Escuela Técnica Superior de Ingeniería en Diseño Industrial

UPM de Madrid.

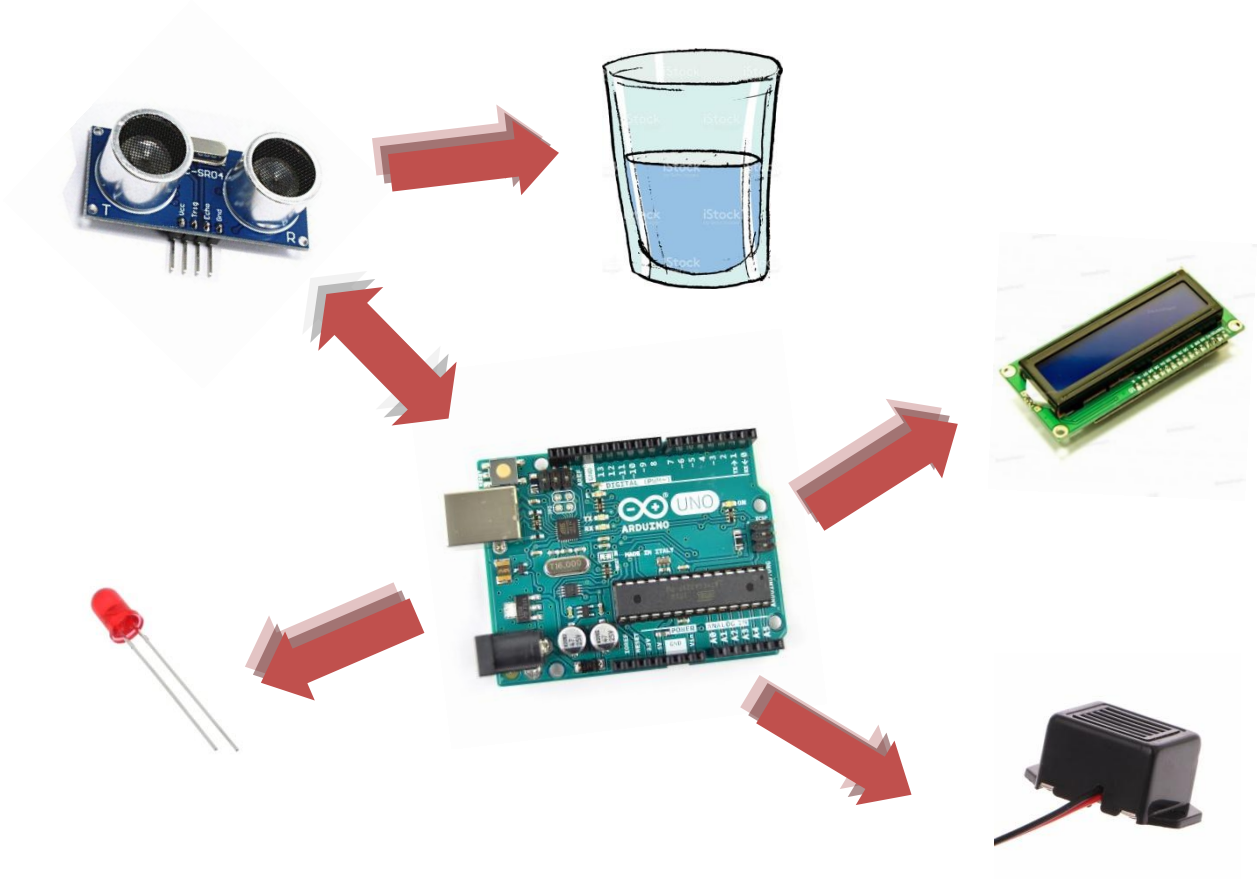
TRABAJO DE INFORMATICA, CURSO 17/18

GRUPO A109, CLASE B11

Alvaro	Merodio Pardo	54090	a.merodiop@alumnos.upm.es
David	Patrizi Garcia	54145	d.patrizi@alumnos.upm.es
Ignacio	Mayoral Sánchez	54276	i.mayorals@alumnos.upm.es

TITULO Y RESUMEN:

El trabajo consiste en un sistema electrónico basado en un micro controlador de arduino, que sirve como ***calculador de cantidad de líquido*** (agua, combustible, etc.) en un recipiente. El micro controlador de la placa de arduino usará distintos dispositivos electrónicos para tomar medidas, calcular datos y mostrar tanto los datos calculados como alertas acústicas y luminiscentes.



