Práctica de la Asignatura
SISTEMAS EMPOTRADOS Y DE TIEMPO REAL

Proyecto Iluminación de Batería (*Drum-Illuminator*)

INTRODUCCIÓN:

La práctica de la asignatura ***Sistemas Empotrados y de Tiempo Real*** propone realizar un proyecto basado en la plataforma ***Arduino UNO***, de libre elección para el alumno.

En este caso he decidido implementar un sistema que controle la iluminación de los componentes de una batería (instrumento musical) cuando éstos son accionados (cuando se toca sobre ellos).

El proyecto se basa en una batería de 6 elementos que pueden iluminarse (6 tambores). Otros elementos de la batería (por ejemplo los platillos) no se verán cubiertos en este proyecto, aunque podrían ser una posible mejora o ampliación del mismo.

FUNCIONAMIENTO GENERAL DEL PROGRAMA:

El objetivo del proyecto es que se iluminen los tambores de una batería acústica cuando éstos son accionados (tocados/golpeados). Para conseguirlo, he planeado instalar unos sensores en cada uno de los elementos a controlar (cada uno de los tambores, en este proyecto) y unos LED que se iluminen como respuesta a la acción sobre los tambores.

Cuando se pulse sobre los elementos controlados, el impulso se recogerá mediante unos elementos *piezoeléctricos*, utilizados como sensores en este caso, que enviarán una señal a la placa ***Arduino UNO***, y ésta estará programada para iluminar el tambor que en cada caso se haya tocado, en un color determinado.

Además, se han incorporado otros accesorios, como un *display LCD* (de 16x2 caracteres) que presenta mensajes informativos y algunos botones (pulsadores) con diferentes funciones como iluminar la batería permanentemente, realizar un test de iluminación o los botones que permiten aumentar o disminuir los umbrales, durante el funcionamiento, que se fijarán para considerar una señal de entrada (proveniente de algún piezoeléctrico) lo suficientemente alta para iluminar el elemento que la originó.

MATERIAL NECESARIO:

- Placa Arduino UNO R3 (una placa clónica, en este caso);
- Breadboard de 830 pines;
- Fuente de alimentación para Breadboard (MB102);
- 6x Elementos Piezoeléctricos (de 27 y/o 35mm);
- 1x Chip Transistores Darlington 8 canales (ULN2803);
- Tira de LED RGB tipo *SMD5050* (60 LED/metro) (2metros, con resistencias incorporadas);
- 4x pulsadores;
- Una pantalla LCD (16x2) con interfaz I2C;
- 6x Resistencia de 1 MΩ (pull-down piezoeléctricos)
- 4x Resistencia de 10 KΩ (pull-down pulsadores)
- 4x Resistencia de 220Ω (Circuito pulsadores)
- Cable de red (para llegar a todos los tambores: unos 25 metros)

FUNCIONAMIENTO DETALLADO:

Al iniciar la placa ***Arduino UNO*** se ejecutan los comandos incluidos en la función "***setup()***". Estas operaciones sólo se ejecutan una vez al iniciar (o reiniciar) la placa, así que se aprovecha la función para realizar comprobaciones iniciales y establecer el modo de operación de los pines de entrada y/o salida.

En primer lugar, se configura la salida serie en caso de que se establezca como true la constante "***__DEBUG***" que, como su propio nombre indica, establece si se debe o no presentar información de *debug* (depuración del programa) por el puerto serie.

Después se comprueba si se utilizará un *display LCD de 16x2* caracteres, que presentará información sobre algunos eventos que se procesan en la placa. Si la constante ***__WITH_LCD*** está establecida como "*true*", se interpreta que hay un display con interfaz I2C conectado en el puerto *0x27* de la placa (*pines A4 [SDA] y A5 [SCL]*). Si hay un dispositivo *I2C*, sólo se comprueban un máximo de 4 entradas analógicas (*A0-A3*) para los detectores piezoeléctricos (por tanto, sólo se soportarían 4 tambores de la batería), porque los pines A4 y A5 son utilizados para la comunicación I2C. Si el display está disponible, se presenta un mensaje de bienvenida y se visualizarán diferentes mensajes como respuesta a algunos eventos. Si el display no está disponible, no se presentan mensajes por él (evidentemente) ni se pierden ciclos de ejecución con tareas relacionadas con la presentación de mensajes por pantalla (se controla con la constante *"__WITH_LCD"*), además de disponer de hasta 6 entradas analógicas para monitorizar los detectores piezoeléctricos (se pueden controlar hasta 6 tambores).

