3 Entradas y salidas

Entradas y Salidas

Vamos a profundizar más en el Arduino y veremos como podemos añadir dispositivos de entrada y salida que nos permitirá la interacción más emocionante

https://giphy.com/embed/Cwtfo6IwfkSbK

via GIPHY

Si tienes dudas técnicas en este capítulo pon un ticket a http://soporte.catedu.es/ y te ayudaremos:

También tienes nuestro whatsapp o telegram en la web www.catedu.es en información (preferible)

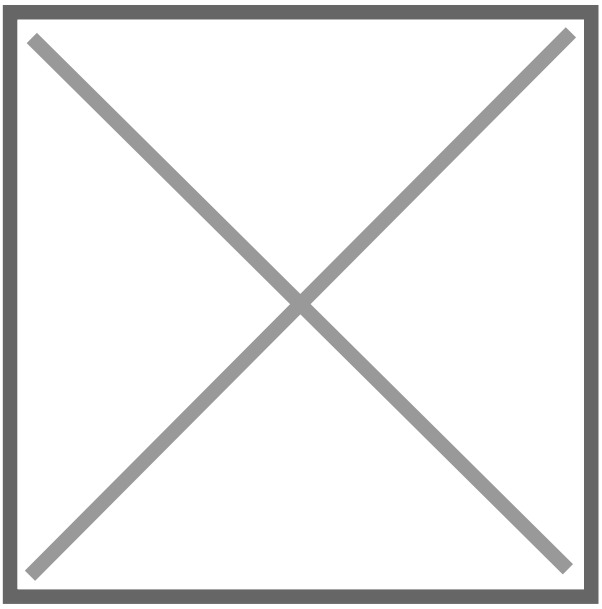
Intensidad LED

Montaje 5: Control de la intensidad de iluminación de un LED

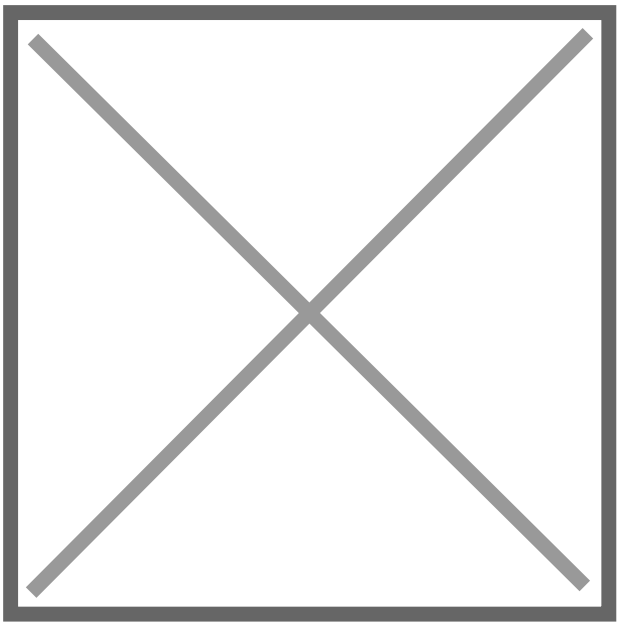
Como ejemplo práctico de la señal PWM vamos a realizar un control de iluminación sobre un diodo led.

Montaje 5 Sin EDUBASICA

En una protoboard montamos el circuito formado por el **potenciómetro** conectado a la entrada analógica A0.



y también montamos el circuito conectado al pin digital D3, al D4 o al D5 como quieras, utilizado como salida PWM, de esta manera nos va a permitir variar la luminosidad del LED.



Aquí en la ilustración puedes ver las conexiones del potenciómetro al A0 y el LED al D5

Montaje 5 CON EDUBÁSICA

En este caso ya tiene integrado un potenciómetro conectado a la entrada A0.

Esquema del potenciómetro en EDUBASICA

Y vamos a utilizar de salida el diodo verde conectado a D3 y a una resistencia ya integrado en EDUBASICA (podemos usar también D4 y D5) :

Esquema de LEDs en EDUBASICA

No tienes que montar nada, la Edubasica lo tiene ya montado.

Un vídeo de cómo queda.

Vamos a ver una pequeña demostración:

https://www.youtube.com/embed/KHeNjn_4Pcg

Cargamos el programa ejemplo, en la placa Arduino y teniendo acoplada la placa EduBásica o montados los circuitos en una placa protoboard, podemos ver su funcionamiento.

Para ver su funcionamiento activamos el monitor serie del IDE de programación y variamos el potenciómetro. El resultado es una variación de luminosidad y variación de valores en el monitor serie.

El siguiente gráfico es una pantalla del monitor serie con los valores leídos y el valor aplicado a la señal PWM para variar la luminosidad.

Montaje 5 Programa. Instrucciones analorRead analogWrite y map

El valor del potenciómetro lo leerá la instrucción **analogRead(pin)** que lee valores analógicos cada 100µs y los convierte por el conversor analógico digital en valores de 10 bits por lo tanto de 0 a 1024. Los pines para leer sólo pueden ser los A0 al A5. (ver+)

Utilizaremos la instrucción **analogWrite(pin,value)** para crear la onda PWM en el pin correspondiente, que sólo puede ser en el Arduino Uno el 3, 5, 6, 9, 10, 11. El valor value es un número entre 0 y 255 y es simulado con una onda cuadrada de 490Hz o 980Hz (depende del pin, ver+).

Como uno lee 0 a 1024 y el otro necesita valores de 0 a 255 necesitamos un traductor o mapeo con la instrucción **map(value, fromLow, fromHigh, toLow, toHigh)** esta instrucción la colocaremos, como suele estar los intérpretes en medio, o sea, entre la instrucción analogRead y la analogWrite

> Veremos más adelante esta instrucción **map**

https://create.arduino.cc/editor/javierquintana/8bee8469-df52-4b88-a406-9f54fe3e20f8/preview

https://create.arduino.cc/editor/javierquintana/8bee8469-df52-4b88-a406-9f54fe3e20f8/preview?embed

La regulación del potenciómetro provocará una variación de voltaje en el pin de entrada analógico 0 de Arduino. Se realizará una conversión analógica-digital en el que los valores de tensión analógicos entre 0 y 5 V se transforma a un rango discreto de valores de 0 a 1023. Para modificar

la intensidad del led D3 rojo le se enviará una señal pseudoanalógica PWM utilizando la salida 5 digital de Arduino. Para ello se enviará un valor de 0 a 255 que marcará el ciclo de trabajo de la onda cuadrada PWM. Previamente habrá que realizar un mapeo (instrucción map) para asignar valores desde el intervalo [0, 1023] al [0, 255].