Método de funcionamiento:

1- El sensor ultrasonidos tomará medidas del vaso vacío.
2- Una vez tengamos las medidas, al introducir agua en el recipiente el ultrasonido tomará datos en tiempo real y micro controlador de arduino calculará las medidas de utilidad real.
3- la pantalla mostrará los datos calculados por el micro controlador (cantidad de agua, porcentaje del vaso lleno)
4- Cuando se hayan alcanzado las condiciones de peligro (más del 80% del vaso con líquido), el arduino conectará el led y el zumbador, mandando así una señal de alarma al usuario.

HARDWARE Y FUNDAMENTOS:

El sensor de ultrasonidos que usaremos para medir las distancias y tomar datos de entrada para el programa será el modelo HC-SR04. El sensor HC-SR04 es un sensor de distancia de bajo costo, su uso es muy frecuente en la robótica, utiliza transductores de ultrasonido para detectar objetos. Los sensores de ultrasonidos son sensores baratos, y sencillos de usar. El rango de medición teórico del sensor HC-SR04 es de 2cm a 400 cm, con una resolución de 0.3cm. En la práctica, sin embargo, el rango de medición real es mucho más limitado, en torno de 20cm a 2 metros. Los sensores de ultrasonidos son sensores de baja precisión. La orientación de la superficie a medir puede provocar que la onda se refleje, falseando la medición.

Su funcionamiento es muy sencillo y para empezar a utilizar el sensor HC-SR04 solo necesitas una placa Arduino. Su funcionamiento consiste en emitir un sonido ultrasónico por uno de sus transductores, y esperar que el sonido rebote de algún objeto presente, el eco es captador por el segundo transductor. La distancia es proporcional al tiempo que demora en llegar el eco.

Se basa simplemente en medir el tiempo entre el envío y la recepción de una onda sónica o pulso sonoro. Gracias a que sabes que la velocidad del sonido es de 343 m/s (tomado en condiciones normales), llegamos a la conclusión de que el sonido viaja a (1/29.2)cm/picosegundos.

$$343\frac{m}{s} \cdot 100\frac{cm}{m} \cdot \frac{1}{1000000}\frac{s}{\mu s} = \frac{1}{29.2}\frac{cm}{\mu s}$$

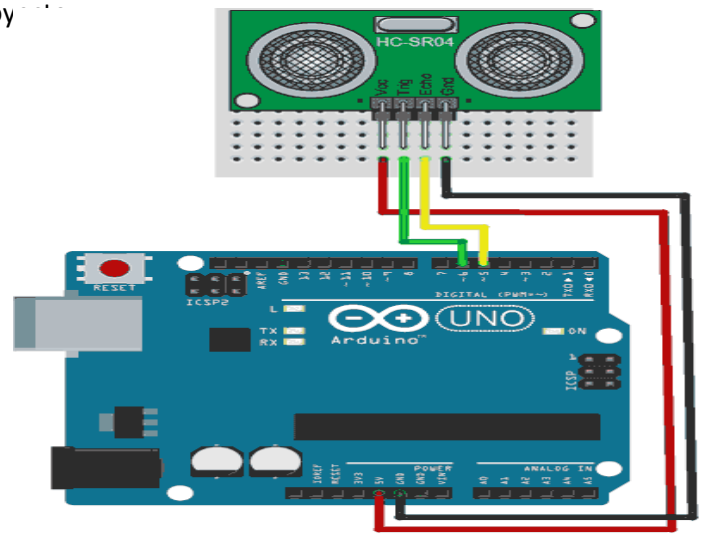
Debido a esto podemos calcular la distancia que hay entre nuestro sensor de ultrasonido y un objeto, que se calcula mediante la relación matemática:

$$Distancia(cm) = \frac{Tiempo(\mu s)}{29.2 \cdot 2}$$

El sensor ultrasónico consta de dos pines:

- Pin “Trigger”, conectado al pin 6 del arduino.
- Pin “Echo”, conectado al pin 5 del arduino.

Después de montar el circuito necesario para el funcionamiento del sensor de ultrasonidos procedemos a probarlo para comprobar cómo funciona y si actúa de la manera correcta. Como se ve en la imagen el circuito es bastante sencillo y no requiere mayor complicación. Este circuito no es el que usaremos el en proyecto final ya que habrá que cambiar algunos pines para ajustarlo a nuestras necesidades con los demás instrumentos del proy



```
// Configuramos los pines del sensor Trigger y Echo
const int PinTrig = 6;
const int PinEcho = 5;

// Constante velocidad sonido en cm/s
const float VelSon = 34000.0;

float distancia;

void setup()
{
  // Iniciamos el monitor serie para mostrar el resultado
  Serial.begin(9600);
  // Ponemos el pin Trig en modo salida
  pinMode(PinTrig, OUTPUT);
  // Ponemos el pin Echo en modo entrada
  pinMode(PinEcho, INPUT);
}
void loop()
{
  iniciarTrigger();

  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este caso a HIGH
  unsigned long tiempo = pulseIn(PinEcho, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en segudos ya que está en microsegundos
  // por eso se multiplica por 0.000001
  distancia = tiempo * 0.000001 * VelSon / 2.0;
  Serial.print(distancia);
  Serial.print("cm");
  Serial.println();
  delay(500);
}

// Método que inicia la secuencia del Trigger para comenzar a medir
void iniciarTrigger()
{
  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);
}
```

Gracias a este programa y al circuito montado, al ejecutarlo y abrir el display el programa nos muestra datos en tiempo real a cerca de las mediciones del sensor.