Después de estas dos comprobaciones, se fija el comportamiento que tendrán los diferentes pines de entrada y/o salida que se utilizan para el sistema.

Existen otras variables y constantes, declaradas en el encabezado del programa principal (el archivo ***DrumIlluminator.ino***), que sirven para ajustar diversos aspectos del comportamiento del programa. En el propio código figuran breves explicaciones de lo que cada una de ellas representa. Para cambiar el comportamiento del programa variando el valor de estas variables (o de las constantes, como las que se mencionan en este documento) es necesario volver a compilar y cargar el programa en la placa *Arduino*.

Una vez finalizada la inicialización del programa, la placa ***Arduino UNO*** ejecuta en un bucle infinito (hasta que se reinicie o hasta que se desconecte la corriente de la placa) la función "***loop()***".

La función ***loop()*** realiza tres tareas, principalmente:

1. Actualizar los contadores y temporizadores que mantienen el control sobre el funcionamiento. Principalmente, se controla si deben apagarse los LED que se hayan encendido previamente (si ha transcurrido el tiempo establecido en la constante ***LIGHT_DELAY***). También se comprueban otros parámetros asociados a envíos de señales y comprobación de los LED.
2. Comprobar si se ha pulsado algún botón con funciones adicionales:
    a. Botones para subir o bajar los umbrales (blanco "bajar" y azul "subir"). Si se ha pulsado alguno de estos botones, los umbrales establecidos para considerar una señal lo suficientemente alta para producir el efecto deseado, sufrirán un incremento o un decremento igual a la cantidad establecida en la constante ***THRES_INTERVAL***. Los umbrales son fijados para cada tambor monitorizado y se encuentran en el array de umbrales ***thresholds***. Con estos pulsadores, se puede modificar el comportamiento del sistema sin tener que compilar ni cargar el programa en la placa y se pueden regular los umbrales para iluminar los LED de forma más apropiada. El objetivo es realizar pequeños ajustes de los umbrales "en caliente" para ajustarse a las vibraciones que

sufren los tambores "por simpatía" cuando suenan ellos mismos u otros y que este efecto no produzca "iluminaciones indeseadas".

b. Botón de Test (verde): Pulsando este botón se hace una comprobación de que todos los LED instalados funcionan.

c. Botón de iluminación constante (rojo): con este botón se activa o se desactiva la iluminación constante de los tambores, aprovechando que los LED son RGB. La iluminación constante será de un color (roja, en este montaje) y cuando se actúe sobre algún tambor, se activará la luz de otros colores en ese tambor.

3. Hacer una lectura de todas las entradas analógicas disponibles (de *0* a *5*). Se leen las entradas analógicas desde ***A0*** hasta ***AX***, siendo ***X*** el valor fijado en la variable ***lastAnalogIndex***. En cada uno de los puertos que se escanean debe haber un sensor piezoeléctrico convenientemente conectado, para obtener un buen funcionamiento del sistema (si están vacíos o mal conectados el resultado es impredecible). Primero se comprueba que haya que realizar la lectura de la entrada analógica: si se ha encendido su LED correspondiente y aún no ha trascurrido el tiempo establecido para que se apague (establecido en la constante *LIGHT_DELAY*), no se realiza lectura de la entrada analógica. En caso contrario, cada vez que se realiza la lectura de una de las entradas analógicas, se comprueba si el valor leído supera un umbral establecido para esa entrada (para el piezoeléctrico correspondiente). Si supera ese valor, se enciende el LED asociado a esa entrada (el del mismo tambor donde está el piezoeléctrico, para conseguir el efecto deseado) y se fija el temporizador que indica cuándo se debe apagar.