En este vídeo se ha modificado el mensaje que sale por el puerto serie a

```
Serial.print("valor analógico leído=");
Serial.println(analogRead(A0));
```

https://www.youtube.com/embed/y4OAnbQkR-c

El potenciómetro

Montaje 6: Lectura de potenciómetro y regulación de la luz sin map

Vamos a realizar un programa para comprobar que al variar el valor de una resistencia mediante un potenciómetro, también variará la cantidad de luz que emite un led. Como se puede ver en el siguiente vídeo, a medida que giramos el potenciómetro el led varía su luminosidad.

https://www.youtube.com/embed/KHeNjn_4Pcg

> **Perooooo si esto ya lo hemos hecho !! aquí**
> Cierto, pero ahora SIN LA INSTRUCCIÓN MAP para comprender bien su funcionamiento.

Para ello el valor que lee 0-1024 lo convertimos a 0-255 que necesita la señal PWM que enviamos al LED simplemente dividiéndolo entre 4. (1024/4 = 256 aproximadamente 255)

Montaje 6 CON EDUBASICA

Vamos a aprovechar el A0 que está conectado al potenciómetro y utilizaremos el D3 que está conectado al LED VERDE

Esquema del potenciómetro y de los diodos en EDUBÁSICA

Montaje 6: SIN EDUBASICA

Pues se necesita hacer el cableado correspondiente A0 con el potenciómetro y D3 a un led:

Montaje 6: Programa. Regulación LED con potenciómetro sin la instrucción map

El programa sería el siguiente

https://create.arduino.cc/editor/javierquintana/c0562793-5b64-490a-8dc4-72e4428de59a/preview

https://create.arduino.cc/editor/javierquintana/c0562793-5b64-490a-8dc4-72e4428de59a/preview?embed

Montaje 7: Leer valores potenciómetro por el puerto serie sin la instrucción map

Vamos ahora leer los valores en voltios que salen del potenciómetro. Pero igual que antes sin la instrucción map.

Para ello tenemos que convertir los valores leídos del potenciómetro que tienen un rango de valores de 0-1024 a valores de voltios, que como está alimentado a 5V, serán de 0-5 por lo tanto vamos a dividirlo **entre 204.6** pues 1024/204.6 = 5.

https://www.youtube.com/embed/mL-8-sFbuR4?rel=0

Montaje 7 SIN EDUBÁSICA

Conectamos la salida de un potenciómetro a A0

Montaje 7 CON EDUBASICA

No hay que hacer nada, ya está !

Montaje 7 Programa

https://create.arduino.cc/editor/javierquintana/e95259e6-9c6f-4079-8c25-fbbbc212ca55/preview

https://create.arduino.cc/editor/javierquintana/e95259e6-9c6f-4079-8c25-fbbbc212ca55/preview?embed

El sensor de luz LDR

Hasta ahora hemos trabajado con resistencias de valor fijo, pero existen una serie de resistencias que varían según distintos parámetros físicos a las que se les somete como presión, luz y temperatura entre otros. Existe una gran variedad que se utilizan para construir lo que llamamos **sensores**.

Montaje 8 Regular el led con la luz

En esta práctica vamos a diseñar un circuito que sea sensible a la luz. El objetivo **será regular la intensidad luminosa de un led con una LDR**, una resistencia sensible a la luz.

https://www.youtube.com/embed/APpJGAgetVo?rel=0

Montaje 8 CON EDUBASICA

En este montaje usaremos la resistencia LDR de la placa Edubásica. Como ya hemos comentado, la LDR modifica su resistencia en dependiendo de la cantidad de luz que incida sobre ella. El siguiente programa mostrará por consola (“Monitor Serial”) las variaciones de luminosidad que detecte la LDR simplemente pasando la mano por encima de ella.

Esquema del LDR en EDUBASICA

> **Valores entre 917 y 1024**
> La configuración es PULL-UP es decir, cuando el valor LDR aumenta (se acerca al OFF) la resistencia superior "tira = pull" hacia arriba = up. Los voltios suben. Si no hay luz, los voltios se acercan al máximo 5V que equivale a la lectura que el Arduino convierte en digital en 1024.
> Cuando hay luz, el LDR baja su resistencia, por lo que los voltios bajan. Pero no llega a 0 Ohm, se queda pues en un valor de voltios bastante alto, 4.5V que corresponde más o menos al valor que lee Arduino en digital 917

Montaje 8 SIN EDUBASICA

Hay que conectar en formato pull-up el LDR a A1 y un led verde a D3, el valor de la resistencia si quieres utilizar los mismos valores que mostramos 917-1024 tendrá que ser 470 Ohm. Cuánto más alto sea, más bajo serán los valores.

Montaje 8 Programa

Como vemos en el esquema, el LDR está conectado a la entrada A1 del Arduino, por lo tanto la instrucción de lectura será ***analogRead(1)*** lo mapearemos correctamente a una variable llamada **luz** y utilizaremos el LED Verde conectado a D3 por lo que la instrucción de salida será **analogWrite(3,luz)**

> Mapearemos con la instrucción **map** para convertir 917-1024 a valores de PWM 0-255

https://create.arduino.cc/editor/javierquintana/326c66b3-c8b8-4879-980c-f81af7ab65d8/preview

https://create.arduino.cc/editor/javierquintana/326c66b3-c8b8-4879-980c-f81af7ab65d8/preview?embed

Internamente la función map
¿Cómo funciona? ¿Cómo se podría quitar en estos anteriores programas?

Vimos en el ejemplo del potenciómetro que los valores a leer eran de 0-1024 y si lo queríamos convertir 0-255 era simplemente dividir por 4 y si es convertirlo de 0-5V se dividía por 204.6

Pero en este caso **EL LDR EMPIEZA DESDE 917 NO DESDE 0** como en el potenciómetro.