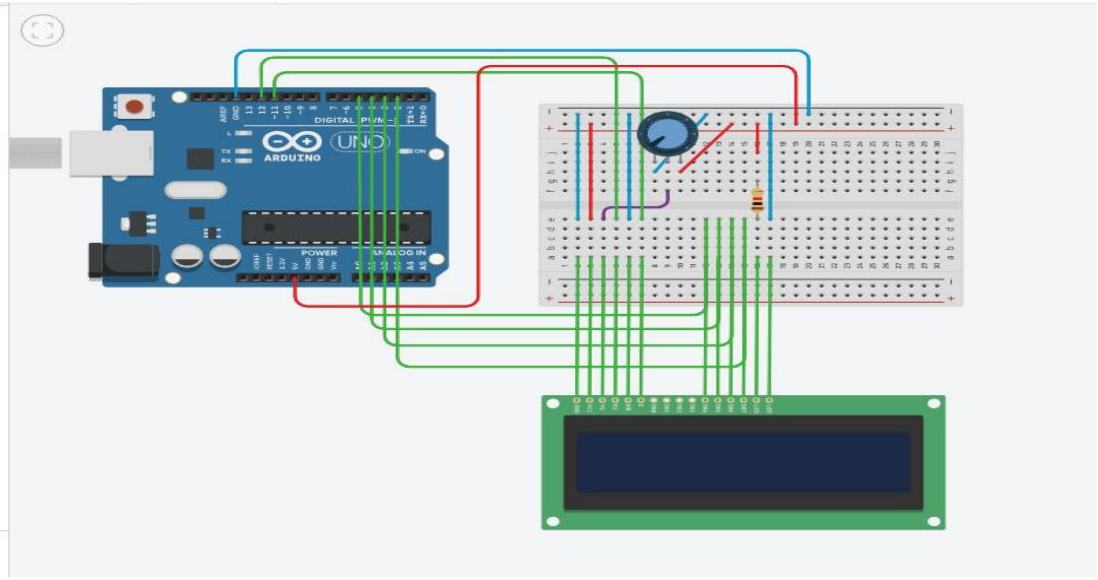
Fuente: https://naylampmechatronics.com/blog/10_Tutorial-de-Arduino-y-sensor-ultras%C3%B3nico-HC-S.html

Pantalla LCD 16x2:

Las pantallas LCD (Liquid Cristal Display) son una de las formas mas sencillas y económicas de dotar de un desplay a un autómata. El Hitchi HD44780 es uno de los controladores de LCDs más ampliamente extendidos por su sencillez y bajo precio. Este está diseñado para controlar LCDs monocromos de hasta 80 caracteres o dibujos. Las pantallas LCD con controlador HR44770 se fabrican en distintos tamaños, siendo los más comunes 16x02 (el que usaremos en este proyecto), 20x02, 20x04, 40x02. Las pantallas LCD disponen de retroiluminacion trasera en azul o verde. El contraste entre la retroilumniacion trasera y las letras o simbolos puede ser variado coectando un potenciómetro al LCD.

Los LCD con controlador HD44770 son componentes baratos. Su precio varía en función del tamaño y el número de caracteres. Podemos encontrar pantallas de tamaño 16x02 por algo más de 1€ en Ebay o AliExpress.

El IDE de arduino incorpora de serie la librería LiguidCrystal, para controlar pantallas LCD HD44770 de forma más sencilla. Existen a demás módulos incorporables para la pantalla LCD que reducen considerablemente el número de pines necesarios para su uso, aunque en nuestro trabajo no hacemos uso de ello ya que sin ellos tampoco es difícil la programación de estos dispositivos.



```
#include <LiquidCrystal.h> // Entre los símbolos <> buscará en la carpeta de librerías configurada

// Lo primero is inicializar la librería indicando los pins de la interfaz
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Definimos las constantes
#define COLS 16 // Las columnas del LCD
#define ROWS 2 // Las filas del LCD

void setup() {
  // Configuramos las filas y las columnas del LCD.
  lcd.begin(COLS, ROWS);
}

void loop() {

  lcd.setCursor(0,0);
  lcd.print("Probando LCD ;)");
  lcd.setCursor(0,1);
  lcd.print("HOLA MUNDO!");
}
```

Como podemos leer en el código, este programa imprime en la pantalla LCD lo siguiente “Probadno LCD ;)” y “HOLA MUNDO!”, en la primera y segunda fila, respectivamente.

