# Universidad Nacional de La Matanza Departamento de Ingeniería e Investigaciones Tecnológicas

**Sistemas Operativos Avanzados**

**SISTEMAS EMBEBIDOS-IoT**

**Días de Cursada:** Miércoles **Turno:** Noche

# Docentes: Lic. Graciela de Luca

**Ing. Waldo Valiente Ing. Esteban Carnuccio Ing. Sebastian Barillaro**

**Integrantes:**

## Agustin, Balart DNI: 35.537.140

Alesandrini, Ernesto DNI: 33.458.573

Famiglietti, Cristian DNI: 29.284.170

D’amico, Rodrigo DNI: 36.817.422

Cruz Rey, Juan DNI: 36.921.33

***Introducción***

A lo largo de este informe se pretende dar una explicación acerca de nuestra experiencia aprendiendo sobre Sistemas Embebidos, uso de sensores y actuadores, uso de placa Arduino, su comunicación a través de una conexión Bluetooth al SO Android, entre otros. El objetivo de este trabajo práctico es alejarse un poco de lo que veníamos acostumbrados a trabajar: solo con software; para poder dedicarnos también al hardware (electrónica). A lo largo de este informe expondremos como y que elementos desarrollamos en nuestro sistema.

***Sistema SafeRoom***

***Desarrollo:***

El problema a resolver es la detección de cantidades peligrosas de monóxido de carbono y altas temperaturas con el objetivo de alertar y actuar en consecuencia. Para ello se usaron los siguientes sensores y actuadores:

**Actuadores:**

* Servo
* Led
* Un cooler de pc
* Buzzer v1.1b
* Display

**Sensores:**

* Sensor de temperatura y humedad
* Sensor de monóxido
* Bluetooth

**Otras componentes:**

* Cable USB-microUSB
* Cable macho/hembra y macho/macho
* Placa Arduino Uno

**Funcionamiento general:**

Si el rango de monóxido y de temperatura es mayor a su respectiva cota inferior y es menor a su respectiva cota superior, se muestra en el display “CUIDADO”, el servo se acciona simulando la apertura de una ventana, y se acciona el cooler para ventilar el abiente. Ahora si ambos superan su respectiva cota superior, el display muestra “PELIGRO”, y se enciende un led, suena el buzzer a modo de alarma y se aumenta la velocidad del cooler para ventilar el ambiente ya que se ha llegado a un valor critico.

**Pasos para la creación del sketch en arduino:**

Se programó un programa en la computadora, estableciendo por código la forma de capturar los datos de los sensores y el accionar de los actuadores como describimos en la etapa de “FUNCIONAMIENTO.” Usamos el concepto de cross/compiler; o sea se compiló el programa en la computadora y se pasó el sketch a la placa arduino. Para eso, previamente tuvimos conectado la computadora al arduino mediante puerto usb, y también tenemos los sensores y actuadores mediante los pines del adruino.

**Funcionamiento del sketch Arduino**

Contamos con los sensores de monóxido, de temperatura y humedad que me indican el nivel de los mismos que hay en una sala, como ser de oficina. Mediante el display se reflejan dichos datos.

Los actuadores actuaran en consecuencia a dichos valores. El accionar de los mismos está automatizado. Entonces si el rango de monóxido y de temperatura es mayor a su respectiva cota inferior y es menor a su respectiva cota superior se ponemos en funcionamiento algunos de nuestros actuadores de la siguiente manera:

* Se abre una ventana con el accionar de un servomotor para permitir que el monóxido salga por la misma.
* La ventilación del aire impuro se realiza mediante el accionar de un cooler.
* Se genera un mensaje de alerta en el display: “CUIDADO”

Ahora si los sensores de monóxido y de temperatura están por encima de su respectiva cota máxima, entonces pondremos en funcionamiento algunos de nuestros actuadores de la siguiente forma:

* Debido a la criticidad de la impureza del aire, el cooler mediante el concepto de pwm, va a variar su velocidad, aumentando la misma al máximo para desparramar esa impureza por la ventana.
* Se enciende y apaga un led para avisar sobre dicha criticidad.
* Suena una alarma para advertir del problema
* Se envía un mensaje de advertencia en el display que dice “PELIGRO”

A todo esto, contamos con un módulo bluetooth HC-06 (esclavo) conectado a la placa Arduino, que se queda esperando peticiones de conexión de otros dispositivos. Cuando detecta que un dispositivo móvil con Android se conectó, dicha placa Arduino le envía los datos de monóxido, temperatura y humedad captados por dicho sensores, previamente solicitado por el celular. Y recibe el accionar de sus actuadores mediante la App SafeRoom instalada en el dispositivo móvil. Tode esto mediante la conexión bluetooth.

**Pasos para la creación de la App SafeRoom con AndroidStudio:**

Se usó el AndroidStudio para crear la aplicación, donde se estableció las diferentes activities y las view que componente cada activity. Y por medio de cross/compiler se compiló el programa y se instaló la APK SafeRoom en el dispositivo móvil.

