1. Toma de contacto

Pensamiento computacional

El origen del término Pensamiento Computacional fue marcado por Jeannette M. Wing. El pensamiento computacional está directamente relacionado con la informática. Representa una aptitud aplicable universalmente para resolver problemas, diseñar de sistemas, pensar de forma paralela y recursiva. Implica conceptualización, abstracción y descomposición de tareas complejas. Jeannette M. Wing, debido a todas las ventajas que aporta el pensamiento computacional, propone la inclusión de este tipo de formación en edades tempranas.

J. M. Wing, 2006. Computational Thinking. Communications Of The ACM. 49 (3).

https://www.cs.cmu.edu/~CompThink/papers/Wing06.pdf

> Tengamos en cuenta antes de empezar, que el pensamiento computacional puede llegar a ser muy variado entre dos personas.

En este curso vamos a ir construyendo la programación **de un juego paso a paso**. Iremos incluyendo bloques partiendo de las necesidades más simples y directas, luego las iremos completando según necesidades más ambiciosas. La secuencia de creación del juego y de programación de los bloques en este curso es bastante lógica y racional.

Pero hay que tener en cuenta que el orden de pensamiento computacional para crear programación puede realizarse de formas muy diferentes. En la realidad, es difícil **que dos personas, de forma independiente y autónoma, sigan exactamente los mismos pasos** para crear un mismo programa siguiendo una misma lógica y secuencialización, ¡Incluso llegando a la misma solución, el orden de razonamiento de ambas personas puede ser muy diferente!

La lógica de razonamiento de cada persona es diferente y esto hace que no haya una forma correcta de programar, sino tantas formas correctas como personas hayan dado con una de las posibles soluciones al problema. También es cierto que habrá soluciones que sean más simples que otras, o que funcionen mejor que otras. Y esto también es un aspecto a valorar y a tener en cuenta en cuanto a una posible evaluación de una estructura de programación.

Es posible que algún estudiante de este curso abordara la solución a este juego de formas muy diferentes, por ejemplo, empezando por los pasos finales de este curso y acabando por los pasos iniciales. Por tanto, si os veis con capacidades para ir construyendo el juego, podéis intentar construir vosotros mismos las soluciones. Si por contra os resulta difícil imaginar qué hacer en cada paso, os podéis dejar guiar por el razonamiento y secuencialización mostrado en este curso.

Es de esperar que al intentar realizar el siguiente proyecto de programación, cada vez seáis más autónomos. No hay que olvidar que para aprender a aplicar el pensamiento computacional para construir aplicaciones o juegos, hay que enfrentarse de forma individual ante nuevos retos, probar

diversas alternativas y acabar dando con la solución de forma autónoma, lo cual incrementará la satisfacción y entrarán ganas de afrontar un reto mayor de programación.

> Y de cara a ir construyendo el juego, cada vez que añadas unos bloques, ejecuta el programa para probar que hacen lo que era de esperar

Oferta de formación en Pensamiento computacional del Centro Aragonés de Tecnologías para la Educación.

https://view.genial.ly/5c546dc28805472c3451861a

> Tenemos un **grupo Telegram Robótica Educativa en Aragón**, si estás interesado en unirte, envía un mensaje por Telegram (obligatorio) a CATEDU 623197587
>
> https://t.me/catedu_es y te añadimos en el grupo

Robótica y accesibilidad

1.- Introducción

Durante mucho tiempo la robótica fue patrimonio de personas y/o instituciones con alta capacidad económica (podían adquirir las placas con microcontroladores comerciales) y capacidad intelectual (podían entender y programar el funcionamiento de las mismas) siempre dentro de los límites establecidos por las marcas comerciales y lo que pudieran “desvelar” de su funcionamiento, vigilando siempre que la competencia no “robara” sus secretos y “copiara” sus soluciones.

Todo esto saltó por los aires en torno a 2005 con la irrupción de un grupo de profesores y estudiantes jóvenes, que decidieron romper con esta dinámica, tratando de poner a disposición de su alumnado microcontroladores económicamente accesibles y que les permitieran conocer su funcionamiento, sus componentes, e incluso replicarlos y mejorarlos. Nacía **Arduino** y el concepto de **Hardware Open Source**. Detrás de este concepto se encuentra la **accesibilidad universal.** En un proyecto Open Source todo el mundo puede venir, ayudar y contribuir, minimizando barreras económicas e intelectuales.

Arduino traslada al hardware un concepto ya muy conocido en el ámbito del software, como es el **software open source o software libre.**

Software libre

Cuando los desarrolladores de software terminan su creación, tienen múltiples posibilidades de ponerlo a disposición de las personas, y lo hacen con condiciones específicas especificadas en una licencia. Esta licencia es un contrato entre el creador o propietario de un software y la persona que finalmente acabará utilizando este software. Como usuarios, es nuestro deber conocer las condiciones y permisos con las que el autor ha licenciado su producto, para conocer bajo qué condiciones podemos instalar y utilizar cada programa.

Existen muchas posibilidades de licencias: software privativo, comercial, freeware, shareware, etc.. Nos centraremos aquí en la de software libre.

GNU (https://www.gnu.org) es una organización sin ánimo de lucro que puso una primera definición disponible de lo que es software libre: Software libre significa que los usuarios del software tienen libertad (la cuestión no es el precio). Desarrollaron el sistema operativo GNU para que los usuarios pudiesen tener libertad en sus tareas informáticas. Para GNU, el software libre implica que los usuarios tienen las cuatro libertades esenciales:

1. ejecutar el programa.
2. estudiar y modificar el código fuente del programa.
3. redistribuir copias exactas.
4. distribuir versiones modificadas.

En otras palabras, el software libre es un tipo de software que se distribuye bajo una licencia que **permite a los usuarios utilizarlo, modificarlo y distribuirlo libremente**. Esto significa que los usuarios tienen libertad de ejecutar el software para cualquier propósito, de estudiar cómo funciona el software y de adaptarlo a sus necesidades, de distribuir copias del software a otros usuarios y de mejorar el software y liberar las mejoras al público.