Fuente: https://www.luisllamas.es/arduino-lcd-hitachi-hd44780/

SONDA DE TEMPERATURA:

Decidimos incluir a demás en nuestro proyecto una sonda de temperatura, que nos será bastante útil para también mostrar por la pantalla la temperatura a la que se encuentra el agua. El DS118B20 es un sensor de temperaturas que proporciona la salida mediante un bus de comunicación digital que puede ser

leído con las entradas digitales de Arduino. El sensor es un sensor barato y, sin embargo, bastante avanzado, que dispone de un rango amplio de medición de -55ºC a +125ºC con una precisión ±0.5ºC.

Una de las ventajas del DS18B20 es que se comercializa tanto en un integrado TO-92 como en forma de sonda impermeable, lo que permite realizar mediciones de temperatura en líquidos y gases. La principal ventaja del bus 1-Wire es que necesita un único conductor para realizar la comunicación (sin contar el conductor de tierra). Los dispositivos pueden ser alimentados directamente por la línea de datos, o mediante una línea adicional con una tensión de 3.0 a 5.5V.

El DS18B20 es un gran sensor para la medición de temperatura, tanto en ambientes domésticos como industriales. Sus características permiten crear redes con gran número de sensores para controlar, por ejemplo, la climatización o el sistema HVAC de un edificio comercial. Es un sensor barato y excelente en calidad / precio. Podemos encontrarlo en su versión integrado TO-92 por 0.45€, y por 1€ en la versión sumergible, buscando en vendedores internacionales en eBay y AliExpress.

```
#include <OneWire.h>
#include <DallasTemperature.h>

const int oneWirePin = 5;

OneWire oneWireBus(oneWirePin);
DallasTemperature sensor(&oneWireBus);

void setup() {
   Serial.begin(9600);
   sensor.begin();
}

void loop() {
    Serial.println("Leyendo temperaturas: ");
   sensor.requestTemperatures();

   Serial.print("Temperatura en sensor 0: ");
   Serial.print(sensor.getTempCByIndex(0));
   Serial.println(" °C");

   delay(1000);
}
```

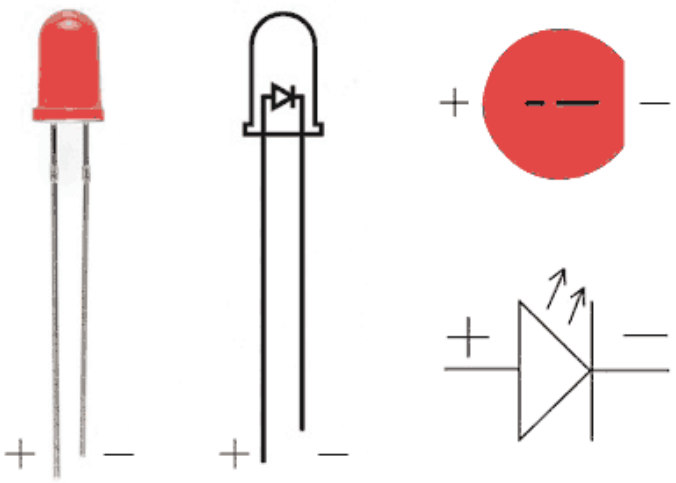
En este ejemplo leeríamos los datos tomados por un único sensor ubicado en el pin número 5 del arduino.

Fuente: https://www.luisllamas.es/temperatura-liquidos-arduino-ds18b20/ ; y curso del club de robótica (CREA) de la universidad.

LED:

Un LED es un diodo emisor de luz, es decir, un tipo particular de diodo que emite luz al ser atravesado por una corriente eléctrica. Un diodo es una unión de dos materiales semiconductores con dopados distintos. Esta diferencia de dopados hace que genere una barrera potencial, que como primera consecuencia hace que el paso de corriente en uno de los sentidos no sea posible. Aquí tenemos la primera característica de los diodos, tienen polaridad, es decir solo dejan pasar la corriente en un sentido. Por tanto, tenemos que conectar correctamente la tensión al dispositivo. La patilla larga debe ser conectada al voltaje positivo (ánodo), y la corta al voltaje negativo (cátodo).

En el momento que superamos la tensión de polarización y dado que la resistencia del diodo es muy péquela, se genera una gran corriente que destruirá el diodo. Por ese motivo necesitamos una resistencia que limite la cantidad de corriente que circula por el diodo.