ACLARACIONES SOBRE EL MONTAJE:

- Se han creado tres archivos para el código fuente que implementa el programa del proyecto. Por claridad y para no introducir una cantidad exagerada de líneas de código en un solo archivo, he creado un archivo auxiliar de *código* ***C*** (***DrumIlluminatorFunctions.cpp***) y otro archivo de *cabecera* (***DrumIlluminatorFunctions.h***) que sirve para enlazarlo al *archivo principal del programa para Arduino* (***DrumIlluminator.ino***). En estos archivos he dejado las funciones relacionadas con la escritura de texto en el display.
- El uso de un display (***LCD de 16x2***) no es estrictamente necesario, pero se ha incorporado para ver el funcionamiento del display con *Arduino*, para ver el funcionamiento de dispositivos con interfaz *I2C* y para aprovechar a mostrar información durante el funcionamiento, que puede ser útil. Para el uso de este dispositivo es necesario importar alguna librería de manejo de LCD con interfaz I2C (como la librería *LiquidCrystal_I2C*). En este proyecto se ha utilizado, concretamente, la librería **LiquidCrystal_I2C, versión 1.1.0, de Marco Schwartz**. Cuando el display no está conectado a la placa *Arduino*, ésta puede seguir funcionando, aunque no admite conexión/desconexión en caliente: como implica el cambio del valor de una constante (*__WITH_LCD*) se hace necesario compilar el código, cargarlo en la placa Arduino y reiniciar. El procedimiento a seguir para instalar la librería desde el IDE de Arduino, es:
  - **Menú *Programa* → *Incluir Librería* → *Gestionar Librerías* → Buscar *LiquidCrystal_I2C***
- He tenido que definir algunas funciones para el uso del display LCD, porque su funcionamiento no era del todo apropiado (en ocasiones se imprimía sólo el primer carácter de un número o de una cadena de caracteres). La definición de estas funciones está en el archivo "*DrumIlluminatorFunctions.cpp*".
- El uso de conjuntos de transistores integrados en un chip (***ULN2803***) se ha hecho necesario porque las tiras de LED que se han utilizado disponen **LED RGB** (tricolores) dispuestos en paralelo y con el **ánodo común**. Esto hace que iluminar los LED de un color concreto exija la conexión de la patilla correspondiente a ese color a tierra (***GND***): la tensión (*5v*) está conectada

permanentemente a la patilla común (*ánodo*). Para conseguir el efecto deseado, se utilizan los transistores como si fuesen interruptores accionados eléctricamente (accionados desde las salidas digitales de *Arduino*, en este caso) que conectan la patilla correspondiente a cada color a tierra, cuando sea necesario.

- La tira de LED que se ha utilizado para la práctica lleva montados las resistencias necesarias para su correcto funcionamiento. Como viene todo integrado en una tira y no dispongo del plano, no sé exactamente si las resistencias están entre el ánodo y cada color del LED o entre cada color del LED y la patilla que va a tierra. En el esquema he puesto las resistencias de este segundo modo porque encaja mejor con los componentes disponibles en ***Fritzing***, aunque podría ser de otro modo.
- La instalación de una fuente de alimentación adicional para el circuito que alimenta a los LED, se ha estimado necesaria para no sobrecargar la placa *Arduino*. El consumo de la tira de LED utilizada (LED tipo ***SMD5050*** con 60 LED por metro) es de **14,4 Watts/metro**. Teniendo en cuenta que se utilizan 1,20metros de esta tira, en total, el consumo con todo encendido al mismo tiempo puede llegar a *17,30 Watts* (aprox.). En cada tambor se instala unos *20cm* de tira (*12 LED*, unos *3W*) y es muy difícil que estén encendidos más de 3 al mismo tiempo, así que podría decirse que el consumo máximo va a ser de *9W* y durante periodos de tiempo relativamente cortos. Siendo ***P=V*I*** y soportando la placa un máximo de *500mA* (consumo total, incluyendo el consumo de la propia placa), la máxima potencia soportada sería de:
    - P = 5v * 0,5A = **2,5W**

  Se ha instalado una fuente de alimentación de ***1A*** y ***9V*** conectada a la corriente (220v), con un transformador a *5V* para alimentar el circuito de los LED (placa MB102). Esto evitará daños en el microcontrolador *Arduino*. El display LCD (en caso de utilizarse) sí puede ir alimentado por la placa directamente (pines 5v y GND). En el esquema, viene representada por una alimentación con pilas.