El software libre se basa en el principio de la libertad de uso, y no en el principio de la propiedad. Esto significa que los usuarios tienen la libertad de utilizar el software de la manera que deseen, siempre y cuando no violen las condiciones de la licencia. El software libre es diferente del software propietario, que es el software que se distribuye con restricciones en su uso y modificación. El software propietario suele estar protegido por derechos de autor y solo se puede utilizar bajo los términos y condiciones especificados por el propietario del software.

Recomendamos la visualización de este video para entender mejor el concepto.

https://www.youtube.com/embed/nlDVZ816zoI

Más adelante, entorno a 2015, en Reino Unido, surgiría también la placa **BBC Micro:bit**, con la misma filosofía de popularizar y hacer accesible en este caso al alumnado de ese país la programación y la robótica. También hablaremos de ella.

2.- ARDUINO o LA ROBÓTICA ACCESIBLE

Arduino es una **plataforma de hardware y software libre**.

Hardware libre

Esto significa que tanto la placa Arduino como el entorno de desarrollo integrado (IDE) son de código abierto. Arduino permite a los usuarios utilizar, modificar y distribuir tanto el software como el hardware de manera libre y gratuita, siempre y cuando se respeten las condiciones de

las licencias correspondientes.

El hardware libre es un tipo de hardware cuya **documentación y diseño están disponibles de manera gratuita y libre** para su modificación y distribución. Esto permite a los usuarios entender cómo funciona el hardware y adaptarlo a sus necesidades, así como también crear sus propias versiones modificadas del hardware.

Arduino surge como solución al **elevado precio de los microcontroladores** allá por el año 2005. En el ámbito de la educación, los microcontroladores solo se utilizaban en la etapa universitaria, y su coste era tan elevado que muchos proyectos de fin de carrera se quedaban únicamente en prototipos virtuales ya que las universidades no podían proveer a cada estudiante con un microprocesador, contando además que en el propio proceso de experimentación lo más habitual era que una mala conexión hiciera que se rompieran. Otro **gran inconveniente era la dificultad de la programación**. Cada fabricante entregaba su manual de programación, lo que hacía que de unos a otros no hubiera un lenguaje estándar, y la consecuente dificultad de interpretación. Además, su programación era a bajo nivel en lenguaje máquina. Generar una simple PWM requería una ardua y minuciosa secuenciación que podía llevar varias horas hasta conseguir el resultado deseado. Por este motivo, el enfoque de Arduino desde el principio fue ser Open Source tanto en hardware como en software. El desarrollo del hardware fue la parte más sencilla. Orientado a educación, sufre algunas modificaciones frente a los microprocesadores existentes para hacer más fácil su manejo y accesibilidad a cualquier sensor o actuador. El mayor esfuerzo se entregó en todas las líneas de código que hacían posible que ya no hubiera que programar a bajo nivel gracias al IDE de Arduino que incluía bibliotecas y librerías que estandarizaban los procesos y hacían tremendamente sencillo su manejo. Ahora el alumnado para mover un motor, ya no tenía que modificar las tramas de bits del procesador una a una, sino que bastaba con decir que quería moverlo en tal dirección, a tal velocidad, o a equis grados.

Acabábamos de pasar de unos costes muy elevados y una programación muy compleja a tener una **placa accesible, open source y de bajo coste** que además hacía muy **accesible su programación y entendimiento**, características fundamentales para su implantación en educación, hasta tal punto que su uso ya no era exclusivo de universidades, sino que se extiende a la educación secundaria.

Este hecho es fundamental para el desarrollo del Pensamiento Computacional en el aula observándose que su accesibilidad y beneficios son tales, que alcanzan a **centros con alumnado de toda tipología** como la aplicación del pensamiento computacional y robótica en aulas con alumnos de necesidades especiales. Una vez más, aparece el concepto de accesibilidad asociado a esta filosofía Open Source.

A este respecto, recomendamos la lectura de este interesante blog, que tiene por título: ROBOTIQUEAMOS...” Experiencia de aproximación a la robótica en Educación Especial (CPEE ÁNGEL RIVIÈRE). También recomendamos los trabajos robótica en Educación Especial (CPEE ÁNGEL RIVIÈRE): http://zaragozacpeeangelriviere.blogspot.com/search/label/ROB%C3%93TICA

Igualmente, la aparición de Arduino supone una gran facilidad para la aplicación de la robótica y la programación en la atención temprana, donde son numerosas sus aplicaciones desde ayudar a mitigar el déficit de atención en jóvenes autistas, hasta ayudar a socializar a los alumnos con dificultades para ello, o ayudar a alumnos de altas capacidades a desarrollar sus ideas.

Por otro lado su accesibilidad económica lo ha llevado a popularizarse en países de **todo el mundo**, especialmente en aquellos cuyos sistemas educativos no disponen en muchas ocasiones de recursos suficientes, lo que supone en la práctica una **democratización del conocimiento y superación de brecha digital.**

Filosofía del Arduino ver vídeo

Arduino y su IDE son la primera solución que aparece en educación con todas las ventajas que hemos enumerado, y esto hace que todos los nuevos prototipados y semejantes tengan algo en común, siempre son compatibles con Arduino

Para entender bien la filosofía de Arduino y el hardware libre, os recomendamos este documental de 30 minutos. Arduino the Documentary

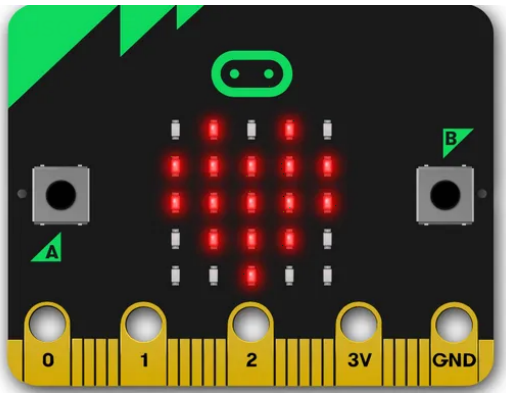
https://player.vimeo.com/video/18390711?h=b5844e7753