El coste de los LED es muy similar, independientemente de su tipo y tamaño, aunque algunos colores pueden ser un poco más caros. En general podemos encontrar un LED realmente barato y es que en tiendas internacionales suelen costar 1 céntimo de euro. En la tienda que lo compramos nosotros (Electrónica Embajadores), especializada en dispositivos electrónicos y con una gran variedad de alternativas en cuanto al uso de arduino costaban entre 30 y 60 céntimos de euro, dependiendo del tipo, tamaño o color.

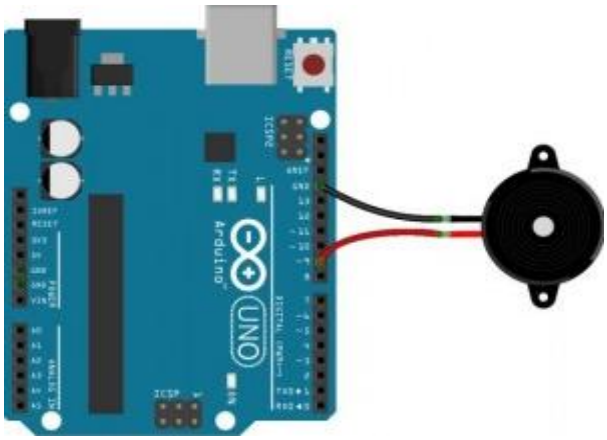
Fuente: https://www.luisllamas.es/encender-un-led-con-arduino/

ZUMADOR:

Los buzzers activos, denominados comúnmente zumbadores, son dispositivos que generan un sonido de una frecuencia determinada y fija cuando son conectados a tensión. Un buzzer activo incorpora un oscilador simple por lo que únicamente es necesario suministrar corriente al dispositivo para que emita sonido. Estos buzzers incluyen en sus componentes un determinado cristal (el cuarzo es uno de estos cristales más conocidos) que si los sometemos a una tensión eléctrica variable vibran.

Existen buzzer en un gran abanico de tamaños y potencias, desde tonos casi imperceptibles hasta alarmas realmente estridentes. Los buzzer de mayor potencia son adecuados para generar alarmas de froma sencilla, por ejemplo sensor de movimiento, sensor de agua o de llamas, entre otros.

Existen una gran cantidad de buzzer o zumbadores que podemos usar para nuestro proyecto con arduino. Podemos encontrarlos desde 0.50€ en páginas webs internacionales como eBay o AliExpress.



```
const int pin = 9;

void setup() {
  pinMode(pin, OUTPUT);  //definir pin como salida
}

void loop(){
  digitalWrite(pin, HIGH);   // poner el Pin en HIGH
  delay(5000);               // esperar 5 segundos
  digitalWrite(pin, LOW);    // poner el Pin en LOW
  delay(20000);               // esperar 20 segundos
}
```

Este sería un ejemplo de código muy sencillo que activa el zumbador durante cinco segundos y posteriormente lo apaga durante 20s.

Fuente: https://www.prometec.net/buzzers/ ; https://www.luisllamas.es/arduino-buzzer-activo/

Proceso de selección:

Cuando empezamos a pensar en el proyecto comenzamos preguntándonos que problemas existen en cuanto a los medidores de líquidos convencionales. Actualmente se suele usar para medir la cantidad de líquido, un vaso que contiene escrito las medidas de agua. Estos vasos son bastante simples pero tienen varios problemas; se borra la tinta de las medidas con el paso del tiempo, necesitas una mesa que esté en nivelada ya que si está inclinada el líquido se inclinará y la medida que tomemos no será correcta y lo más importante, es poco preciso.

