ESCUELA DE INGENIERÍA INFORMÁTICA

Grado en Ingeniería Informática

PERIFÉRICOS E INTERFACES

PRÁCTICAS DE LABORATORIO

Módulo 3 – Práctica 3

Programación de interfaces paralelos

Última actualización: 16 abril 2018

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. En concreto, se propone alcanzar los siguientes objetivos:

1. Conocer la estructura interna de un interfaz típico de entrada/salida a través del estudio y manejo de los puertos de una tarjeta Arduino.
2. Poner en práctica los conocimientos básicos sobre las entradas/salidas paralelas en un sistema computador.
3. Conocer y entender la funcionalidad de los diferentes pines de los puertos de E/S del microcontrolador así como los múltiples aspectos relativos a la conexión con el hardware externo.
4. Conocer los aspectos prácticos de funcionamiento de los diferentes componentes básicos a utilizar en la práctica.
5. Entender la integración y conexión de componentes básicos para diseñar periféricos sencillos que cumplan con una funcionalidad determinada.
6. Analizar, diseñar e implementar el software de control del periférico para su correcto funcionamiento en base a la funcionalidad especificada.
7. Verificar y depurar la integración hardware/software realizada para la aplicación hasta alcanzar un funcionamiento satisfactorio y fiable. Rediseñar si fuese necesario.

2 Documentación previa

La documentación básica a utilizar para la realización de esta práctica está disponible en la página web de la asignatura ubicada en el Campus Virtual de la ULPGC. 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 la presentación de la práctica
- 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 desarrollo de software de control de un dispositivo periférico sencillo, con una funcionalidad similar a la de un "Turnomatic" (dispositivo para establecer un turno en la prestación de algún servicio: supermercado, bancos, ...) que estará formado por los siguientes componentes: un elemento visualizador de cuatro dígitos (usaremos un módulo de 4 dígitos del que solo usaremos dos aunque el estudiante podrá hacer uso de más dígitos dentro del apartado de mejoras), dos pulsadores (up/down), un teclado y un altavoz o zumbador. Estos componentes serán controlados a través de los pines de entrada/salida del Arduino que actúa como controlador hardware del dispositivo y cuya funcionalidad queda determinada por el software de control. La funcionalidad básica del dispositivo consiste en poder incrementar, decrementar o poner a cero el número de turno visualizado en el display de 7-segmentos a través de los pulsadores. Asimismo, cada vez que cambie el número del turno habrá de oírse un pitido de una frecuencia determinada. Esta frecuencia podrá ser seleccionada por el usuario, a través del teclado, según determinadas secuencias de pulsaciones descritas más adelante. La figura 1 muestra un diagrama de bloques simplificado del dispositivo a implementar.

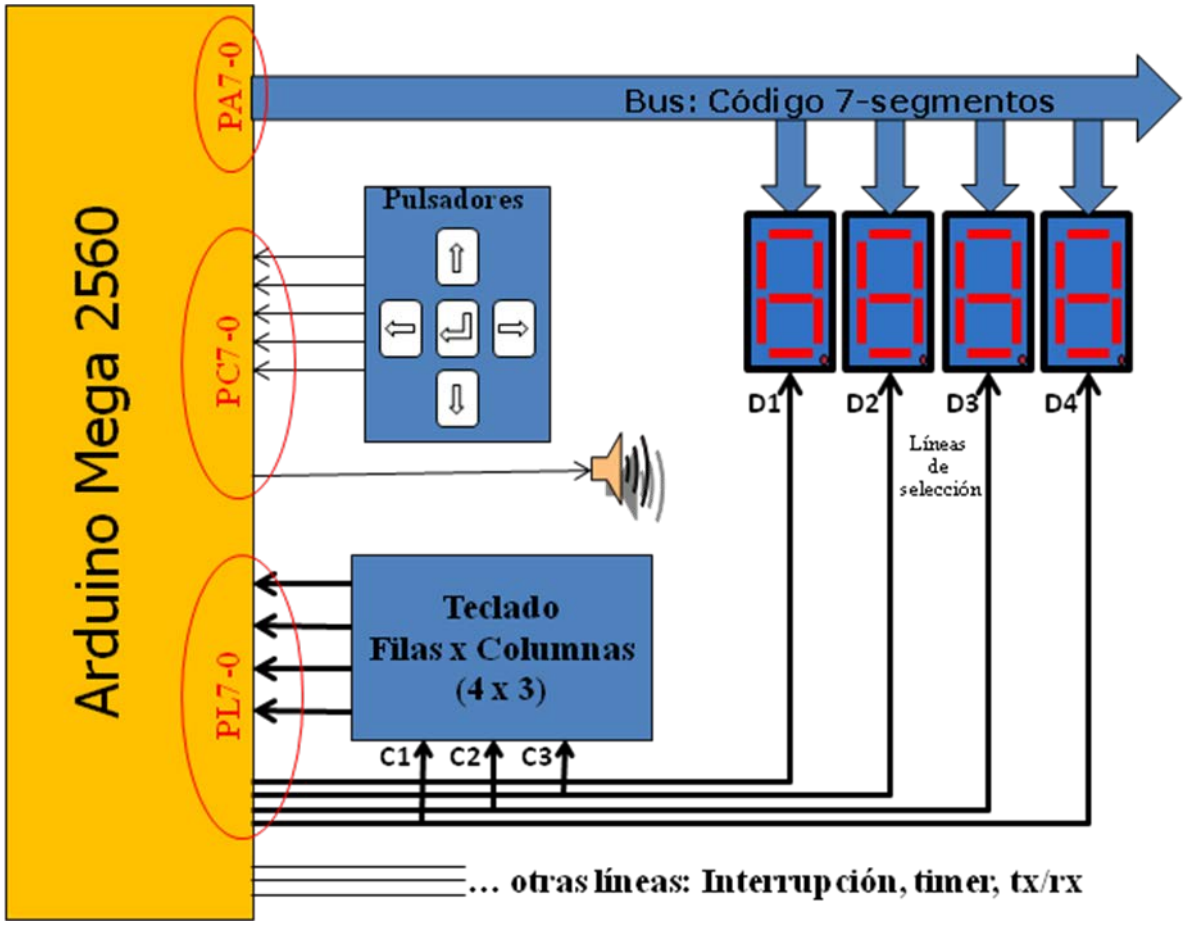


Figura 1. Diagrama de bloques del Turnomatic

Como se puede observar, 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 pulsando los dos pulsadores a la vez (up+down). 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 seleccionable a través del teclado. Por otra parte y de forma repetitiva, se hará un barrido del display para mostrar las unidades y decenas del número del turno y la exploración del teclado para detectar cualquier pulsación. Observe como las unidades y decenas se envían al display por las mismas líneas PA7-0 de forma multiplexada en el tiempo.