**Funcionamiento de la APK SafeRoom:**

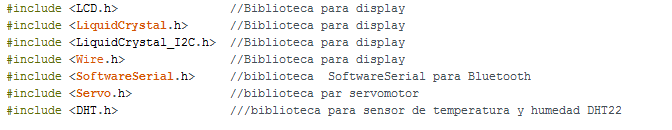
Dicha aplicación necesita ser conectada mediante bluetooth a la placa arduino. Cuando presiono el botón “información” de la placa Arduino, el móvil le envía una solicitud de los datos necesitados a la placa. Y la misma responde trayendo valores de monóxido, humedad y temperatura. El usuario visualiza estos datos en la app. Si el mismo detecta un nivel elevado de impureza en el aire, puede tomar decisiones que producirá el accionar de algunos actuadores conectados al arduino vía bluetooth:

* Si se desea ventilar la impureza del aire, el usuario presiona el botón “Encender ventilador”.
* Si el valor es crítico, el usuario optará por presionar el botón “Activar alarma”

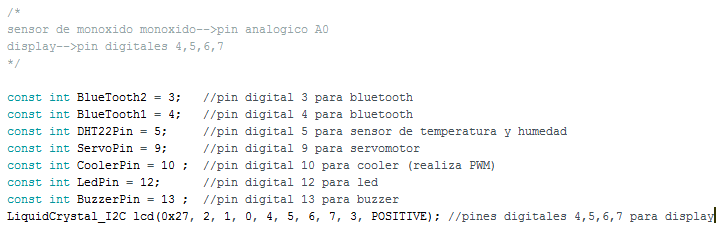
***Capturas del código de Arduino***

**Bibliotecas**

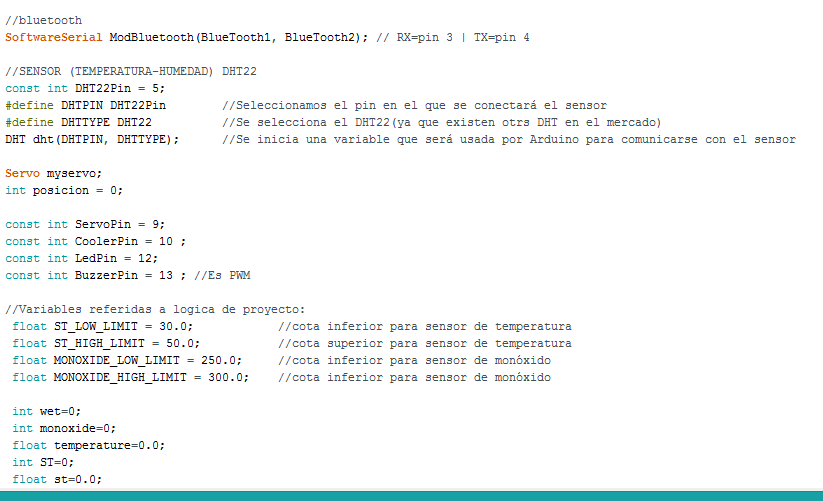
Por defecto el bluetooth usa los pines 0 (Rx) y 1 (Tx). Pero esto no me deja debuggear porque imposibilita hacer serial.Print por consola. Esto lo decidimos solucionar, usando la biblioteca SoftwareSerial que nos permite elegir otros pines digitales para usar el bluetooth

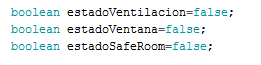


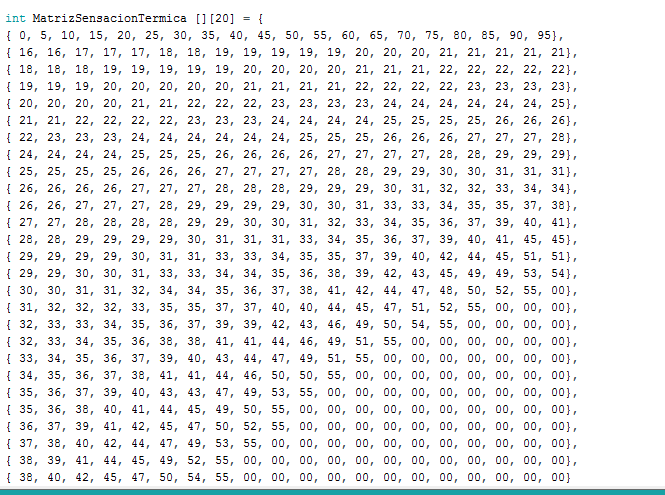
**Pines asignados a cada sensor o actuador**