Scratch: software libre para el desarrollo del pensamiento computacional

Scratch es un lenguaje de programación visual desarrollado por el grupo Lifelong Kindergarten del MIT Media Lab. Scratch es un software libre. Esto significa que está disponible gratuitamente para todos y que se distribuye bajo una licencia de software libre, la Licencia Pública General de Massachusetts (MIT License). Esta licencia permite a los usuarios utilizar, modificar y distribuir el software de manera libre, siempre y cuando se respeten ciertas condiciones. Entre otras cosas, la licencia de Scratch permite a los usuarios utilizar el software para cualquier propósito, incluyendo fines comerciales. También permite modificar el software y distribuir las modificaciones, siempre y cuando se incluya una copia de la licencia y se indique que el software ha sido modificado. En resumen, Scratch es un software libre que permite a los usuarios utilizar, modificar y distribuir el software de manera libre y gratuita, siempre y cuando se respeten las condiciones de la licencia. De hecho, gracias a que está licenciado de esta forma, han surgido decenas de variaciones de Scratch para todo tipos de propósitos, eso sí, siempre educativos y relacionados con las enseñanzas de programación y robótica

3. BBC micro:bit y la Teoría del Cambio

BBC micro:bit, a veces escrito como Microbit o Micro Bit, es un pequeño ordenador del tamaño de media tarjeta de crédito, creado en 2015 por la BBC con el fin de promover el desarrollo de la robótica y el pensamiento computacional entre la población escolar del Reino Unido. Actualmente s es de escolares de 7 a 16 años de más de 60 países.



Tarjeta BBC micro:bit V1. Fuente: https://microbit.org. CC BY-SA 4.0.

Aunque el proyecto fue iniciado por la BBC, su desarrollo fue llevado a cabo por 29 socios tecnológicos de primera línea. Por ejemplo, la implementación del Bluetooth integrado en la tarjeta corrió a cargo de la fundación propietaria de la marca, Bluetooth SIG, una asociación privada sin ánimo de lucro.

El hardware y el software resultantes son 100% abiertos, y están gestionados por una fundación sin ánimo de lucro que comenzó a funcionar en el año 2016, la Micro:bit Educational Foundation. La fundación basa sus actuaciones en su Teoría del Cambio,

Teoría del cambio y más sobre microbit

Teoría del cambio puede resumirse en tres principios:

- El convencimiento de que la capacidad de comprender, participar y trabajar en el mundo digital es de vital importancia para las oportunidades de vida de una persona joven.
- La necesidad de emocionar y atraer a las personas jóvenes por medio de BBC micro:bit, especialmente a las que podrían pensar que la tecnología no es para ellas.
- Diversificar a los estudiantes que eligen las materias STEM a medida que avanzan en la escuela y en sus carreras, para hacer crecer una fuente diversa de talento, impulsando la equidad social y contribuyendo a crear una tecnología mejor.

Para desarrollar sus principios, la fundación trabaja en tres líneas de acción:

- El desarrollo de hardware y software que contribuyan a despertar el entusiasmo en las personas jóvenes hacia la tecnología y hacia las oportunidades que presenta.
- La creación de recursos educativos gratuitos y fáciles de usar que permitan al profesorado enseñar de forma atractiva y creativa.
- La colaboración con entidades asociadas que compartan una misma visión para ofrecer programas educativos de alto impacto en todo el mundo.

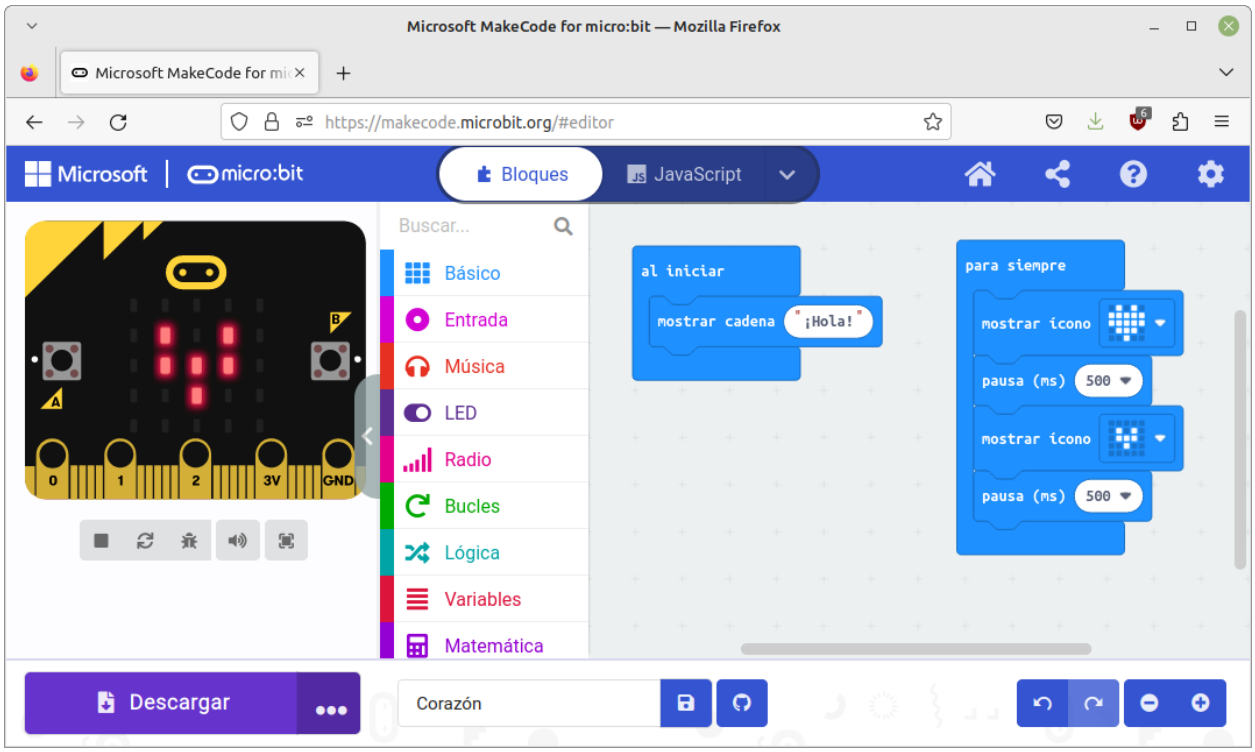
> Uno de los objetivos de la Micro:bit Educational Foundation es llegar a 100 millones de escolares en todo el mundo.