- Se ha procurado minimizar las tareas innecesarias (por ejemplo, refrescar el estado de los LED cuando no es necesario) y las esperas (evitando en lo posible la función "*delay*") para intentar que la placa esté la mayor parte del tiempo dedicada a la lectura de las entradas analógicas, que es su principal cometido: estar a la espera de un evento para reaccionar cuando éste ocurre. Este es el motivo del uso de los múltiples temporizadores utilizados.

POSIBLES MEJORAS:

A continuación se exponen una serie de posibles mejoras que no han sido implementadas, por falta de tiempo y/o material (además de que había que establecer unos límites al proyecto), pero que podrían aumentar la funcionalidad del sistema propuesto:

- Ampliar el número de entradas analógicas para monitorizar más elementos de la batería (por ejemplo, los platillos). Requeriría la instalación de multiplexadores analógicos (por ejemplo el 74HC4051) y precisaría un replanteamiento del algoritmo (la función) que realiza la lectura de las entradas analógicas y establece el estado de las salidas. Su uso con Arduino puede consultarse en: http://playground.arduino.cc/Learning/4051 .
  Además, habría que utilizar algún sistema para ampliar el número de salidas para dar respuesta a todas las entradas. En el esquema propuesto se han utilizado todos los pines digitales del *Arduino UNO*, aunque podría liberarse el pin 13. Los pines 0 y 1 se reservan para no interferir en las comunicaciones serie. Las salidas digitales podrían ampliarse, por ejemplo,

con registros de desplazamiento del tipo 74HC595. Su uso puede consultarse en: https://www.arduino.cc/en/Tutorial/ShiftOut .

- Conectar la alimentación de los LED a una fuente de alimentación de mayor potencia para dejar un margen de confianza. Según los cálculos, la fuente de alimentación que se ha instalado puede resultar excesivamente ajustada o incluso deficitaria. Podrían considerarse, incluso, opciones como el uso de tiras de LED que funcionen a otras tensiones (12v, por ejemplo) para poder utilizar fuentes de alimentación con menor exigencia de potencia (en caso de cambiar la tensión, debería adaptarse el esquema para los pulsadores, que van a las entradas digitales de *Arduino*).
  Lo ideal, en definitiva, sería instalar una tira de LED por todo el perímetro interior de cada uno de los tambores y alimentarlo con una fuente de alimentación bien dimensionada. El esquema de montaje no variaría con respecto del propuesto, porque se separan los circuitos de alimentación de los LED y los de *Arduino*.
- Una mejor instalación física en general: el cableado de los tambores, la instalación de los LED, de la sujeción de los sensores a los tambores,...

PUBLICACIÓN DEL PROYECTO:

El proyecto, completo, se ha publicado en GitHub:

https://github.com/jorgeblnz/DrumIlluminator

También se ha publicado un video demostrativo del funcionamiento en YouTube:

https://youtu.be/wYUKeTOndPg

FUENTES CONSULTADAS:

http://www.miguelgrassi.com.ar/arduino/PiezoMIDI.htm

https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads

https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home#!downloading-and-installation

http://www.sebyc.es/reess/componentes/uln2803.htm

http://www.electrontools.com/Home/WP/2016/03/09/registro-de-desplazamiento-74hc595/