3.2.- Componentes básicos

3.2.1.- Tarjeta Arduino Mega 2560

El hardware del "Turnomatic" se conectará a las líneas de entrada/salida de un Arduino Mega 2560 basado en el microcontrolador ATmega2560 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 Mega y en ella podemos apreciar los conectores correspondientes a las líneas o pines de entrada-salida multifuncionales que a su vez mapean en puertos de 8 bits (PA7-0, PB7-0, ... C-D-E-F-G-H-J-K-L). Si bien la mayoría de los pines admiten varias funcionalidades (programables) podemos establecer 4 grandes grupos de pines inicialmente orientados a:

- Comunicaciones seriales: TX0/RX0, TX1/RX1, TX2/RX2, TX3/RX3, SDA/SCL
- Salidas de control de potencia (PWM): pines 2-13
- Entrada/salida digitales: 22-53
- Entradas analógicas: A15-A0

Las líneas analógicas, **A15-A0**, son sólo de entrada y permiten la lectura de señales analógicas realizándose una conversión analógico-digital con 10 bits de resolución (0v --> 0000000000; 5v --> 1111111111). Sin embargo, las líneas analógicas se pueden reprogramar para que su comportamiento sea de tipo digital, de entrada o de salida, (haciendo uso de pinMode() o DDRx) con lo que se aumentaría el número de líneas digitales de entrada/salida.

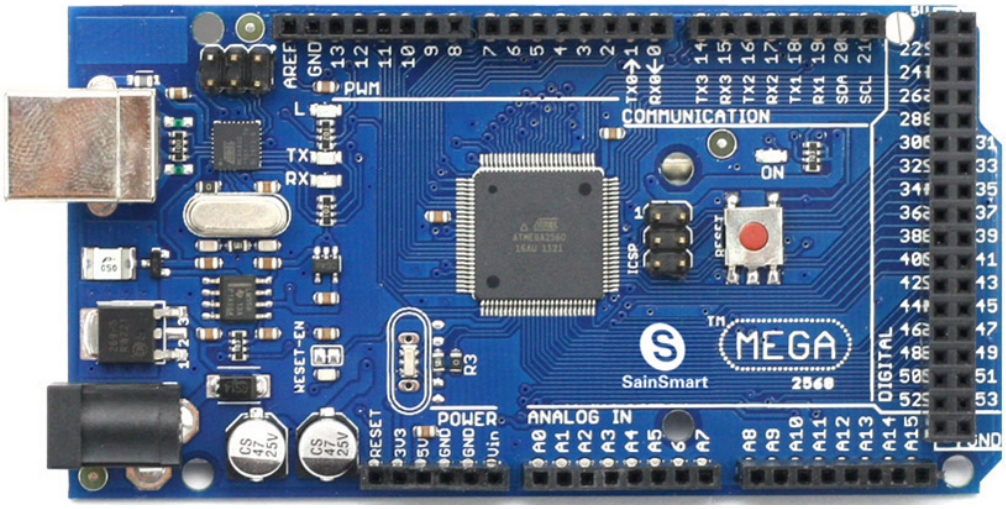


Figura 2. Tarjeta Arduino Mega 2560 (microcontrolador ATmega2560)

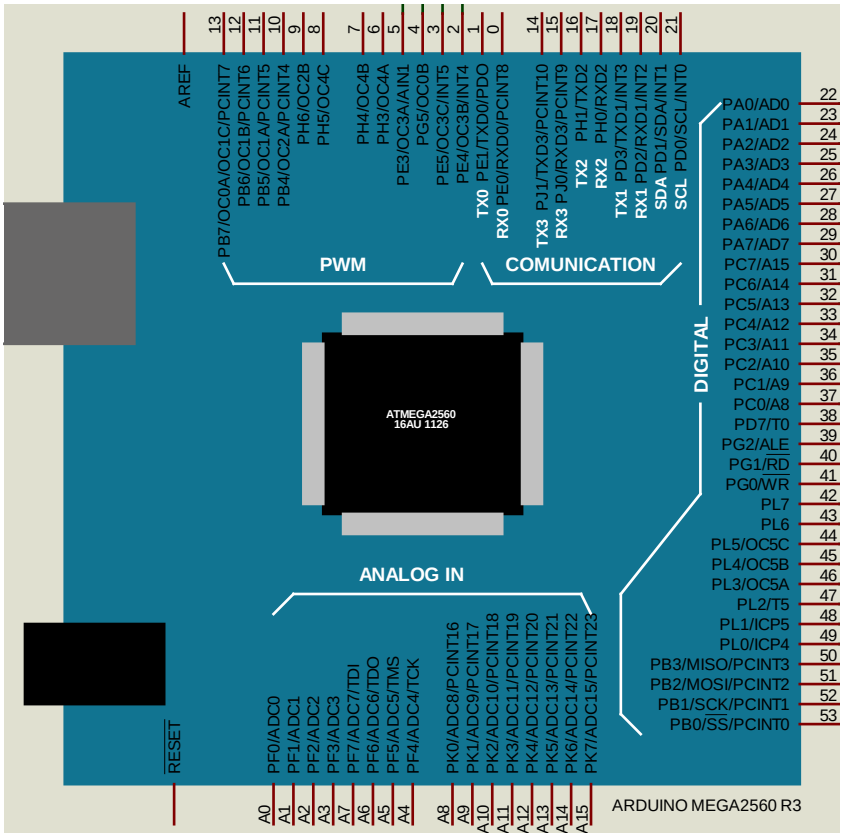


Figura 3. Tarjeta Arduino Mega 2560 (asignación de pines)

Como puede observarse en las figuras 2 y 3 cada línea de entrada-salida tiene un número asignado de modo que particularizando a las que usamos en la práctica tendríamos:

PORTA (PA7-PA0): Pines 29-22
PORTB (PB7-PB0): Pines 13-12-11-10-50-51-52-53
PORTC (PC7-PC0): Pines 30-37
PORTL (PL7-PL0): Pines 42-49
Interrupciones (int 0-1-2-3-4-5): Pines 2-3-21-20-19-18 (ojo: no coincide con la Figura 3!!!)

Programación del modo de funcionamiento de las líneas de entrada/salida digitales

Programar el modo de funcionamiento pin a pin:

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

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

También se pueden programar varias líneas a la vez utilizando el registro de control interno (DDRx) asociado a cada uno de los puertos: DDRA (para el puerto PORTA), DDRB (para el puerto PORTB), DDRC (para el puerto PORTC) y DDRL (para el puerto PORTL). En estos registros, 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 PORTB de salida:

```
DDRB = B11111111; o DDRB = DDRB | B11111111;
```

Programar todas las líneas del puerto PORTB de entrada:

```
DDRB = B00000000;
```

Reprogramar todas las líneas analógicas A7-A0 (PF7-PF0) como líneas digitales de entrada:

```
DDRF = B00000000; nº líneas o pines: 61(A7) ... 54(A0)
```

Nota: Después de un reset del microcontrolador las líneas analógicas quedan por defecto como entradas analógicas siendo necesaria su reprogramación para utilizarlas como líneas digitales de entrada o de salida. El número de línea o pin es correlativo a las digitales (última línea digital: pin 53, -PB0-) por lo que el pin A0 sería la línea 54 y el pin A7 la línea 61.
.