En correspondencia con las líneas de acción y con los principios expuestos, el sistema resultante es muy económico: tanto las placas como los accesorios producidos por terceras empresas tienen un precio muy contenido. Además, dado el carácter abierto del proyecto, están disponibles algunos clones totalmente compatibles, como Elecrow Mbits o bpi:bit. Estos clones son incluso más potentes y económicos que la placa original.

El universo micro:bit destaca por su **alta integración de software y hardware**: basta un clic de ratón para cargar las librerías necesarias para que funcione cualquier complemento robótico, como sensores, pantallas, tarjetas de Internet de las Cosas, robots, casas domóticas, etc.

La programación de la placa se realiza desde un ordenador a través de un navegador cualquiera, estando disponibles **12 lenguajes de programación**. De nuevo, por ser un sistema abierto, existen múltiples soluciones de programación, aunque las más común es

MakeCode.



Captura de pantalla del editor MakeCode, https://makecode.microbit.org/#.

El sitio web MakeCode permite programar con bloques y también en Python y en Java, traduciendo de un lenguaje a otro instantáneamente. No se necesita ningún registro en la plataforma para poder programar.

Los programas también pueden guardarse descargados en el ordenador compilados en código de máquina. Al subir de nuevo el programa al editor, se realiza una decompilación automática al lenguaje de bloques, Python o Java. Los programas guardados en código de máquina se pueden cargar directamente en micro:bit, que en el escritorio de un ordenador se maneja como una simple unidad de memoria USB.

MakeCode contiene además múltiples recursos como tutoriales, vídeos, fichas de programación, cursos para el profesorado, ejemplos y propuestas de proyectos y experimentos, todo ello en varios idiomas y clasificado por edades desde los 7 años.

Otra solución muy usada para programar micro:bit es MicroPython, creada por Python Software Foundation, otra organización sin ánimo de lucro.

MicroCode permite que los más pequeños, a partir de los 6 años de edad, programen micro:bit mediante un sistema de fichas dispuestas en líneas de acción. Están disponibles un tutorial introductorio en 20 idiomas, una guía del usuario y muchos ejemplos. El proyecto es de código abierto.

Micro:bit también es programable en **Scratch** con sólo añadir una extensión al editor.

Todos los entornos de desarrollo descritos disponen de un simulador de micro:bit, por lo que ni siquiera resulta necesario disponer de una tarjeta física para aprender a programar.

Una vez realizada la programación, la placa y sus complementos pueden funcionar desconectados del ordenador por medio de un cargador de móvil, una batería externa o un simple par de pilas alcalinas.

Versiones y características de micro:bit

A pesar de su pequeño tamaño, micro:bit es un sistema potente. Existen dos versiones de la placa. La más moderna, llamada micro:bit V2, tiene las siguientes características:

- Procesador de 64 MHz.
- 512 KB de RAM Flash y 128 KB de RAM.
- Matriz de 5 x 5 LED rojos.
- Dos pulsadores mecánicos y un tercer pulsador de apagado y reset.
- Un pulsador táctil.
- Micrófono y altavoz.
- Acelerómetro y brújula.
- Sensores de luz y de temperatura.
- Comunicación con otras placas por Bluetooth de bajo consumo.
- Alimentación a 3 V o por USB.
- 25 pines de entradas y salidas para conectar motorcitos, sensores, placas de Intenet de las Cosas, robots y, en general, cualquier otro tipo de accesorio.
- 200 mA de intensidad de corriente disponibles en las salidas para alimentar accesorios.

4.- LA IMPORTANCIA DEL OPEN SOURCE / CÓDIGO ABIERTO EN EDUCACIÓN

La creación, distribución, modificación y redistribución del hardware y software libre así como su utilización, están asociados a una serie de valores que deberían ser explicados en la escuela a nuestros alumnos para dar una alternativa a la versión mercantilista de que cualquier creación es creada para obtener beneficios económicos.

En GNU, pusieron especial énfasis en la difusión del software libre en colegios y universidades, promoviendo una serie de valores fundacionales:

Valores GNU

Compartir

El código fuente y los métodos del hardware y software libre son parte del conocimiento humano. Al contrario, el hardware software privativo es conocimiento secreto y restringido. El código abierto no es simplemente un asunto técnico, es un asunto ético, social y político. Es una cuestión de derechos humanos que la personas usuarias deben tener. La libertad y la cooperación son valores esenciales del código abierto. El sistema GNU pone en práctica estos valores y el principio del compartir, pues compartir es bueno y útil para el progreso de la humanidad. Las escuelas deben enseñar el valor de compartir dando ejemplo. El hardware y software libre favorece la educación pues permite compartir conocimientos y herramientas.

Responsabilidad social

La informática, electrónica, robótica... han pasado a ser una parte esencial de la vida diaria. La tecnología digital está transformando la sociedad muy rápidamente y las escuelas ejercen una influencia decisiva en el futuro de la sociedad. Su misión es preparar al alumnado para que participen en una sociedad digital libre, mediante la enseñanza de habilidades que les permitan tomar el control de sus propias vidas con facilidad. El hardware y el software no debería estar bajo el poder de un desarrollador que toma decisiones unilaterales que nadie más puede cambiar.

Independencia

Las escuelas tienen la responsabilidad ética de enseñar la fortaleza, no la dependencia de un único producto o de una poderosa empresa en particular. Además, al elegir hardware y software libre, la misma escuela gana independencia de cualquier interés comercial y evita permanecer cautiva de un único proveedor. Las licencias de hardware y software libre no expiran

Aprendizaje

Con el open source los estudiantes tienen la libertad de examinar cómo funcionan los dispositivos y programas y aprender cómo adaptarlos si fuera necesario. Con el software libre se aprende también la ética del desarrollo de software y la práctica profesional.

Ahorro

Esta es una ventaja obvia que percibirán inmediatamente muchos administradores de instituciones educativas, pero se trata de un beneficio marginal. El punto principal de este aspecto es que, por estar autorizadas a distribuir copias de los programas a bajo costo o gratuitamente, las escuelas pueden realmente ayudar a las familias que se encuentran en dificultad económica, con lo cual promueven la equidad y la igualdad de oportunidades de aprendizaje entre los estudiantes, y contribuyen de forma decisiva a ser una escuela inclusiva.