En una lectura previa de lo que nos devuelve **el LDR** y nos devuelve unos valores mínimo y máximo por ejemplo: ***917, 1024*** lo llamaremos **A1min y A1max** y queremos traducir estos valores al rango **0-255** pues es lo que podemos darle a un LED. Que lo llamaremos **luxmin y lumax**

Podemos hallar la pendiente m y el corte con el eje y n de la recta y = mx + n que convierte los valores analógicos 917-1024 a los valores 0-255. En el programa en Arduino si no queremos usar la función **map**, tenemos que escribir esa ecuación.

Aqui tienes una hoja de cálculo para calcular la pendiente y la intersección con el eje

Bobinas-altavoz

El altavoz es una simple bobina o electroimán que mueve una membrana, si la membrana se mueve repetidamente puede producir un sonido.

Este sonido es audible si está dentro de nuestro rango auditivo, suele ser entre 20Hz y 20kHz

> ¿Sabías que a medida que creces el margen de agudos (20kHz) baja?

https://giphy.com/embed/4njYZD26XnY7C

via GIPHY

Montaje 9 Pitido

Vamos a incorporar un altavoz y realizar una sirena. Al ser una bobina, es conveniente utilizar una amplificación por medio de un transistor, por D6 enviaremos la señal cuadrada a la base del transistor.

Montaje 9 SIN EDUBASICA

Montaje 9 Con EDUBÁSICA

Conectaremos el altavoz en el terminal X2 y el interruptor V1 en ON para que esté alimentado

Montaje 9: Programa

El código es muy sencillo, simplemente es una intermitencia por **D6** que en este caso se ha elegido **1mseg** ¿Qué pasaría si aumentamos este valor?

https://create.arduino.cc/editor/javierquintana/59845f8c-fb1f-4d8e-b04b-845687aad4a3/preview

https://create.arduino.cc/editor/javierquintana/59845f8c-fb1f-4d8e-b04b-845687aad4a3/preview?embed

El resultado es :

https://www.youtube.com/embed/7SsKMj2WMSw?rel=0

Montaje 9: SIN EDUBÁSICA Y SIN TRANSISTOR A LO BRUTO !

Bueno, vamos a conectarlo DIRECTAMENTE a D6 (el otro extremo a GND) no es muy conveniente pero a ver el resultado (con el mismo código):

https://www.youtube.com/embed/wcJBEfr8hNo?rel=0

¿Cuál suena más? Premio entrada a dinópolis Teruel quien acierte..

https://giphy.com/embed/3o6gb18J2gERJiLmmc

via GIPHY

Montaje 10 Alarma

Teniendo en EDUBASICA los LEDs, el LDR que nos puede servir como sensor y el altavóz amplificado con un transistor, y nosotros que somos expertos programadores, NOS ESTÁ PIDIENDO A GRITOS hacer una alarma:

Enunciado:
Cuando el LDR esté tapado, tiene que sonar un pitido intermitente de un segundo, con visualización también en los LEDs

Montaje 10: SIN EDUBASICA

Pues hay que poner el LDR en A1, las luces (por simplicidad uno), el altavoz y el transistor con la conexión en la base por D5:

Montaje 10: CON EDUBASICA

Se simplifica mucho la conexión sólo el altavoz tal y como está conectado en el montaje 14

Montaje 10: Programa:

https://create.arduino.cc/editor/javierquintana/e5766acc-9256-4277-a7a5-464ae1ba2976/preview

https://create.arduino.cc/editor/javierquintana/e5766acc-9256-4277-a7a5-464ae1ba2976/preview?embed

Montaje 10: Resultado

https://www.youtube.com/embed/Eit6hQzr57U?rel=0

LCD

La pantalla LCD

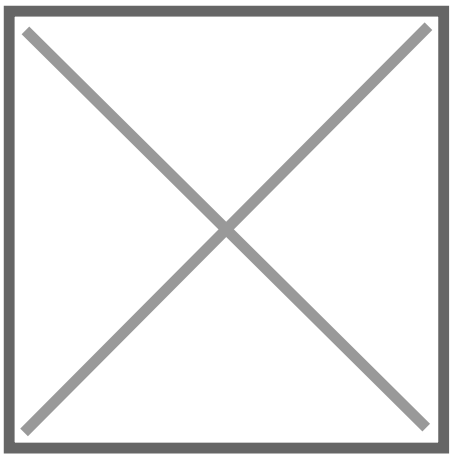
Las pantallas Liquid Cristal Display es la forma más sencilla de poner una interfaz de texto a nuestro Arduino.

https://giphy.com/embed/7RrSGWsK3BKRG

via GIPHY

La pantalla LCD tiene un precio muy bajo, el más común es el Hitachi HD44780 monocromo con configuración 16 caracteres y 2 líneas (16x2) pero también se venden 20×02, 20×04 y 40×02.

Su conexión **DIRECTA** con el Arduino no es recomendable por la cantidad de cables que se necesitan y el código elaborado, pero tiene la ventaja de tener total libertad en creación de caracteres y control. Si quieres ver esta opción puedes ver la página de Luis LLamas



Conexión I2C

Como hemos dicho en La pantalla LCD no es recomendable su conexión directa con Arduino, para ello está este controlador que permite su conexión **utilizando sólo dos cables**. Los dos componentes pueden salir por menos de 5€.

Su conexión con el LCD tiene que ser tal y como indica la imagen, y soldando los terminales con cuidado: 

CC-BY-SA Luis Llamas

> *Mejor que el Display,* ***es el OLED,*** *tiene un consumo más bajo, mejor visibilidad en ambientes con luz, y ya estan listas para conectar con Arduino con el protocolo I2C*
>
> *Recomendamos esta página https://www.luisllamas.es/conectar-arduino-a-una-pantalla-oled-de-0-96/*

Conexión con el Arduino

La conexión tiene serie tiene que ser en los pines A4 y A5 exclusivamente en el Arduino Uno pues son los dedicados para el protocolo serie I2C que veremos más adelante. Para otras placas ver la página de Luis Llamas.

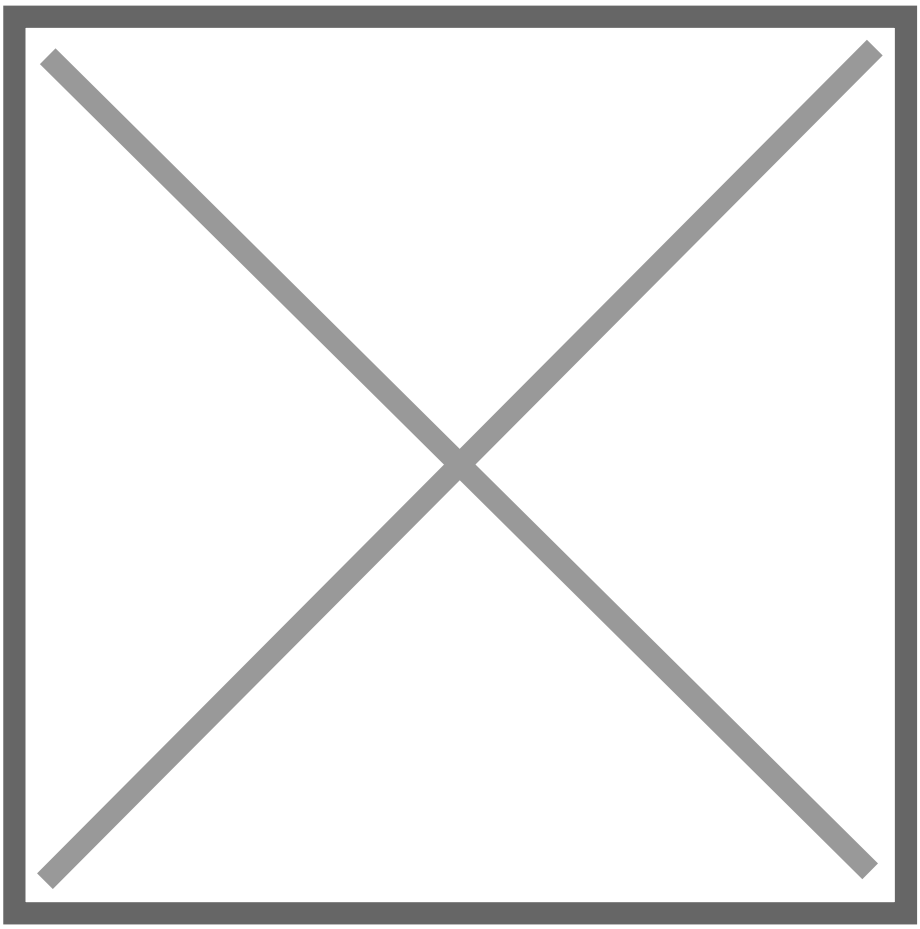
El esquema es muy sencillo:

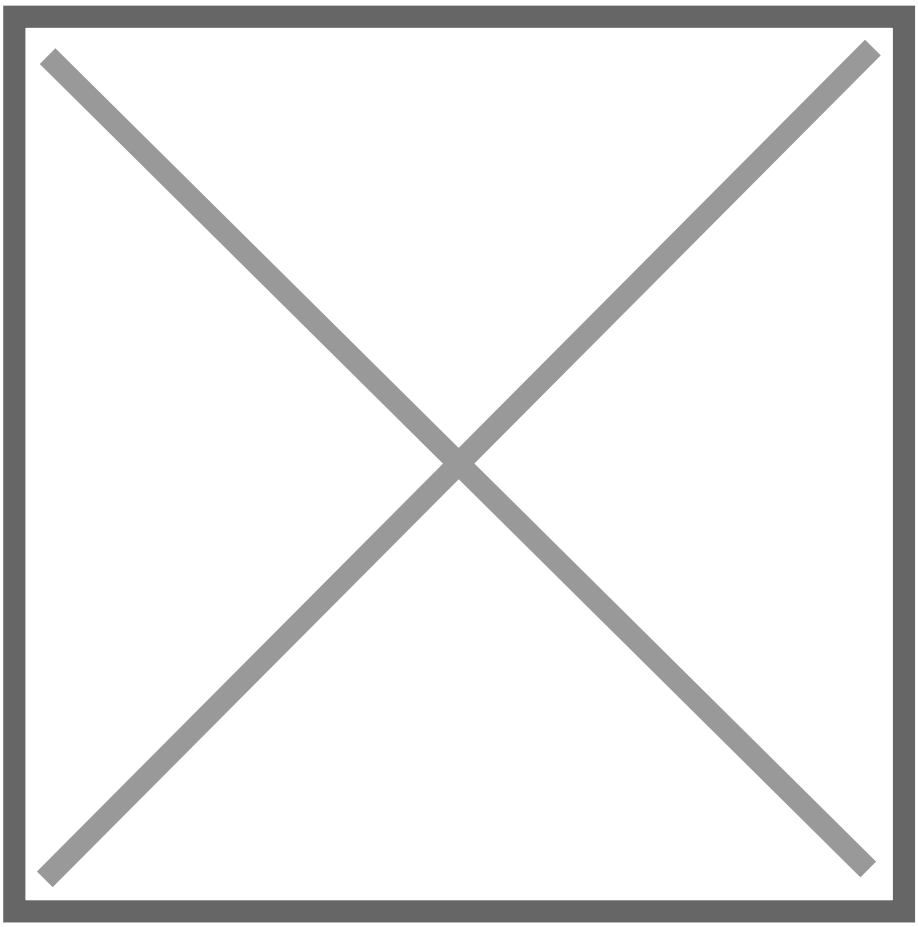


CC-BY-SA Luis Llamas

Contraste

Tiene un potenciómetro azul para regular el contraste, bastante sensible por cierto (una poca variación hace que nuestro texto no se vea correctamente):





Montaje 11 Escaneo

El bus I2C está vinculado en el Arduino Uno a los Pines A4 y A5 como hemos visto en la anterior página, el protocolo serie I2C necesita **la librería Wire.h (no ejecutar el editor online, no esta wire.h)** pero la **dirección de dispositivo no lo sabemos** PARA ELLO HAY QUE EJECUTAR ESTE CÓDIGO:

https://create.arduino.cc/editor/javierquintana/0f2a1666-cec2-43ad-a978-e83c421316a5/preview

https://create.arduino.cc/editor/javierquintana/0f2a1666-cec2-43ad-a978-e83c421316a5/preview?embed

Extraído de Arduino.cc o también de Luis Llamas

Nos tiene que salir lo siguiente:

En mi caso como ves la dirección me ha salido **0x3F** pero puede ser cualquier otro, por ejemplo otro valor muy típico es **0x27**

Librería LiquidCrystal

Sólo nos falta incorporar la librería LiquidCrystal_I2C que te lo puedes descargar aquí

Una vez descargado, es un fichero comprimido .zip o .rar **no lo descomprimas** directamente desde el menú del entorno de programación lo incorporas de esta manera :

Principales funciones

LiquidCrystal_I2C(lcd_Addr, lcd_cols, lcd_rows) Crea una variable (informáticamente un objeto de la clase LiquidCrystal_I2C) para poder utilizar sus funciones, hay que indicar entre paréntesis la dirección, columnas y filas indicadas. Por ejemplo LiquidCrystal_I2C lcd(0x3F,16,2);

> *¿No sabes la dirección?. Eso es que te has saltado lo que hemos explicado anteriormente. E*n mi caso es 0x3F.

Después de crear esa variable hay que inicializarlo con lcd.**init()**

> **lcd** es el nombre de la variable, puedes poner el nombre que quieras

- lcd.**clear()** Borra la pantalla y posiciona el cursor en la esquina superior izquierda (0,0).
- lcd.**setCursor(columna, fila)** Posiciona el cursor del LCD en la posición indicada por columna y fila.
- lcd.**print(**"texto"**)** Escribe el texto
- lcd.**scrollDisplayLeft()** Se desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.

- lcd.**scrollDisplayRight()** Se desplaza el contenido de la pantalla (texto y el cursor) un espacio a la derecha.
- lcd.**backlight()** Enciende la Luz del Fondo del LCD
- lcd.**noBacklight();** Apaga la Luz del Fondo del LCD
- lcd.**createChar (num, datos)** Crea un carácter personalizado permite crear hasta 8.

  Para usar esta función ver esta página.

Montaje 12 Texto en LCD

Vamos a realizar un ejemplo para practicar:

- El encendido y apagado de la pantalla
- Visualización de texto
- Visualización de texto con desplazamiento

Practicaremos qué código tiene que estar en el código dentro de *setup* y qué código dentro de *bool* :

https://www.youtube.com/embed/CBcGb9WWihY

El código es sencillo, la primera parte que sólo lo hace una vez, tiene que estar en setup y en loop sólo la parte de que se desplaza contínuamente.

OJO CAMBIA TU DIRECCIÓN 0x3F si no es esa

https://create.arduino.cc/editor/javierquintana/8c633ec1-0e2a-4c5f-8fa0-a0a1bb31645c/preview

https://create.arduino.cc/editor/javierquintana/8c633ec1-0e2a-4c5f-8fa0-a0a1bb31645c/preview?embed

Sensor de ultrasonidos

Manejar este tipo de sensores que son muy comunes en las aplicaciones de robótica para medir distancias. Para ello aprenderás a:

- Realizar las conexiones necesarias sobre el sensor ultrasonidos HC-SR04.
- Conocer el funcionamiento de un radar.
- Cómo convertir el tiempo de rebote de un sonido en distancia.

Montaje 13: Medición de la distancia

Este tipo de sensores también nos permite conocer la distancia a un objeto. Es más preciso que el de infrarrojos visto en la sección anterior y su rango de funcionamiento también es mayor. Funciona desde los 2cm hasta los 3 metros.

Podemos usar un sensor de ultrasonidos para obtener la distancia a un objeto. Este sensor se basa en el envío de una señal acústica y la recepción del eco de dicha señal. Lo que haremos después, al igual que hace un radar , un aparato de ecografías o un murciélago es calcular la distancia en función del tiempo que ha tardado el rebotar el sonido y la velocidad del sonido. Podemos encontrar las especificaciones en la página del fabricante. Uno de los modelos más comunes es el HC-SR04:

> No es un sensor muy preciso. Si el obstáculo presenta caras oblicuas ya falla en la lectura del eco. Pero por el precio que tiene y la sensibilidad, no esta mal para utilizarla en la robótica educativa.

El sensor tiene 2 partes como puedes ver en la figura. Una se encarga de enviar un sonido (a una frecuencia alta que no podemos escuchar), y la otra parte detecta cuando ese sonido vuelve.

Este sensor es muy útil en robots móviles para diversas acciones como no chocar o mantenerse a cierta distancia de una pared.

El sensor HC-SR04 que usamos en estos ejemplos tiene 4 pines que corresponden (de izquierda a derecha):

- GND , Vcc (a +5V)
- Trig: es el que emite el ultrasonido
- Echo: Es el que recibe el rebote

(Algunos modelos solo tienen 3 pines -HCSR05- indicándonos por el tercer pin ya directamente un valor proporcional con la distancia.)

> **No aconsejamos usar la Shield de Edubasica, sino conectar diréctamente**, en este caso no nos supone un ahorro de cableado, no como en los motores, leds, ldr, etc...:

El programa es:

https://create.arduino.cc/editor/javierquintana/e3bac5c7-0bf1-49d9-b267-94b628e04f2e/preview

https://create.arduino.cc/editor/javierquintana/e3bac5c7-0bf1-49d9-b267-94b628e04f2e/preview?embed

El resultado :

https://www.youtube.com/embed/cYiOaTwq2E8?rel=0

Montaje 14 Visualización distancia en el LCD

Vamos a repetir el anterior programa pero que lo visualice el LCD

https://www.youtube.com/embed/eZvLMnHBypQ

Montaje 14 Conexión:

Conectar el sensor de Ultrasonidos

Y el LCD

Montaje 14 programa

https://create.arduino.cc/editor/javierquintana/60770327-de2a-41d0-ac11-ddcddb1da0c9/preview

https://create.arduino.cc/editor/javierquintana/60770327-de2a-41d0-ac11-ddcddb1da0c9/preview?embed

Temperatura y humedad

Vamos a utilizar dos sensores para medir estas variables: **DHT12** y su hermano pequeño **DHT11**.

Ya explicamos estos sensores en el capítulo de sensores https://libros.catedu.es/books/programa-arduino-mediante-codigo/page/sensores

image-1644407103198.png

¿Cómo se conecta?

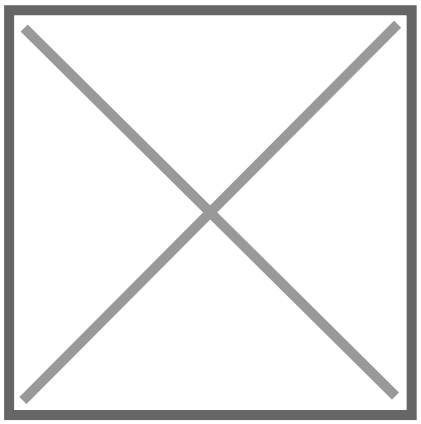
Para ello tendrás que realizar las conexiones necesarias para poder obtener los datos desde Arduino. Se trata de un sensor digital que utiliza 3 pines: Alimentación +5V, tierra/GND (-) y pin de datos (out) por donde se envían los datos de humedad y temperatura.

> A veces el sensor viene sobre 4 pines, en este caso uno de ellos no se conecta.

Como las medidas de humedad y temperatura van por un solo pin, la información se transmite como un tren de pulsos en serie, por lo tanto, necesitamos un programa que "extraiga" eso dos datos de forma diferenciada. Para ello vamos a usar una librería referenciada por **DHT11.h**. A través del monitor serie del IDE de Arduino podremos ver las medidas obtenidas.

Esquema DHT11 :

Es un modelo "conectar y listo" que ya vienen con los cables preparados, pero si te fijas son en este orden : GND - 5V -NC - D2 donde NC significa NO CONECTADO y D2 son los datos



Lo conectamos en el Arduino sin necesidad de placa Protoboard:

> Para este montaje, la colocación de la placa Edubásica no implica ninguna ventaja.

utilizando cables macho-macho

Esquema de conexión:

Esquema DHT12

Es un sensor que necesita 3.3V pero si trabajas con 5V necesitas hacer este puente con una resistencia de más o menos 10K

O con la placa Protoboard

Fuente Cactus.io CC BY-NC-SA 3.0

Librería DHT11.h

Para poder utilizar este dispositivo, necesitas esta librería, lo primero que tienes que hacerlo es descargarlo, puedes hacerlo desde https://github.com/adafruit/DHT-sensor-library o este enlace de Drive (zip - 4,29 KB), o simplemente buscando DHT11 library en Internet.

Una vez descargado, tienes que incorporarlo en tu librería, aquí tienes cómo hacerlo

https://docs.google.com/presentation/d/e/2PACX-1vSTHyelOzP8IRe6NORUJzw0IgxPzlyGoeiDS4eRxOeprKd1BClBy7LzF36hPPc-MESzZdEGj37_DBUx/embed?start=falseloop=false&delayms=3000

Librería DHT12.h

Para poder utilizar este dispositivo, necesitas esta librería, lo primero que tienes que hacerlo es descargarlo, puedes hacerlo desde https://github.com/adafruit/DHT-sensor-library o simplemente buscando DHT12 library en Internet.

Una vez descargado, tienes que incorporarlo en tu librería, aquí tienes cómo hacerlo

https://docs.google.com/presentation/d/e/2PACX-1vSTHyelOzP8IRe6NORUJzw0IgxPzlyGoeiDS4eRxOeprKd1BClBy7LzF36hPPc-MESzZdEGj37_DBUx/embed?start=falseloop=false&delayms=3000

Montaje 15 Medición T y H por puerto serie con DHT11:

https://create.arduino.cc/editor/javierquintana/57997bda-7323-4f8a-9157-92f9611dbdba/preview

https://create.arduino.cc/editor/javierquintana/57997bda-7323-4f8a-9157-92f9611dbdba/preview?embed

El resultado se puede ver en este vídeo, simplemente soplando nuestro vaho pasamos de 20% de humedad y 22ºC a 93% y 24ºC.

> Si lo hacéis con niños, enseguida se les ocurre ponerlo en el sobaco, menos mal que solo son 5V ;) _

https://www.youtube.com/embed/gLxdSaxOJbY?rel=0

Montaje 16 Medición T y H por puerto serie con DHT12

La librería de este sensor es más potente y nos puede decir la sensación térmica:

https://create.arduino.cc/editor/javierquintana/66a222b4-8e95-4693-b8db-df5a1fdb68a3/preview

https://create.arduino.cc/editor/javierquintana/66a222b4-8e95-4693-b8db-df5a1fdb68a3/preview?embed

El resultado es:

Montaje 17 H y T por LCD

Ahora vamos a conectarlo por LCD :

https://www.youtube.com/embed/6J8PKruEcD4

Conexiones

Si tienes el DHT11 o si tienes el DH12 lo has visto ya Ahora añade el LCD con el I2C

Programa

En este caso lo hacemos con el DHT12 ya sabes que si utilizas DHT11 no mide la humedad y la sensación térmica

https://create.arduino.cc/editor/javierquintana/10dcd8cb-1128-4959-8d78-70464cea939f/preview

https://create.arduino.cc/editor/javierquintana/10dcd8cb-1128-4959-8d78-70464cea939f/preview?embed

Processing

Es un programa similar al IDE de Arduino que has estado manejando hasta ahora, sólo cambia en un botón de PLAY y en otro de STOP. Es software abierto desarrollado en Java por *Ben Fry y Casey Reas* a raíz de una reflexiones en un congreso donde se detecto esta necesidad. Te lo puedes descargar de http://processing.org

¿Para qué sirve?

Es muy común tener la necesidad te representar los datos que nos da Arduino en un entorno visual mucho más atractivo que el monitor serie que nos ofrece IDE Arduino. Si quieres saber las instrucciones que tiene y más información consulta esta página.

Saber el puerto de conexión

Para empezar a utilizar Processing con nuestro Arduino necesitamos saber en qué puerto se conecta, una forma fácil es cargar y ejecutar este código con el Arduino conectado y que liste los puerto, esta instrucción **printArray(Serial.list());** nos lo puede decir

Y el resultado puedes ver que sale abajo en la consola [0] "COM4" luego es el 0 en mi caso

Montaje 18 Representación gráfica de medidas con Processing.

Una vez obtenidos los datos de temperatura y humedad a través del sensor DHT11 desde Arduino, enviamos, a través del puerto serie, estos datos al PC, donde tenemos ejecutando un programa en Processing que está "escuchando" el puerto serie, obteniendo los datos y representándolos en pantalla. Simultáneamente guardamos los datos en un archivo de texto que posteriormente podremos analizar en una hoja de cálculo.

Programa a cargar en el ARDUINO

Cargamos este programa, fíjate que sólo ponemos un valor de la temperatura, si queremos representar la humedad, quitamos el comentario de la temperatura y ponemos el de la humedad

> **IMPORTANTE**: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing.

Si fuera un DHT12 en vez de un DHT11 poner comentarios a las 4 primeras líneas delante // y quitárselas a las 3 siguientes

https://create.arduino.cc/editor/javierquintana/09ac70ea-0d4b-4d26-bce4-4b45bb45d3cb/preview

https://create.arduino.cc/editor/javierquintana/09ac70ea-0d4b-4d26-bce4-4b45bb45d3cb/preview?embed

Programa a Cargar en PROCESING

Aquí lo tienes (rar - 1,96 KB), sólo representa un valor, está puesto en el puerto 0 puertoArduino = new Serial(this, Serial.list()[0], 9600);

El resultado puedes verlo aquí abajo para la temperatura, el aumento se debe a aplicar vaho al sensor:

Otro programa de visualización de datos

En este caso no vamos a representar los datos en forma de gráfica, sino por colores, y además vamos a añadir un botón que encienda un LED conectado por simplicidad en el pin 13

Programa a cargar en el ARDUINO

El programa lee la temperatura y lo escribe en el puerto serie en forma de byte. También lee el puerto serie para cambiar el estado del led.

```
#include "DHT.h"
#define DHTPIN 2
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
```

```
boolean status=LOW; //Estado del led
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  dht.begin();
}

void loop() {
  delay(100);

  //float h = dht.readHumidity();
  int temp = dht.readTemperature();
  Serial.write(temp); //Enviamos los datos en forma de byte
  if(Serial.available()>0)//Si el Arduino recibe datos a través del puerto serie
  {
    byte dato = Serial.read(); //Los almacena en la variable "dato"
    if(dato==65)  //Si recibe una "A" (en ASCII "65")
    {
      status=!status; //Cambia el estatus del led
    }
    digitalWrite(13,status);
  }

}
```

En el Arduino tenemos que poner el sensor de temperatura y humedad tal y como se ha explicado en el Montaje 8 y además un led en el 13

Programa en Processing

te lo puedes descargar aquí (rar - 31,02 KB)(recuerda cambiar port = new Serial(this, Serial.list()[0], 9600); por tu puerto)

dweet.io

Disponemos del portal web **dweet.io** que nos ofrece un servicio para enviar y representar datos en la nube sin necesidad, ni si quiera, de registrarnos en la plataforma.

Vamos a ver los pasos a seguir:

1. Probamos la plataforma introduciendo un dato, para ello en el navegador tecleamos por ejemplo (cambia **catedu** por tu nombre):

   https://dweet.io/dweet/for/catedu?temperatura=20
2. Abre otra pestaña del navegador o utiliza un móvil para seguir el dato:

   https://dweet.io/follow/catedu
3. Prueba añadiendo otra variable, en este caso la humedad:

   https://dweet.io/dweet/for/catedu?temperatura=20&humedad=8

Automatizamos el proceso de recogida de datos desde Arduino con un programa en Processing, que enviará datos a través del navegador a dweet.io.

IMPORTANTE: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing.

REPRESENTACIÓN DE DATOS EN EL NAVEGADOR:

Dweet.io nos ofrecerá los datos de la siguiente manera:

Si queremos algo más vistoso podemos utilizar el servicio freeboard.io aunque en este caso nos tendremos que registrar en la web.

Una vez registrados podemos crear paneles indicadores configurados a nuestro gusto para visualizar la información. Primero habrá que añadir como fuente de datos Dweet.io y nuestro nombre utilizado allí (jorgeroden en el ejemplo).

Después creamos un panel indicando que la fuente de datos que queremos utilizar y la variable en cuestión a visualizar.

¡Y resultado puede ser de este tipo!

Montaje 19 Visualización de datos en la nube. Internet de las cosas.

> Consejo: Si quieres una visualización más sencilla de los datos que recoge un Arduino y llevarlas a la nube, te recomendamos el curso ESP32 EN EL AULA
>
> https://libros.catedu.es/books/esp32-en-el-aula el apartado IoT por ejemplo con Telegram

Monitorizar los datos de temperatura y humedad obtenidos del sensor DHT11 en la nube.

Para ello disponemos del portal web **dweet.io** que nos ofrece un servicio para enviar y representar datos en la nube sin necesidad, ni si quiera, de registrarnos en la plataforma.

Vamos a ver los pasos a seguir:

1. Probamos la plataforma introduciendo un dato, para ello en el navegador tecleamos por ejemplo (cambia **CATEDU** por tu nombre):

   https://dweet.io/dweet/for/CATEDU?temperatura=20
2. Abre otra pestaña del navegador o utiliza un móvil para seguir el dato:

   https://dweet.io/follow/CATEDU

3. Prueba añadiendo otra variable, en este caso la humedad:
   https://dweet.io/dweet/for/CATEDU?temperatura=20&humedad=8

> ¿Quieres ver las cosas que ahora mismo se están difundiendo por dweet? Mira
> https://dweet.io/see

Automatizamos el proceso de recogida de datos desde Arduino con un programa en Processing, que enviará datos a través del navegador a dweet.io.

> **IMPORTANTE**: No hay que tener abierto el monitor serie del IDE de Arduino porque ocupa el puerto y, por lo tanto, no deja leer los datos a Processing.