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

Lectura línea a línea (bit a bit o pin a pin):

```
Val = digitalRead(nº línea)
```

Lectura de todas las líneas asociadas a un puerto (8 bits):

```
ValA = PINA;    // lectura del puerto A, PA7-0, pines 29-22
ValC = PINC;
ValL = PINL;
```

Escritura línea a línea (bit a bit):

```
digitalWrite(nº línea, HIGH);
digitalWrite (nº línea, LOW);
```

Escritura de todas las líneas asociadas a un puerto (8 bits):

```
PORTA = 28;
PORTC = val;
PORTL = B00110011;
```

3.2.2.- Display de 7 segmentos

Como dispositivo de visualización utilizaremos displays de 7 segmentos. La Figura 4 muestra la estructura interna de un display 7-segmentos (HP 5082-7740) formada 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 será de cátodo común por lo que para encender los segmentos será necesario aplicar una tensión positiva a la entrada del segmento y conectar a tierra (0 voltios) la pata o pin de cátodo común. 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.

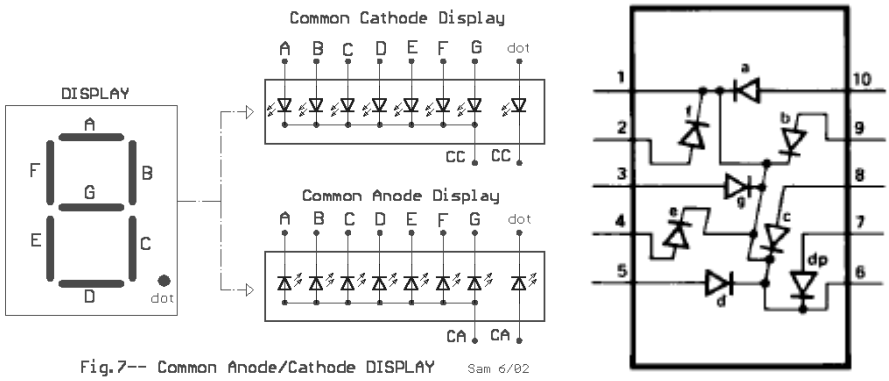


Figura 4. a) Display de 7 segmentos; b) Pines del HP 5082-7740)

También se podrá hacer uso de un display de 4 dígitos con un conexionado interno preestablecido que simplifica enormemente el diseño cuando se necesitan varios dígitos en el display. Esta es la solución adoptada en la tarjeta prototipo utilizada en el laboratorio. En la Figura 4c se muestra un display de cátodo común de 4 dígitos (Digit1, Digit2, Digit3, Digit4), modelo SMA420564, con las entradas de cada segmento (a, b , c ,d e, f, g, dp) y con los pines de selección correspondientes al cátodo común de cada uno de ellos: pin 12 (Digit1), pin 9 (Digit2), pin 8 (Digit3) y pin 6 (Digit4). 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.

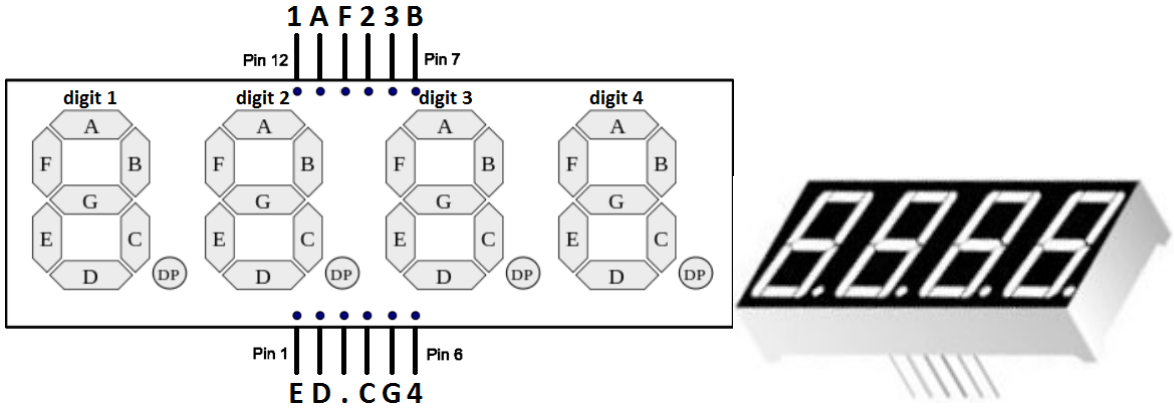


Figura 4. c) Display de 4 dígitos SMA420564

3.2.3.- Pulsadores

Los pulsadores son elementos muy sencillos que cuando se pulsan establecen una conexión entre dos terminales. Se comercializan en distintas calidades en base al número de pulsaciones que son capaces de soportar funcionando correctamente o sin fallos de conexión.

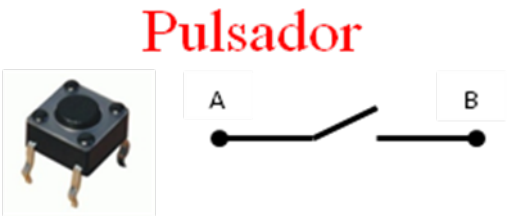


Figura 5. Pulsador

3.2.4.- Teclado

Otro de los componentes a utilizar en esta práctica será un teclado matricial organizado en 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 5 se muestra el teclado que utilizaremos en la práctica así como las señales que podemos encontrar en su conector.

