actualización: 15 abril 2016

1 Competencias y objetivos de la práctica

La tercera práctica de la asignatura Periféricos e Interfaces se centra en la puesta en práctica de conocimientos teóricos asociados al módulo 3 de la asignatura relativos a los periféricos de entrada y salida de datos si bien es necesario el bagaje adquirido en las prácticas y módulos teóricos anteriormente estudiados. En esta práctica se plantea la programación y control de periféricos a bajo nivel proponiendo al estudiante la implementación de un periférico sencillo, un turnomatic, haciendo uso de los puertos de entrada salida de una placa Arduino.

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 facilite el uso del dispositivo externo a un usuario final.
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. Conocer la estructura interna de un interfaz típico de entrada/salida a través del estudio y manejo de los puertos paralelos o entradas/salidas de una placa tipo Arduino.
2. Conocer y entender la funcionalidad de cada uno de los bits de los puertos de E/S y cómo se relacionan con las señales a nivel de conexiones externas.
3. Conocer las particularidades de las diferentes señales físicas que aparecen en los conectores de la placa a efectos de poder conectar dispositivos correctamente.
4. Saber programar y desarrollar una aplicación sencilla, haciendo uso del entorno de programación de Arduino y el lenguaje de programación C para controlar dispositivos conectados a los puertos.
5. Conocer el funcionamiento de algunos componentes electrónicos básicos y la forma de conectarlos.
6. Diseñar y desarrollar una aplicación de complejidad media para implementar una determinada funcionalidad contemplando los aspectos tanto el hardware como el software de control de bajo nivel.

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 y 3.
- Transparencias de presentación de la práctica 3
- Hojas de características de los componentes electrónicos (subdirectorio "Doc" de la práctica 3)
- Página web de Arduino: http://www.arduino.cc/

-

3 Descripción del dispositivo periférico (turnomatic)

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 similar a un "turnomatic" (dispositivo para establecer un turno en la prestación de algún servicio: supermercado, bancos, ...) que estará 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. Estos componentes serán controlados a través de los pines de entrada/salida del Arduino para lo que será necesario desarrollar el software correspondiente para obtener una funcionalidad similar a la de un "turnomatic". Este dispositivo visualizador presenta un turno (número de 2 dígitos) que podemos incrementar, decrementar o poner a cero a través de pulsadores y que avisa con una señal acústica cada vez que cambia el estado del contador. Esta frecuencia ha de ser variable y seleccionable por el usuario a través del teclado. La figura 1 muestra el diagrama de bloques del dispositivo a implementar.