Calidad

Estable, seguro y fácilmente instalable, el software libre ofrece una amplia gama de soluciones para la educación.

Para saber más

En los años 90, era realmente complicado utilizar un sistema operativo Linux y la mayoría de la cuota del mercado de los ordenadores personales estaba dominada por Windows. Encontrar drivers de Linux para el hardware que tenía tu equipo era casi una quimera dado que las principales compañías de hardware y de software no se molestaban en crear software para este sistema operativo, puesto que alimentaba la independencia de los usuarios con respecto a ellas mismas.

Afortunadamente, y gracias a la creciente presión de su comunidad de usuarios, estas situaciones pertenecen al pasado, y las compañías fabricantes de hardware han tenido que variar el rumbo. Hoy en día tenemos una gran cantidad de argumentos en los que nos podemos basar para dar el salto hacia cualquier sistema operativo basado en Linux. Tal y como podemos leer en educacionit.com, podemos encontrar las siguientes ventajas:

- Es seguro y respeta la privacidad de los usuarios: Aunque hay compañías linuxeras, como Oracle, Novell, Canonical, Red Hat o SUSE, el grueso de distribuciones y software Linux está mantenido por usuarios y colectivos sin ánimo de lucro. De esta forma, podemos confiar en que una comunidad que tiene detrás millones de usuarios, pueda validar el código fuente de cualquier de estas distribuciones, asegurándonos la calidad de las mismas, compartir posibles problemas de seguridad, y sobre todo, estar bien tranquilos con la privacidad y seguridad de nuestros datos e información personal, aspecto que debería ser crítico y determinante a la hora de trabajar con los datos de menores de edad en las escuelas y colegios.
- Es ético y socialmente responsable: La naturaleza de Linux y su filosofía de código abierto y libre hace posible que cualquier usuario con conocimientos pueda crear su propia distribución basada en otras o probar las decenas de versiones que nos podemos encontrar de una distribución Linux. Este es el caso de Ubuntu por ejemplo. Gracias a esta democratización de los sistemas operativos, incluso han podido aparecer en nuestras vidas nuevos dispositivos basados en software y hardware libre como Arduino y Raspberry Pi.
- Es personalizable: el código abierto permite su estudio, modificación y adaptación a las necesidades de los diferentes usuarios, teniendo así no un único producto sino una multiplicidad de distribuciones que satisfacen las necesidades de los diferentes colectivos a los que se dirijan. Especialmente útiles son las distribuciones educativas libres, que pueden ser adaptadas a las necesidades de las escuelas.

- Está basado en las necesidades de los usuarios y no en las de los creadores de hardware y software
- Es gratis. La mayoría de las distribuciones Linux son gratuitas y de libre descarga
- Es fácil de usar. Una de las barreras que durante años ha evitado a muchos usar Linux es su complejidad. Las distribuciones orientadas al consumo doméstico cumplen los estándares de simplicidad y necesidades que cualquier usuario sin conocimientos de tecnología pueda necesitar. El entorno gráfico es sencillo, intuitivo, e incluso se puede customizar para que se pueda parecer a los más conocidos como Windows y MacOS. Además, vienen con la mayoría de aplicaciones que cualquier usuario puede necesitar: ofimáticas, edición de audio y vídeo y navegación por Internet.
- Es suficiente. Tiene su propio market de aplicaciones. Como el resto de sistemas operativos ya sea para ordenadores o dispositivos móviles, también podemos encontrar un lugar único donde poder descargar cientos de aplicaciones para todos los gustos y necesidades.

Por estas razones, el software libre se ha expandido por toda la comunidad educativa en los últimos años de manera exponencial. Un buen ejemplo de lo que estamos hablando es **Bookstack**, este sistema de edición de contenidos para cursos que utiliza Aularagón así como el uso de **Moodle** como plataforma de enseñanza y aprendizaje. En cuanto a sistema operativo para ordenadores, en Aragón disponemos de nuestra propia distribución Linux: Vitalinux EDU. Tal y como podemos leer desde su página web: **Vitalinux EDU (DGA)** es la distribución Linux elegida por el Gobierno de Aragón para los centros educativos. Está basada en Vitalinux, que se define como un proyecto para llevar el Software Libre a personas y organizaciones facilitando al máximo su instalación, uso y mantenimiento. En concreto Vitalinux EDU (DGA) es una distribución Ubuntu (Lubuntu) personalizada para Educación, "tuneada" por los requisitos y necesidades de los propios usuarios de los centros y adaptada de forma personalizada a cada centro y a la que se ha añadido una aplicación cliente Migasfree. De ésta forma, obtenemos:

1. Un **Sistema Ligero.** Permite "revivir" equipos obsoletos y "volar" en equipos modernos. Esto garantiza la sostenibilidad de un sistema que no consume recursos de hardware innecesariamente ni obliga a la sustitución del hardware cada poco tiempo en esa espiral de obsolescencia programada en la que se ha convertido el mercado tecnológico.
2. **Facilidad en la instalación y el uso** del sistema mediante programas personalizados.
3. Un Sistema que **se adapta al centro** y/o a cada aula o espacio, y no un centro que se adapta a un Sistema Operativo.
4. **Gestión de equipo y del software de manera remota** y desatendida mediante un servidor Migasfree.
5. **Inventario** de todo el hardware y software del equipo de una forma muy cómoda.
6. Soporte y apoyo de una **comunidad** que crea, comparte e innova constantemente.

¿Qué herramientas tenemos?