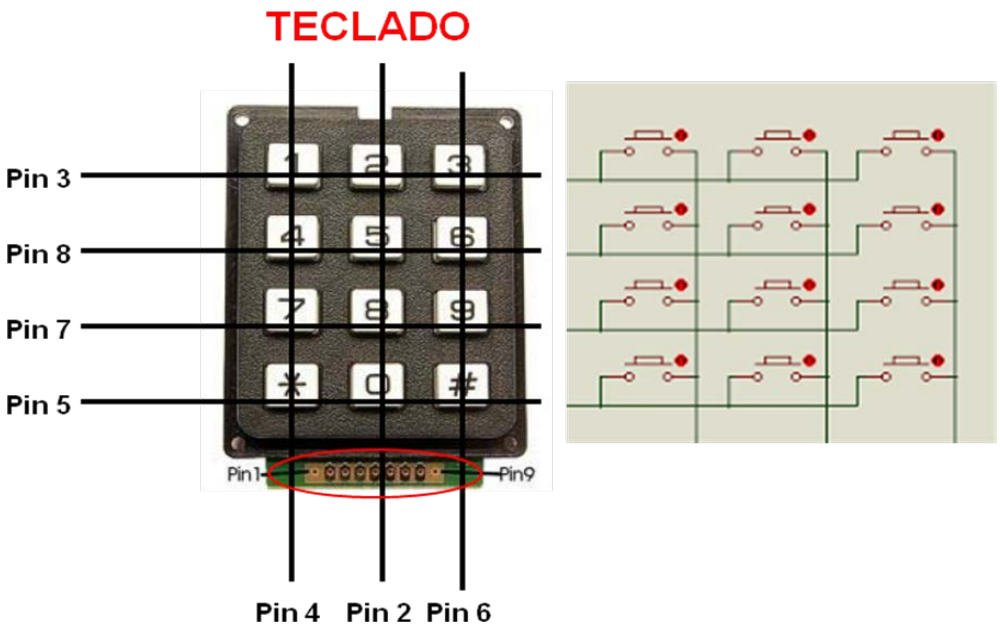


Figura 6. Teclado

3.2.5.- Zumbador y/o altavoz

Como avisador acústico del "Turnomatic" se hará uso de un altavoz o zumbador. Este dispositivo transductor convierte la energía eléctrica en acústica a través de una membrana o elemento similar. Puede generar sonidos de diferentes frecuencias según la frecuencia de la señal eléctrica aplicada.

Figura 7. Altavoz/zumbador

3.2.6.- Otros componentes

Además de los componentes anteriormente citados se utilizan otros que son necesarios para el funcionamiento conjunto del sistema. Básicamente, nos estamos refiriendo a resistencias discretas o en formato array (DIL: Dual In Line), transistores PNP como drivers de los displays, conectores varios y cables de conexionado. Las resistencias se utilizan como limitadores de corriente mientras que los transistores, en este caso, se utilizan como interruptores de paso que pueden manejar corrientes de hasta 500 mA.

La línea de selección de cada uno de los dígitos del display (cátodo común) no debe conectarse para su control directamente a un pin del Arduino ya que la corriente por este pin del cátodo común puede alcanzar los 96-120 mA (suponiendo 12-15 mA por segmento) cuando están todos los segmentos encendidos lo cual es excesivo para un pin del Arduino que según las especificaciones puede trabajar con valores típicos de hasta 20 mA. A partir de los 40 mA se corre el riesgo de quemar el pin o línea. Es aquí donde utilizamos un transistor que actúa como un interruptor controlado por una señal de mando generada por el Arduino. Con ello podremos controlar corrientes de colector (Ic) de 100-200 mA a partir de una señal de mando del orden de 10-20 mA aplicada a la base (Ib). Esta corriente de base es la que tendría que gobernar (dar o recibir) el pin del Arduino que como puede observarse es 10

veces más pequeña que la corriente del terminal de colector o corriente a controlar (proveniente del cátodo común del display 7 segmentos).

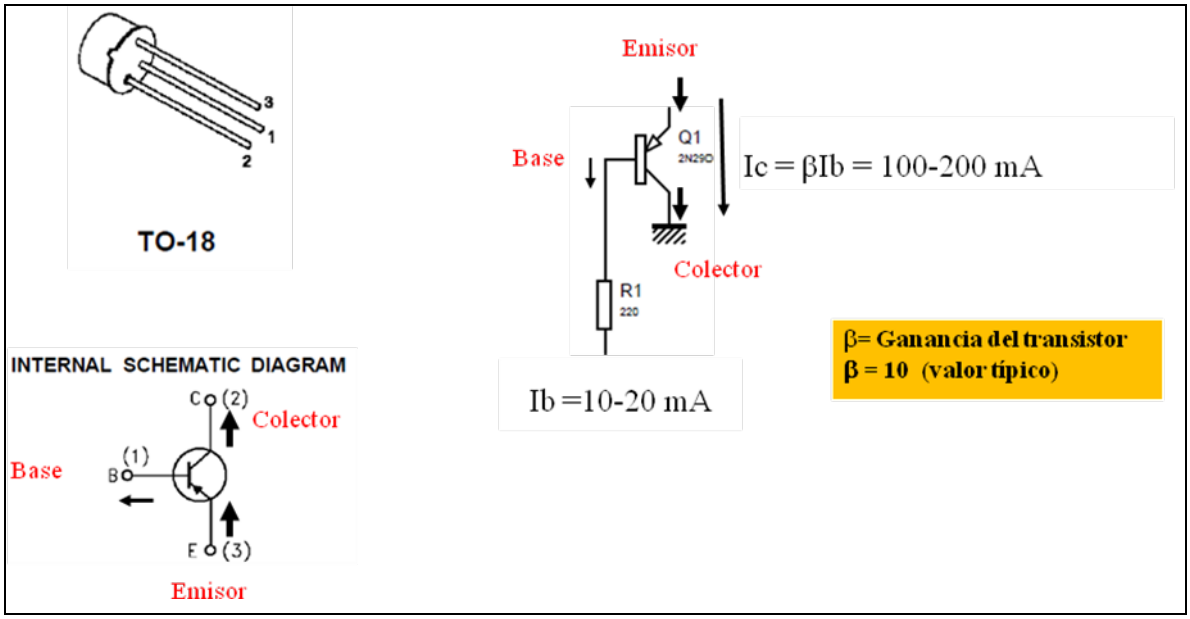


Figura 7. Transistor PNP 2N2907

3.3.- Esquema general de la tarjeta prototipo

Una vez descritos los componentes básicos a utilizar en la práctica vamos a continuar con la descripción de la tarjeta prototipo que es una tarjeta diseñada por el equipo docente de la asignatura que integra, entre otros componentes, los descritos anteriormente y que son suficientes para implementar la funcionalidad del "Turnomatic" u otras aplicaciones. El estudiante habrá de estudiar y analizar esta tarjeta prototipo 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 8 muestra un esquema de todas las interconexiones entre los diferentes componentes de que consta la tarjeta prototipo que se suministra al estudiante para el desarrollo de la práctica. Opcionalmente, el estudiante que lo desee podrá montar su propio prototipo a partir de recursos particulares para lo que podrá hacer uso de las placas de desarrollo de prototipos o breadboards.

La tarjeta dispone de más componentes de los que se necesitan para implementar el "Turnomatic" pero podrán ser utilizados para la implementación de mejoras o para otras prácticas de la asignatura.

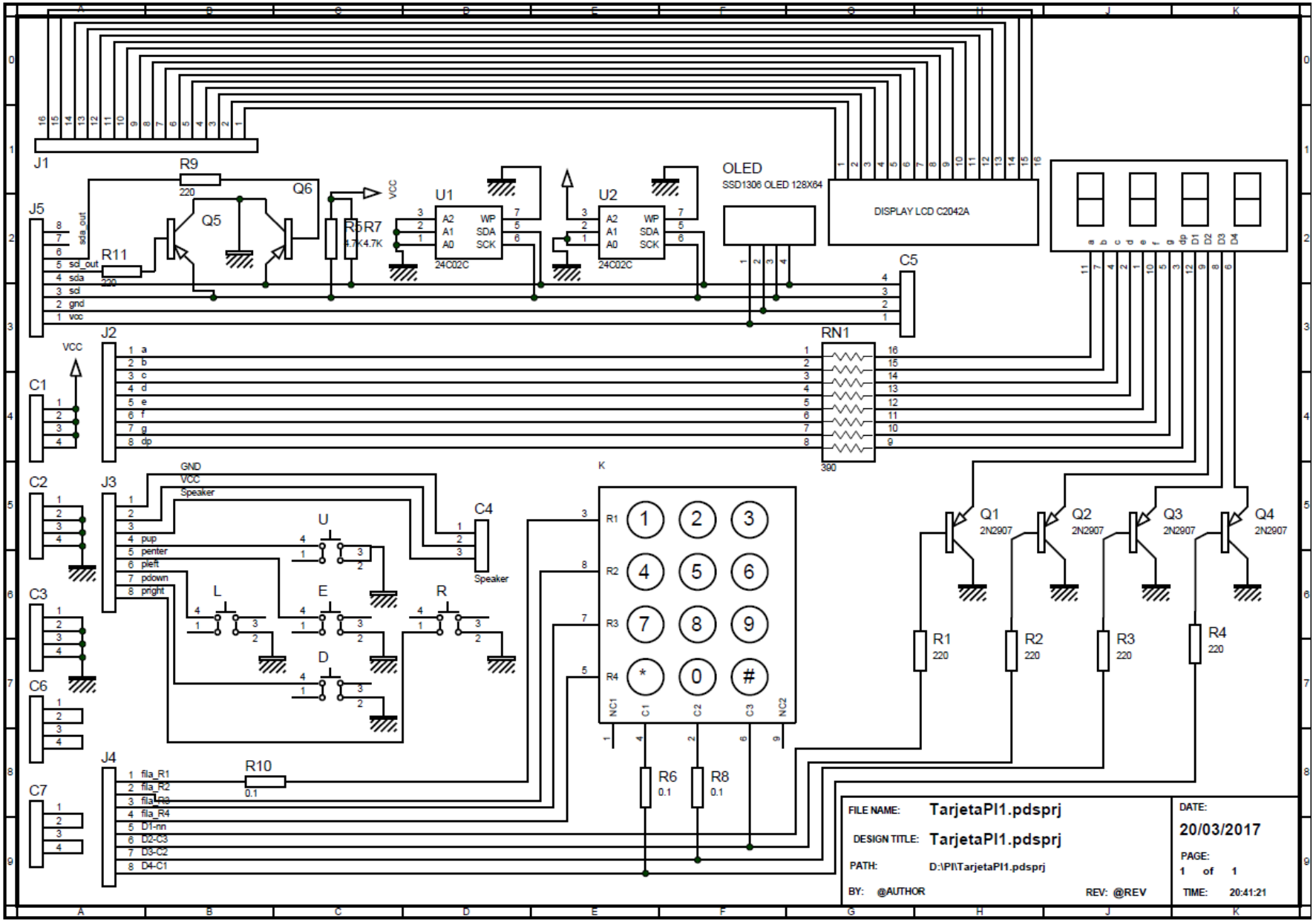


Figura 8.- Diagrama de conexiones en la tarjeta prototipo

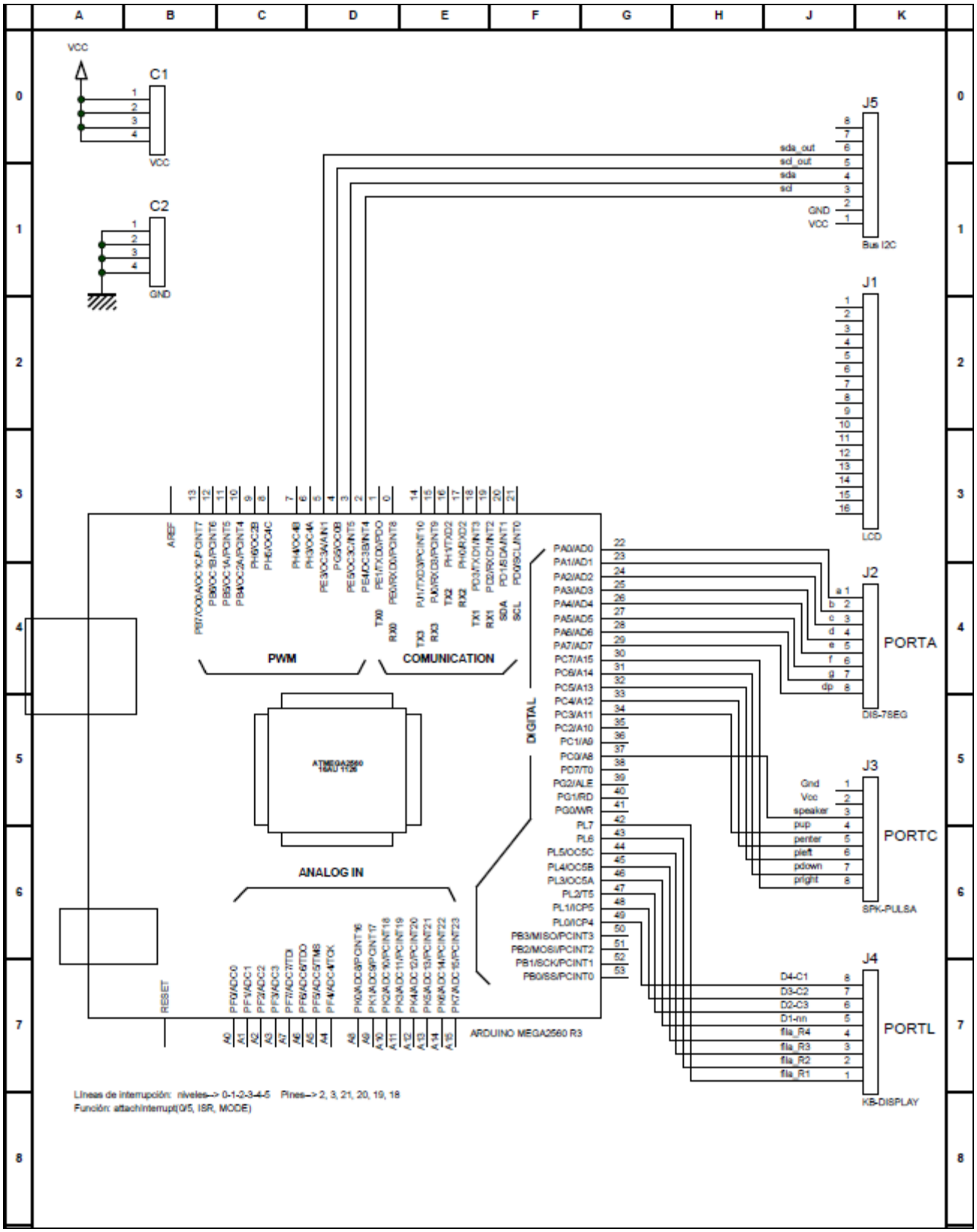


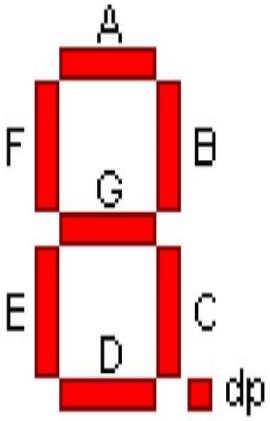
Figura 9.- Interconexiones entre el Arduino Mega 2560 y la tarjeta prototipo

3.4.- Aspectos básicos de implementación

3.4.1.- Display 7-segmentos

El visualizador del diseño básico del Turnomatic consta de solo dos dígitos implementados con displays de 7 segmentos conectados al PORTA (bus 7-seg, salida) del Arduino Mega 2560. Véase como las conexiones de los segmentos al puerto de datos se hace a través de una red de resistencias de 330 ohmios para limitar la corriente que circula por los segmentos a efectos de protegerlos (para 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 7-seg. Sin embargo, las conexiones del cátodo común de los displays no son comunes como se puede apreciar en la Figura 8. 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 *D4-C1* (pin 49, PL0) 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 *D3-C2* (pin 48, PL1) la encargada de activar el display. Aún quedan dos señales de control más, D2-C3 (pin 47, PL2) y D1-nn (pin46, PL3), para el control de los dígitos de las centenas y unidades de millar. Estos dígitos no se contemplan en la realización de la práctica básica pero pueden ser objeto de uso en la realización de mejoras para enriquecer el Turnomatic o implementar alguna otra funcionalidad.

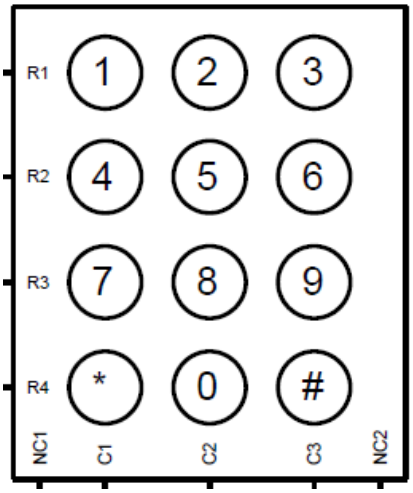
Para visualizar los dígitos del 0 al 9 será necesario depositar en el bus 7-seg. (PORTA) la combinación de "unos y "ceros" adecuada (código de 7 segmentos) para formar el dígito deseado. La siguiente tabla muestra la correspondencia entre los segmentos del display y los pines del PORTA así como los valores a enviar al puerto para visualizar los dígitos decimales 0, 1 y 2. El resto de la tabla se deja como ejercicio para el estudiante.



	Puertos Arduino								**Valor**
bit →	**PA7**	**PA6**	**PA5**	**PA4**	**PA3**	**PA2**	**PA1**	**PA0**	
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									

3.4.2.- Teclado

El teclado, de 4 filas por 3 columnas, será explorado síncronamente con el display de 7-segmentos de modo que cada vez se seleccione un dígito para visualizarlo se seleccionará una columna del teclado. 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 "fila_R1...fila_R4", pines PL7 ... PL4 del Arduino (PORTL) para detectar si en alguna de ellas hay un "cero". Para ello, se habrá de programar las entradas PL7-PL4 como entradas digitales, se conectarán las resistencias de pull-up internas (ver apartado de pulsadores de cómo conectar las resistencias de pull-up) y se referenciarán como las entradas digitales: 42-43-44-45. Éstas se podrán leer línea a línea o en conjunto con la función "PINL". Cuando se detecte una pulsación se codificará la tecla con el código que corresponda (ASCII, por ejemplo) y se almacenará en memoria para su posterior uso.



Exploración de Columna	R1 PL7	R2 PL6	R3 PL5	R4 PL4	Tecla Pulsada
D4-C1 = 0	0	1	1	1	**1**
	1	0	1	1	**4**
	1	1	0	1	**7**
	1	1	1	0	*
D3-C2 = 0	0	1	1	1	**2**
	1	0	1	1	**5**
	1	1	0	1	**8**
	1	1	1	0	**0**
D2-C3 = 0	0	1	1	1	**3**
	1	0	1	1	**6**
	1	1	0	1	**9**
	1	1	1	0	#

PORTL			
Nº pin	**Puerto**	**Nombre señal**	**Función**
42	PL7-in	fila_R1	Leer fila R1 del teclado
43	PL6-in	fila_R2	Leer fila R2 del teclado
44	PL5-in	fila_R3	Leer fila R3 del teclado
45	PL4-in	fila_R4	Leer fila R4 del teclado
46	PL3-out	D1-nn	activar D1
47	PL2-out	D2-C3	activar D2 + col. 3 teclado
48	PL1-out	D3-C2	activar D3 + col. 2 teclado
49	PL0-out	D4-C1	activar D4 + col. 1 teclado

Nota: Orden de los dígitos del display --> D1-D2-D3-D4 (menos significativo)

El teclado se utilizará para seleccionar la frecuencia del sonido que se emite por el zumbador cuando cambia el número o cuenta del Turnomatic. La frecuencia será seleccionada mediante la pulsación de una secuencia de dos teclas de acuerdo a lo siguiente:

*0 → = silencio (no se genera sonido cuando cambia el número del Turnomatic)
*1 → = 200 Hz
*2 → = 400 Hz
*3 → = 600 Hz
*4 → = 800 Hz
*5 → = 1000 Hz
*6 → = 1200 Hz
*7 → = 1400 Hz
*8 → = 1600 Hz
*9 → = 1800 Hz

Opcionalmente, podrán utilizarse otras frecuencias como por ejemplo las que correspondas a notas musicales.

3.4.3.- Pulsadores

La tarjeta prototipo dispone de 5 pulsadores aunque solo se utilizan dos en la realización de la práctica básica del Turnomatic: pup y pdown. Los pulsadores son conectados a puertos de entrada del Arduino (PC7-PC3) de forma que cuando se pulsan colocan un “0” en la línea de entrada y un "1" cuando no están pulsados. **Será necesario activar las resistencias de pull-up internas del Arduino para definir el estado HIGH de un pulsador cuando éste no esté pulsado.**

La conexión de los pulsadores a los pines de PORTC del Arduino, responde a la siguiente tabla:

PORTC			
Nº pin	**Puerto**	**Nombre señal**	**Función**
30	PC7-in	pright	Derecha
31	PC6-in	pdown	Abajo
32	PC5-in	pleft	Izquierda
33	PC4-in	penter	Enter
34	PC3-in	pup	Arriba
35	PC2	-	
36	PC1	-	
37	PC0-out	speaker	Altavoz

Para activar las resistencias de pull-up asociadas a los puertos de entrada 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.

Como es necesario conectar las resistencias de pull-up de las líneas de entrada donde estén los pulsadores, habrá que introducir en el setup() del sketch del Turnomatic las siguientes líneas de código:

```
DDRC = B00000001; // Definimos el PORTC de entrada salvo PC0 (salida)
PORTC= B11111000; // activamos el pull-up interno de las líneas entrada PC7-PC3
```

Finalmente, la función asociada a los pulsadores del Turnomatic en su versión básica (solo dos pulsadores) será la siguiente: avance (pup), retroceso (pdown) y puesta a cero pulsando los dos a la vez (pup + pdown). Dentro del apartado de mejoras el estudiante podrá añadir nuevas funcionalidades haciendo uso de todos los pulsadores.

3.4.4.- 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)
```

3.5.- 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);
}
void function_1() { … }
void function_2() { … }

// 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();
function_2();
.....
}
```

El software del "Turnomatic" se organizará en base a las dos tareas básicas o módulos de software que se describen a continuación:

Tarea 1: Tarea principal

Exploración de pulsadores, actualización contador y generación de señales acústicas

La tarea principal de la aplicación estará ubicada en la función loop() del programa y su función será 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 (pulsando pup), decrementar (pulsando pdown) o poner a cero (pulsando simultáneamente pup y pdown). 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 mediante una secuencia de dos teclas tal y como se explicó en el apartado del teclado.

Por otra parte, esta tarea principal será interrumpida cada 10 ms por una señal de petición de interrupción (onda cuadrada, 100 Hz) generada por un pin del Arduino con la función tone() o bien haciendo uso de algún timer de los que dispone el Arduino Mega 2560 (timer0,...,-timer5). La interrupción será atendida mediante una rutina de servicio de interrupción cuya funcionalidad será la de explorar el teclado y display a un ritmo constante fijado por la interrupción.

Tarea 2.- Rutina de servicio de interrupción

Exploración del display y teclado

El sistema consta de dos dígitos 7-seg. para visualizar las decenas y unidades del "Turnomatic". En cada instante sólo habrá un dígito activo (decenas o unidades) que será seleccionado con su correspondiente línea de selección. Cuando se selecciona un dígito los segmentos se encienden de acuerdo a la información que esté en el bus 7-seg (PORTA). Así, por ejemplo, para visualizar las unidades se habrá de inhabilitar ambos dígitos, colocar la información de las unidades (codificada en 7 seg.) en el PORTA, y luego seleccionar el dígito de unidades. Para mostrar el contador completo, con sus dos dígitos, bastará con alternar la visualización de las decenas y unidades al ritmo suficiente para que el usuario no detecte parpadeos (por ejemplo, cada 10 ms).

Como se comentó en el apartado de la tarea principal, la alternancia puede ser fijada mediante una interrupción que se produce cada 10 ms a través del **pin 21** (interrupción de **nivel 2**). Esta interrupción puede ser generada por algún pin del Arduino (pin20, por ejemplo) mediante la función tone() que permite generar una señal por un pin de la

frecuencia que se desee. El inconveniente que tiene este método es que si se necesita generar otra señal por algún otro pin (para el altavoz, por ejemplo) hay que suspender la petición de interrupción para poder generar la señal de altavoz por el nuevo pin. Por ello, es mejor solución hacer uso de un timer interno para generar una interrupción y dejar la función tone() solo para el altavoz .

La rutina de servicio de la interrupción determinará, en base al valor de alguna variable, la información que se ha de visualizar (unidades o decenas). Así, por ejemplo, si programamos que se genere una interrupción cada segundo, se deberían ver las unidades durante un segundo, y luego, en la siguiente interrupción, conmutaríamos a la visualización de las decenas y, así, sucesivamente. De este modo, veríamos la cuenta del contador con parpadeo: unidades-decenas-unidades-decenas a intervalos de 1 segundo. Para evitar este parpadeo, molesto, aumentaríamos el ritmo al que se muestra las unidades y las decenas del contador para lo que es necesario aumentar la frecuencia de la señal que genera las interrupciones. Como ya hemos dicho, con una frecuencia de 100 Hz (10ms de periodo) sería más que suficiente para "engañar" al sistema visual humano que percibiría la información como si se visualizaran las unidades y las decenas del contador a la vez y sin parpadeos.

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 instante que se aprovecha para leer el estado de las filas y 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.

En resumen, esta tarea 2 (o rutina de servicio de la interrupción de nivel 2 por el pin 21) 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 visualizador no parpadee. Si este periodo de tiempo lo alargamos veremos parpadear los displays 7-seg del visualizador y, posiblemente, se perderá alguna que otra pulsación al no explorarse el teclado con la suficiente rapidez.

Generación de una petición de interrupción

La petición de interrupción se podrá realizar a través de una interrupción externa o una interrupción interna. Para la interrupción externa necesitaremos una señal que interrumpa y para la interrupción interna utilizaremos un timer que cuando alcanza una determinada cuenta interrumpe.

Método 1: Interrupción externa

Para generar por el pin 20 la señal que interrumpe usaremos la función tone(). La señal de petición de interrupción se generará por la línea o pin 20 mediante la función:

```
tone(nº línea, frecuencia)
Ejemplo: tone(20,100)
```

Esta función genera una señal de onda cuadrada de frecuencia 100 Hz (10 ms periodo) por el pin 20 del Arduino. Es necesario establecer un puente entre los pines 20-21 para que la señal generada por el pin 20 genere una petición de interrupción por el pin 21, de nivel 2. La siguiente tabla muestra los niveles de interrupciones y los pines asociados:

Nivel:	int0	int1	int2	int3	int4	int5
Pin:	2	3	21	20	19	18

Por último, es necesario indicar al Arduino que rutina de servicio se ha de ejecutar cuando se interrumpe por un determinado nivel. Las funciones disponibles para ello son:

A) Activar/conectar una interrupción:

```
attachInterrupt(nivel_de_interrupción, rutina de servicio, modo)
Ejemplo: attachInterrupt(2, tarea2, FALLING)
```

B) Desactivar/desconectar una interrupción:

```
detachInterrupt(nivel_de_interrupción)
Ejemplo: detachInterrupt(2)
```

D) Habilitar interrupciones

```
interrupts()
sei()
```

C) Deshabilitar interrupciones

```
noInterrupts()
cli()
```

Método 2: Interrupción interna

Haremos uso del timer1 de 16 bits para generar una interrupción. Cuando el registro de cuenta del timer1, TCNT1, alcance el valor programado en el registro OCR1A se generará una interrupción y el registro TCNT1 se pondrá a cero para iniciar un nuevo ciclo.

Código para programar el timer1 para que interrumpa cada 10 ms (100Hz):

```
void setup(){
..........
..........
// ************************************************************
// disable interrupts
  cli();
  // modo normal de funcionamiento
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0; // cuenta inicial a cero
  // mode CTC
  TCCR1B |= (1<<WGM12);
  // prescaler  N = 1024
  TCCR1B |= (1<<CS12)|(1<<CS10);
  // fintr = fclk/(N*(OCR1A+1)) --> OCR1A = [fclk/(N*fintr)] - 1
  // para fintr = 100Hz --> OCR1A = [16*10^6/(1024*100)] -1 = 155,25 --> 155

  OCR1A = 155; // para 200 Hz  programar OCR1A = 77

  // enable timer1 compare interrupt
  TIMSK1 |=(1<<OCIE1A);
  // habilitamos interrupciones
  sei();
// ************************************************************
}
```

Una vez programado el timer1 es necesario especificar la rutina de servicio que se ha de ejecutar cuando interrumpa el timer1 para la cuenta OCR1A. Para ello, usaremos la función ISR:

```
ISR(TIMER1_COMPA_vect){

// Aquí va la rutina de servicio a ejecutar: barrido teclado-display
// Se ejecuta cada 10ms (100Hz) o 5 ms (200 Hz) según programación

}
```

4 Realización práctica

Las tareas a desarrollar serán las siguientes:

Realización práctica básica:

1. Revisar el montaje del circuito comprobando todas las interconexiones de acuerdo al esquema del circuito de las Figuras 8 y 9. Realizar 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. Implementar un contador de dos dígitos que cuente de 00 a 99, usando los pulsadores (pup-pdown) para proporcionar la siguiente funcionalidad: Comenzar la cuenta (pulsar pup), detener la cuenta (pulsar pdown) y poner a cero el contador (pulsar pup y pdown, simultáneamente).
4. Implementar la tarea 2 y comprobar su funcionamiento conjuntamente con la interrupción.
5. Implementar la tarea 1 y comprobar el funcionamiento global del Turnomatic.

Realización práctica mejorada:

Proponga alguna mejora o nueva funcionalidad del Turnomatic. Podrá utilizar los 4 dígitos del display, los cinco pulsadores y algún otro dispositivo disponible en la tarjeta de desarrollo.

Para realizar las conexiones utilice, exclusivamente, **cables especiales suministrados** para tal fin como los mostrados en la figura 10. Otros cables pueden partirse y quedar dentro de los conectores de la placa Arduino inutilizando el pin afectado.

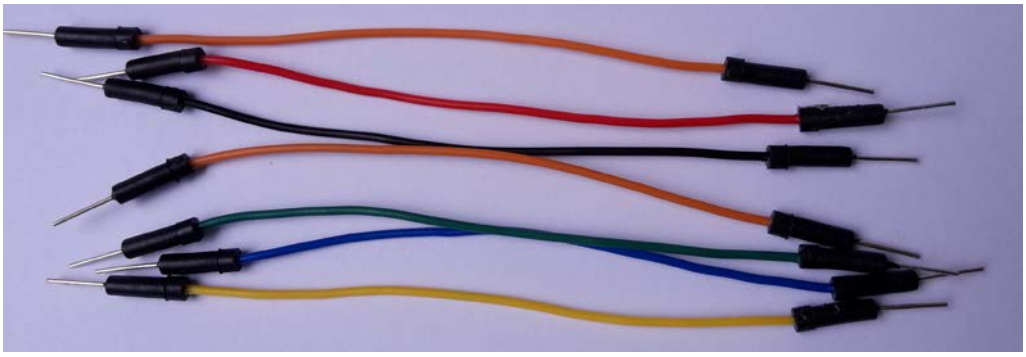


Figura 10.- Esquema del circuito del "Uurnomatic"

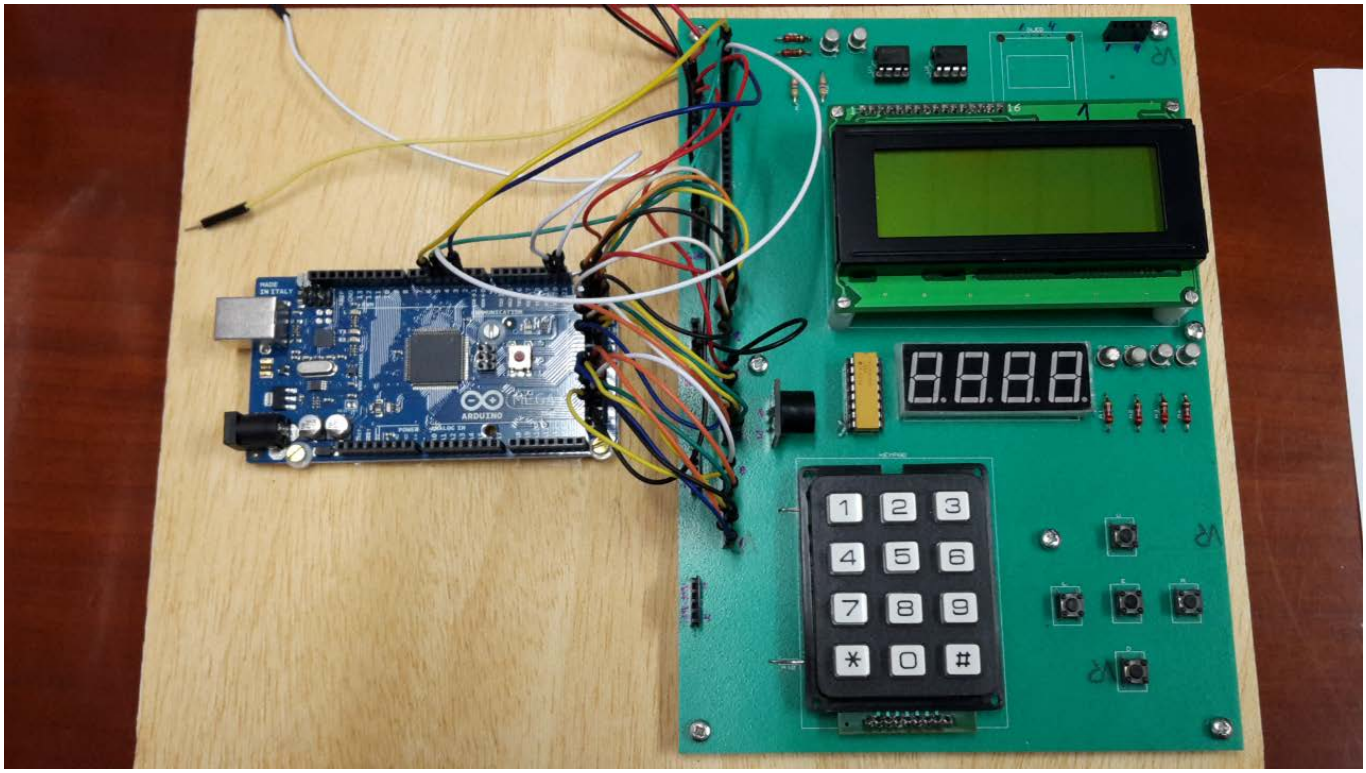
5 Defensa y entrega de la memoria

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.

ANEXO

Fotos de los montajes de las prácticas

A) Diseño con el Arduino Mega 2560 y tarjetaPI (a partir del curso 2016-2017)



B) Diseño con el Arduino Uno Rev. 3 (cursos anteriores)