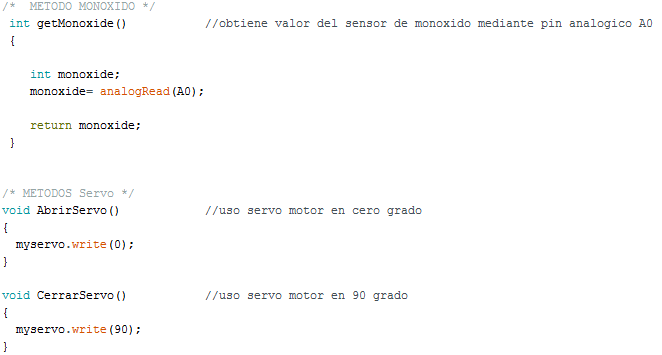
**Inicialización de variables**

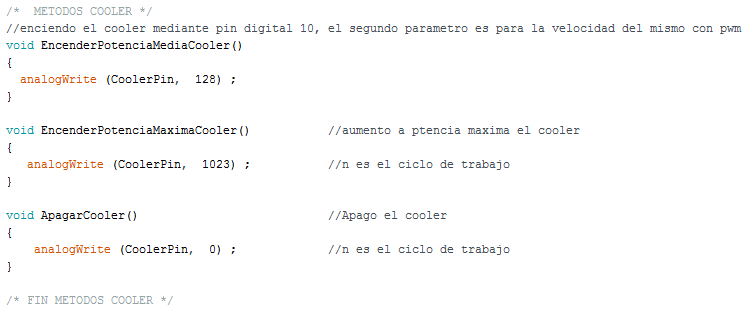


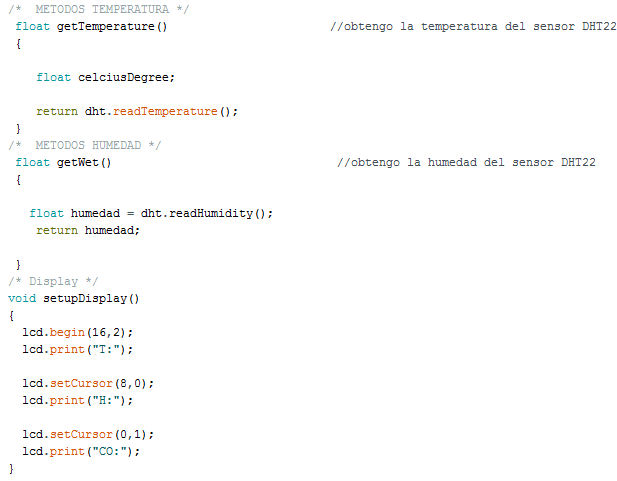


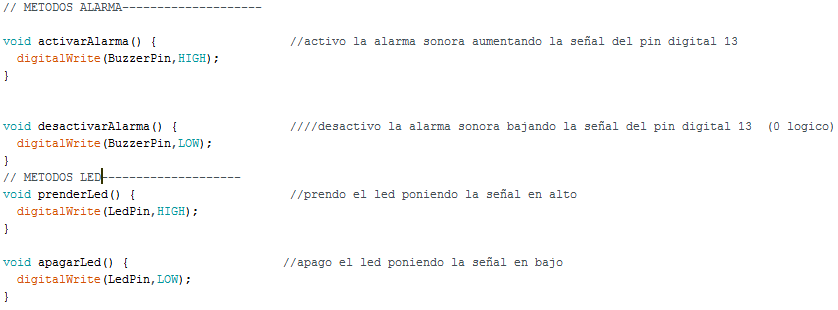


**Métodos**

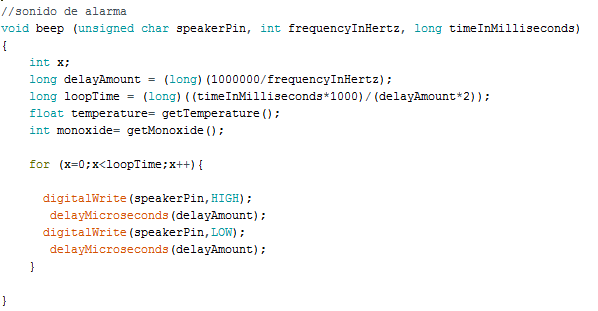


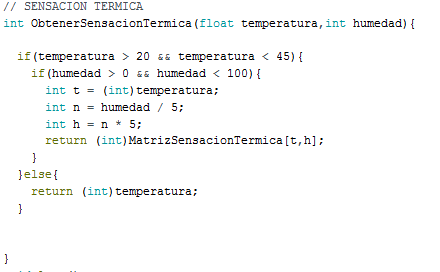


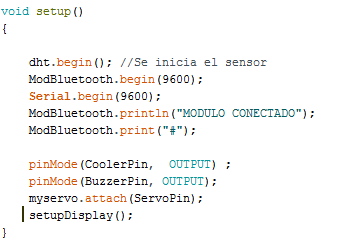




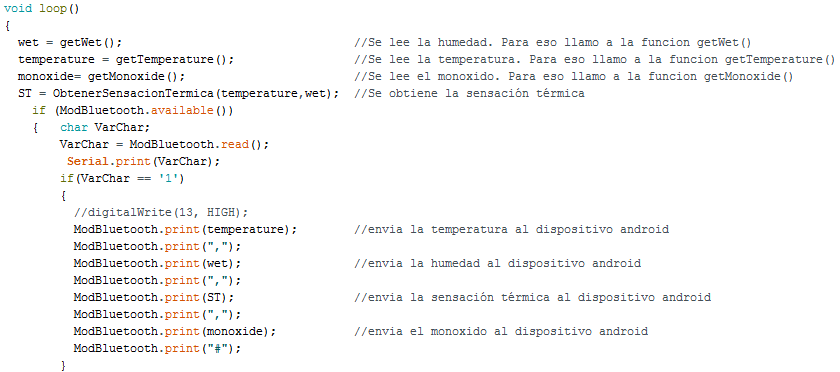
**Setup**

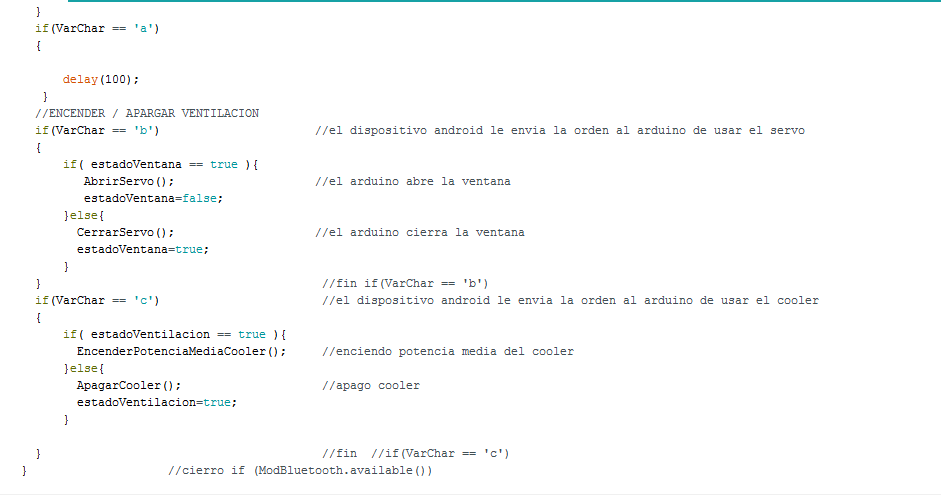


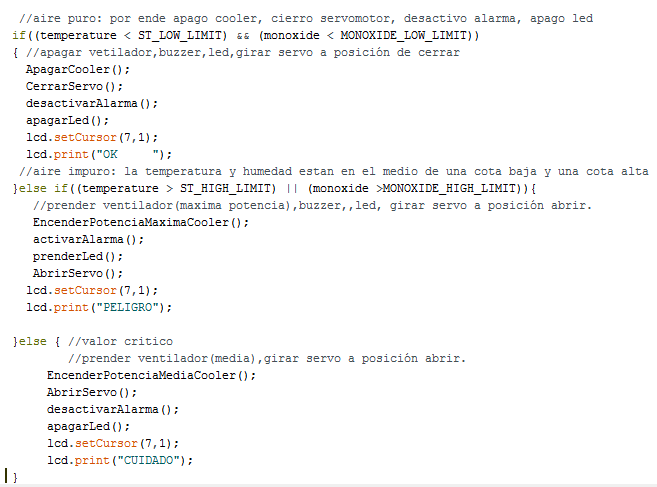


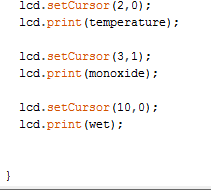


**Loop**









***Explicaciones de actuadores***

**Servomotor Tower Pro SG92R 9G**

El Servomotor Tower Pro SG92R 9G tiene un ángulo de giro de hasta 180 grados.

Los 3 cables en el conector están distribuidos de la siguiente forma: Rojo =Alimentación (+), Café = Alimentación (–) o tierra, Naranja= Señal PWM.

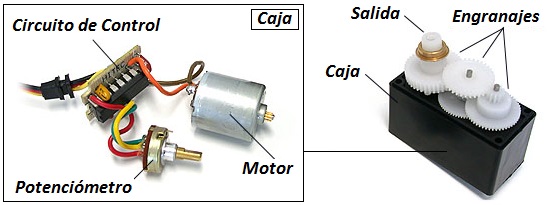
Tiene requerimientos de energía bastante bajos y se permite alimentarlo con la misma fuente de alimentación que el circuito de control. Por ejemplo, si se conecta a una tarjeta Arduino, se puede alimentar durante las pruebas desde el puerto USB del PC sin mayor problema.

**Características**

* Micro Servo Tower-pro
* Velocidad: 0.10 sec/60° @ 4.8V
* Torque: 1.8 Kg-cm @ 4.8V
* Voltaje de funcionamiento: 3.0-7.2V
* Temperatura de funcionamiento: -30 ℃ ~ 60 ℃
* Ángulo de rotación: 180°
* Ancho de pulso: 500-2400 µs