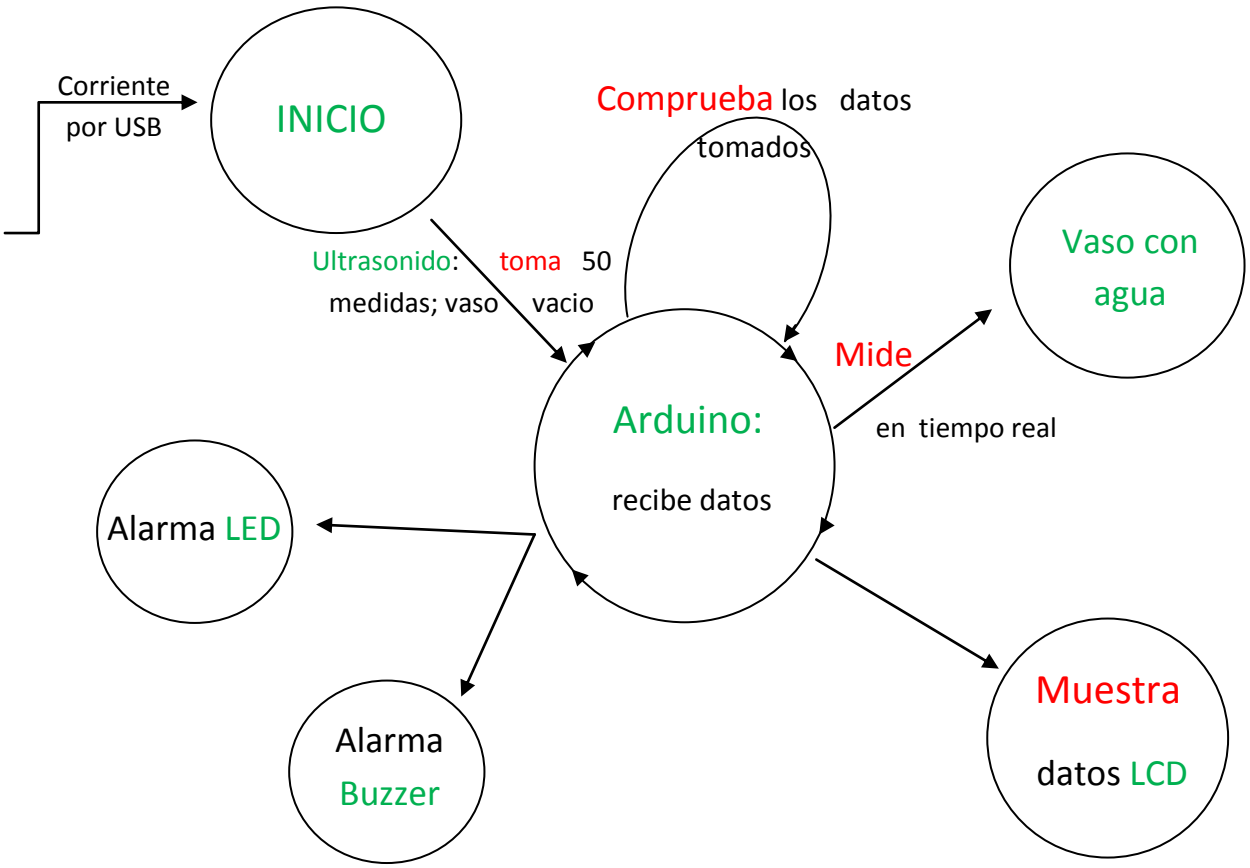
Por eso decidimos incluir en nuestro “invento” varios elementos que nos proporcionan la solución a todos estos problemas. El invento no lleva tinta ni nada que se desgaste con el tiempo por lo que podrá ser usado durante el tiempo que quieras sin necesidad de comprar otro instrumento de medida debido al desgaste o la vejez. El sensor ultrasonidos nos proporciona una medida mucho más precisa a cerca de la cantidad de agua que hay en el recipiente. El proyecto puede ser aplicado a cualquier vaso cilíndrico ya que el mismo sensor toma las medidas del vaso vacío. Si la mesa está inclinada el sensor también lo estará y la medida no se verá afectada.

A demás de todo esto, añadimos al sistema dos elementos como el zumbador y el LED que nos proporcionarán una alarma tanto visual como acústica de la cantidad de agua cuando sobrepasemos determinado nivel de líquido en el vaso en cuestión.

DISEÑO DEL SOFTWARE:

Al tratarse de elementos electrónicos intercomunicados con una pantalla LCD nos permite mostrar los datos tomados por el sensor ultrasonidos y procesado por el Arduino por la pantalla LCD a tiempo real, sin necesidad de pulsar ningún botón ni encender el dispositivo. Simplemente conectando el arduino a corriente por el puerto USB el programa se inicia al momento y comienza a "trabajar". Inmediatamente el programa se ejecuta y ya está listo para su uso.

A grandes rasgos este sería el esquema sobre el software de funcionamiento del programa, en el que intervienen 5 procesos fundamentales que serán explicados posteriormente.



Desarrollo del proceso de funcionamiento:

Como se puede ver en el esquema anterior el proceso electrónico se basa en cinco sub-procesos de comunicación entre el Arduino y los distintos elementos de hardware explicados anteriormente. Al pasarle corriente a la placa de Arduino por medio de la conexión USB insertada en la placa, el programa se inicia automáticamente.

1- El Ultrasonido toma 50 medidas, del vaso vacío, y realiza la media aritmética de los datos tomados.

2- El arduino, gracias al código implantado, comprobará que las medidas tomadas por el sensor de ultrasonidos coinciden con las introducidas por el usuario.

3- A medida que el usuario añade líquido al vaso los sensores toman medidas a tiempo real a cerca de la diferencia de distancia con el vaso vacío.

4- Después de pasar por el arduino y realizar los cálculos pertinentes los resultados se muestran en la pantalla LCD 16x02.