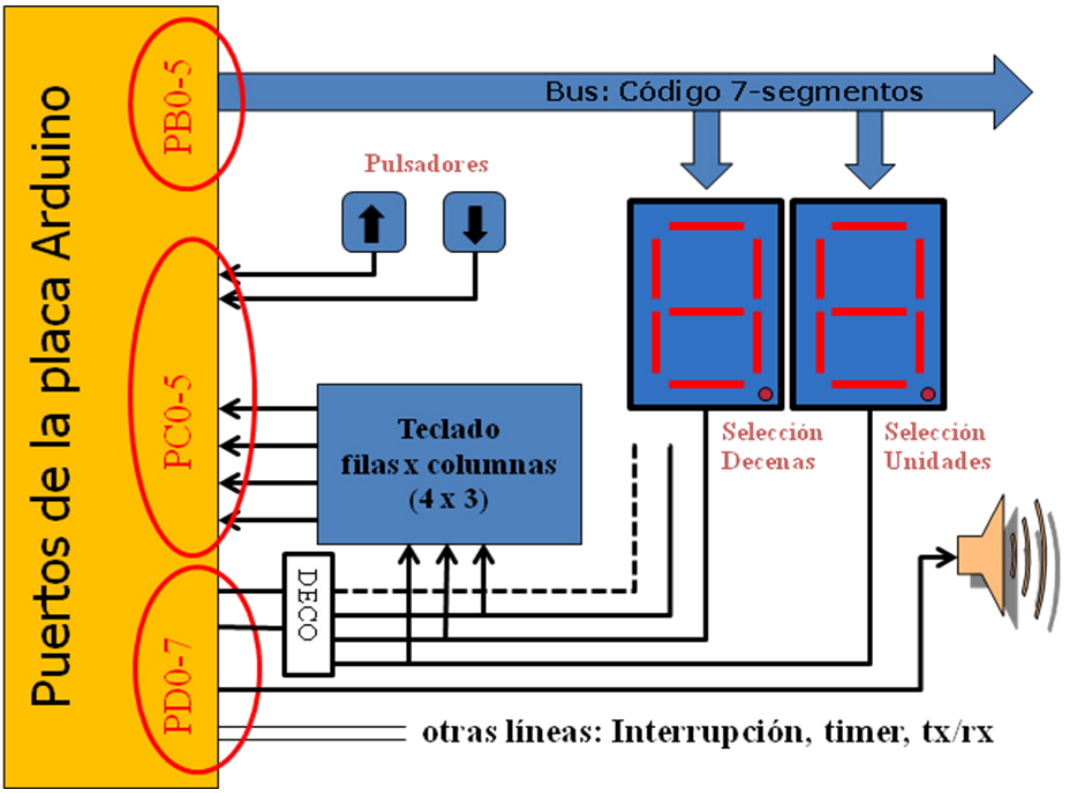


Figura 1. Diagrama de bloques del Turnomatic

Como se puede ver todos los componentes serán controlados a través de los puertos del Arduino. Los pulsadores se utilizarán para modificar el número mostrado en el display: avanzar (up), retroceder (down) o poner a cero (reset) pulsando los dos pulsadores a la vez. Cada vez que se modifique el estado del contador se oirá por el altavoz o zumbador del sistema un sonido de corta duración (beep) de una frecuencia determinada. Las operaciones de vigilancia de los pulsadores y la generación del sonido pueden estar asociadas a una tarea principal en la función loop() que siempre se estará ejecutando. Esta tarea principal será interrumpida cada 10 ms para dar paso a la ejecución de una tarea (rutina de servicio de interrupción) encargada de visualizar el número del turno en el display y de explorar el teclado para la detección de posibles pulsaciones.

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, 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.

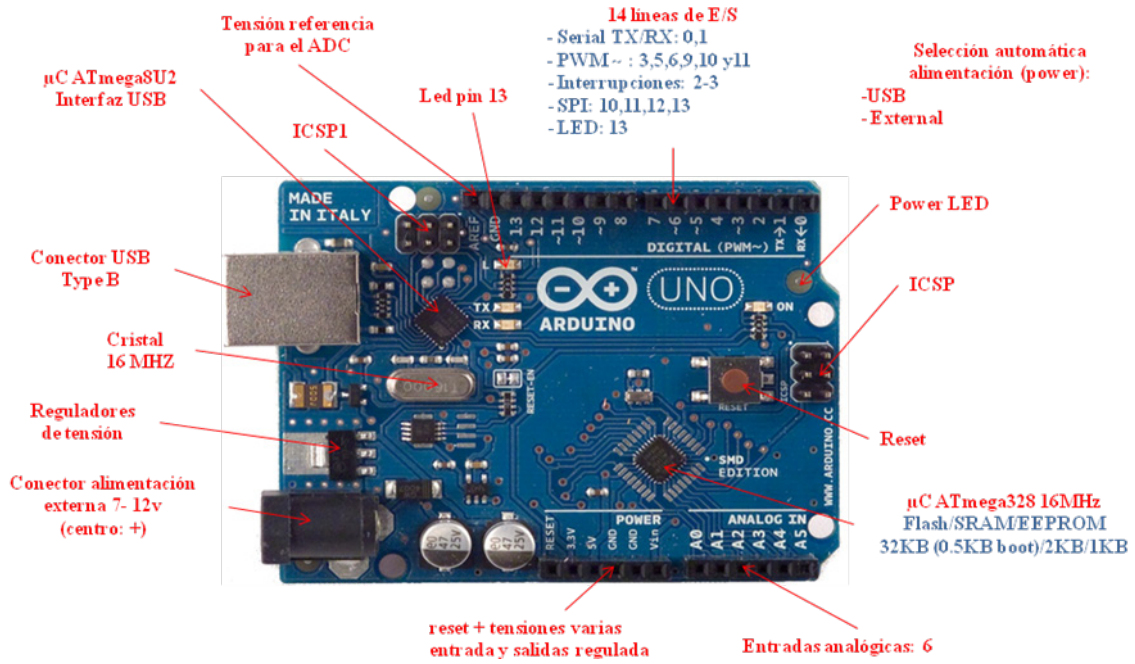


Figura 2. Placa Arduino Uno

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 (n° línea, OUTPUT);
pinMode (n°línea, 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.

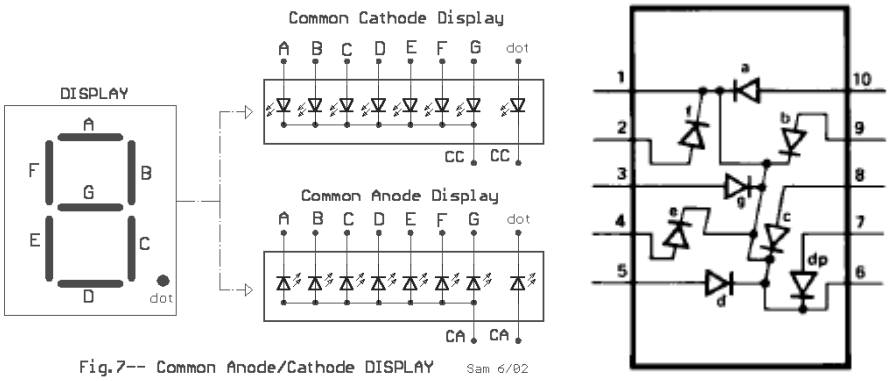


Figura 3. a) Display de 7 segmentos; b) Pines del HP 5082-7740)

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.

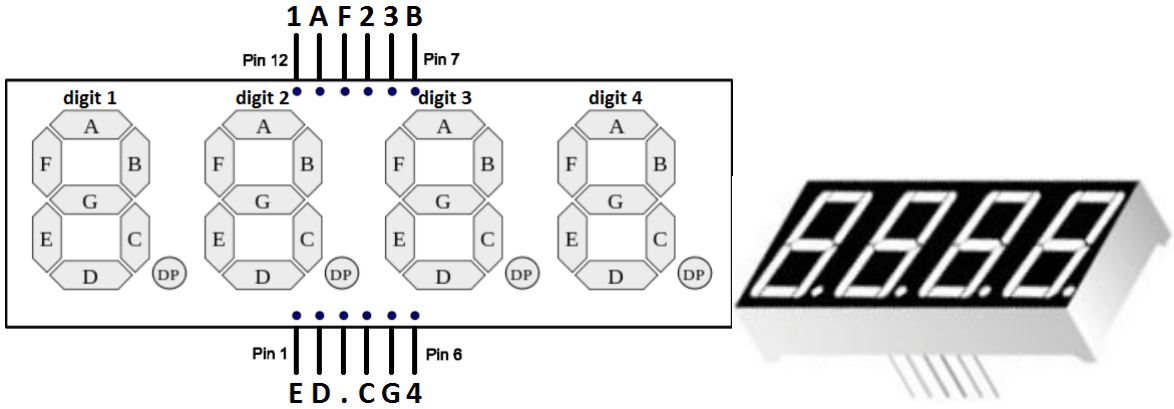


Figura 3. c) Display de 4 dígitos SMA420564

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.

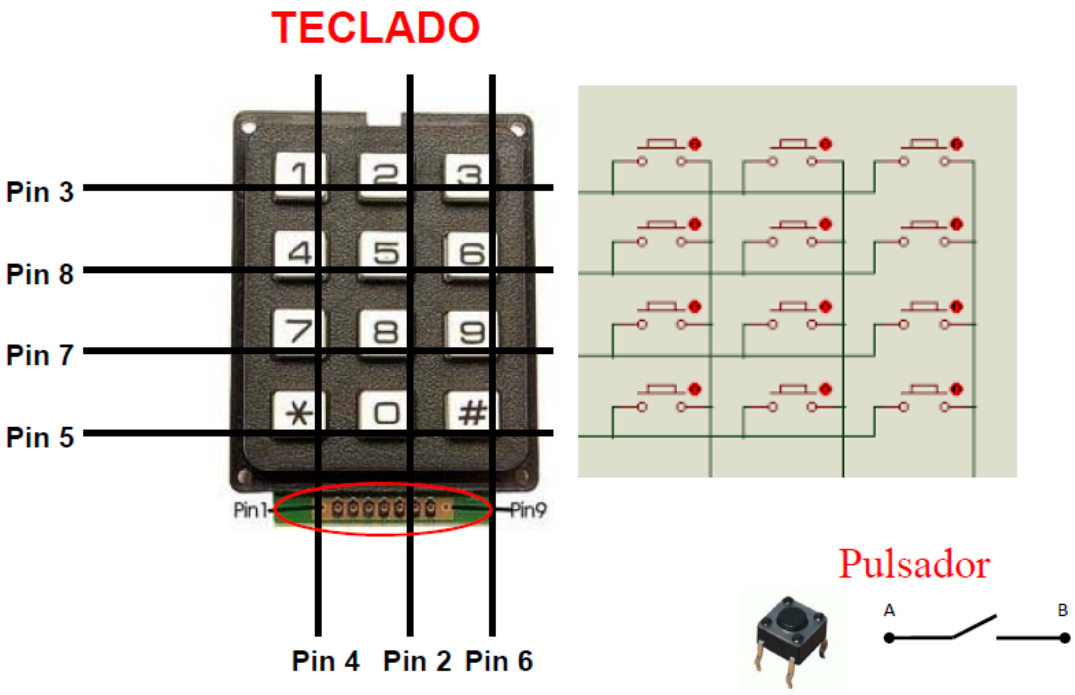


Figura 4. Teclado y pulsador sencillo

3.3.- Esquema general del circuito del "turnomatic"

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 un "turnomatic" verificando su correcto funcionamiento. La figura 5 muestra un esquema de todas las interconexiones entre los diferentes dispositivos del turnomatic que será montado en una placa para el desarrollo de prototipos (breadboard) y conectado a los puertos del Arduino.