* Longitud de cable de conector: 24.5cm

Los servos son motores de corriente continua (CC), pero en lugar tener un giro continuo que podamos aprovechar (para mover un molino por ejemplo), están preparados para moverse a un ángulo fijo en respuesta a una señal de control, y mantenerse fijos en dicha posición.

Un servo principalmente está formado por un conjunto reductor (engranajes), un motor de CC y por último por un circuito de control, aunque en la práctica se comporta como un bloque funcional que posiciona su eje en un ángulo preciso en función de la señal de control.

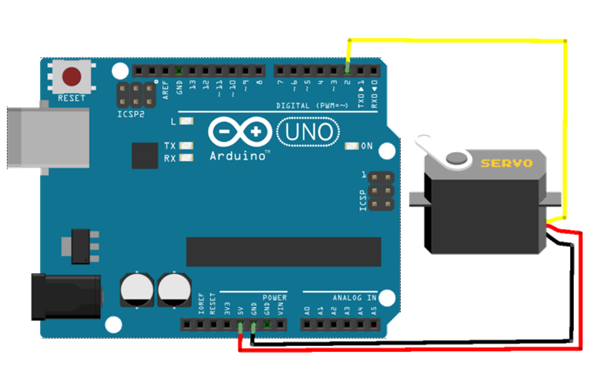


El control se realiza mediante una señal de control PWM, en la que el ancho el pulso indica el ángulo que deseamos que adopte el eje.

Tiene dos características especiales:

Por un lado, nos permite **mantener una posición** que indiquemos, siempre que esté dentro del rango de operación del propio dispositivo. Por otro lado nos permite **controlar la velocidad de giro**, podemos hacer que antes de que se mueva a la siguiente posición espere un tiempo.

## Conexionado con Arduino



**DISPLAY LCD 16X2 BACKLIGHT AZUL**

LCD alfanumérico de 2 filas por 16 columnas. Esto significa que puede mostrar 2 líneas x 16 caracteres cada una.

Tiene Controlador integrado LCD estándar HD44780.

Opera con 5V DC.

Consumo de corriente con luz de fondo total: 25 mA máx

Dimensión del módulo: 80mm x 35mm x 11mm.