5- Cuando alcanzamos las condiciones de emergencia el Arduino manda una señal al usuario en forma de luz del LED y sonido del zumbador.

Descripción del código:

Lo primero en el código es definir todas las variables que después usará el programa para realizar los cálculos pertinentes. Lo primero que hacemos es lo necesario para el LCD ya que es la que más cables y pines necesita es fundamental para ver si está funcionando correctamente el sensor. Incluimos la librería "LiquidCristal.h" necesaria para el funcionamiento de la pantalla LCD e inicializamos la librería anterior definiendo los pines de la pantalla. También definimos el número de columnas (16) y de filas (2) del LCD.

A continuación definimos los datos del sensor ultrasónico, poniendo a que pin del arduino va cada una de las conexiones del sensor. A demás de definir otras constantes como la velocidad del sonido, el número de lecturas que tomará al comienzo del programa el sensor ultrasónico, las medidas del vaso vacío y del vaso cuando tiene 100ml de agua (que no sería necesario, es únicamente para mejorar la precisión del medidor).

```
// Incluímos la libreria externa para poder utilizarla
#include <LiquidCrystal.h> // Entre los símbolos <> buscará en la carpeta de librerías configurada

// Lo primero is inicializar la librería indicando los pins de la interfaz
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Definimos las constantes
#define COLS 16 // Columnas del LCD
#define ROWS 2 // Filas del LCD


// Configuramos los pines del sensor Trigger y Echo para tenerlos en los pines 8 y 9
const int PinTrig = 9;
const int PinEcho = 8;

// Número de lecturas de la cual vamos a realizar la media
const int numeroLecturas = 50;


// Guardamos la constante de velocidad de sonido en centimetros/sg
const float VelSonido = 34000.0;


// Tomamos las distancia del recipiente cilíndrico que vamos a utilizar a los 100 ml y al recipiente vacío, para después calcular el porcentaje.
const float distancia100 = 6.3;
const float distanciaVacio= 16;
```

A continuación definimos algunas otras variables que usaremos durante el programa. Contadores de: lecturas para almacenar las lecturas tomadas por el sensor (un vector), lecturaActual y total para saber por qué lectura vamos y en qué momento hay que parar de medir e inicializamos la media a 0.

```
float lecturas[numLecturas]; // Vector para almacenar lecturas
int lecturaActual = 0; // Lectura por la que vamos
float total = 0; // Total de las que llevamos
//un contador para saber cuando dejar de tomar leecturas.
float media = 0; // Media de las medidas
bool primeraMedia = false; // Para saber que ya hemos calculado por lo menos una
```

Todos los elementos del medidor de líquidos: inicio del monitor para mostrar los resultado por él, definir los pines Trig y Echo del sensor ultrasonidos (introducidos anteriormente en los pines) como salida y entrada, respectivamente, inicializar el vector numLecturas (definido también previamente) para que vaya guardando las medidas que toma el sensor de ultrasonidos y la configuración de las filas y columnas del LCD; se configuran en la función setup ( ):

```
void setup()
{
  // Iniciamos el monitor serie para mostrar el resultado
  Serial.begin(9600);
  // Ponemos el pin Trig en modo salida
  pinMode(PinTrig, OUTPUT);
  // el pin Echo en modo entrada
  pinMode(PinEcho, INPUT);
```

```
  // Inicializamos el vector
   int i;
   for (i = 0; i < numLecturas; i++)
   {
     lecturas[i] = 0;
   }
   // Configuramos las filas y las columnas del LCD en este caso 16 y 2 filas
   lcd.begin(COLS, ROWS);
}
```

Es aquí cuando comienza la función “void loop()”, que se repite constantemente en el proceso de ejecución y toma datos y los procesa en el momento.

```
void loop()
{
  |
  // Eliminamos la última medida
  total = total - lecturas[lecturaActual];

  iniciarTrigger();

  // La función pulseIn obtiene el tiempo que tarda en cambiar entre estados, en este caso a HIGH
  unsigned long tiempo = pulseIn(PinEcho, HIGH);

  // Obtenemos la distancia en cm, hay que convertir el tiempo en segudos ya que está en microsegundos
  // por eso se multiplica por 0.000001
  float distancia = tiempo * 0.000001 * VelSon / 2.0;

  // Almacenamos la distancia en el vector
  lecturas[lecturaActual] = distancia;
```