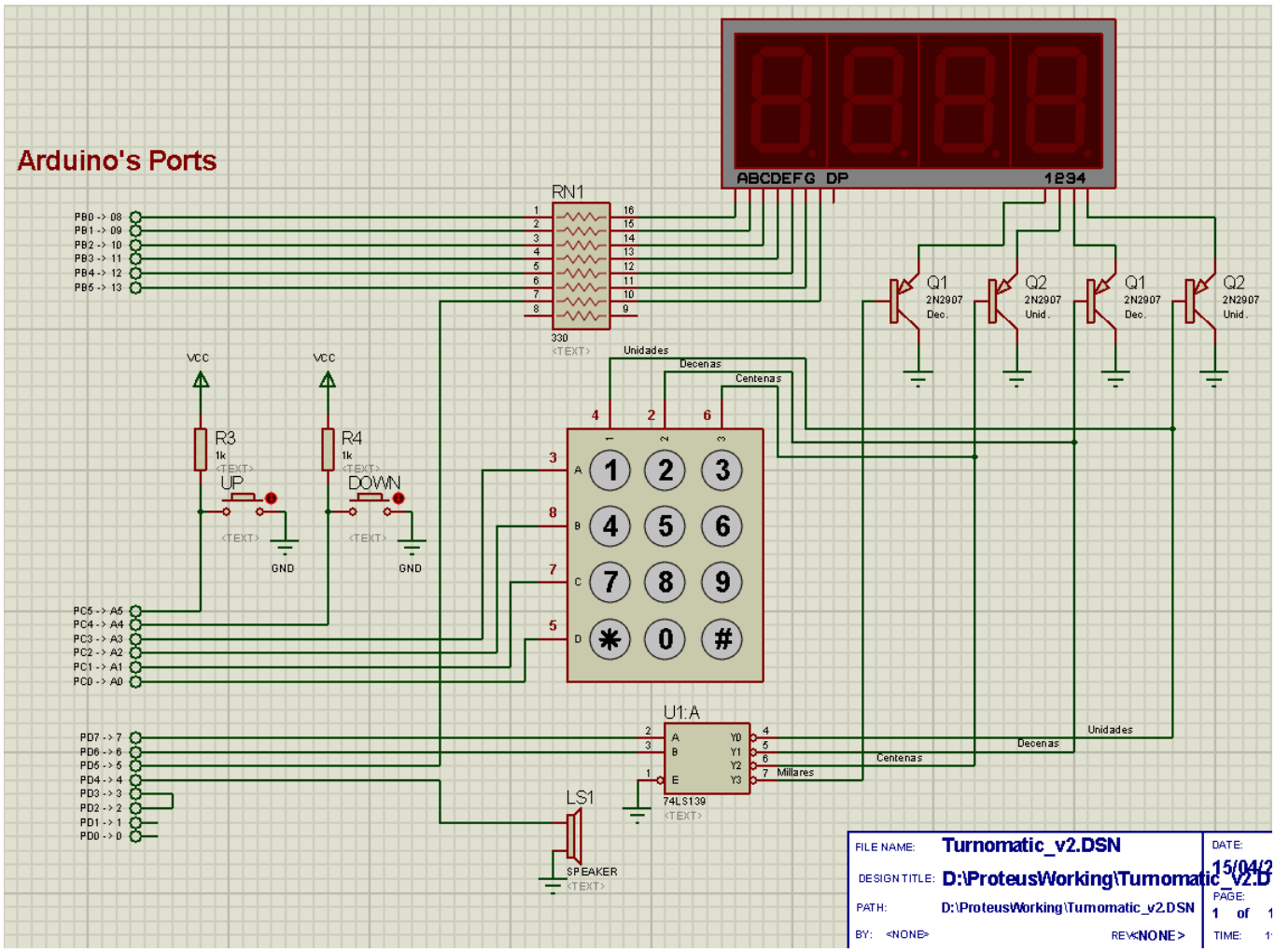
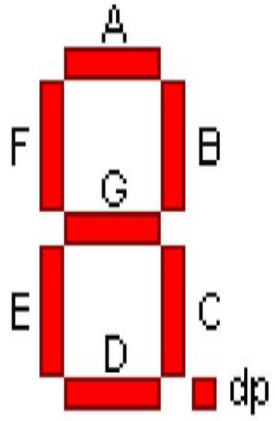


Figura 5.- Esquema del circuito del "Turnomatic"

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 quemen por exceso de corriente). Los dos dígitos del "turnomatic" correspondientes a las decenas-unidades se han conectado al mismo bus de datos. Sin embargo, las conexiones del cátodo común de los displays no son comunes como se puede apreciar en la figura 5. 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 *PD6* del puerto de control. 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 *PD7* la encargada de activar el display.

Para visualizar los dígitos del 0 al 9 en cada uno de los displays será necesario depositar en el bus o puerto de datos 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 displays al bus de datos así como los valores que se han de enviar al puerto 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									

El resto de la tabla se deja como ejercicio para el estudiante.

Una versión de display con cuatro dígitos integrados se muestra a continuación. El principio de funcionamiento es el mismo siendo 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.

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 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 seleccionar la frecuencia del sonido que se deberá oír cuando se modifique el estado del contador del Turnomatic con objeto de avisar a los clientes. La frecuencia será seleccionada pulsando dos teclas según las siguientes secuencias:

*0 → frecuencia base = 200Hz
*1 → frecuencia base + 200 = 400 Hz
*2 → frecuencia base + 400 = 600 Hz
*3 → frecuencia base + 600 = 800 Hz
*4 → frecuencia base + 800 = 1000 Hz
*5 → frecuencia base + 1000 = 1200 Hz
*6 → frecuencia base + 1200 = 1400 Hz

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 ello será necesario activar la líneas de pull-up internas del microcontrolador)

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).`

La función asociada a los pulsadores del turnomatic será: avance (señal A5/PC5), retroceso (A4/PC4) y puesta a cero pulsando los dos a la vez. Es importante destacar que los pulsadores y las filas del teclado se han conectado al PORTC en el que cada uno de sus bits o líneas, por defecto, son entradas analógicas por lo que 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"

```
Val = digitalRead(19); // pulsador UP conectado a A5 (línea 19)
```

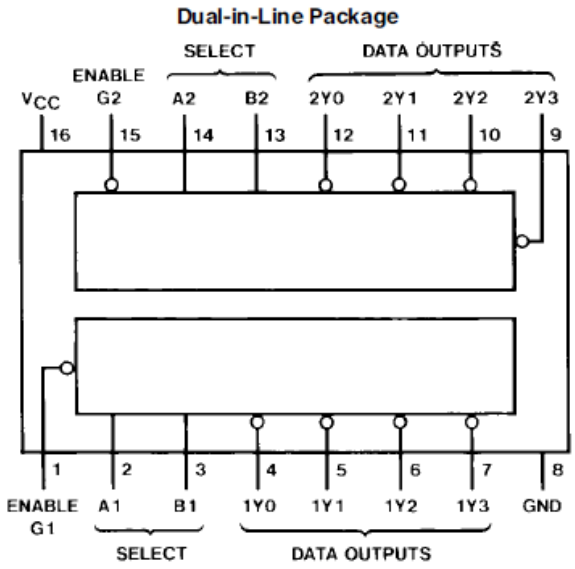
Altavoz

A cada acción de los pulsadores corresponde una señal acústica (beep) emitida por el altavoz del sistema. La señal a emitir ha de ser de una frecuencia determinada por el usuario a través del teclado. Ver el apartado de teclado. Para generar una señal de un tono determinado se podrá hacer uso de la función:

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

Decodificador de 2 a 4

Las líneas de selección de los dígitos del display (unidades y decenas) y de exploración del teclado se obtienen de los puertos PD6 y PD7 del Arduino. Sin embargo, si quisiéramos controlar un display de 4 dígitos así como explorar un teclado de, por ejemplo, 16 teclas (organización de 4x4) necesitaríamos entonces 4 líneas de selección en total: una por cada dígito o columna del teclado. Para ello, tendríamos que recurrir a utilizar más pines del Arduino (en nuestro caso ya no podemos porque están prácticamente todos en uso) o utilizar un decodificador de 2 a 4 de modo que con dos líneas del Arduino podríamos obtener hasta 4 señales de selección. Un posible decodificador sería el chip 74LS139 que tiene dos decodificadores 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

3.4.- Software del "turnomatic"

El software del "turnomatic" 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 de exploración de pulsadores y generación de señales acústicas

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 detectar la pulsación de los pulsadores para en base a ello actualizar el estado del contador interno del "turnomatic" (decenas-unidades). El contador se podrá incrementar (up), decrementar (down) y poner a cero (reset). Cada vez que cambie el contador se ha de oír un tono por el altavoz de corta duración

y una determinada frecuencia. La frecuencia será seleccionada pulsando secuencialmente dos teclas ("*" y un dígito "0-8") pertenecientes a las dos primeras columnas del teclado. La secuencia de teclas a detectar será:

*0 → frecuencia base = 200Hz
*1 → frecuencia base + 200 = 400 Hz
*2 → frecuencia base + 400 = 600 Hz
*3 → frecuencia base + 600 = 800 Hz
*4 → frecuencia base + 800 = 1000 Hz
*5 → frecuencia base + 1000 = 1200 Hz
*6 → frecuencia base + 1200 = 1400 Hz
*7 → frecuencia base + 1400 = 1600 Hz
*8 → frecuencia base + 1600 = 1800 Hz

Tarea 2.- Exploración del display y teclado (rutina de servicio de la interrupción)
El sistema consta de dos displays de 7-seg. para visualizar las decenas y unidades del "turnomatic". Ambos se conectan al puerto PB0-5 a través de unas vías de comunicación (hilos conductores) por donde se suministra el código 7-seg de cada uno de los dígitos. 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 mostrar las decenas y unidades de forma alternativa y con la suficiente rapidez como para que el contador no parpadee.

Las líneas de selección de los displays (activas a nivel bajo o "0") también serán aprovechadas para explorar las columnas del teclado de modo que cuando se visualizan las unidades también se está explorando la primera columna del teclado. A continuación se leerá el estado de las filas para detectar si alguna de ellas está a cero como consecuencia de la pulsación de alguna de las teclas de la primera columna. En caso afirmativo se procederá a codificar la tecla en función de la columna y fila en la que se haya producido la detección.

Esta tarea se ha de activar (ejecutar) cada 10ms o menos para que no se pierda ninguna pulsación realizada por el usuario del teclado y para que el contador no parpadee. Si este periodo de tiempo lo alargamos veremos parpadear el display y se perderán pulsaciones al no explorarse el teclado con la suficiente rapidez.

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 5. Realice los ajustes que estime oportunos.
2.- Probar 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 se detiene la cuenta.
3.- Implementar la tarea 2 y comprobar su funcionamiento conjuntamente con la interrupción.
4.- Implementar la tarea 1 y comprobar el funcionamiento del turnomatic.

Mejoras (optativo) (30%):
Proponga alguna mejora o nueva funcionalidad del turnomatic. Podrá utilizar los 4 dígitos del display.

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.

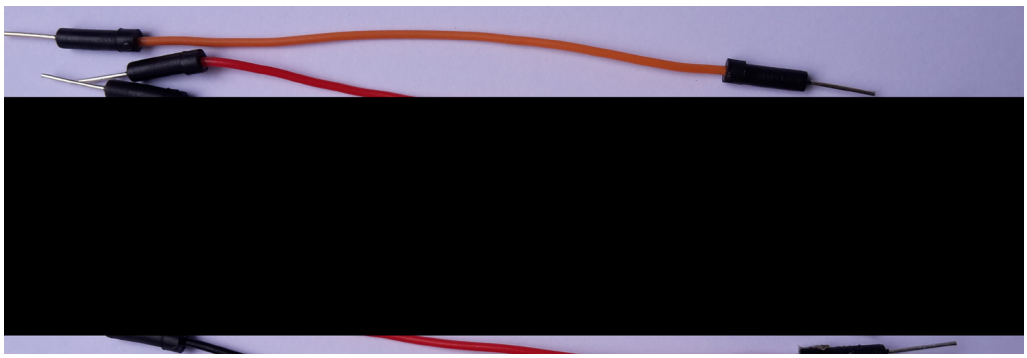


Figura 6.- Esquema del circuito del "turnomatic"

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