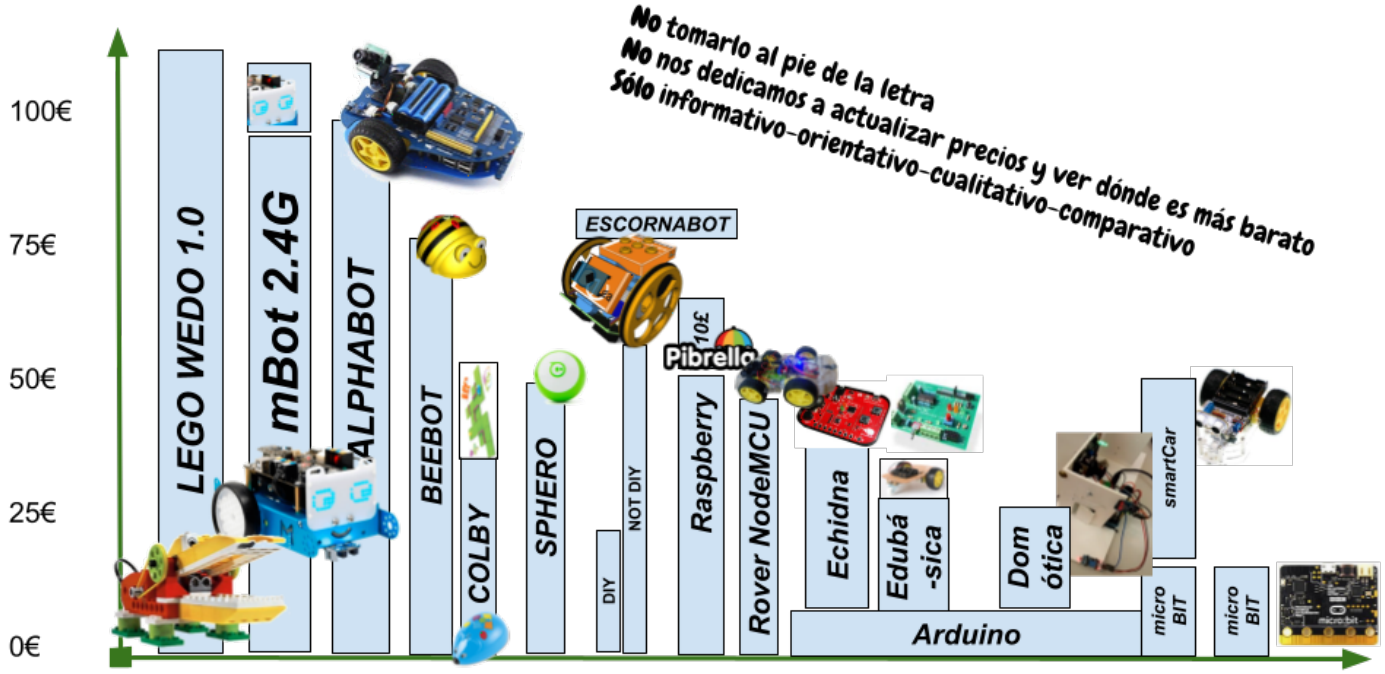
- App Inventor: http://appinventor.mit.edu/explore/
- Code: https://code.org/
- Blockly: https://developers.google.com/blockly/
- https://blockly-demo.appspot.com/static/demos/index.html
- https://blockly-games.appspot.com/?lang=es

Propuesta de pensamiento computacional

Hay muchas herramientas, pero no todas son válidas ni a cualquier edad. En CATEDU hemos elaborado esta hoja de ruta seleccionando algunas y clasificándolo por edades (no se tiene que tomar al pie de la letra, sólo a nivel orientativo) para trabajar el pensamiento computacional, podríamos elegir otros modelos de robots, pero estos son los que trabajamos en CATEDU. **Puedes ver dónde encaja Scratch**:

CENTRO ARAGONÉS de TECNOLOGÍAS para la EDUCACIÓN
CATEDU

Coste aproximado de los packs de robótica



Como ves SCRATCH JUEGA UN PAPEL MUY IMPORTANTE en toda la primaria y secundaria, y **CODE.ORG** es una herramienta muy buena como base si no se quiere empezar de lleno en Scratch

También vemos que SCRATCH es un lenguaje apropiado para la robótica.

Instalación

Dos formas de trabajar en SCRATCH

La web oficial de Scratch es http://scratch.mit.edu/

La versión actual de Scratch es Scratch 3.0.

Hay dos caminos para crear proyectos con Scratch :

1. **Sin necesidad de conexión a Internet: mediante Scratch Offline Editor**
   - Es el editor de proyectos Scratch que no necesita conexión a Internet para funcionar.
   - Recomendada en caso de existir problemas de conexión.
   - Los proyectos Scratch se guardan en archivos con extensión .sb3 dentro de nuestro ordenador.
   - Lamentablemente en este momento desde la página oficial solo se suministra esta versión para sistemas operativos Windows.
   - Web oficial de descarga de Scratch
   - Existen disponibles en esa misma web las versiones offline anteriores (Scratch 1.4 y Scratch 2) que son necesarias para el caso de querer abrir ficheros de extensión .sb y .sb2
   - Para el caso de equipos con Linux, existe la opción de descargar siempre la última versión actualizada de Scratch desde el proyecto **Scratux**.
   - Web oficial de descarga de Scratux
   - Para los equipos vitalinux se encuentra disponible en Vitalinux Play.
2. **Con necesidad de conexión a Internet: mediante** Scratch 3 Online
   - **Se realiza desde la página oficial http://scratch.mit.edu/ en CREAR. (Acceso directo: https://scratch.mit.edu/projects/editor/)**
   - Los proyectos Scratch se guardan en la Web de Scratch http://scratch.mit.edu/
   - Para guardar los proyectos realizados en la Web, es necesario crearse una cuenta de usuario en la web de Scratch .
   - Los proyectos Scratch se pueden exportar y acabar guardados en nuestro ordenador.
   - Permite compartir nuestro proyecto a la comunidad de "Scratchers".
   - Permite "embeber" el proyecto en una página web, proporcionando el código html.

¿Qué opción es la más recomendable?

Lo mejor es la versión OnLine, sólo recomendamos utilizar la versión Offline si hay problemas de conexión de Internet.

On-Line

Scratch on-line es la opción que recomendamos al menos que en tu centro tenga problemas de conexión de Internet.

Registro como docente

Si te registras como docente puedes crear como alumno, y además crearte tus clases y controlar a tus alumnos. Si te registras como alumno no puedes cambiar como docente luego.

Aquí tienes un tutorial para registrarte como docente, si ese es tu caso.

> ATENCIÓN si quieres más información sobre las opciones como docente, entonces consulta esta página

Registro como alumnado

Aquí tienes un tutorial para registrarte como alumnado, si ese es tu caso.