http://www.prometec.net/shift-registers/

http://www.educachip.com/led-rgb-arduino-anodo-comun/

https://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

http://toshiba.semicon-storage.com/info/docget.jsp?did=7064&prodName=ULN2803APG

http://www.luisllamas.es/2014/09/leer-un-pulsador-con-arduino/

http://playground.arduino.cc/Main/I2cScanner

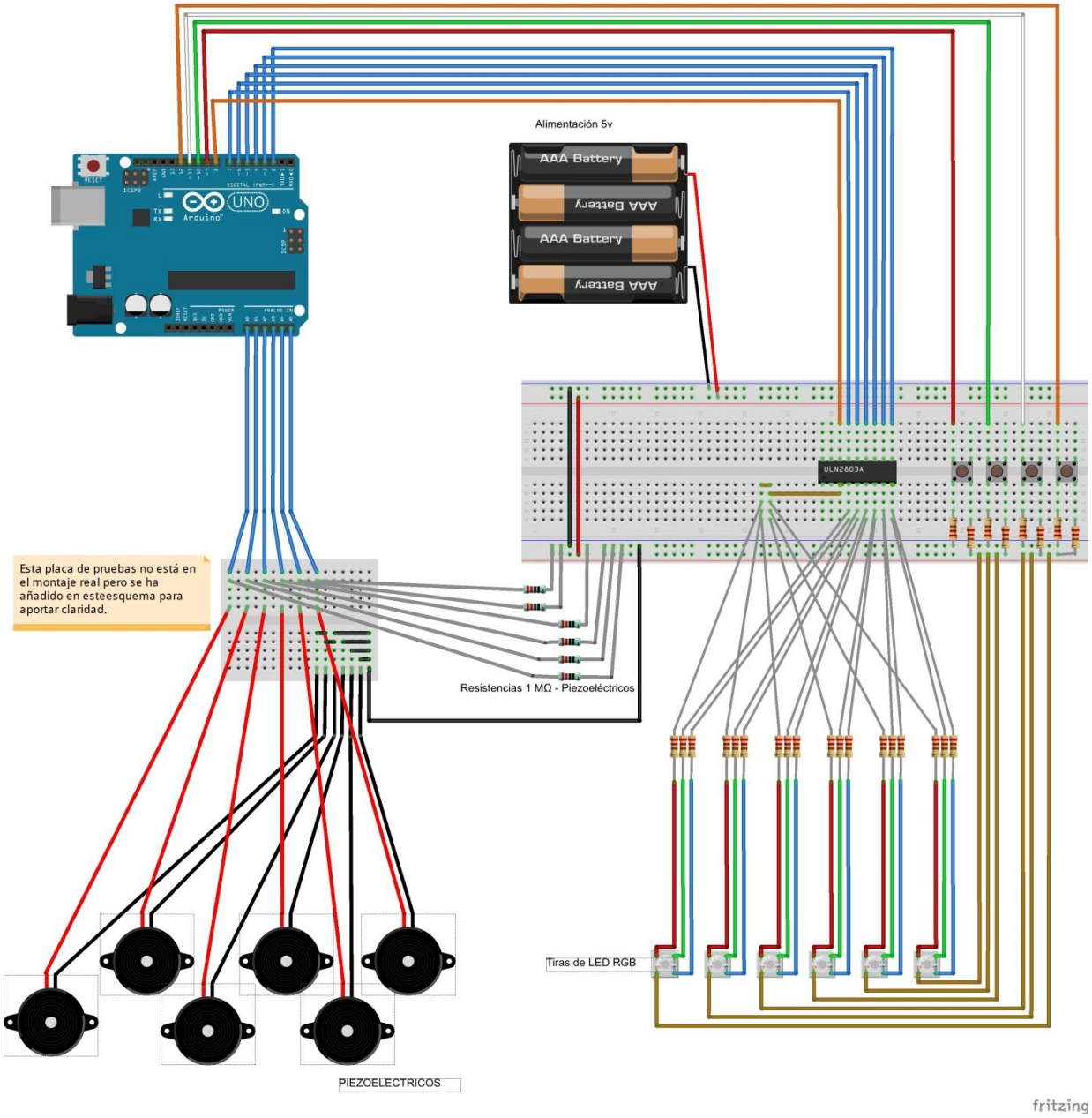
https://arduino-info.wikispaces.com/LCD-Blue-I2C

https://github.com/arduino-libraries/LiquidCrystal

http://downloads.arduino.cc/libraries/marcoschwartz/LiquidCrystal_I2C-1.1.2.zip

ESQUEMAS:

Esquema Sin LCD:



Esquema Con LCD:

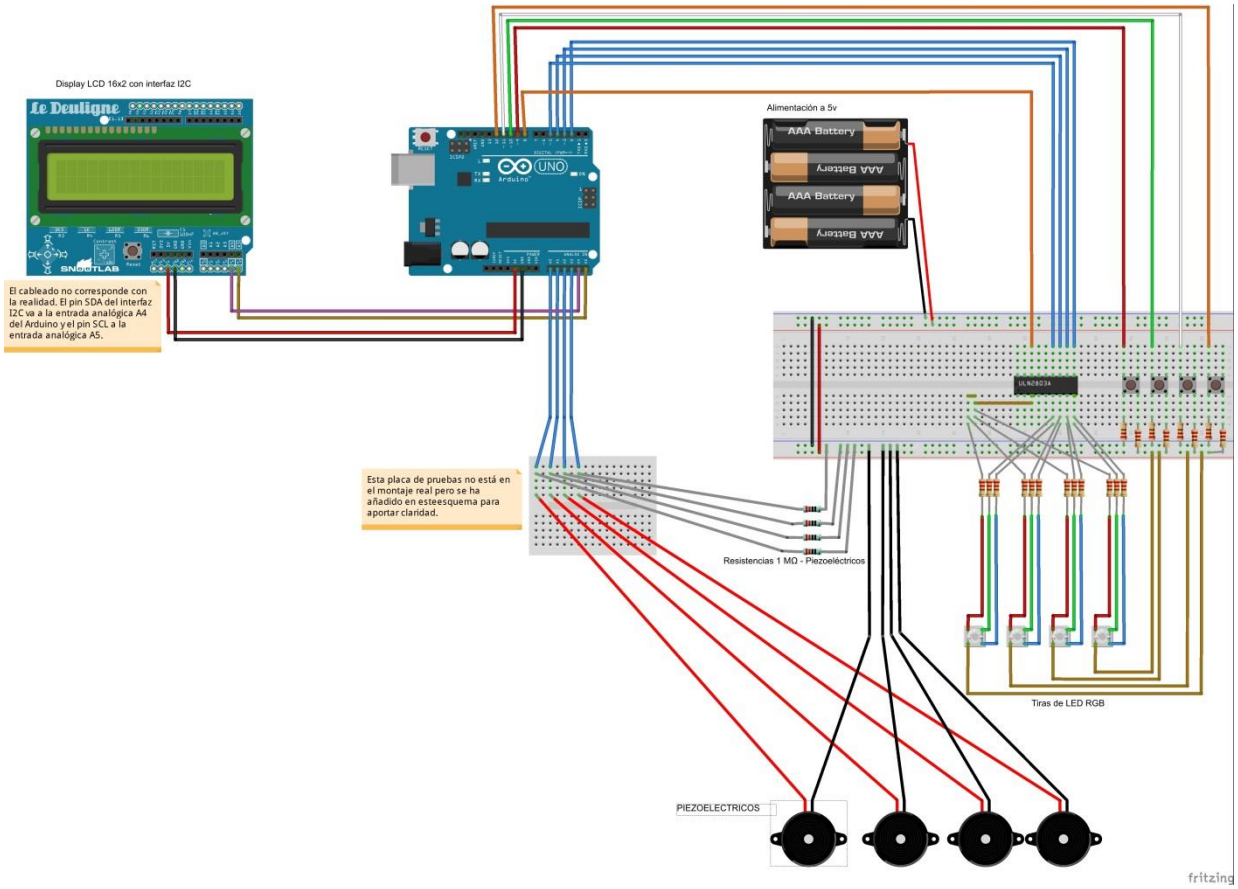
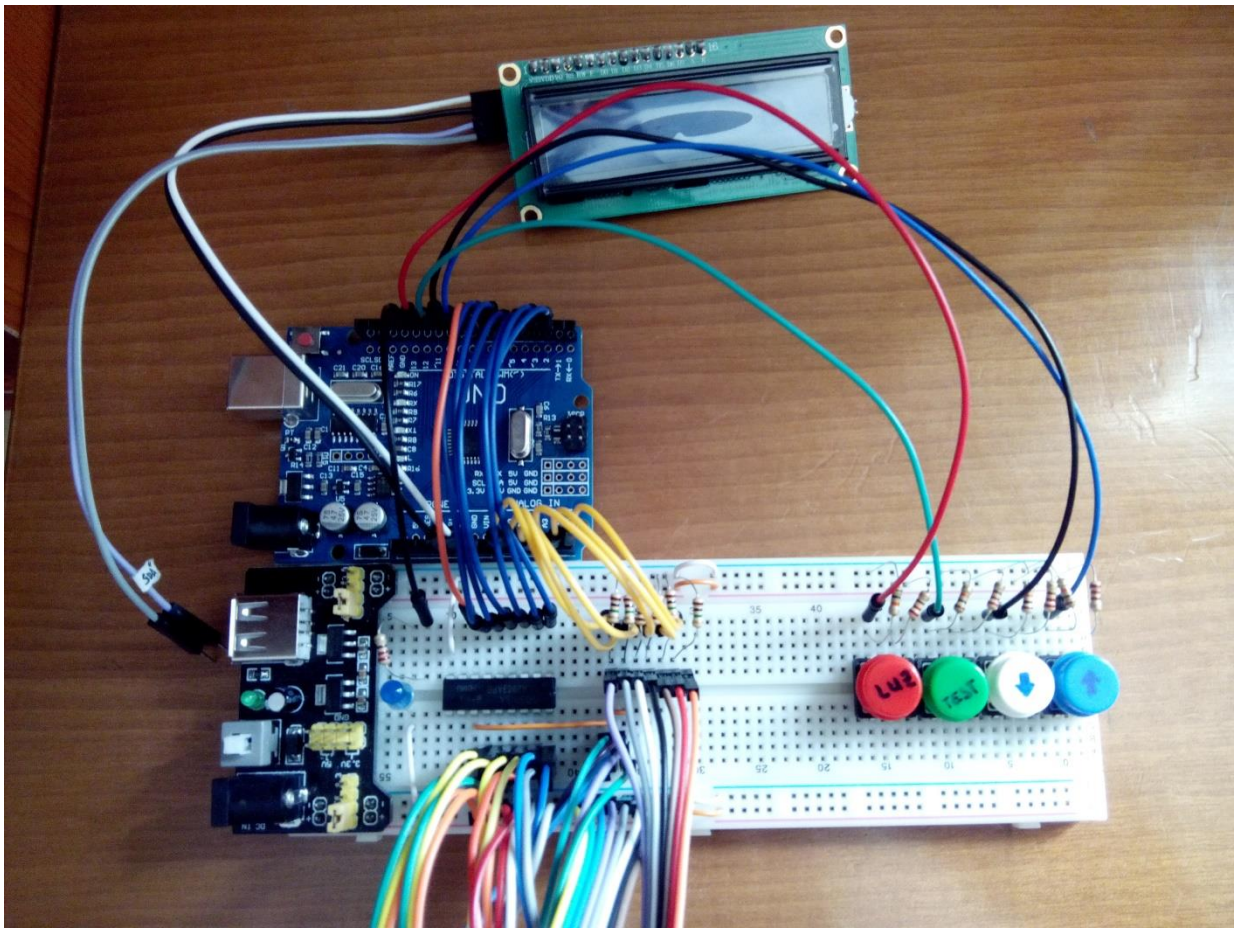


Foto del Montaje Real:



ATENCIÓN: Para conectar el Display LCD se deben desconectar los cables amarillos de las entradas A4 y A5 y conectar los pines del display: SDA a la entrada A4 (o al pin SDA de Arduino) y SCL a la entrada A5 (o al pin SCL de Arduino). NO PUEDEN ESTAR CONECTADOS AL MISMO TIEMPO todos los pines (SDA, SCL, A4 y A5): se puede utilizar únicamente la comunicación I2C (SDA-SCL) o utilizar únicamente las entradas analógicas (A4-A5) para lecturas de los sensores piezoeléctricos.