ARDUINO Y COMPUTACION FISICA

¿Qué es la computación física?

Se centra en diseñar, y utilizar dispositivos, objetos e incluso entornos que permitan establecer un canal de comunicación entre el mundo físico y el mundo virtual. Aunque esto suene a algo relacionado con el futuro, la realidad es que llevamos conviviendo con esta disciplina mucho tiempo. Solo tenemos que pensar en un ordenador o en un dispositivo móvil. Estos dispositivos ponen en contacto nuestro mundo, el mundo físico, y el mundo virtual de las máquinas y ordenadores.
El diálogo se realiza a través de interfaces hardware como un teclado, ratón, micrófono, pantallas, altavoces etc… La finalidad de la computación física es diseñar estas interfaces para que sean capaces de detectar alteraciones en el medio físico y traducirlas a señales que entiendan las máquinas. Esto se hace tanto a través de software como de hardware.
En la actualidad, cuando desarrollamos un software para un ordenador o un dispositivo móvil, estamos aplicando computación física. El problema que existe es que solo tocamos una de las patas de esta disciplina. Las interfaces hardware de los dispositivos actuales, nos limita a la hora de diseñar software para esas interfaces. Gracias al Open Hardware, que está relacionada de forma directa con la filosofía del opensource, esto está cambiando.
En resumen, para avanzar en el mundo de la computación física, necesitamos de ciertos componentes hardware (transductores) que conviertan los cambios de energía producidos por las alteraciones en el medio físico, en señales eléctricas entendibles por los ordenadores y máquinas. Aquí es donde entran en juego los sensores. Son los encargados de transformar una magnitud física en una señal eléctrica. Pero como esta comunicación es bidireccional, también necesitaremos actuadores que convierten las señales eléctricas en magnitudes físicas.
Los ordenadores, ya sean microprocesadores o microcontroladores, son los encargados del control de los sensores y actuadores. Deben ser capaces a la vez, de comunicar con otras máquinas para mostrar los datos en pantallas multimedia o almacenar información en base de datos o en la nube.

Diseño de interfaces hardware en la computación física

Como ya hemos visto, una de las partes importantes dentro de la computación física es el hardware. Aquí se ve involucrado tres componentes de hardware a nivel básico.
Los sensores serán los encargados de detectar esas alteraciones en el medio físico y transformarlas en señales eléctricas. Tenemos como ejemplos el sensor de temperatura LM35 o el DHT11 que además mide la humedad. Dentro de esta categoría podemos encontrar también el sensor de ultrasonidos HC-SR04 con el que podemos hacer un sensor de nivel de agua.
Los actuadores harán lo contrario, convertir señales eléctricas en magnitudes físicas que activan procesos. Los motores paso a paso de las impresoras 3D o los relés son ejemplos de actuadores. Incluso un LED funciona como tal.

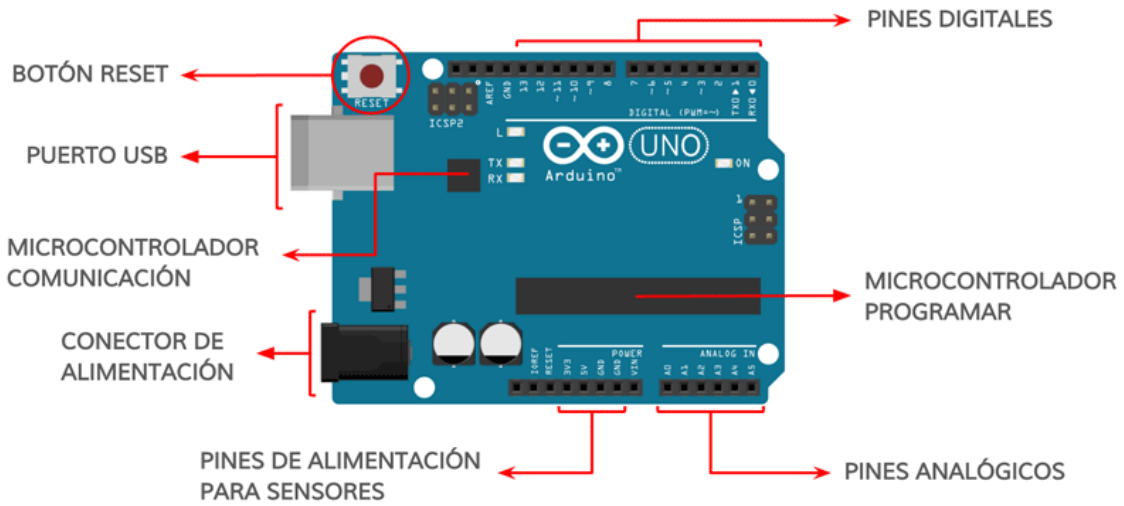
El microcontrolador será el encargado de controlar y gestionar los sensores y actuadores. Podríamos pensar en utilizar un microprocesador, nos daría más potencia y funcionalidades y es verdad, pero los microcontroladores tienen ciertas características por las cuales es la opción más óptima. Dentro del propio chip viene todo integrado, memoria RAM donde se almacenan datos temporales, memoria ROM o Flash para almacenar los programas y dispone de entradas y salidas que nos permitirán conectar los sensores y actuadores.
Además, los requerimientos que vamos a necesitar no son muy exigentes, solo procesaremos la señal de entrada y salida. Los microcontroladores son más económicos y más compactos que los microprocesadores. Por último, nos permite comunicar con otras máquinas a través de diferentes protocolos, una característica que se requiere a la hora de diseñar interfaces hardware para la computación física.
Y cuando hablamos de microcontroladores, se nos viene a la cabeza Arduino. Sin duda alguna es la mejor opción para realizar nuestros proyectos.

¿Qué es Arduino?

Arduino es una placa microcontroladora para el prototipado. Fue creada en el año 2005 en el Instituto IVREA, en Italia por cinco personas: Massimo Banzi, David Cuartielles, David Mellis, Tom Igoe y Gianluca Martino. En principio estaba destinada a estudiantes para que pudieran conectar componentes y hacer sus desarrollos.
Dos características hacen única a esta placa, su bajo costo y su sencillez a la hora de utilizarlo tanto a nivel de software como de hardware. Además, cuando adquirimos un Arduino no solo estamos comprando una placa, también tenemos a nuestra disposición una plataforma totalmente abierta. Tenemos un entorno de desarrollo de código abierto, podemos bajarnos su código fuente, y una documentación muy extensa.
Por lo tanto, si vas a adquirir uno, piensa antes de comprar una copia, ¿quieres que el proyecto y la plataforma sigan siendo libres y de acceso a todo el mundo?
Pero quizás la idea más fantástica que tuvieron estos cinco señores fue publicar el diseño de la placa microcontroladora con licencias libres Creative Commons. Esto significa que cualquiera puede fabricar una replica o modificar y adaptar el diseño a sus requerimientos sin pedir permiso. Es más, podemos llegar a comercializar la placa sin ningún problema eso si, siempre respetando la imagen de marca.
Dentro de la gama de productos que nos ofrece Arduino, el más utilizado y comercializado es el Arduino UNO.

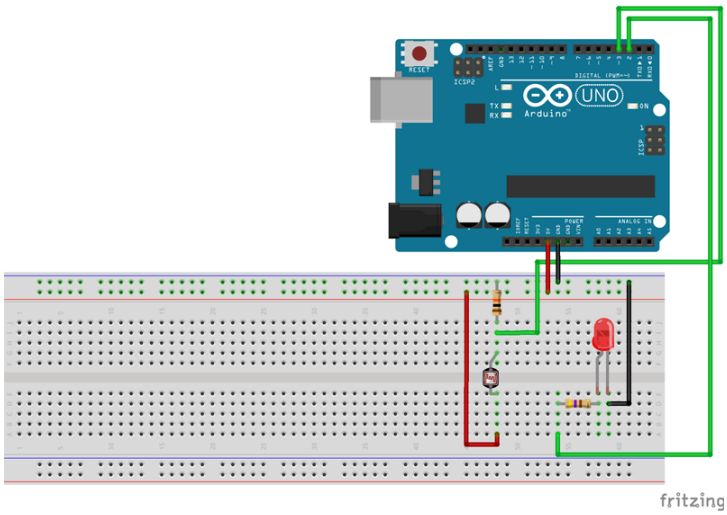
Arduino UNO, los puntos clave para conocer y controlar la placa

Tanto como si te estás iniciando como si ya te has iniciado, es recomendable que recuerdes los 7 puntos clave para tener un control total sobre Arduino UNO.



Pines digitales: sirven para conectar los sensores y actuadores digitales.
Botón reset: pulsando este botón podremos reiniciar la ejecución del programa. Esto no borra el programa ya cargado.
Puerto USB: nos servirá para cargar los programas y para alimentar la placa.
Conector de alimentación: gracias a este conector, podremos alimentar la placa de forma externa, con pilas o con batería.
Pines de alimentación para sensores: estos pines suministran tensiones de 3,3V o 5V, además de la toma de tierra.
Pines analógicos: para conectar sensores analógicos.
Microcontroladores: dispone de dos, uno que se encarga de la comunicación a través del USB y otro para almacenar la lógica de la placa.

En el siguiente ejemplo, te muestro cómo podemos utilizar Arduino para conectar sensores y actuadores. Se trata de utilizar una fotoresistencia, la cual cambia sus propiedades dependiendo de la luz. Si hay mucha luz disminuye la resistencia y por lo tanto sube el voltaje de la señal eléctrica. Si hay poca luz aumenta la resistencia y por lo tanto baja el voltaje. Este componente hará de sensor. El actuador será un LED. Se encenderá cuando la fotorresistencia capte muy poca luz y se apagará en caso contrario. Veamos como sería el montaje del circuito



Con Arduino ya tenemos resuelto todo lo necesario con respecto a la interfaz hardware que comunique el mundo físico y el mundo virtual. Pero como todas las máquinas, estas deben ser programadas. Como introducción a la programación de Arduino, miremos como es el código necesario para hacer funcionar el anterior circuito.

```
// Número de pin
const int LEDPin = 2;
const int LDRPin = 3;

void setup()
{
  // Pin LED modo salida
  pinMode(LEDPin, OUTPUT);
  // Pin fotoresistencia modo entrada
  pinMode(LDRPin, INPUT);
}

void loop()
{
  // Obtenemos valor del pin de la fotoresistencia
  int value = digitalRead(LDRPin);

  // Si hay poca luz, estado bajo, encendemos el LED durante 5 segundos
  if (value == LOW)
  {
    digitalWrite(LEDPin, HIGH);
    delay(5000);
    digitalWrite(LEDPin, LOW);
  }
}
```

Con 17 líneas somos capaces de crear un sistema de encendido de luces automático. Esto es lo realmente increíble de Arduino, su facilidad tanto a nivel hardware como de software, haciendo que prototipar y crear proyectos, esté al alcance de todo el mundo.
Ahora veremos cómo programar el software necesario para nuestros proyectos.

Programando Arduino de diferentes formas

Ya hablamos de dónde venía el código nativo de Arduino. El resumen es que se nutre de dos plataformas creadas por el MIT (Massachusetts Institute of Technology), Processing y Wiring. Además, a mi me gusta decir que también se alimenta de GitHub, la plataforma de código abierto que nos proporciona multitud de librerías para este lenguaje.

Comencemos a ver las diferentes formas de programar Arduino empezando por el código nativo. Es recomendable tener al menos nociones básicas en algún lenguaje de programación para iniciarse con este método.

Código nativo de Arduino

Entorno de desarrollo oficial

Es el primer paso que debemos dar si lo que queremos es programar con código nativo de Arduino basado en C/C++. Lo podemos conseguir de forma gratuita en la web oficial y tenemos a nuestro alcance todo el código fuente. Es muy sencillo e intuitivo, la gran pega es que no dispone de herramientas como Intellisense o Debugger para poder programar con más sencillez.
Es multiplataforma, tenemos versiones para Windows, Linux y Mac.

https://www.arduino.cc/en/Main/Software

Visual Micro

Visual Micro es una extensión para Visual Studio. Esto no quiere decir que vayamos a programar en otro lenguaje de programación que no sea el código nativo de Arduino. Es un complemento que instalamos a Visual Studio que nos permite programar para Arduino. El código es igual que en el IDE oficial, la gran diferencia es que aquí si que tenemos Intellisense y Debugger, una gran ventaja para programar.
Solo está disponible para Windows.

http://www.visualmicro.com/

Code Bender

Es una plataforma web donde tenemos las mismas opciones que en el IDE oficial. La ventaja es que podemos almacenar nuestros programas en la nube.
Es multiplataforma.

https://codebender.cc/

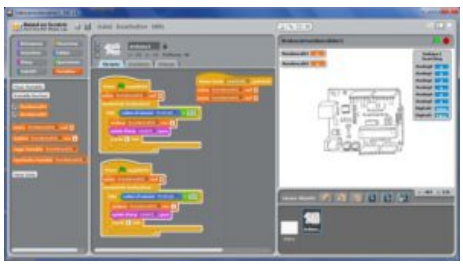
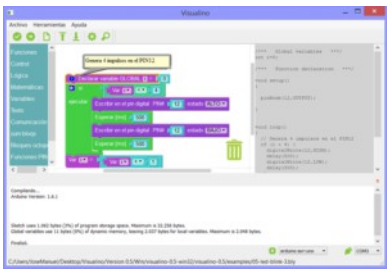
PlatformIO

Este entorno de desarrollo es más complicado de utilizar y de configurar. Está destinado a crear proyectos del IoT (Internet of the Things o Internet de las Cosas) pero además nos permite programar microcontroladores entre ellos, Arduino. Es una modificación del entorno de desarrollo Atom de GitHub.
Es multiplataforma.

http://platformio.org/

Lenguajes de programación visual

Si tu experiencia con los lenguajes de programación es prácticamente inexistente, la mejor manera de introducirte en el mundo de la computación física es empezar por los lenguajes de programación visual.
Los entornos de desarrollo más sencillos y populares son **Scratch4Arduino** y **Snap4Arduino**.
Los dos fueron creados por el grupo Smalltalk del Citilab. Como todo lenguaje de programación visual, utiliza bloques para codificar las diferentes sentencias. Arrastrando y soltando podemos llegar a crear nuestras aplicaciones. Gracias a esto nos abstraemos de la sintaxis de un lenguaje de programación, enfocándonos en lo realmente importante, el pensamiento computacional.
El gran inconveniente de estas plataformas, es que debemos tener conectado siempre el Arduino al ordenador donde tengamos instalado el entorno de desarrollo de Scratch4Arduino o Snap4Arduino. No permite que Arduino sea independiente o autónomo. Esto es debido a que utilizan el protocolo de comunicaciones estándar Firmata, más adelante veremos para qué sirve.
Para resolver este gran problema, podemos utilizar **Visualino**. Esto, junto a que podemos ver el código nativo según vamos añadiendo bloques, hace que sea le mejor opción para la gente que se quiera introducir en la rama de la programación dentro de la computación física.


Herramientas para el prototipado

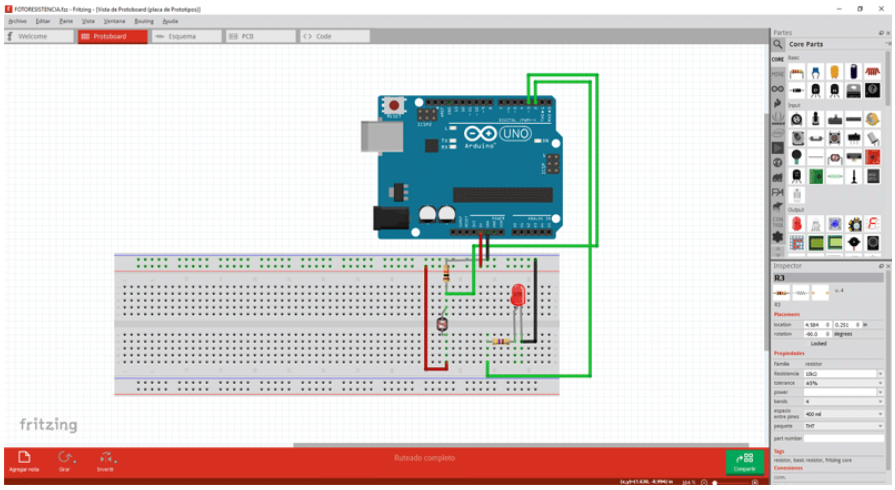
Las herramientas de software para el prototipado nos van a servir para dos cosas, para documentar y diseñar nuestros proyectos y para hacer simulaciones. Dentro del gran abanico que encontramos en Internet, la mas utilizada es Fritzing

http://fritzing.org/

Se trata de un software libre muy útil para documentar, diseñar y prototipar nuestros proyectos eléctricos. El inconveniente que tiene es que no simula. Las cuatro funcionalidades básicas son.

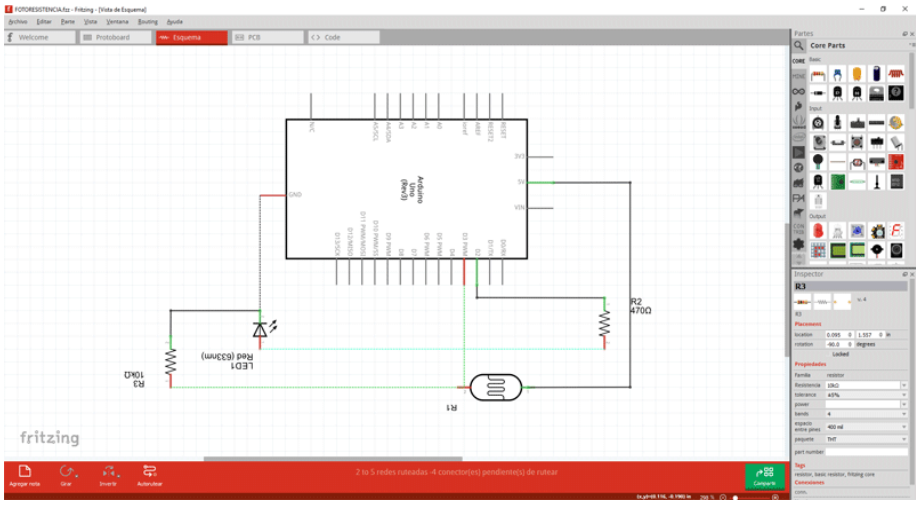
Protoboard

Diseña tu circuito arrastrando y soltando componentes dentro de la protoboard. Te permite cablear y documentar las conexiones del prototipo.



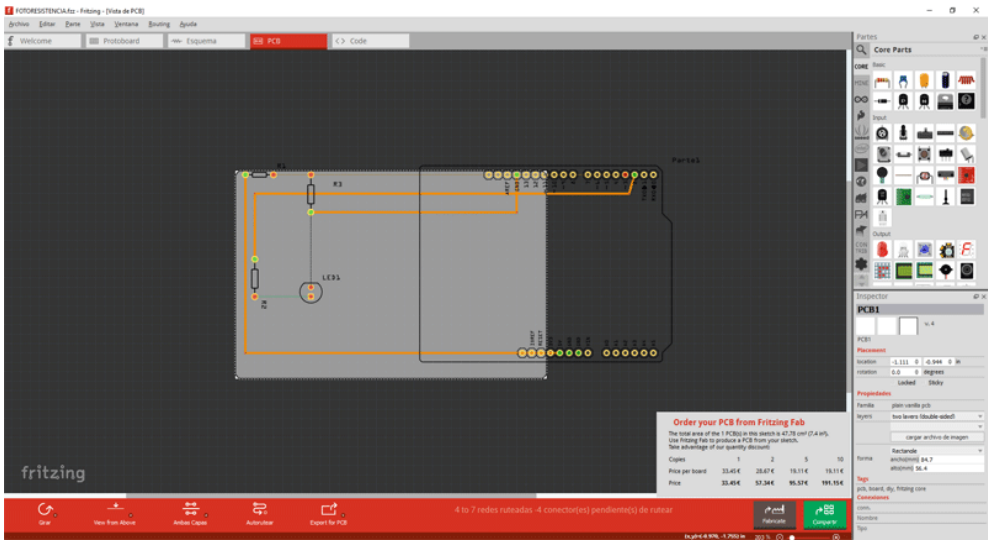
Esquema eléctrico

Todo lo creado en la pestaña de protoboard, te lo muestra como un esquema eléctrico tradicional.



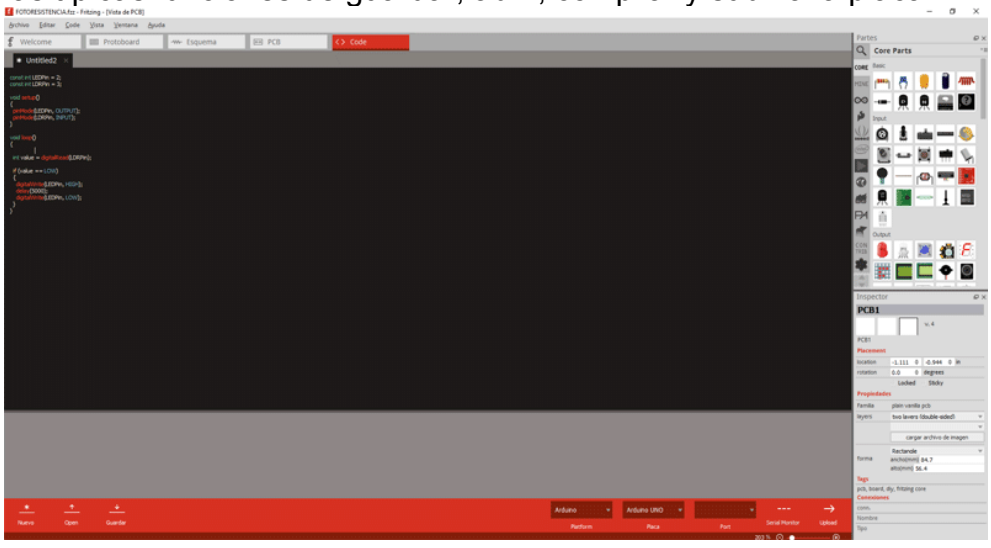
PCB (Printed Circuit Board)

A la vez, te muestra como quedaría en una PCB. En este punto tenemos la posibilidad de hasta enviar a que fabriquen la placa.



Código

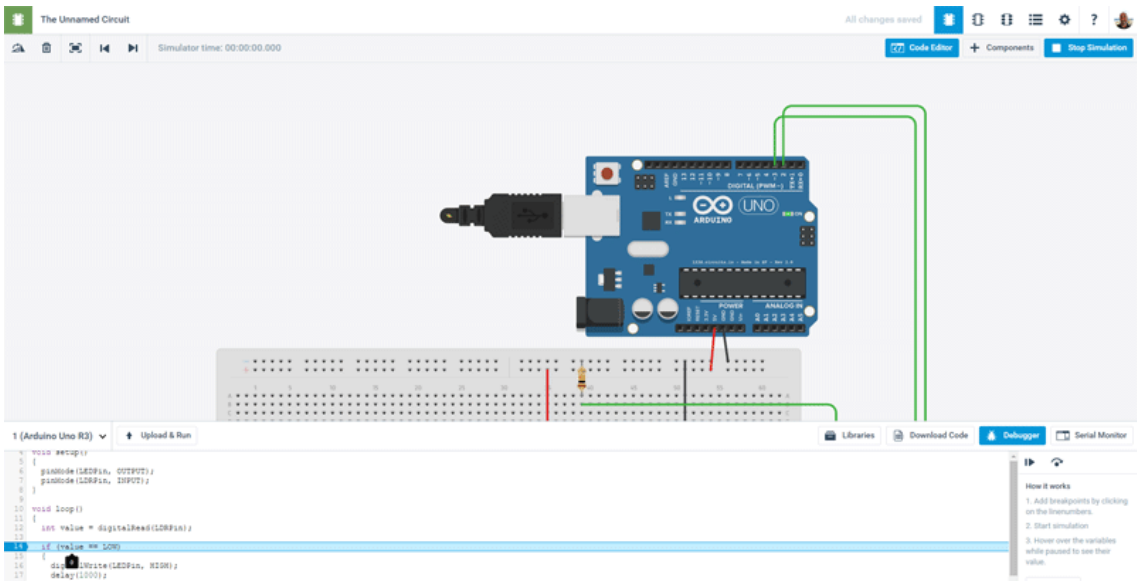
Por último, te permite tener algo parecido al entorno de desarrollo oficial de Arduino. Con las típicas funciones de guardar, abrir, compilar y subir a la placa.



CircuitsIO

https://circuits.io/

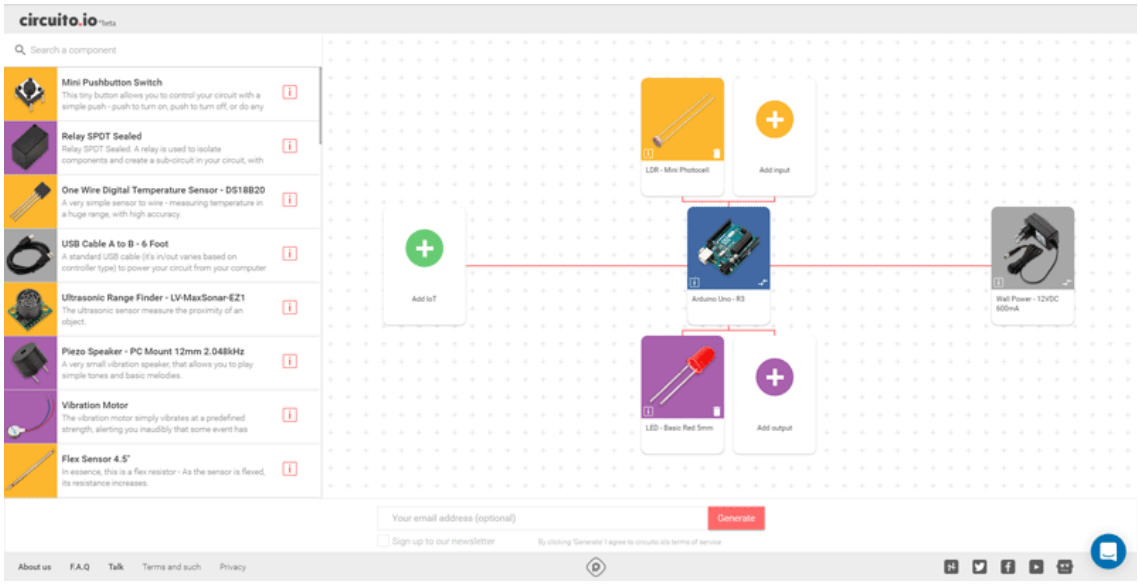
Este es el nuevo nombre que le han dado. Al software 123D Circuits de la empresa Autodesk. Es una aplicación web donde podremos arrastrar componentes a una protoboard para diseñar, crear esquemas eléctricos y PCBs.
La características más interesante que tiene esta herramienta es que te permite simular a nivel de código y eléctrico. Podrás cambiar las propiedades de tus sensores para ver la respuesta y debuggar el código, mirar en tiempo real el flujo del programa y el valor de las variables. Sin duda alguna una buena herramienta para probar antes de montar.



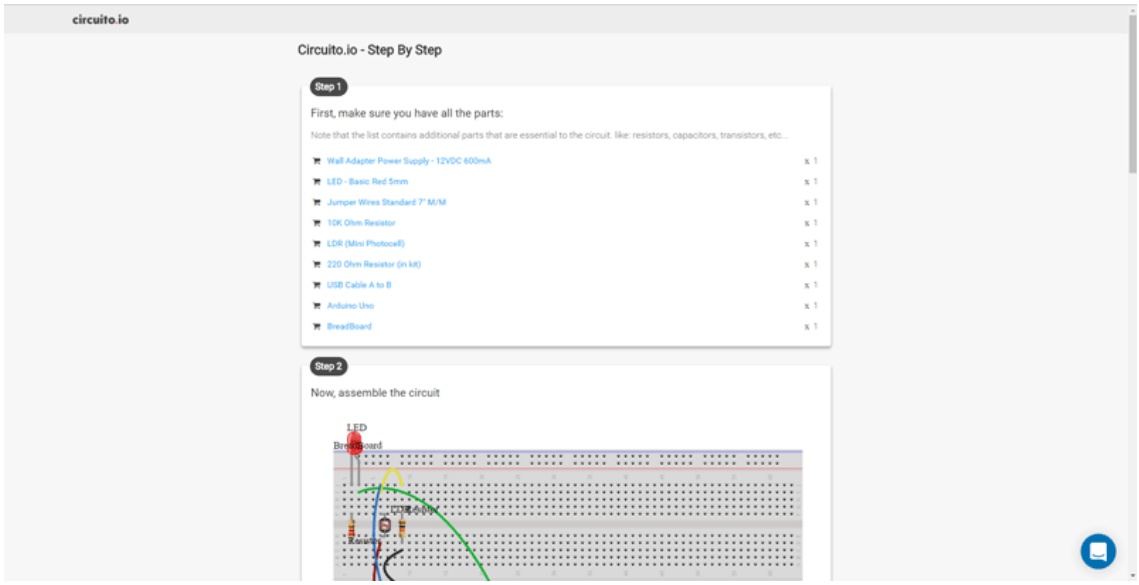
CircuitoIO

https://circuito.io/

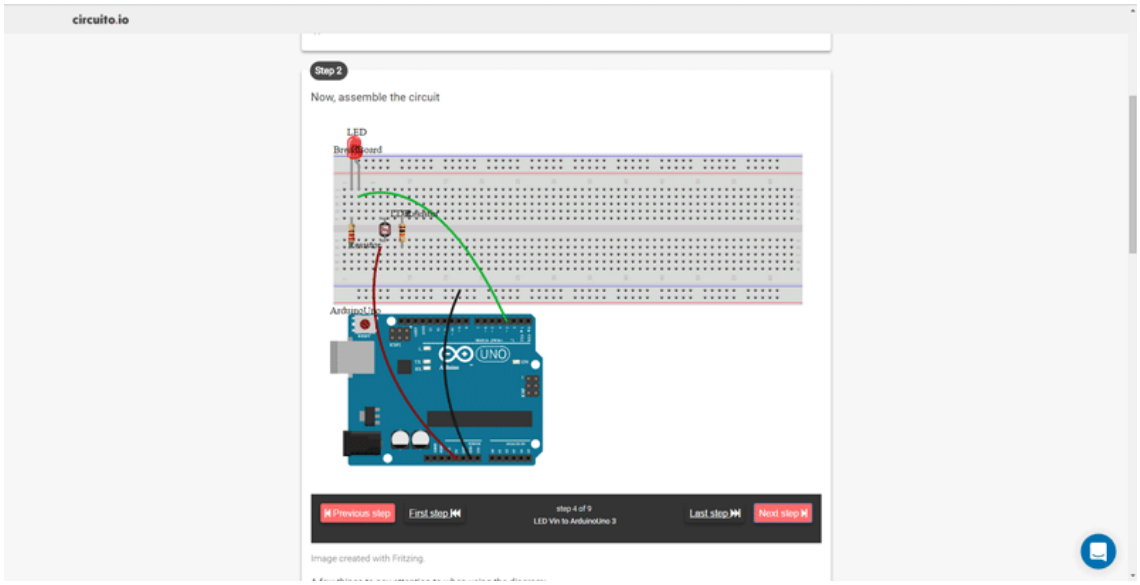
Se trata de una plataforma que nos reduce el esfuerzo a la mínima expresión. Hasta hace bien poco estaba en una fase alpha, pero ya parece que ha sido lanzado una versión beta más estable. Aquí prácticamente la herramienta prototipa por nosotros. Lo único que tenemos que hacer es arrastrar los componentes a un panel donde separamos los sensores de los actuadores. Una vez elegidos, generamos el proyecto y la aplicación es capaz de decidir donde debe conectar cada componente y si necesita algún tipo de resistencia para su funcionamiento. Todo esto te lo dice en cuatro pasos.
El primer paso es seleccionar los componentes.



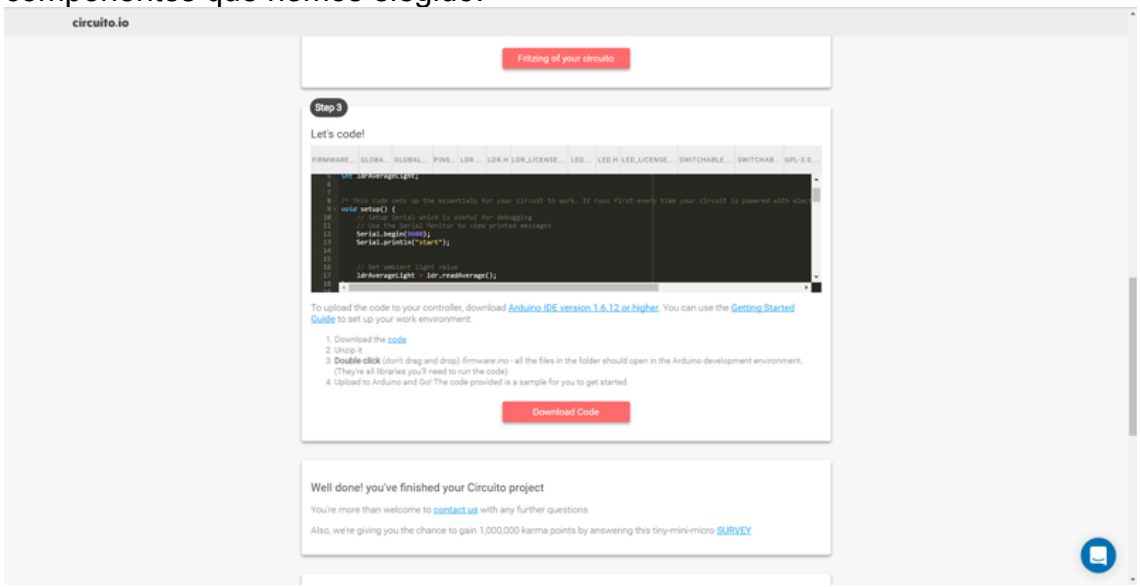
El segundo paso son los materiales y componentes que debes tener. El carrito que se muestra al lado de cada componente, enlaza con una empresa que colaboran con ellos Sparkfun.



El tercer paso es el cableado. En principio no aparece ningún cable. Si pulsas el botón next step, te muestra cómo conectar el primer cable. Así sucesivamente hasta que todos los cables estén conectados.



El cuarto paso es el código. Con los componentes elegidos te propone una programación que, en la mayoría de los casos programa la funcionalidades típicas del grupo de componentes que hemos elegido.



Tenemos a nuestra disposición plataformas tanto hardware como software, libres y de fácil acceso. También disponemos de las herramientas suficientes para que cualquier persona se pueda iniciar. Nunca ha sido tan barata ni tan accesible la computación física, electrónica, programación y demás áreas. Aprovechemos el momento y creemos todo aquello que se nos ocurra.

Solo depende de nosotros tener un mundo mejor y solo lo podemos mejorar si lo conocemos.