En esta parte del programa iniciamos desde 0 el número de lecturas ya que esta función se repetirá indefinidas veces y cada vez que comience a medir empieza otra vez a contar el numero de medidas desde 0. “pulseIn()” lo usamos para que el sensor ultrasonidos deje un tiempo entre cada distancia que mide. Obtenemos la distancia desde el ultrasonido hasta el agua, que se usa para saber la cantidad de agua que contiene el vaso. La cantidad de agua del vaso se calcula mediante la siguiente formula. Gracias a ella ya almacenamos la cantidad de agua que hay.

```
// Añadimos la lectura al total
total = total + lecturas[lecturaActual];

// Avanzamos a la siguiente posición del vector
lecturaActual = lecturaActual + 1;

// Comprobamos si hemos llegado al final del vector
if (lecturaActual >= numLecturas)
{
  primeraMedia = true;
  lecturaActual = 0;
}

// Calculamos la media
media = total / numLecturas;

// Solo mostramos si hemos calculado por lo menos una media
if (primeraMedia)
{
  float distanciaLleno = distanciaVacio - media;
  float cantidadLiquido = distanciaLleno * 100 / distancia100;
  int porcentaje = (int) (distanciaLleno * 100 / distanciaVacio);
```

Añadimos una distancia para que sepa por cual va y termine, es decir, avanzamos una cantidad en el número de distancias. Con el “if()” lo que hacemos es que cuando almacenemos el numero de distancias, el programa sabrá cuando tiene que parar de medir. El programa calcula a continuación la media de las medidas tomadas, para mejorar así la precisión de las medidas tomadas.

Finalmente calculamos la cantidad de líquido que tenemos a demás del porcentaje del vaso lleno, lo cual será mostrado después por el LCD para que lo vea el usuario.

```
//Vamos a poner un led  que avise cuando el vaso esté a un nivel del 50%
if(porcentaje>50)
  digitalWrite (9, HIGH);
else
  digitalWrite (9,LOW);


// Mostramos en la pantalla LCD los datos que se van obteniendo.
lcd.clear();
// Cantidad de líquido
lcd.setCursor(0, 0);
lcd.print(String(cantidadLiquido) + " ml");
```

```
  // Porcentaje
  lcd.setCursor(0, 1);
  lcd.print(String(porcentaje) + " %");


  // Mostramos las medidas por el monitor del arduino con el serial print, donde vamos a tener la media
  // la cantidad de liquido que se encuentra en ese momento en el vaso, y el porcentaje de llenado.

  Serial.print(media);
  Serial.println("cm");

  Serial.print(cantidadLiquido);
  Serial.println("ml");
```

Para finalizar el programa ponemos la condición para que se encienda el LED e imprimimos los datos tomados y calculados (cantidad de agua y porcentaje del vaso lleno) y los mostramos por la pantalla LCD.

```
    // Porcentaje
    lcd.setCursor(0, 1);
    lcd.print(String(porcentaje) + " %");


    // Mostramos las medidas por el monitor del arduino con el serial print, donde vamos a tener la media
    // la cantidad de liquido que se encuentra en ese momento en el vaso, y el porcentaje de llenado.

    Serial.print(media);
    Serial.println("cm");

    Serial.print(cantidadLiquido);
    Serial.println("ml");

    Serial.print(porcentaje);
    Serial.println("%");
  }
  else
  {
    lcd.setCursor(0, 0);
    lcd.print("Calculando: " + String(lecturaActual));
  }
  delay(100);
}

void(loop)
{
   sensor.requestTemperatures();  //Hace las lecturas de la temperatura.

   Serial.print("Temperatura en sensor 0: ");
   Serial.print(sensor.getTempCByIndex(0));
   Serial.println(" °C");

   delay(500);
}

// Método que inicia la secuencia del Trigger para comenzar a medir
void iniciarTrigger()
{

  // Comenzamos poniendo el pin Trigger en estado bajo
  digitalWrite(PinTrig, LOW);

  // Ponemos el Triiger en estado bajo y esperamos 2 ms
  digitalWrite(PinTrig, LOW);
  delayMicroseconds(2);

  // Ponemos el pin Trigger a estado alto y esperamos 10 ms
  digitalWrite(PinTrig, HIGH);
  delayMicroseconds(10);
}
```

Por último el programa comienza otra vez a tomar medidas para que el proceso se repita indefinidamente.

Circuito electrónico:

Tras probar todos los elementos del hardware para comprobar su funcionamiento, montamos el circuito electrónico acorde a las necesidades. Hemos usado cables macho hembra para hacer los cables del ultrasonido más largos y facilitar la manejabilidad del sensor. Para montarlo hemos usado las mismas conexiones de los de hardware individual, cambiando los pines de cada uno de acuerdo con el programa que ejecutaremos en la placa de Arduino.

Para añadir el LED simplemente hemos conectado un cable al positivo y la patilla larga del LED, pasando antes por la resistencia de 220 ohmios para no quemarlo. EL zumbador También tiene una conexión muy simple al circuito por lo que no fue difícil conectarlos posteriormente.

Producto final: