ARDUINO

APROXIMACIÓN A LA UTILIZACIÓN DEL ENTORNO

ARDUINO Y SU IDE

¿QUÉ ES ARDUINO?

Es una plataforma de creación de electrónica de código abierto.

Está basada en hardware y software libre, flexible y fácil de usar para creadores y desarrolladores.

El hardware libre son los dispositivos, especificaciones y diagramas de acceso público, de manera que cualquier persona puede replicarlos.

Esto quiere decir que cualquier persona o empresa puede crear sus propias placas, que pueden ser diferentes entre ellas, pero totalmente funcionales porque parten de la misma base.

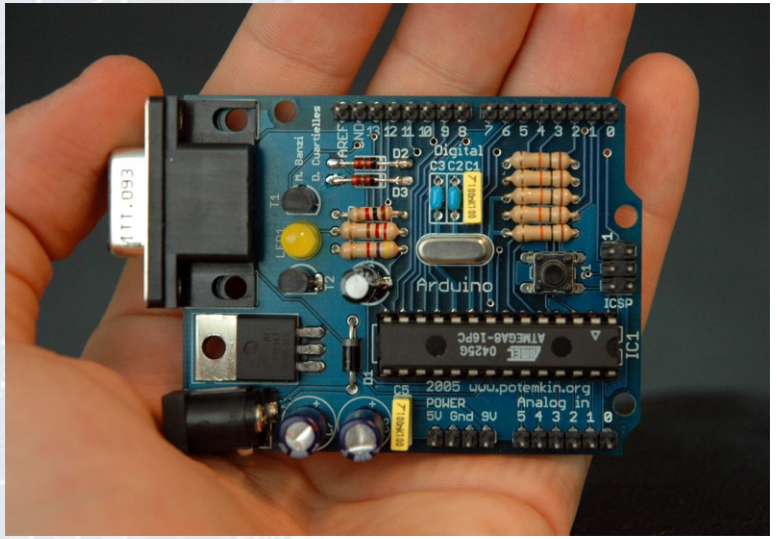
El software libre tiene un código accesible para cualquiera que quiera utilizarlo o modificarlo. Arduino ofrece la plataforma Arduino IDE (que es un entorno de desarrollo integrado) en que cualquier persona puede crear aplicaciones para loas placas Arduino para darle todo tipo de utilidades.

HISTORIA

El proyecto nació en el año 2003

Diversos estudiantes del Instituto de Diseño Interactivo de Ivrea (Itàlia) hicieron una placa electrónica que era una alternativa a las populares BASIC Stamp, que por entonces tenían un precio de unos 100 $.

El proyecto se denominó Wiring y contaba con la posibilidad de poder hacer servir un entorno de programación de código abierto multiplataforma, que se puede utilizar en Linux, Windows, Mac OSX, GetApp i ARM.



El nombre Arduino proviene del lugar dónde se reunían los estudiantes, un bar de Ivrea dedicado a Arduino de Ivrea, que fue rey de Italia entre Los años 1002 y 1014.

El primer diseño que vio la luz pública de forma comercial fue introducido el año 2005 en el mercado ofreciendo un bajo coste y facilidad de uso.

TIPOS DE PLACA

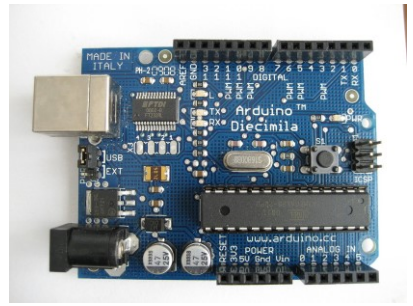
A lo largo de su vida útil ha tenido diferentes tipos de procesadores, Atmel AVR, ARM Cortex o Intel Quark. Aunque la filosofía es la misma, el código fuente para hacer funcionar estas placas difiere, por lo que hay diferentes versiones del software.

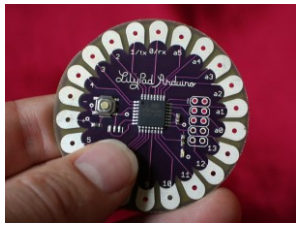
La placa principal es del tipo Single Board Computer, contiene memoria SRAM y una Flash EEPROM. Dependiendo de su forma de comunicaciones puede tener puerto serie, USB, Bluetooth, o de prototipo.

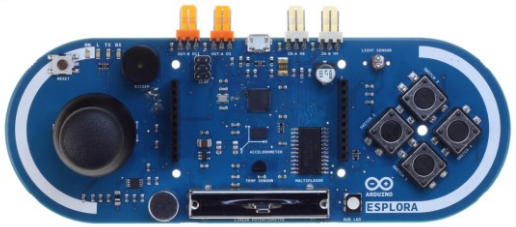
Dependiendo del fabricante hay infinidad de placas, pero la más utilizada es la Arduino Uno.

Una cosa importante es que su bus a cada lado posibilita que puede tener extensiones conectadas y hace que pueda controlar desde robots a impresoras 3D.

Lógicamente, también sirve para controlar todos los dispositivos que hay en una maqueta...







BUS SERIE

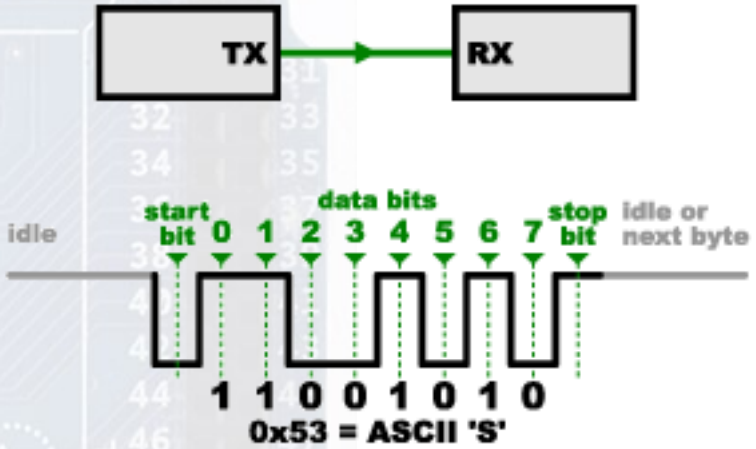
En les placas Uno, Nano, Mini i Mega los pines 0 y 1 se usen para comunicarse con el ordenador.

Las comunicaciones en los pines TX (transmisión)/RX (Recepción) utilizan niveles lógicos TTL de 5 voltios o 3,3 voltios dependiendo de la placa.

Estos pines no se pueden conectar directamente a un puerto RS232, ya que estos puertos operan a 12 voltios.

Al margen de los dos pines también tenemos un puerto (o más) que utiliza la comunicación serie (excepto en Arduino Pro mini), que es el USB que lo conecta con el ordenador, que sirve para introducir la programación al Arduino.

En este Puerto es muy importante que transmisor y receptor funcionen a la misma velocidad (de 9600 a 115.400 baudios).



BUS SPI

El bus *Serial Peripheral Interface* fue desarrollado por Motorola en 1980, y es un estándar de facto debido a las ventajas que tiene sobre otros sistemas.

Tiene una arquitectura maestro/esclavo en que el maestro puede iniciar la comunicación con uno o diferentes esclavos y enviar y recibir datos de ellos. Los dispositivos esclavos no pueden iniciar la comunicación ni comunicarse entre ellos.

La comunicación es del tipo síncrona, en la que hay una señal de reloj que tiene todos los dispositivos sincronizados, evitando lo problemas de velocidad que tiene el BUS serie.

Como mínimo tiene 3 líneas:

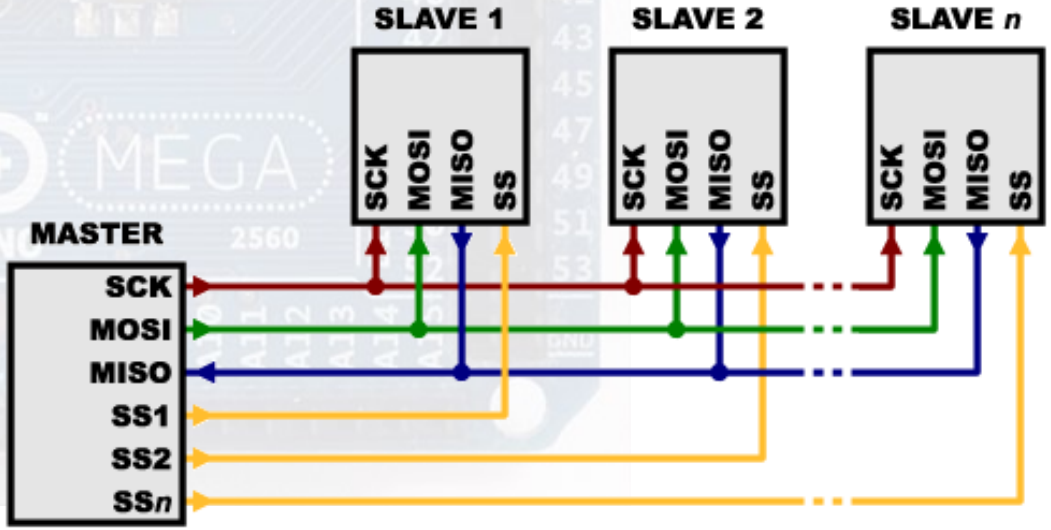
MOSI: comunicación maestro a esclavo

MISO: comunicación esclavo a maestro

SCK: Señal de reloj del maestro

SS: Slave-select (indica que el mensaje es para él)

Es posible hacer una conexión en cascada en que cada Esclavo transmite datos al siguiente.



BUS I2C

El Inter-Integrated Circuit (I2C) es una comunicación estándar. Fue diseñado para facilitar la comunicación entre diferentes tipos de dispositivos, entre los cuales se encuentran los microcontroladores.

La comunicación a través del bus no puede ser anárquica, sino que tiene que seguir las normas que dictaminan el protocolo de comunicaciones, que en este caso es el I2C.

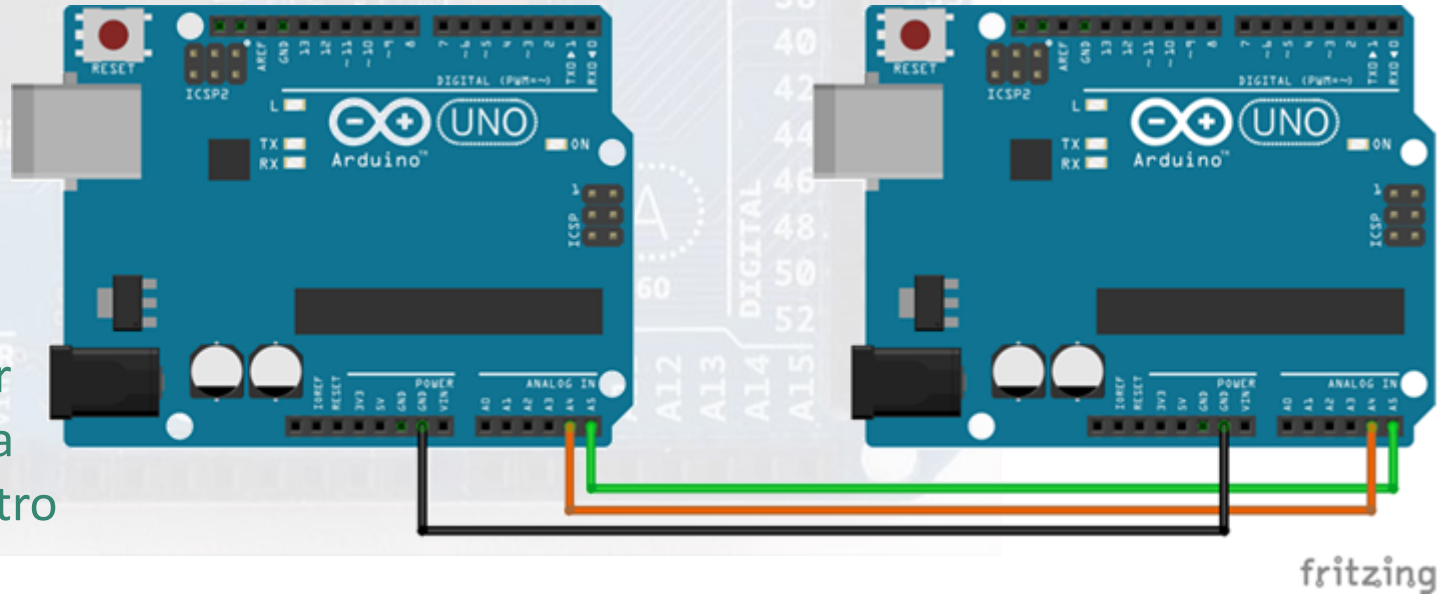
El tipo de comunicación es del tipo síncrona, en la que hay una señal de reloj que en el Arduino puede llegar a los 100 kbits/s y tiene tres señales:

SCL: reloj del sistema

SDA: señal de dades

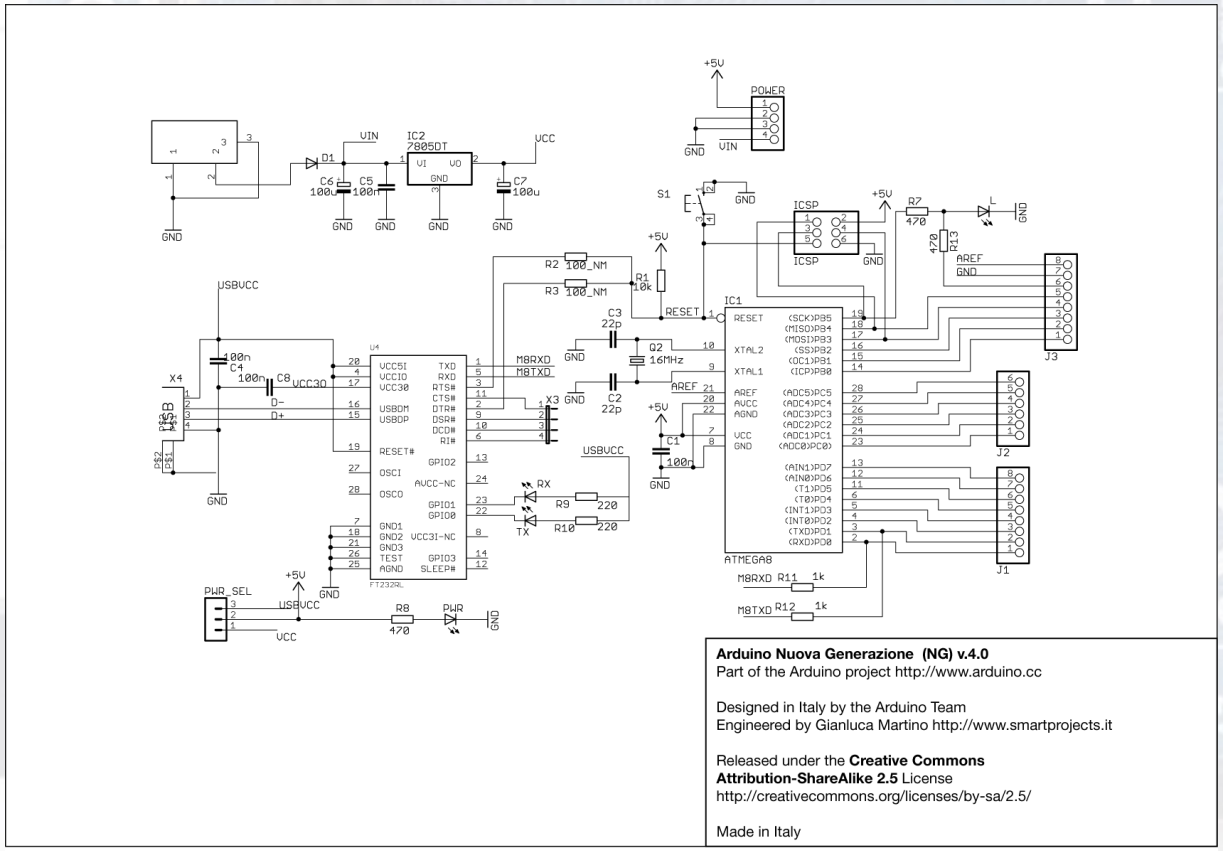
GND: masa de la señal

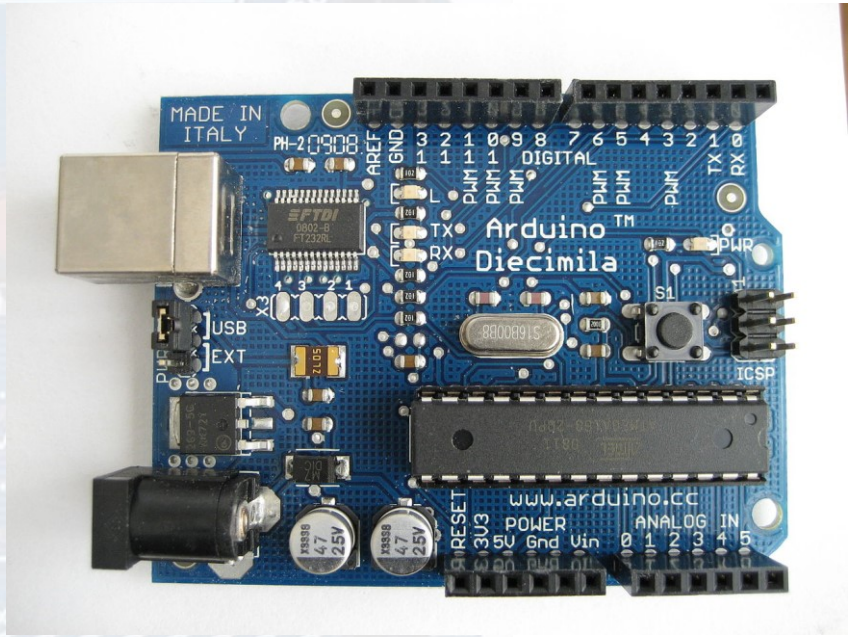
Estas señales se tienen que compartir entre todos los dispositivos que tenga el Bus. Un ejemplo, un Arduino maestro Y un esclavó, como en la foto.



ARDUINO UNO

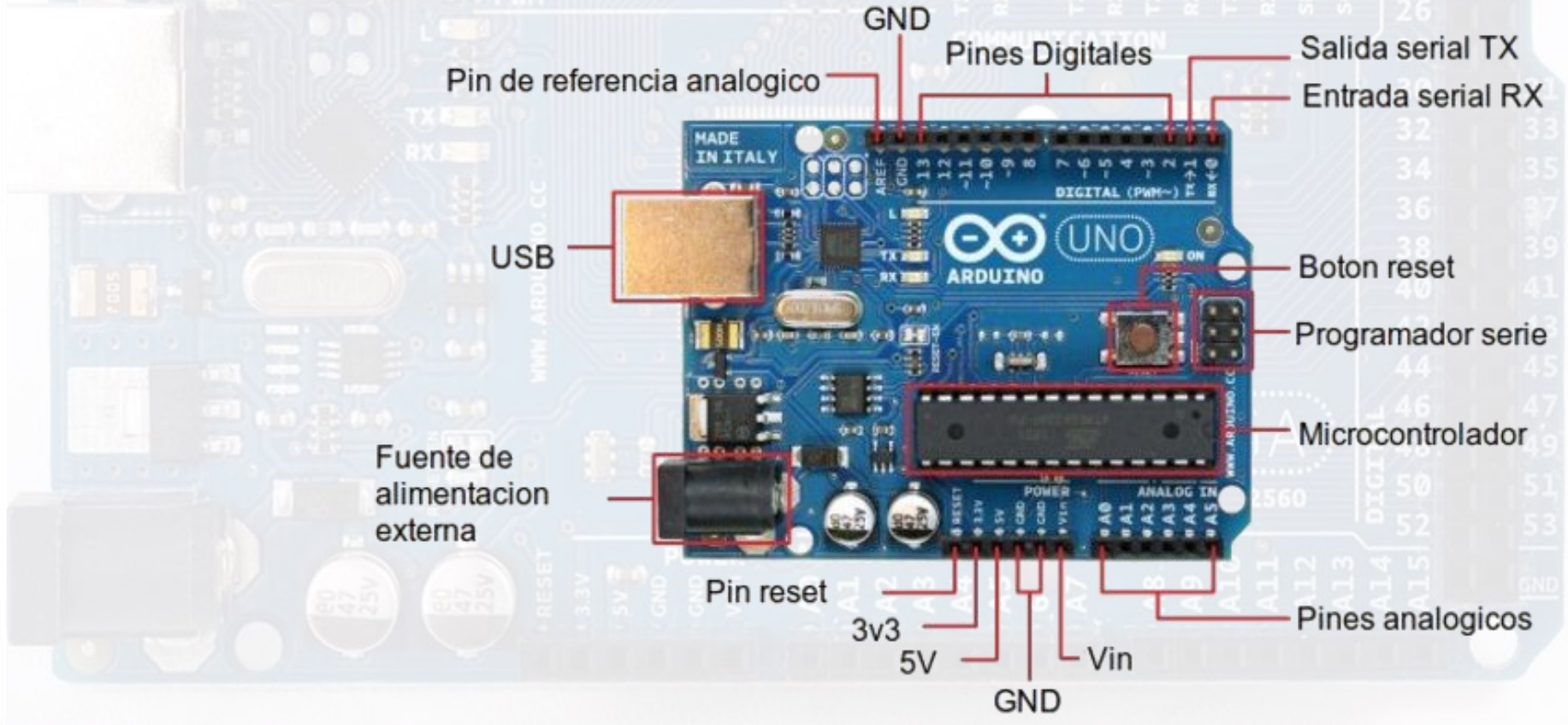
La más asequible de todas las placas y la que prácticamente encontraremos en todos los kits es la placa Arduino Uno. En la foto se ve el esquema Creative Commons, y a la derecha versión para USB.





ARDUINO UNO

La placa electrónica Arduino Uno R3 puede alimentarse de diferentes formas, con un cable USB del tipo B conectado al ordenador o con una fuente de alimentación externa (adaptador DC 7-12 voltios).



ARDUINO UNO

El microcontrolador es un ATmega328P.
Es un microcontrolador de arquitectura RISC avanzado de Atmel, de alto rendimiento, bajo consumo y optimizado para compiladores de lenguaje C.

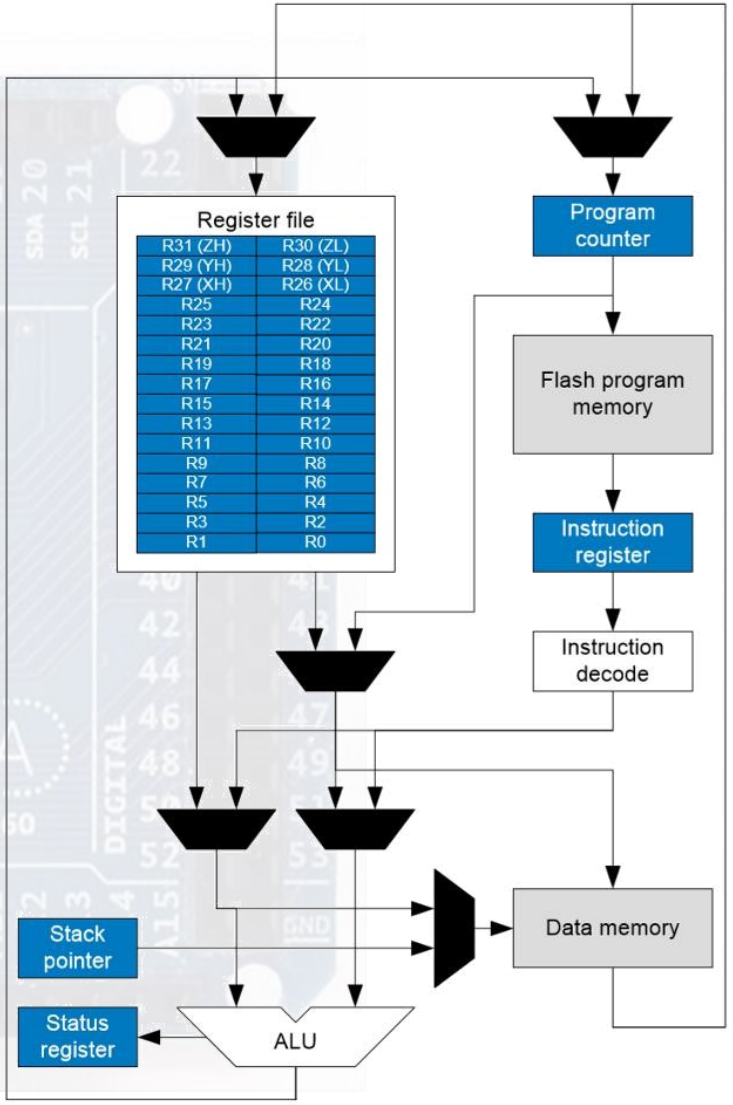
La arquitectura interna (figura de la derecha) es del tipo Hardward, con memorias y buses separados para programa y datos. La CPU utiliza una pipeline, en la que mientras se ejecuta una instrucción ya se está buscando la próxima desde la memoria del programa.

De esta manera es puede ejecutar las instrucciones en un solo ciclo de reloj.

También posee un banco de registros de propósito general de 32x8 bits (prácticamente hace 16x16) que permite que las instrucciones aritméticas y lógicas se hagan en un solo ciclo de reloj.

El microcontrolador tiene una memoria reprogramable Flash de 32 kBytes de almacenamiento.

La memoria se divide en dos espacios de seguridad:
- La sección de inicio de carga (Boot Loader)
- La sección de programa de la aplicación



ARDUINO UNO

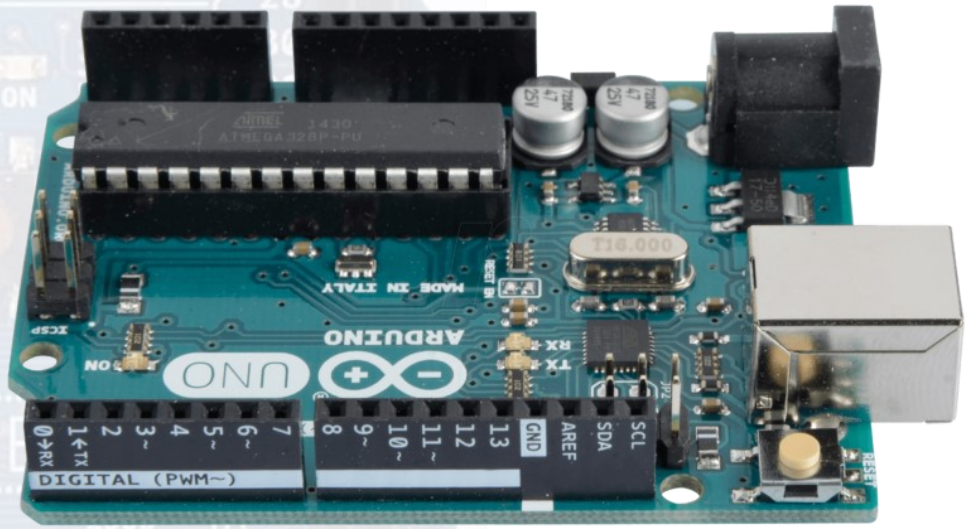
La placa está dispuesta en grupos:

Bajo el procesador hay un grupo de alimentación en que se ven una serie de tensiones y masa con el pin de reset. Algunas tensiones pueden ser de entrada pero también de salida dependiendo de la utilización.

También debajo están las entrada analógicas PC0 a PC5, que proporcionan 10 bits, denominados de resolución.

En la cabecera del procesador están los pines de programación (ICSP) y el pin o el botón de reset, que permite al Arduino volver a su estado original.

En el lado contrario de las entradas analógicas es encuentran las entradas y/o salidas digitales en número de 14, que ofrecen una tensión de 5 voltios.



Dos de ellas la 0 RX i la 1 TX sirven para recibir y transmitir datos en serie.

Los fabricantes tienen la posibilidad de cambiar determinados componentes físicos, pero básicamente aunque cambie la fisonomía de la placa, por debajo permanece la misma filosofía inicial.

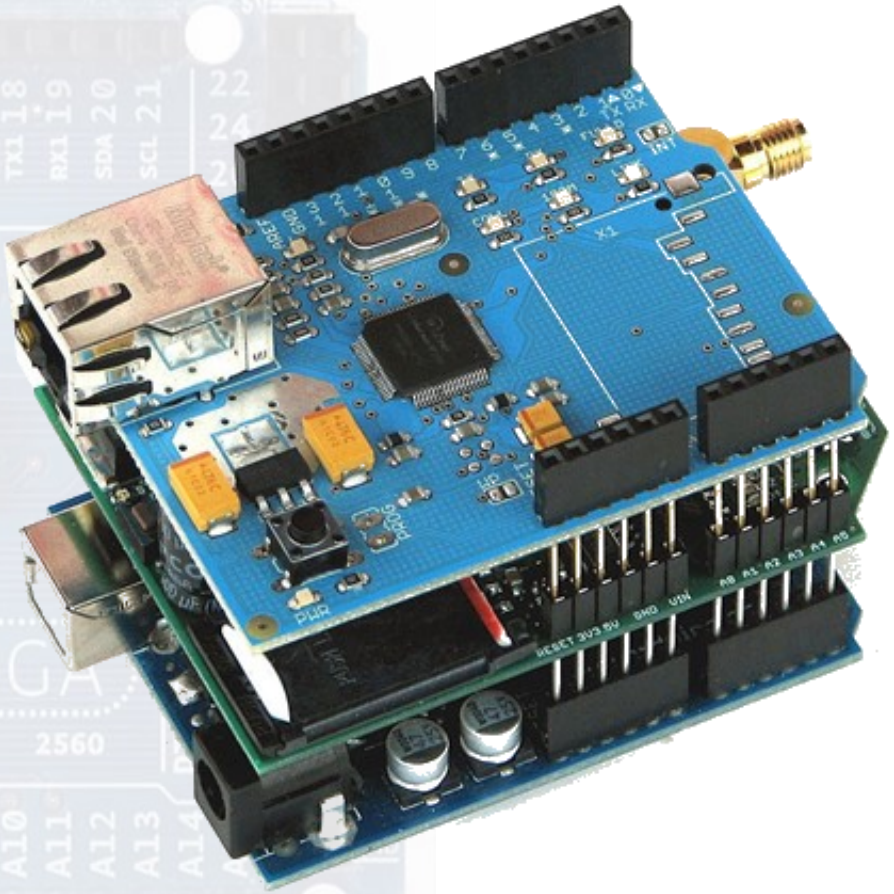
SHIELD

Las placas Shield son placas modulares que se montan unes sobre otras para dar funcionalidad extra a un Arduino. Son apilables.

Cada shield debe tener la misma forma que el Arduino para que sólo tenga una forma de encajar en él.

Es posible que se puede apilar otro shield encima o no, y esto viene dado por los pines de conexión.

Es muy importante leer su documentación para saber si utiliza un bus o inhabilita algunas entradas/salidas. Habitualmente el shield viene con una librería para su utilización, aunque puede utilizar librerías genéricas.



PROGRAMACIÓN DE ARDUINOS

Todos los Arduinos pueden utilizarse específicamente para una finalidad concreta. Ya sea para el control de un robot, una impresora 3D, domótica de la casa, etc.

Para que haga cosas concretas se le tiene que introducir un programa en su memoria flash para que responda de una determinada manera a impulsos de entrada.

La manera más habitual de hacerlo servir es como centralita en que una serie de botones hacen que determinados dispositivo colgados en la salida funcionen, y lo hagan como está previsto en la programación.

En teoría programar cualquier elemento electrónico es introducir un listado con una serie de funciones y datos que funcionen con una cadencia, como hemos dicho antes, a base de impulsos.

El lenguaje de programación permite poner comentarios al desarrollo del programa de tal forma que tenemos la idea de lo que hacen las líneas sucesivas (en neutro tenemos las explicaciones y en color las funciones):

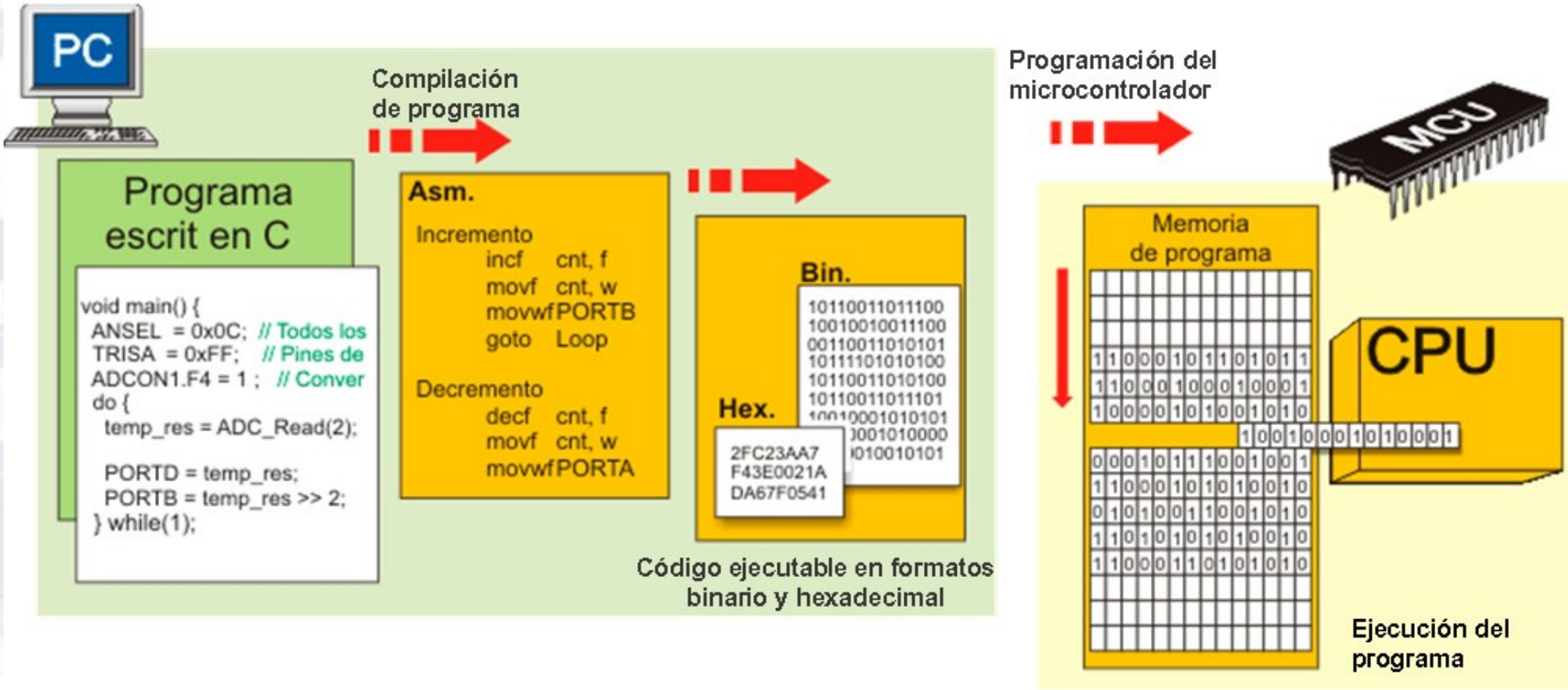
```
void store_log (void)
{
  /* we continously update the "current" entry */
  int logno = get_minutes_since_midnight () / LOG_INTERVAL;
  if ((logno < 0) ||
      (logno >= (24*60/LOG_INTERVAL)))
    logno = 0;
//  datalog[logno].temperature = cached_temperature;
//  datalog[logno].humidity = cached_humidity;
//  datalog[logno].moisture = moisture_read();//cached_moisture * 100 / moisture_calib;
}

void setup(){

    reset_settings (config);
```

COMPILAR Y CARGAR EL PROGRAMA

Para hacer que el microcontrolador entienda que debe hacer, el programa se debe compilar en assembler, binario y hexadecimal. Una vez hechos este paso intermedio ya se puede programar el microcontrolador.



IDE DE ARDUINO

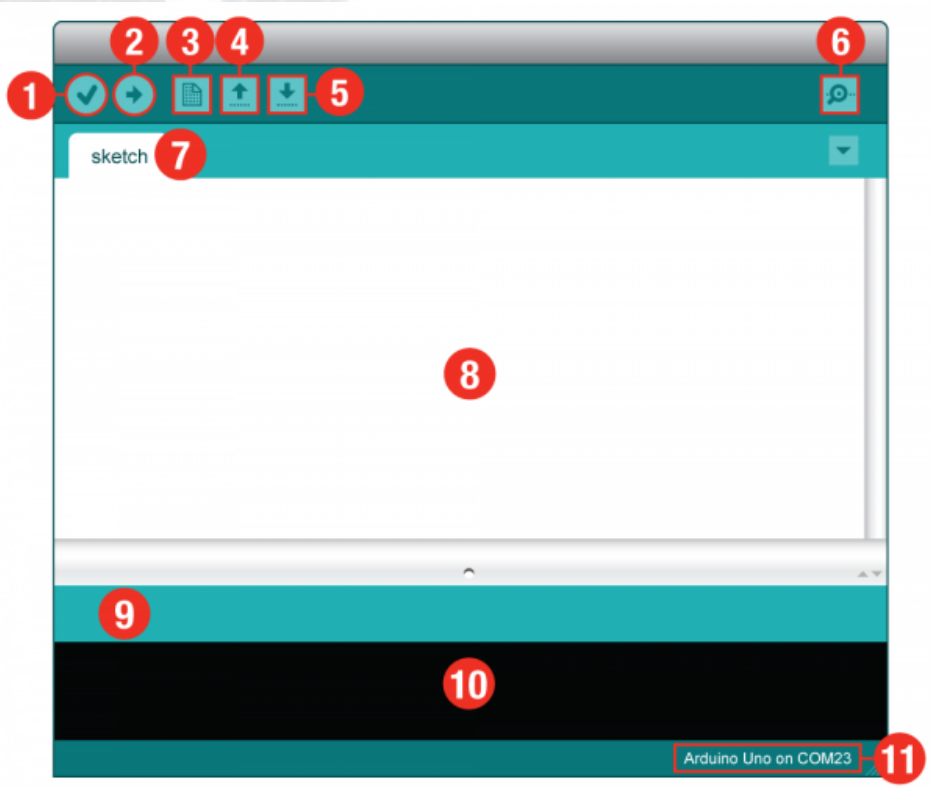
Como Arduino se conecta al ordenador, tenemos una herramienta para programar su funcionamiento de forma rápida y fácil. El IDE.

El IDE es un programa que se instala en el ordenador (hay diferentes versiones según el sistema operativo) y en él se puede hacer el programa que irá al aparato. En el programa se pueden copiar códigos de terceros o hacer el nuestro propio.

Recuerde la filosofía inicial de compartirlo todo.

La ventana tiene unos parámetros básicos en el menú típico del programa:

1. Verificar el código.
2. Cargar el código en la placa.
3. Nuevo: Abre una nueva pestaña de código.
4. Abrir: Abre un archivo ya existente.
5. Guardar: Almacena el archivo que está activo.
6. Monitor serie: Abre una ventana de información serie. Es útil para depurar.
7. Nombre del archivo existente.
8. Área de código, es la que comprende el código.
9. Área de mensajes, dónde se indica si hay errores en el código.
10. Consola de texto, que muestra los mensajes de error completos. Útil para depurar.
11. Placa y puerto serie: muestra el tipo de placa y el puerto al que se conecta.



DESCARGAR EL PROGRAMA

En la web de Arduino se puede descargar el programa que nos haga falta, en su última versión.

La página principal es: https://www.arduino.cc/

Degraciadamente està en inglés o chino.

El IDE sirve para todas las places de Arduino.

INSTALAR EL PROGRAMA

Después de bajar el programa de la web principal, y tras pedir ayuda (si le es necesaria). Para Windows antiguo y Linux es un programa, pero en Windows 8.1-10 ya es una App, y se instala y configurar como tal.

La ruta dónde se instala es Program Files\WindowsApp, y normalmente no podemos tocar allí por problemas de permisos.

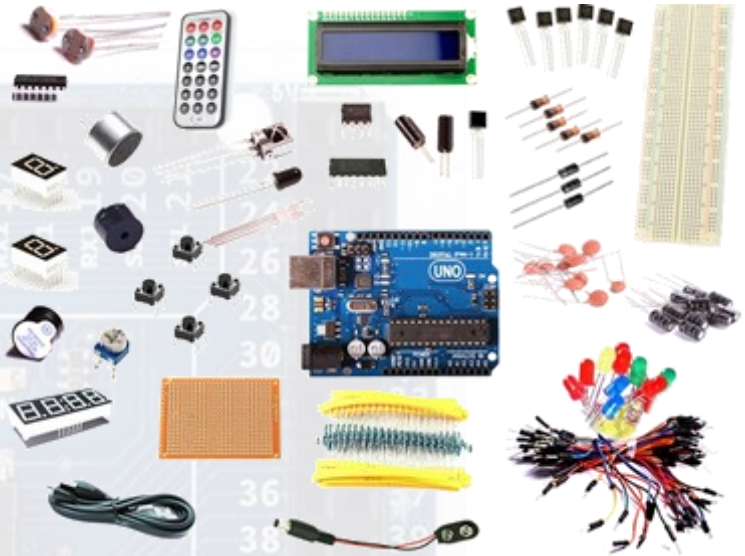
Si se utiliza la placa Arduino Picaro serán necesarios los controladores del chip FTDI.

Ya tenemos el programa instalado y se puede hacer servir para programar el Arduino.

PRIMEROS PASOS

Lo mejor cuando nos ponemos a manejar cualquier cosa nueva es hacer pruebas.
Una buena solución sería comprar un kit con una placa Arduino Uno con accesorios, especialmente la placa “breadboard” que serà la base de operaciones para “pinchar” todos los elementos que haremos servir en nuestras prácticas.

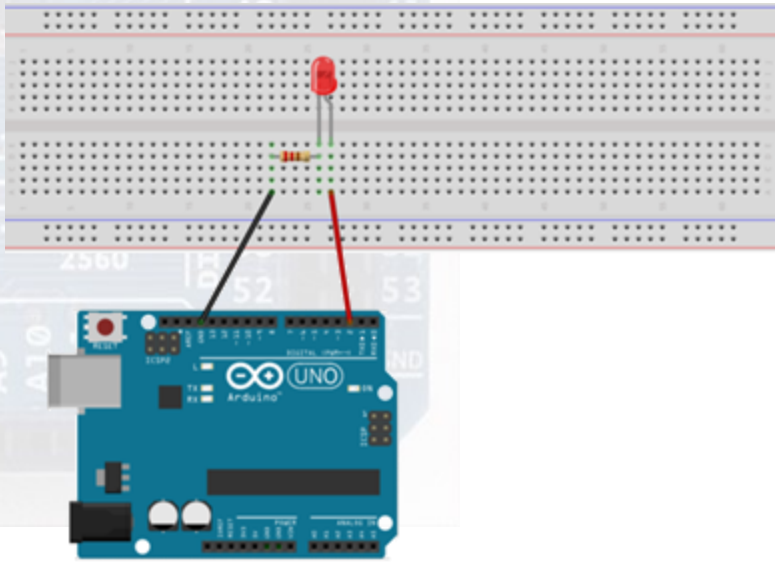
Aunque encontraremos todos los esquemas en Internet, y gratis, puede ser factible comprar el kit de inicio con manual y prácticas.



PRIMERA PRÁCTICA

Describo aquí de forma rápida lo que debería ser nuestra primera práctica con Arduino (a la derecha el montaje, debajo la programación)

- Conectar un LED a una de las salidas.
- Mediante la programación posibilitar que haga intermitencias.




```
1. void loop() {
2.   digitalWrite(13, HIGH);
3.   delay(500);
4.   digitalWrite(13, LOW);
5.   delay(500);
6. }
```

CONTROL DE ACCESORIOS CON ARDUINO

PASOS BÁSICOS PARA UTILITZAR ARDUINO

EN UNA MAQUETA DE TREN MINIATURA

ESTRATEGIA

Llevar a cabo un proyecto para Arduino implica una serie de parámetros a tener en cuenta:

1. Saber evaluar nuestra pericia, porque no es lo mismo comenzar sabiendo cosas de electrónica que desde cero. A veces un buen manual puede servir.
2. Saber evaluar la complejidad del proyecto propio, ya que poner en marcha pequeños gadgets de manual no es lo mismo que iniciar un proyecto propio que tenga cierta complejidad i/o dificultad.
3. Saber evaluar las diferentes fases de trabajo para terminar el proyecto, recordando que Arduino no hará aquello que nos hemos olvidado.
4. Comparar les diferentes soluciones tecnológicas, tanto en la parte del hardware como en la del software.
5. Tenemos que saber encontrar la documentación adecuada.
6. Copiar del todo o inspirarse en el trabajo de otros.
7. Per cada fase se tiene que trabajar el algoritmo, el cuadro sinóptico de lo que se tiene que hacer en cada momento.
8. Evaluar los recursos y escoger el Arduino adecuado.
9. Escribir el código en diferentes versiones nos puede ayudar a encontrar el método de control más preciso.
10. Establecer un calendario y saber dónde encontrar ayuda.

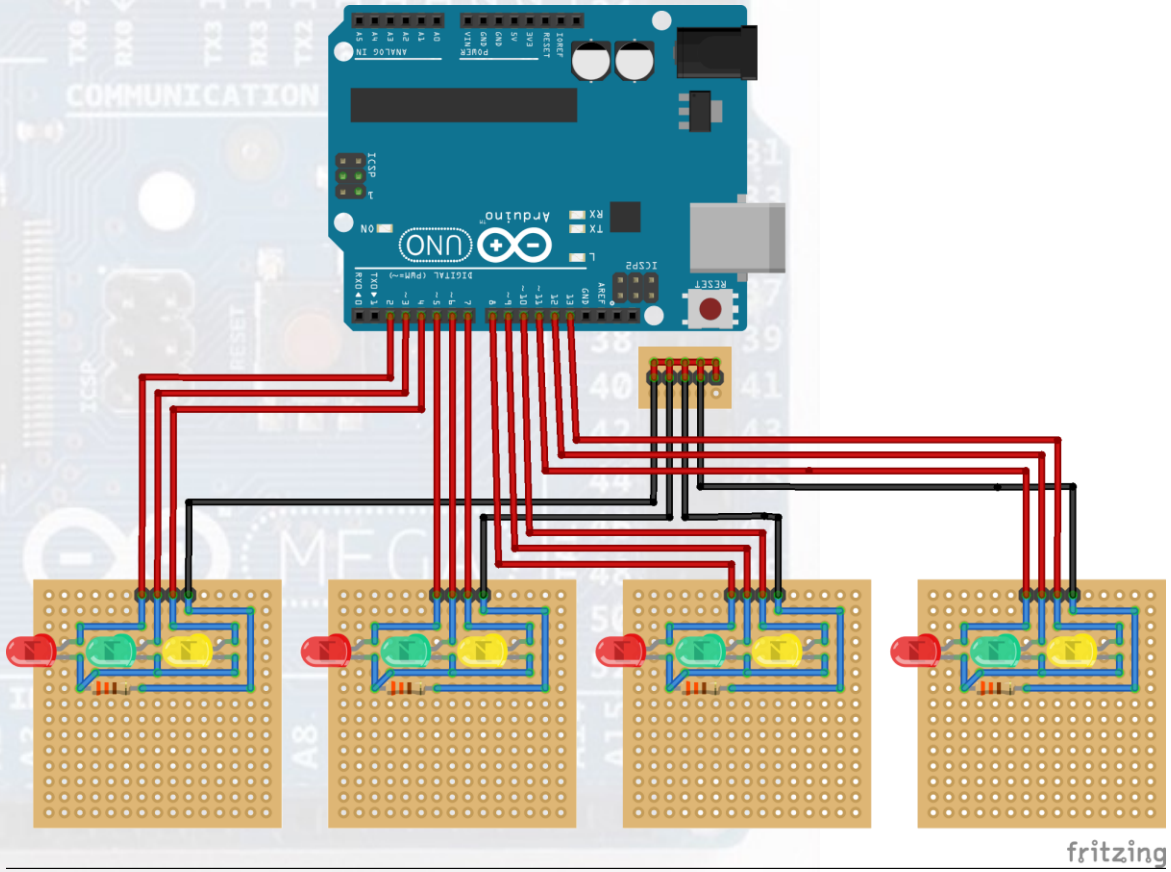
CONTROL DE UN SEMÁFORO DE CRUCE

Una de las aplicaciones más sencillas que se puede hacer con Arduino son los semáforos, esta aplicación consiste en crear una secuencia de LED de color rojo, amarillo y verde, simulando el funcionamiento de cuatro semáforos y controlar un cruce de calles.

El código es bastante sencillo, hace la secuencia correspondiente para cada semáforo, en la sección de constantes puedes variar el tiempo de duración de cada luz, que se expresa en milisegundos.

La secuencia completa ya no es tan sencilla como la del LED intermitente, pero es muy repetitiva y la forma seria dar unas constantes en primer lugar, unas conexiones en las siguientes líneas y el reinicio, i posteriormente está la secuencia para los cuatro semáforos.

En total se escribe 134 líneas de código.



CONTROL DE UN SERVO

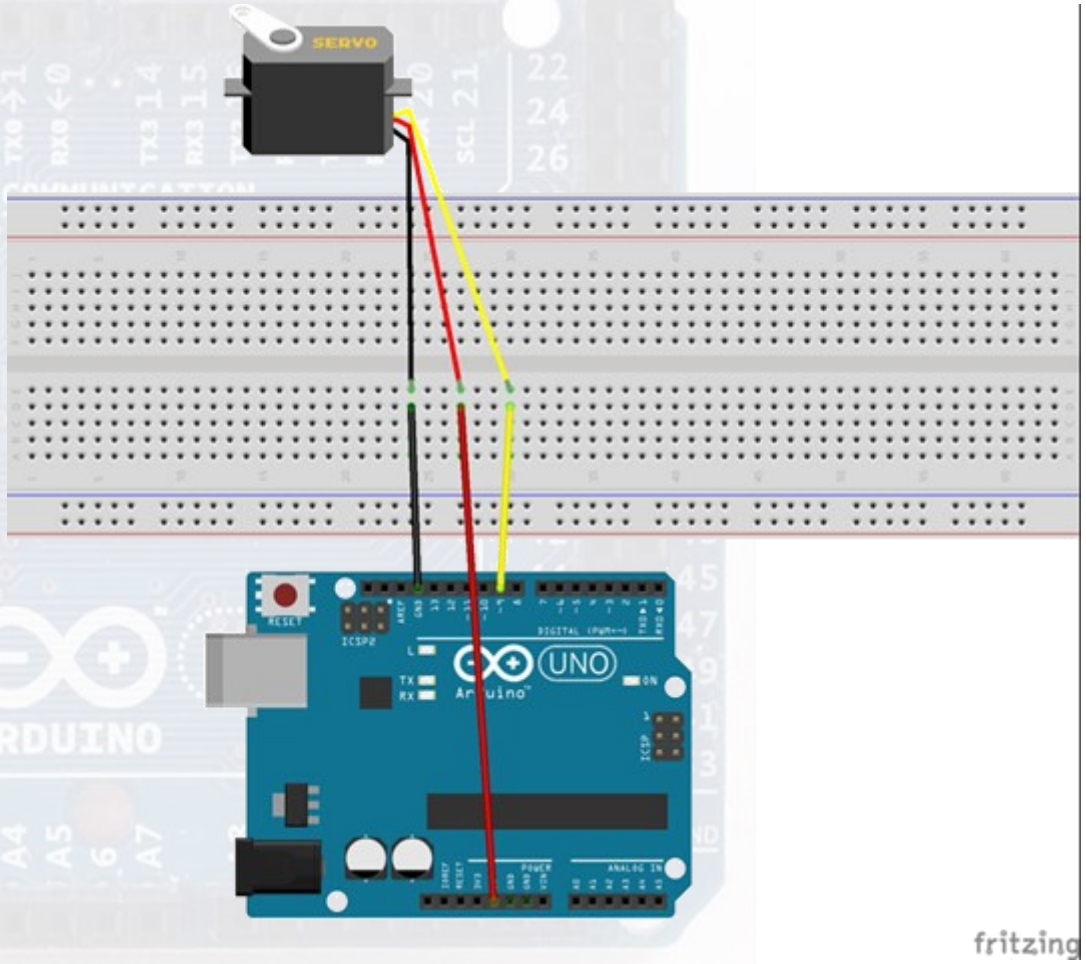
El servomotor es un dispositivo motorizado que permite hacer un ángulo de giro de 180 grados.

Se puede mover con una resolución de más de un grado, pero éste es el máximo de resolución que puede dar un Arduino UNO.

Este tipo de motores funcionan con una señal PWM con un pulso de Trabajo entre 1 s y 2ms, con un periodo de 20 ms (50 Hz). Esto quiere decir que es la velocidad máxima en la que podremos mover los servos con un Arduino.

Todos los motores servo tiene que tener 3 cables, uno irá a tierra o masa, el otro es la alimentación positiva a 5 voltios y el tercero es un pin PWM que da la señal para mover el servo.

También se puede hacer servir un shield para controlar los servomotores, si éstos no pudieran ser controlados por el Arduino.



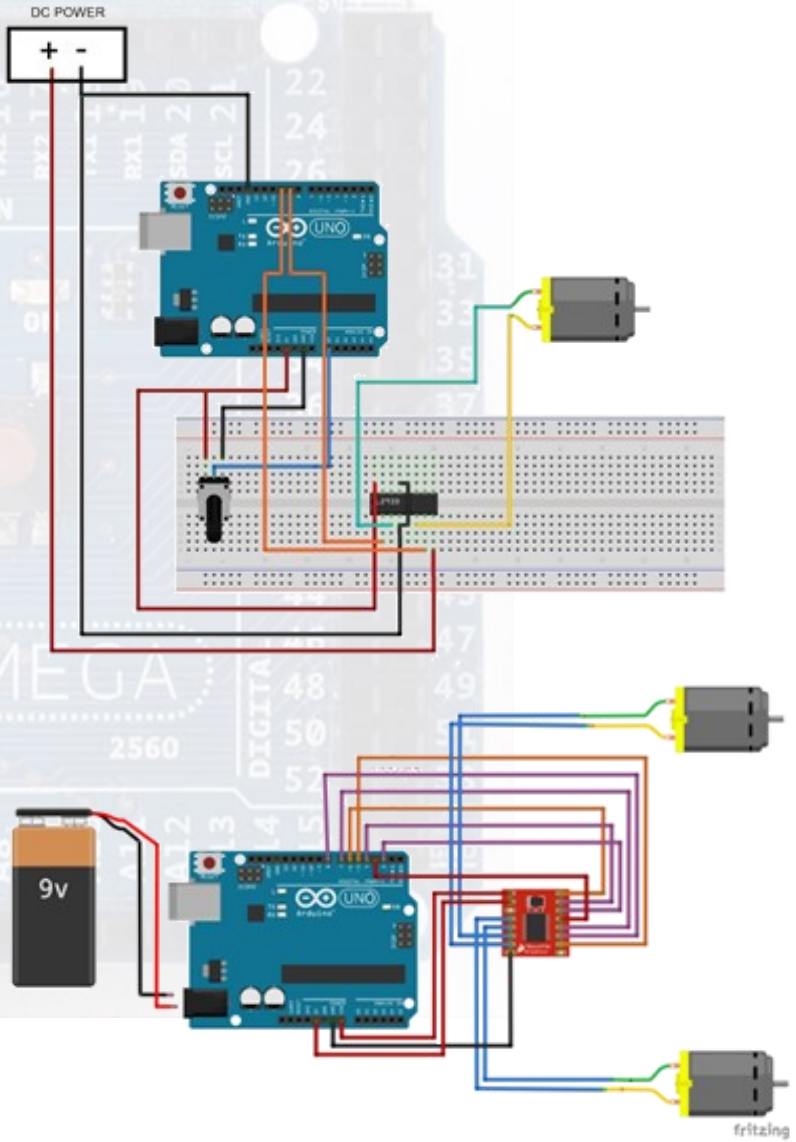
CONTROL DE UN MOTOR CC

El motor de corriente continua se puede controlar directamente por el Arduino a través de un montaje.

El motivo es que el Arduino no tiene potencia suficiente para mover motores, sino que normalmente da la orden para moverse.

Se puede hacer servir un potenciómetro para graduar la velocidad y sentido del motor mediante un L293N, o si sólo es en un sentido mediante un transistor de potencia.

Lo más habitual es utilizar un L298N que sirve para controlar motores de corriente continua (aunque también lo puede hacer con motores paso a paso). La corriente que puede suministrar es de 2ª y de 3 a 35V. Pero con un inconveniente, que se calienta para poder perder 3 voltios para llegar al motor.



CONTROL DE UN MOTOR PASO A PASO

El motor paso a paso (también llamado stepper) es un dispositivo que convierte impulsos eléctricos en movimientos mecánicos de rotación. Es decir, se mueven un paso por cada impulso de reciben.

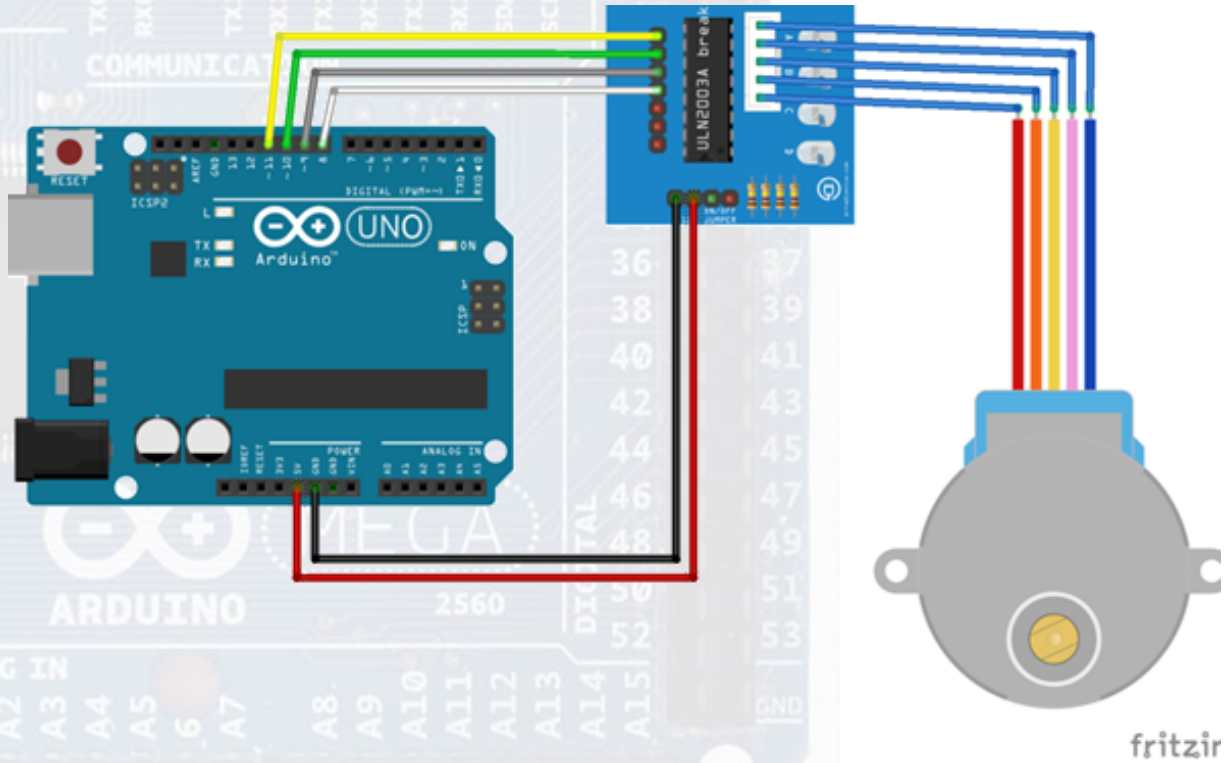
Hay dos tipos de motores paso a paso:

- Unipolar que contiene 4 bobinas con común entre parejas de bobinas (6 terminales)
- Bipolar, que contiene 2 bobinas (4 terminales)

Dependiendo del tipo de motor se hace un montaje u otro para hacer el control del motor paso a paso.

Lo más habitual es utilizar un ULN2803A (transistor Darlington) que sirve para dar potencia a las señales provenientes del Arduino 8uns 500 mA).

Hay unos shields expresamente concebidos para los motores paso a paso que se utilizan en las impresoras 3D qe tiene entre 4 y 5 salidas para controlar todos los motores a la vez.



ENLACES

PASOS BÁSICOS PARA UTILITZAR ARDUINO

EN UNA MAQUETA DE TREN MINIATURA

LOCODUINO Arduino per el tren miniatura, en francés: https://www.locoduino.org/

7 TRAINT PROJECTS https://create.arduino.cc/projecthub/projects/tags/train

ARDUINORAILWAYCONTROL https://arduinorailwaycontrol.com

APRENDIENDO ARDUINO https://aprendiendoarduino.wordpress.com/

FRITZING Open hardware Initiative: http://fritzing.org/home/

ARDUINO Página oficial en inglés: https://www.arduino.cc/

EDUCALAB Taller de robòtica: http://educalab.es/documents/10180/640047/TallerRoboticaLibreArduino.pdf/c77adbfd-606a-4fbe-acd4-11630927b5a4