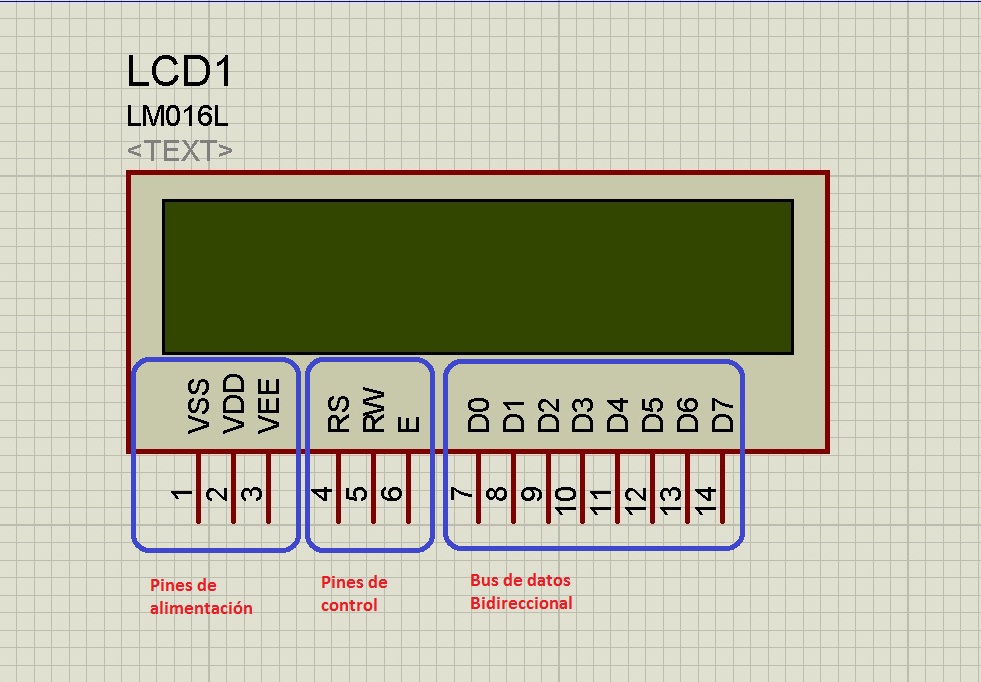
Tamaño de la zona de display: 64.5mm x 16mm

En esta entrada vamos a explicar el funcionamiento y conexionado de un LCD con una resolución de 16x2.

El LCD (Liquid Crystal Dysplay) o pantalla de cristal líquido es un dispositivo empleado para la visualización de contenidos o información de una forma gráfica, mediante caracteres y símbolos. Está gobernado por el microcontrolador HD44780, el cual dirige todo su funcionamiento.

**¿Cómo es su conexionado?**

Lo  podemos dividir en los Pines de alimentación, pines de control y los pines del bus de datos bidireccional.



**Pines de alimentación:**

***Vss:*** Gnd  
***Vdd:*** +5 voltios  
***Vee:*** corresponde al pin de contraste, lo regularemos con un potenciómetro de 10K conectado a Vdd.

**Pines de control:**

***RS:*** Corresponde al pin de selección de registro de control de datos (0) o registro de datos(1). Es decir el pin RS funciona paralelamente a los pines del bus de datos. Cuando RS es 0 el dato presente en el bus pertenece a un registro de control/instrucción. y cuando RS es 1 el dato presente en el bus de datos pertenece a un registro de datos o un carácter.  
***RW:*** Corresponde al pin de Escritura(0) o de Lectura(1). Nos permite escribir un dato en la pantalla o leer un dato desde la pantalla.  
***E:*** Corresponde al pin Enable o de habilitación. Si E (0) esto quiere decir que el LCD no esta activado para recibir datos, pero si E (1) se encuentra activo y podemos escribir o leer desde el LCD.

**Pines de Bus de datos:**

El Bus de datos bidireccional comprende desde los pines D0 a D7. Para realizar la comunicación con el LCD podemos hacerlo utilizando los 8 bits del bus de datos(D0 a D7) o  empleando los 4 bits mas significativos del bus de datos(D4 a D7).

**DDRAM y CGROM?**

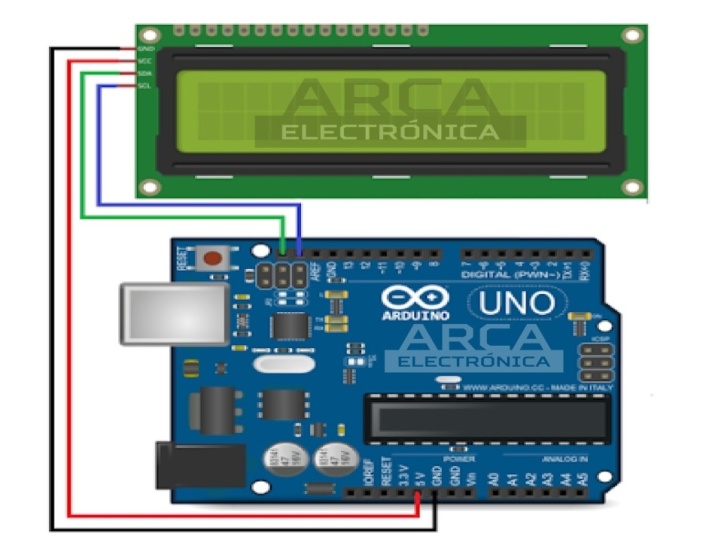
Son las dos zonas de memoria del LCD.

La memoria DDRAM (Data Display Ram): corresponde a una zona de memoria donde se almacenan los caracteres que se van a representar en pantalla. Es decir es la memoria donde se almacenan los caracteres a mostrar con su correspondiente posición.

La memoria CGROM es una memoria interna donde se almacena una tabla con los caracteres que podemos visualizar en el lcd.

La memoria CGRAM (Character Generator Ram): en ella se pueden almacenar nuestros propios caracteres.

**Conexión de display al arduino**



**BUZZER PIEZOELÉCTRICO PASIVO V1.1B**

Primero pasaremos a definir que significa piezoeléctrico. El mismo es un fenómeno que ocurre en determinados cristales que, al ser sometidos a tensiones mecánicas, adquieren una polarización eléctrica y aparece una diferencia de potencial y cargas eléctricas en su superficie que generan una tensión eléctrica.

Este fenómeno también ocurre a la inversa: se deforman bajo la acción de fuerzas internas al ser sometidos a un campo eléctrico. El efecto piezoeléctrico es normalmente reversible: al dejar de someter los cristales a un voltaje exterior o campo eléctrico, recuperan su forma.

Es decir, que son materiales (el cuarzo es el más conocido) que si los sometemos a una tensión eléctrica variable (como una señal PWM, que ya nos son familiares) vibran.

En los circuitos electrónicos digitales, suelen disponer de un reloj interno que vibra a una velocidad patrón, basados en cristales de cuarzo piezoeléctrico. El cristal de Arduino late a 16Mhz por segundo y la flecha de la foto de abajo indica la posición del mismo.



Si conectamos un piezo con una señal digital, vibran a una frecuencia sigue bastante fielmente la variación eléctrica con que los excita, y si vibran a la frecuencia audible, oiremos el sonido que producen. A un componente que hace esto, le llamamos Buzzer o zumbador. Recordemos que el oído humano escucha en un rango de frecuencia audible, aproximadamente entre 20 Hz – 20 kHz.

Con esto, es suficiente para generar tonos audibles (como la típica alarma de los despertadores digitales) e incluso tonos musicales que podemos secuenciar.

La función Beep() de arduino me permite provocar un sonido. Lo único que beep () hace es poner una señal PWM en el pin 9 de 20 sobre 255.

¿Qué pasa si quiero generar señales de tono variable para hacer una melodía? En tal caso Arduino Uno dispone de la función tone() que genera una señal de la frecuencia de audio y duración indicadas, y notone() que la corta.

. El piezo puede conectarse a salidas digitales y emitirá un tono cuando la salida sea ALTA. Alternativamente, se puede conectar a una salida de modulación analógica de ancho de pulso para generar varios tonos y efectos.

**Especificaciones**

|  |  |
| --- | --- |
| Tensión de funcionamiento | 4-8V |
| Salida de sonido | ≥85dB |
| Frecuencia de resonancia | 2300 ± 300Hz |

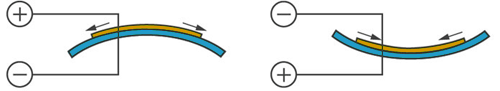
Encontramos 2 tipos de buzzer:

· **Buzzer ACTIVO**: Esta clase de zumbador, emite sonido cuando lo conectamos directamente a la fuente de alimentación.  
  
· **Buzzer PASIVO**: En cambio, esta clase de zumbador, solo emite sonido si lo excitamos mediante un tren de pulsos.

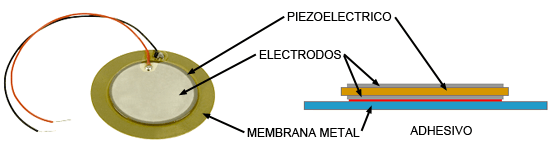
En nuestro proyecto decidimos usar buzzer pasivo. Entraremos a profundizar sobre estos mismos. Los buzzer pasivo son dispositivos que **permiten convertir una señal eléctrica en una onda de sonido**. Estos dispositivos no disponen de electrónica interna, por lo que tenemos que proporcionar una señal eléctrica para conseguir el sonido deseado. En oposición, los buzzer activos disponen de un oscilador interno, por lo que únicamente tenemos que alimentar el dispositivo para que se produzca el sonido. Pese a tener la complejidad de proporcionar y controlar nosotros la señal eléctrica, los buzzer pasivos **tienen la ventaja de que podemos variar el tono emitido** modificando la señal que aplicamos al mismo, lo que nos permite generar melodías.

Frecuentemente, el buzzer pasivo se acompaña en una placa para facilitar su conexión, que incorpora un transistor y resistencias necesarias para hacer funcionar el buzzer pasivo sin más que conectarlo.

Como hemos mencionado anteriormente, los buzzer son **dispositivos que convierten señales eléctricas en sonido**. **Los buzzer son transductores piezoeléctricos**. Los materiales piezoeléctricos tiene la propiedad especial de variar su volumen al ser atravesados por corrientes eléctricas.

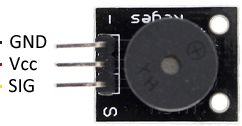


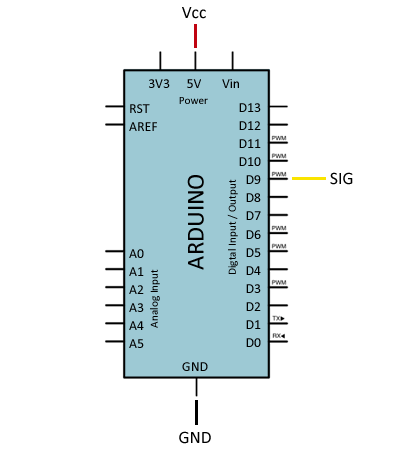
Un buzzer aprovecha este fenómeno para hacer vibrar una membrana al atravesar el material piezoeléctrico con una señal eléctrica.



**CONECCION DE BUZZER CON ARDUINO**

La conexión es tan simple como conectar negativo a GND y positivo al pin 9. De todas maneras hay que tener cuidado. Los piezos tienen polaridad y hay que asegurarse de conectarlos correctamente.Si los conectáis al revés, simplemente no sonará, y tendréis que dar la vuelta. Ahora nos resta conectar el VCC al arduino





COOLER

Introducción

Para nuestro proyecto, decidimos usar un cooler de informática, ya que los mismos están estandarizados, lo que evita posibles errores de conexión. Esto es así salvo en marcas concretas que personalizan sus componentes, algo que tiende a desaparecer en la actualidad. Aun así tenemos cooler con distintas estandarización:

* Cooler de 2 cables
* Cooler de 3 cables
* Cooler de 4 cables

Cooler de 2 cables

Lo habitual es que sean de color rojo y negro, indicando el polo positivo y negativo respectivamente.

Cooler de 3 cables

Los ventiladores de tres pines suelen tener los cables de color negro, rojo y amarillo. Las líneas roja y negra siguen siendo las de alimentación, es decir positivo y negativo, y la tercera es el llamado "tacómetro". Esta línea, en las placas que dispongan de ella, indica las vueltas que las aspas del ventilador pueden dar durante un minuto. Se denomina principalmente con la abreviatura R.P.M.

Cooler de 4 cables

Esta es la última especificación introducida en ventiladores para informática. Los colores pueden tener cualquiera de estas combinaciones:

1. Negro - Rojo - Amarillo - Azul
2. Negro - Amarillo- Verde - Azul
3. Negro - Rojo - Verde - Azul

Los más comunes son el 1 y el 2. Hay que tener en cuenta que a veces el amarillo que está en tercer lugar puede ser de color blanco. Por lo demás sus funciones son siempre las mismas.

El negro será como siempre el negativo, el rojo en segundo lugar el positivo y el amarillo o verde en tercer lugar el tacómetro.

El cuarto cable es el controlador de velocidad de tecnología PWM, que por medio de pulsos digitales controla la velocidad de giro. Esto es particularmente útil cuando necesitamos controlar automáticamente su velocidad.

En nuestro proyecto usamos la siguiente especificación:

Negro - Amarillo- Verde - Azul

***Explicación de sensores***

**Sensor MQ**

Antes de explicar sensor de monóxido, vamos a aprender sobre los MQ en general.

Los sensores de gas de la serie MQ son sensores analógicos.

Estos sensores son electroquímicos y varían su resistencia cuando se exponen a determinados gases, internamente posee un calentador encargado de aumentar la temperatura interna y con esto el sensor pueda reaccionar con los gases provocando un cambio en el valor de la resistencia. El calentador dependiendo del modelo puede necesitar un voltaje entre 5 y 2 voltios, el sensor se comporta como una resistencia y necesita una resistencia de carga (RL) para cerrar el circuito y con este hacer un divisor de tención y poder leerlo desde un microcontrolador.

Estos módulos también tienen una salida digital la cual internamente trabaja con un comparador y con la ayuda de un potenciómetro podemos calibrar el umbral y así poder interpretar la salida digital como presencia o ausencia del gas.

La diferencia entre los distintos tipos de sensores MQ es la sensibilidad a cierta gama de gases, más sensibles a algunos gases que a otros, pero siempre detectan a más de un gas.

### El calentador

El voltaje para el calentador interno es muy importante. Algunos sensores usan 5V para el calentador, otros necesitan 2V. Este calentador es necesario para elevar la temperatura del sensor. . Mientras el calentador no alcance la temperatura de funcionamiento, la lectura del sensor no será fiable.

El tiempo de calentamiento depende de cada modelo de sensor. En la mayoría de modelos es suficiente para con unos pocos minutos.

### Resistencia de carga

El sensor necesita una resistencia de carga en la salida a tierra. Su valor podría ser de 2kOhm a 47kOhm. Cuanto menor es el valor, menos sensible. Cuanto mayor sea el valor, menos precisa será para concentraciones más altas de gas.

Si se mide solo un gas específico, la resistencia de carga se puede calibrar aplicando una concentración conocida de ese gas.

Todos los modelos MQ disponen de un calentador necesario para elevar la temperatura del sensor, y que sus materiales adquieran la sensibilidad. Mientras el calentador no alcance la temperatura de funcionamiento, la lectura del sensor no será fiable.

El tiempo de calentamiento depende de cada modelo de sensor. En la mayoría de modelos es suficiente para con unos pocos minutos

**Sensor de Monóxido de Carbono MQ7**

Este sensor es de alta sensibilidad al monóxido de carbono (CO), pero también es sensible al H2 (Dihidrógeno, de fórmula química H2, gas cuya molécula se compone de 2 átomos de hidrógeno).

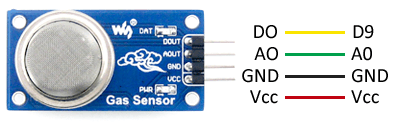
Este es un sensor fácil de usar para detección de Monóxido de Carbono (CO), ideal para detectar concentraciones de CO en el aire. El MQ-7 puede detectar concentraciones en el rango de 20 a 2000ppm.  
  
Este sensor tiene una lata sensibilidad y un corto tiempo de respuesta.

El módulo posee una salida analógica que proviene del divisor de voltaje que forma el sensor y una resistencia de carga. También tiene una salida digital  que se calibra con un potenciómetro, esta salida tiene un led indicador. Todo lo que necesitas es proporcionarle 5V, añadir una resistencia de carga y conectar la salida a una entrada ADC de un microcontrolador.

 El calentador usa una tensión alterna de 5V y 1.4V. Y para q el sensor arroje los valores deseados, se necesita esperar a dicho calentador por 2 minutos.

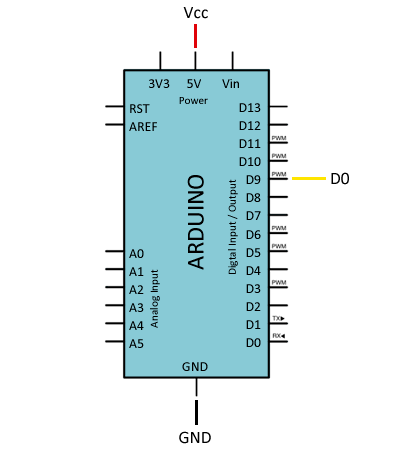
**Conexión mq7 con arduino**

El esquema eléctrico es sencillo. Alimentamos el módulo conectando GND y 5V a los pines correspondientes de Arduino.

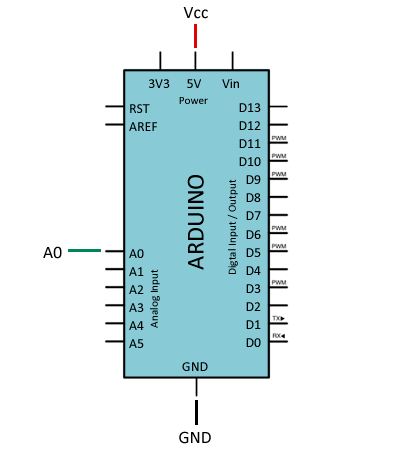


Ahora si queremos usar la lectura digital, conectamos la salida DO a una de las [entradas digitales](https://www.luisllamas.es/entradas-digitales-en-arduino/) de Arduino.

La conexión vista desde Arduino quedaría así,



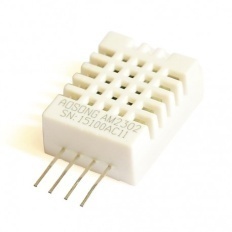
Si quisiéramos emplear el valor analógico, simplemente conectaríamos la salida AO del sensor a una [entrada analógica](https://www.luisllamas.es/entradas-analogicas-en-arduino/) de Arduino. (En nuestro proyecto usamos salida analógica)



**SENSOR DHT22**

El DHT22 es un  sensores digitales de Temperatura y Humedad, fáciles de implementar con cualquier microcontrolador.  Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante y solo un pin para la lectura de los datos. Tal vez la desventaja de estos es la velocidad de las lecturas y  el tiempo que hay que esperar para tomar nuevas lecturas (nueva lectura después de 2 segundos), pero esto no es tan importante puesto que la Temperatura y Humedad son variables que no cambian muy rápido en el tiempo.

El rango de  medición de temperatura es de  -40°C a 80 °C con precisión de ±0.5 °C y rango de humedad de 0 a 100% RH (humedad relativa) con precisión de 2% RH, el tiempo entre lecturas debe ser de 2 segundos.



**FUNCIONAMIENTO**

### ****CICLO DE OPERACIÓN.****

Es el tiempo que el sensor tarda en ofrecer una respuesta desde que se le pide.Formato de datos de un solo bus para la comunicación y sincronización entre MCU y el sensor. El proceso de comunicación es de 4 ms aproximadamente.   
Una transmisión de datos completa es de 40 bits. Donde obtenemos la temperatura y la humedad. 

Ejemplo: Recibimos 40 bits:

* **0011 0101 0000 0000 0001 1000 0000 0000 0100 1101**

**Los primeros 8 bit son de humedad en alto-high**

**Los segundos 8 bit son de(humedad en bajo-low**

**Los terceros 8 bit son de temperatura en alto-high**

**Los cuartos 8 bit son de temperatura en bajo-low**

**Los últimos 8bit son de paridad**

**Tenemos un total de=40 bits**

Calculando：

* **0011 0101+0000 0000+0001 1000+0000 0000= 0100 1101**

Datos correctos recibidos：

* Humedad：0011 0101 = 35H = **53%RH**
* Temperatura：0001 1000 = 18H = **24℃**

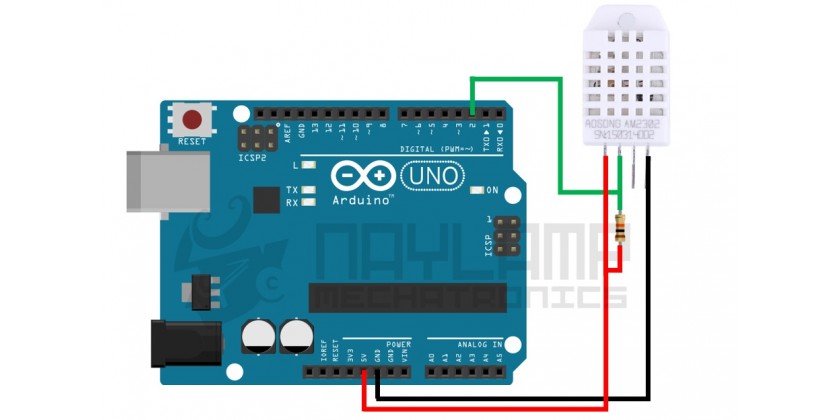
El microcontrolador externo y el microcontrolador que lleva integrado el sensor, se hablan entre sí de la siguiente manera:

* Se inicia la comunicación.
* El sensor responde estableciendo un nivel bajo de 80us y un nivel alto de 80us.
* El sensor envía 5 bytes con la información de temperatura y humedad.

**Conexiones entre Arduino y el sensor DHT22**



Tiene 4 pines. Pero para nuestro proyecto nos basta con usar el pin 1 (para voltaje), pin 2 (para lectura de datos digitales). Este pin se conecta al microcontrolador y el pin 4 (GND)



**BLUETOOTH**

# Bluetooth HC-05 y HC-06

Los módulos de**bluetooth HC-05 y HC-06 sirve para la comunicación entre distintos dispositivos.**

Además de recibir conexiones desde un dispositivo bluetooth, también es capaz de generar conexiones hacia otros dispositivos bluetooth. Ahora Un módulo Bluetooth HC-06 se comporta como esclavo, esperando peticiones de conexión. Presentan un modo de configuración y un modo de utilización y se configuran mediante comandos AT.