PROGRAMA A CARGAR EN ARDUINO:

Si fuera un DHT12 en vez de un DHT11 poner comentarios a las 4 primeras líneas delante // y quitárselas a las 3 siguientes

https://create.arduino.cc/editor/javierquintana/8154af00-c514-4936-81fb-b03fad1963f8/preview

https://create.arduino.cc/editor/javierquintana/8154af00-c514-4936-81fb-b03fad1963f8/preview?embed

PROGRAMA A EJECUTAR EN PROCESSING :

```
// El puerto serie
Serial myPort;

void setup() {
 // Lista todos los puertos serie
 printArray(Serial.list());
 // OJO: Elige el puerto donde tengas conectado Arduino.
 // Cambia el "0" de Serial.list()[0] por el orden que
 // tu puerto ocupe en la lista (0, 1, 2,...).
 // Si no lo tienes claro qué puerto ocupa Arduino mira
 // en el IDE Arduino en "Herramientas" mira el puerto que esté seleccionado.
```

```
 //Fíjate que tenemos la velocidad del puerto a la misma que pusimos en Arduino
 myPort = new Serial(this, Serial.list()[0], 9600);
}
void draw() {
 while (myPort.available() > 0) {

    String lectura = myPort.readStringUntil(lf);
    if (lectura != null) {
        println(lectura);
       //IMPORTANTE! cambia CATEDU por tu nombre
       // visualiza los resultados en esta web https://dweet.io/follow/CATEDU

      loadStrings("https://dweet.io/dweet/for/CATEDU?"+lectura);

    }

  }
}
```

REPRESENTACIÓN DE DATOS EN EL NAVEGADOR:

https://dweet.io/follow/CATEDU nos **ofrecería** los datos de la siguiente manera:

> No lo hagas, pues NO LO TENGO CONECTADO !! no sale nada !! por eso pone "*nos ofrecería*"

El resultado es espectacular

https://giphy.com/embed/sjDV6YTbw8tig

via GIPHY

Sensor de infrarrojos CNY70

Uno de los sensores más utilizados en robótica, o infinidad de aplicaciones industriales es el CNY-70.

Su nombre técnico es optoacoplador. Se basa en la acción conjunta de un diodo que emite una luz infrarroja (que no vemos) y un fototransistor que detecta el rebote de esta cuando incide sobre algún objeto.

Inicialmente es un sensor analógico y nos da un valor de voltaje proporcional a la luz rebotada, pero podemos utilizarlo también de manera digital. El transistor y el diodo hay que alimentarlo a través de una resistencia, el diodo del orden de Ohmios para dar una señal razonable y el transistor del orden de k para que trabaja en la zona activa.

Su funcionamiento es sencillo, si el receptor recibe la señal del emisor, el transistor conduce, por lo que recibiremos un '1' lógico en el Arduino:

Montaje 20 detección línea blanca

El problema del CNY70 es que tiene los pines muy juntos que no se puede poner en medio de la placa protoboard, tenemos pues que utilizar dos opciones:

- Utilizar cables Dupond macho-hembra
- Ponerlo inclinado aprovechando que dos extremos de la diagonal tienen que estar conectados a 5V

Lo mejor es utilizar cables M-H pero si no se tienen, vamos a utilizar la segunda opción, este es el esquema:

Utilizaremos la Edubásica el led rojo, si no tienes, simplemente añade un led al pin 13

El programa a cargar en el Arduino es:

https://create.arduino.cc/editor/javierquintana/9b8fc1a0-03fc-42d9-a85c-ca10e9f26588/preview

https://create.arduino.cc/editor/javierquintana/9b8fc1a0-03fc-42d9-a85c-ca10e9f26588/preview?embed

El resultado es:

https://www.youtube.com/embed/d6QXe9t__wE?rel=0

Y en el monitor serie sale:

Linea negra
Linea negra
Linea negra
Linea blanca
Linea blanca
Linea blanca
Linea blanca
Linea blanca
Linea negra
Linea negra
Linea negra
Linea ...

Contador Geiger

> Este material no esta en el kit, pero se muestra por ser un buen ejemplo del uso obligatorio de las **interrupciones**

Un módulo contador Geiger puede salir por unos 36€.

Lo más importante es el tubo. Ver cuadro de tubos comerciales. Ojo que trabaja con aprox 400V.

Si tu tubo es más pequeño, aquí tienes como hacerte un adaptador casero.

2024-06-03 20_59_31-Venta de Módulo de contador Geiger Geekcreit ensamblado Tubo GM de tubo

La conexión es fácil simplemente realiza un impulso y lo puede recoger el pin 2

2024-06-03 21_01_30-Arduino DIY Geiger Counter _ 12 Steps (with Pictures) - Instructables.png
Imagen de Hisehf Murchinson en https://www.instructables.com/Arduino-DIY-Geiger-Counter/

> PERO **¿COMO HACEMOS QUE CUENTE DE FORMA ASINCRONA?** es decir, no podemos poner en un bucle tipo
> loop(){
> if (digitalRead(2) == HIIGH){ CNT++;}
> }
> Pues si en ese momento NO hay chasquido, no cuenta
> Esto es debido a que el chasquido puede producirse en cualquier momento, y muy breve
> Arduino no puede mandar sobre el pin2 sino el pin2 tiene que mandar sobre el Arduino.

> SOLUCION: LAS INTERRUPCIONES

Con la instrucción **attachInterrupt**, cuando haya un chasquido, atenderá a la función **GetEvent**

(las interrupciones en ArduinoUNO sólo son válidos en los pines 2 y 3, ver+)
(el tercer parámetro FALLING significa que haga caso en el flanco de bajada, cuando va de 1 a 0)

attachInterrupt(digitalPinToInterrupt(2), GetEvent, FALLING);

y en **GetEvent** simplemente ponemos un contador

```
void GetEvent() {  // Get Event from Device
  CNT++;
}
```

Añadimos al Arduino un Display conectado en serie I2C para evitar el cableado, tal y como lo hemos conectado aquí https://libros.catedu.es/books/programa-arduino-mediante-codigo/page/lcd por lo tanto, el código completo es el siguiente:

Código https://app.arduino.cc/sketches/b1b88357-7588-4b07-9f83-bf32093358e3?view-mode=preview

https://app.arduino.cc/sketches/b1b88357-7588-4b07-9f83-bf32093358e3?view-mode=embed

Aquí en este vídeo podemos ver que los conteos por minuto CPM son 10-20 que es la radiactividad natural 2.4mSv año (en teoría 15.6 CPM) pero si le acercamos un resto de camisa de camping gas (fabricadas antes del 2.000) que tienen Torio radiactivo, sube el CPM a 350, Otra muestra asequible son los detectores de humo que llevan Americio, unos 519CPM.

El detector no es lo suficiente sensible para medir la radiactividad de un plátano (que tiene potasio pero solo 0.036mSv año)

https://www.youtube.com/embed/iXtnPy4BoHk