B2 – Introducción y presentación de información

Objetivo

El objetivo de esta sesión de práctica es aprender a realizar programaciones adecuadas para (i) manejar teclados matriciales y pantallas LCD y (ii) ejecutar bucles repetitivos.

Variables de tipo *byte*

Una variable de tipo "**byte**" se codifica mediante 8 bits. Puede tomar 2^8=256 valores (enteros): los comprendidos entre 0 y 255. Los valores de una variable "byte" no pueden ser negativos.

Variables de tipo *char*

El valor que puede tomar una variable de tipo "**char**" es siempre un solo carácter (una letra, un dígito, un signo de puntuación...). Para asignar explícitamente a una variable de tipo "char" un determinado valor, es decir, un carácter, debemos escribir ese carácter *entre comillas simples*.

Por ejemplo:

char mivariable='A';

Bloque repetitivo "*for*"

El bucle "*for*" se usa para ejecutar un conjunto de instrucciones un número concreto de veces. La sintaxis general del bucle "for" es:

```
for (valor_inicial_contador;condicion_final;incremento){
  // Instrucciones que se repetirán un número determinado de veces
}
```

Entre paréntesis se deben escribir tres partes diferentes, separadas por punto y coma. Estas tres partes son opcionales (se pueden omitir cualesquiera de ellas).

Valor inicial del contador: En esta parte se asigna el valor inicial de una variable entera que se utilizará como contador en las iteraciones del bucle.

Condición final del bucle: En esta parte, se especifica una condición. Justo antes de cada iteración, se comprueba que sea cierta, para pasar a ejecutar el grupo de sentencias internas. Si la condición se evalúa como falsa, se finaliza el bucle "for", continuando el programa tras su llave de cierre.

Incremento del contador: En la última de las tres partes, es donde se indica el cambio de valor que sufrirá, al inicio de cada iteración del bucle, la variable usada como contador. Este cambio se expresa con una asignación.

Bloques condicionales "*if*" y "*if/else*"

Un bloque "*if*" sirve para comprobar si una condición determinada es cierta ("true", 1) o falsa ("false", 0). **Si la condición es cierta**, se ejecutarán las instrucciones escritas en el interior del bloque "*if*" (es decir, dentro de las llaves de apertura y cierre). **Si la condición es falsa**, puede no pasar nada, o bien, si existe tras el bloque "*if*" un bloque "*else*" (opcional), se ejecutarán las instrucciones escritas en el interior del bloque "*else*".

Si solo escribimos el bloque "*if*", el sketch tendrá respuesta solamente si se cumple la condición. **Si además del bloque "*if*", escribimos un bloque "*else*"**, el sketch tendrá respuesta si se cumple la condición y también si no se cumple la condición.

En general, la sintaxis del bloque "*if/else*" es:

```
if (condición) {
    //Instrucciones –una o más– que se ejecutan si la condición es cierta –"true",1–
} else {
    //Instrucciones –una o más– que se ejecutan si la condición es falsa –"false",0–
}
```

Tanto si se ejecutan las sentencias del bloque "*if*" como si se ejecutan las sentencias del bloque "*else*", cuando se llega a la última línea de esa sección (una u otra), se salta a la línea inmediatamente posterior a su llave de cierre, para continuar desde allí la ejecución del programa.

Funciones propias

Una *función* es un trozo de código al que se identifica con un nombre. Escribiendo el nombre de una función en el lugar deseado de nuestro sketch, se puede ejecutar todo el código incluido dentro de ella.

Para crear una *función propia*, debemos "declararla". Esto se hace en cualquier lugar fuera de "*void setup()*" y "*void loop()*" –antes o después de ambas secciones– , siguiendo la siguiente sintaxis:

```
tipoRetorno nombreFuncion (tipo param1, tipo param2, …){
            // código interno de la función
}
```

El **código interno de la función** está delimitado por las llaves de apertura y cierre.

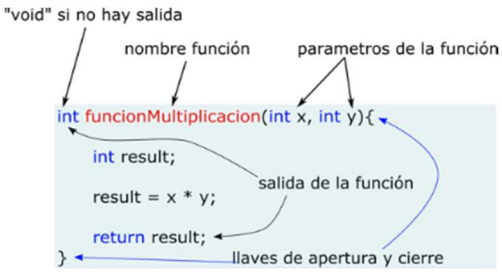
"***tipoRetorno***" es el tipo de valor ("*int*", "*float*", etc.) que la función devolverá al sketch principal una vez ejecutada. Este valor devuelto se puede guardar en una variable para ser usada en el sketch principal, o se puede ignorar. Si no se desea devolver ningún dato, se puede utilizar como tipo de retorno uno especial llamado "*void*" o bien no especificar ninguno. Para devolver el dato, se ha de utilizar la instrucción '***return valor;***', que tiene como efecto "colateral" el fin de la

ejecución de la función. Por ello, la instrucción "*return*" es normalmente la última que se escribe dentro del código de la función. Si la función no retorna nada, no es necesario escribirla.

"***tipo param1, tipo param2,...***" son las declaraciones de los parámetros de la función. Los valores iniciales de los parámetros se asignan explícitamente en la "llamada" a la función (cuando es invocada dentro del sketch principal). Los valores de los parámetros pueden variar dentro del código interno de la función. Al finalizar la ejecución de la función, todos sus parámetros son destruidos de la memoria del microcontrolador.

Ejemplo:

```
int funcionMultiplicacion(int x, int y){
   int result;
   result = x * y;
   return result;
}
```



Librerías Arduino oficiales

Una *librería* sirve para proporcionar funcionalidad extra en nuestros sketches. Junto con el IDE oficial de Arduino, se instalan por defecto una serie de *librerías oficiales*. Para poder utilizar las funciones que proveen, la librería en cuestión primero se ha de importar, bien mediante la opción ***"Sketch"->"Import library"*** del menú del IDE, o bien incluyendo al principio del código del sketch la sentencia ***#include*** pertinente.

Teclados matriciales de membrana

Un teclado matricial es un dispositivo que agrupa varios pulsadores en filas y columnas, formando una matriz. Podemos emplear teclados matriciales en proyectos de electrónica y robótica, por ejemplo, para cambiar el modo de funcionamiento de un montaje, para solicitar una contraseña, como teclas de dirección para controlar un brazo robótico o un vehículo, o proporcionar instrucciones a un robot.

El esquema de conexión es sencillo. Simplemente conectamos todos los pines a *entradas digitales* de Arduino.

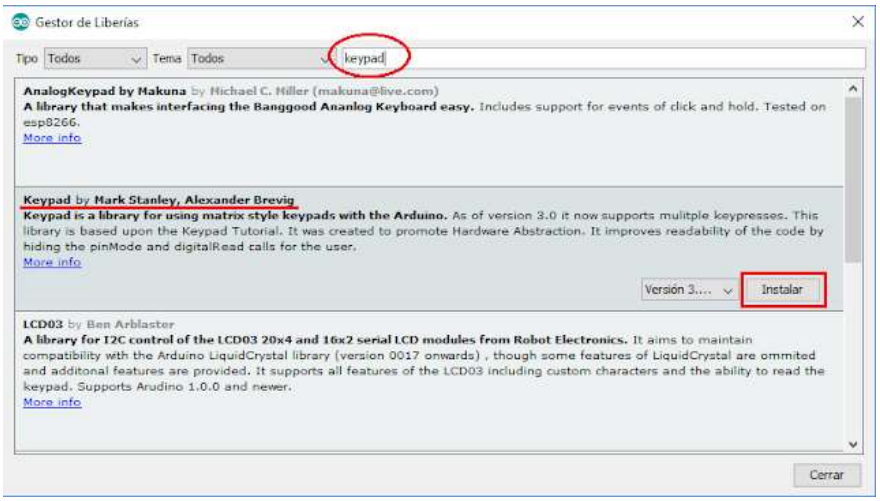
Librería *Keypad*

Para facilitar el uso de teclados matriciales, lo más recomendable es utilizar la librería "*Keypad*", librería no oficial, pero disponible dentro de la propia web de Arduino (concretamente, en http://arduino.cc/playground/Code/Keypad).

En IDEs modernos (1.6.2 o superior), la librería *Keypad* está disponible a través del administrador de librerías del IDE de Arduino. Simplemente, usar el menú:

Programa-> Incluir librería-> Administrar bibliotecas…

Luego buscar *Keypad*. Una vez encontrado, hacer clic en su entrada, y posteriormente clicar el botón de instalación.



Una vez instalada como cualquier otra librería, y conectado cada pin del teclado matricial a un pin-hembra digital de la placa Arduino configurado como entrada, ya podemos empezar a usarla.

makeKeymap():

Transforma el array 2D de teclas (*keys*) en un mapa de teclas que la librería puede entender.

Sintaxis:

```
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

getKey():

Retorna la tecla presionada, o *NO_KEY* si no hay teclas presionadas.

Sintaxis:

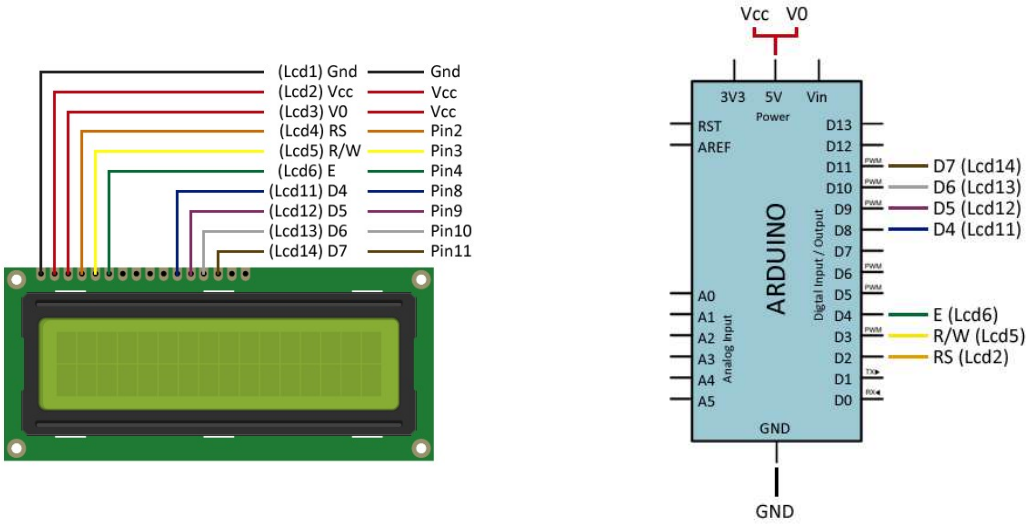
```
char key = keypad.getKey();
```

Pantallas LCD

Las pantallas LCD (*Liquid Cristal Display*) son una de las formas más sencillas y económicas de dotar de un pantalla a un autómata. El Hitachi HD44780 es uno de los controladores de LCDs más ampliamente extendidos por su sencillez y bajo precio. El HD44780 está diseñado para controlar LCDs monocromos de hasta 80 caracteres alfanuméricos y símbolos. También dispone de una pequeña memoria RAM para configurar nuestros propios caracteres o dibujos.

Las pantallas LCD disponen de retroiluminación trasera en azul o en verde. Se puede variar el contraste conectando un potenciómetro a la LCD.

Conectar un Arduino a una pantalla LCD HD44770 requiere una gran cantidad de cables. La figura siguiente muestra los pines de la LCD y su conexión a Arduino.



Si quisiéramos controlar el contraste de la pantalla, añadiríamos un potenciómetro para regular la tensión en V0.

Librería *LiquidCrystal*

La librería *LiquidCrystal* permite controlar pantallas de cristal líquido (LCDs) basadas en el chip HD44780 de Hitachi (o compatibles, como el KS0066 de Samsung o el ST7065C de Sitronix, entre otros). Este modelo de chip se encuentra en la mayoría de LCDs de caracteres. La librería puede trabajar tanto en modo 4-bit como en modo 8-bit.

Lo primero que debemos hacer para poder utilizar pantallas LCD compatibles con la librería oficial *LiquidCrystal* es declarar una variable global de tipo "LiquidCrystal", la cual representará dentro de nuestro sketch al objeto LCD que queremos controlar. La declaración se debe realizar usando la siguiente sintaxis (suponiendo que llamamos "*milcd*" a dicha variable-objeto):

LiquidCrystal milcd(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7);

donde todos los parámetros especificados entre paréntesis en realidad son valores numéricos que representan:

rs :	Pin de la placa Arduino conectado al pin "RS"- *Register Select* de la LCD.
rw:	Pin de la placa Arduino conectado al pin "RW"- modo *Read Write* de la LCD. (Opcional.)
enable:	Pin de la placa Arduino conectado al pin "ENABLE" de la LCD.
d0... hasta ***d7***:	Pines de la placa Arduino conectados a los pines de datos correspondientes de la LCD. Los parámetros *d0, d1, d2* y *d3* son opcionales: si se omiten, la LCD estará controlada mediante cuatro líneas (*d4*, *d5*, *d6*, *d7*) en vez de 8.

Una vez creado el objeto *milcd* con la línea anterior, lo primero que debemos hacer es establecer el *tamaño de la pantalla* para poder trabajar con ella. Esto se hace mediante la siguiente función:

milcd.begin(): especifica las dimensiones (columnas y filas) de la pantalla.

Tiene dos parámetros: el primero es el número de columnas que tiene la pantalla, y el segundo es el número de filas. No tiene valor de retorno.

A partir de aquí, podemos escribir caracteres en la pantalla y manipularla con alguna función de la librería.

Por ejemplo:

milcd.setCursor(): posiciona el cursor en la columna y fila especificadas como parámetros para escribir a partir de allí el próximo texto. Su primer parámetro es la columna en la que se quiere situar el cursor (la primera es la número 0) y su segundo parámetro es la fila (la primera es la número 0 también). No tiene valor de retorno.

milcd.print(): escribe un dato (de cualquier tipo) en la pantalla. Como primer parámetro tiene ese dato, que puede ser tanto un carácter de tipo "char" como una cadena de caracteres, pero también puede ser numérico entero ("int", "long", etc.). Opcionalmente, se puede especificar un segundo parámetro (consultar ayuda librería). Su dato de retorno es de tipo "byte" y vale el número de bytes escritos, aunque no es obligatorio utilizarlo.

milcd.clear(): borra la pantalla y posiciona el cursor en su esquina superior izquierda para escribir a partir de allí el próximo texto. No tiene ni parámetros ni valor de retorno.

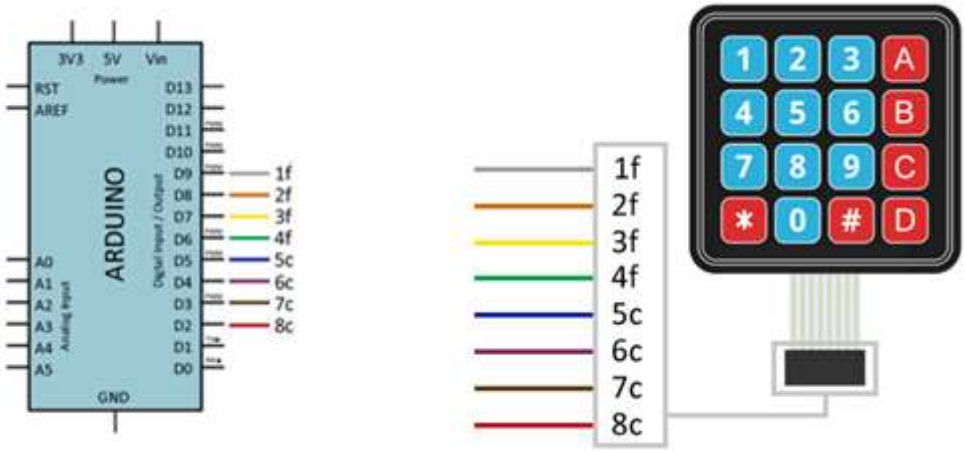
En el siguiente enlace se puede encontrar un ejemplo de uso de una LCD:

https://www.tinkercad.com/things/e3oNLwZYAaN-terrific-blad-curcan/editel?tenant=circuits

Prácticas a realizar

B2a – Teclado matricial (salida por monitor serie)

Conectar el teclado a la placa Arduino. **NO SEPARAR LOS CABLES DE CONEXIÓN (DEJARLOS UNIDOS).**



Probar el teclado, mostrando cada tecla presionada en el monitor serie.

Se pide, mediante programación:

- Leer la tecla presionada.
- Mostrar la tecla presionada en el monitor serie.

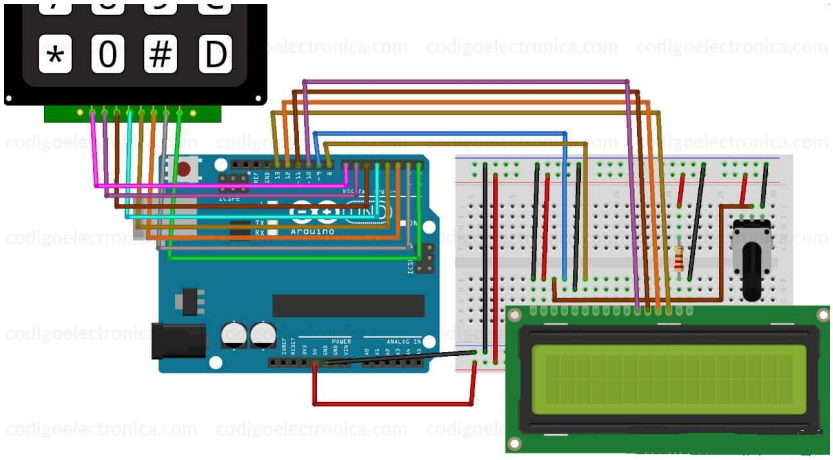
(A veces sucede que las lecturas no coinciden con las teclas que se están presionando. Esto ocurre si se conectan las filas donde van las columnas y viceversa.)

B2b – Teclado matricial (salida por pantalla LCD)

Conectar el teclado y la pantalla LCD a la placa Arduino. Utilizar un potenciómetro de **100 kΩ** (para regular el brillo) y una resistencia de **330 Ω** (para proteger un LED interno).

Para realizar la conexión, consultar la "Datasheet" correspondiente al modelo de pantalla (1602G, LMB162HBC, etc.) que se esté empleando.

Ejemplo de conexión de una pantalla modelo "1602":



Para conexiones "pin de protoboard" a "pin de protoboard" utilizar cables normales, disponibles en los cajones.

Escribir en la pantalla LCD utilizando el teclado matricial.

Se pide, mediante programación:

- Leer la tecla presionada.
- Mostrar la tecla presionada en la pantalla LCD.

B2c - Secuencia del coche fantástico

Con la serie luminosa de LEDs, reproducir la secuencia del “coche fantástico” en la que un haz de luz se dirige de derecha a izquierda y viceversa. Crear varias secuencias de iluminación utilizando *funciones propias*.