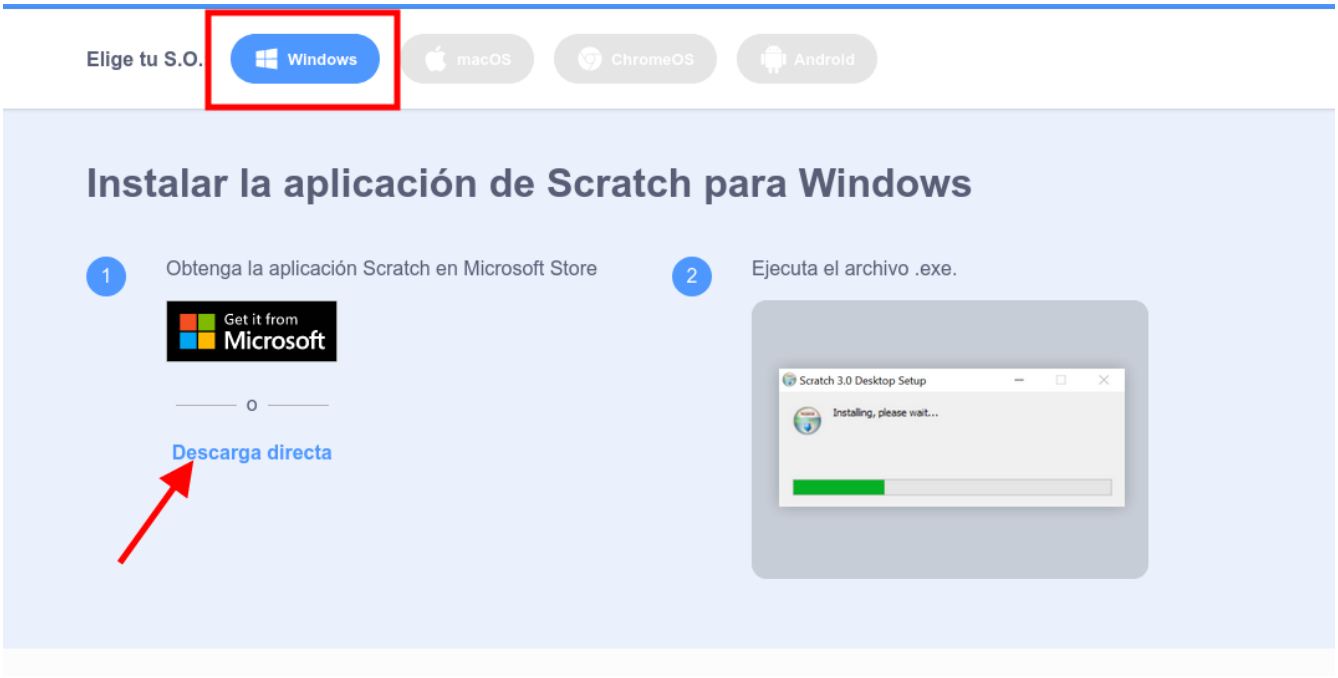
También existe una versión para Android llamada ScratchJr [Ver]. Es una versión reducida pensada para alumnos de infantil.

Off-Line

Actividad: Instalar Scratch 3 Offline Editor

Si tienes Windows:

Entra en la web de descargas de Scratch https://scratch.mit.edu/download y descarga desde allí el ejecutable .exe

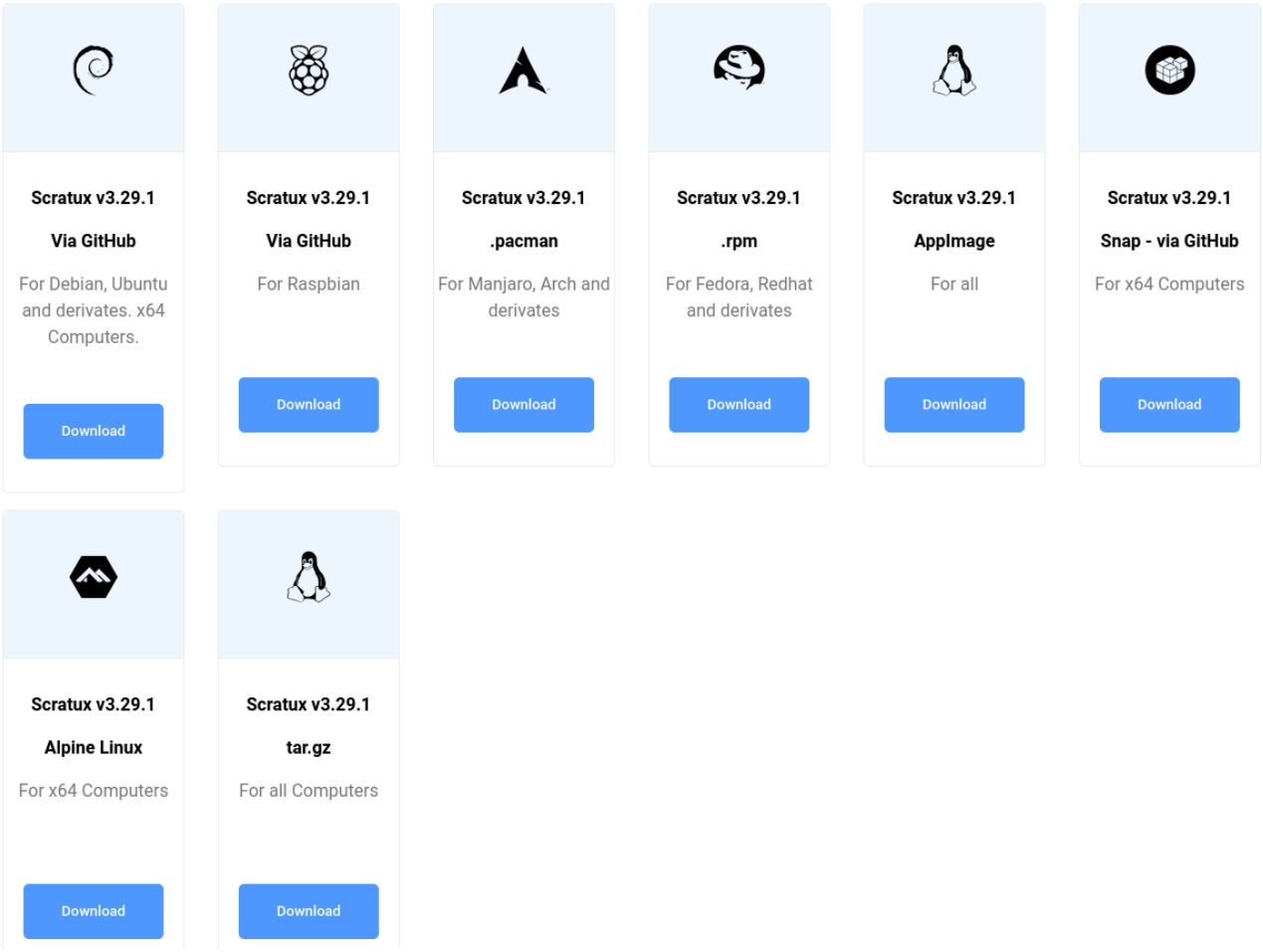


Si tienes vitalinux:

Ve al Vitalinux Play y descárgate desde allí Scratux: Scratch3 offline

Si tienes linux pero no vitalinux:

Ve a la web oficial de Scratux y descárgate tu opción



Sobre las versiones anteriores Scratch2 y Scratch 1.4:

- Los proyectos de Scratch 1.4 se guardan en archivos con extensión .sb
- Los proyectos de Scratch 2 se guardan en archivos con extensión .sb2
- Un proyecto realizado en Scratch 2.0 (extensión .sb2) puede convertirse en 1.4 (extensión .sb) usando Retro Converter pero algunos módulos propios de la versión 2.0 no se convertirán. Ver página con información.
- Un fichero .sb se puede convertir en un ejecutable exe de Windows o App en Mac o en un JAR Java ejecutable Ver página.

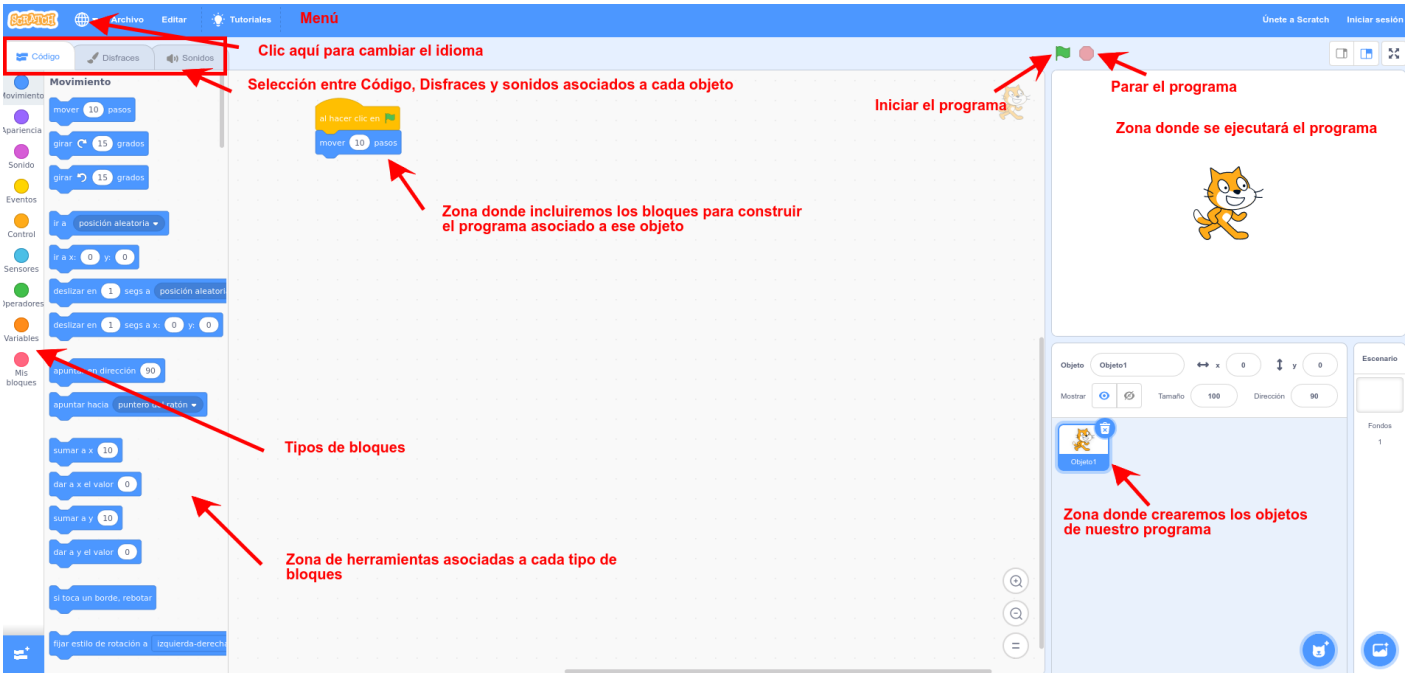
Entorno Scratch

Una vez que hayas arrancado Scratch en tu ordenador, o que hayas entrado en la web online de Scratch, verás el entorno de trabajo de Scratch. En la siguiente imagen se muestran las diferentes zonas de trabajo de Scratch.

En el primer ítem del menú con forma de bola del mundo se puede cambiar el idioma a español.

Glosario:

- **SPRITE:** Es el objeto sobre el cual programamos. En la figura de abajo es el gato, pero podemos incorporar más objetos predefinidos de la biblioteca de Scratch, pintarlos con mapa de bits o vectorial, cargarlos desde archivo desde nuestro ordenador o incluso desde cámara web.
- **EVENTO:** Suceso provocados por el usuario, los sprite o por el sistema, y sobre el cual queremos que el programa se ejecute o haga un comportamiento determinado, por eso en Scratch se dice que su programación es "orientada a eventos". Ejemplo de eventos provocados por ...
  - **Sprite**: si el gato toca el borde, pues que haga...
  - **Usuario:** Si se pulsa la tecla espacio que ...
  - **Sistema**: Si hora actual = 12 que ....



Respecto a la **accesibilidad** de esta herramienta para **alumnado con Necesidades Educativas Especiales**, recomendamos consultar el siguiente artículo de la Universidad Complutense de

Madrid, o esta presentación realizada por personal de ONCE para el caso específico de alumnado con deficiencia visual.

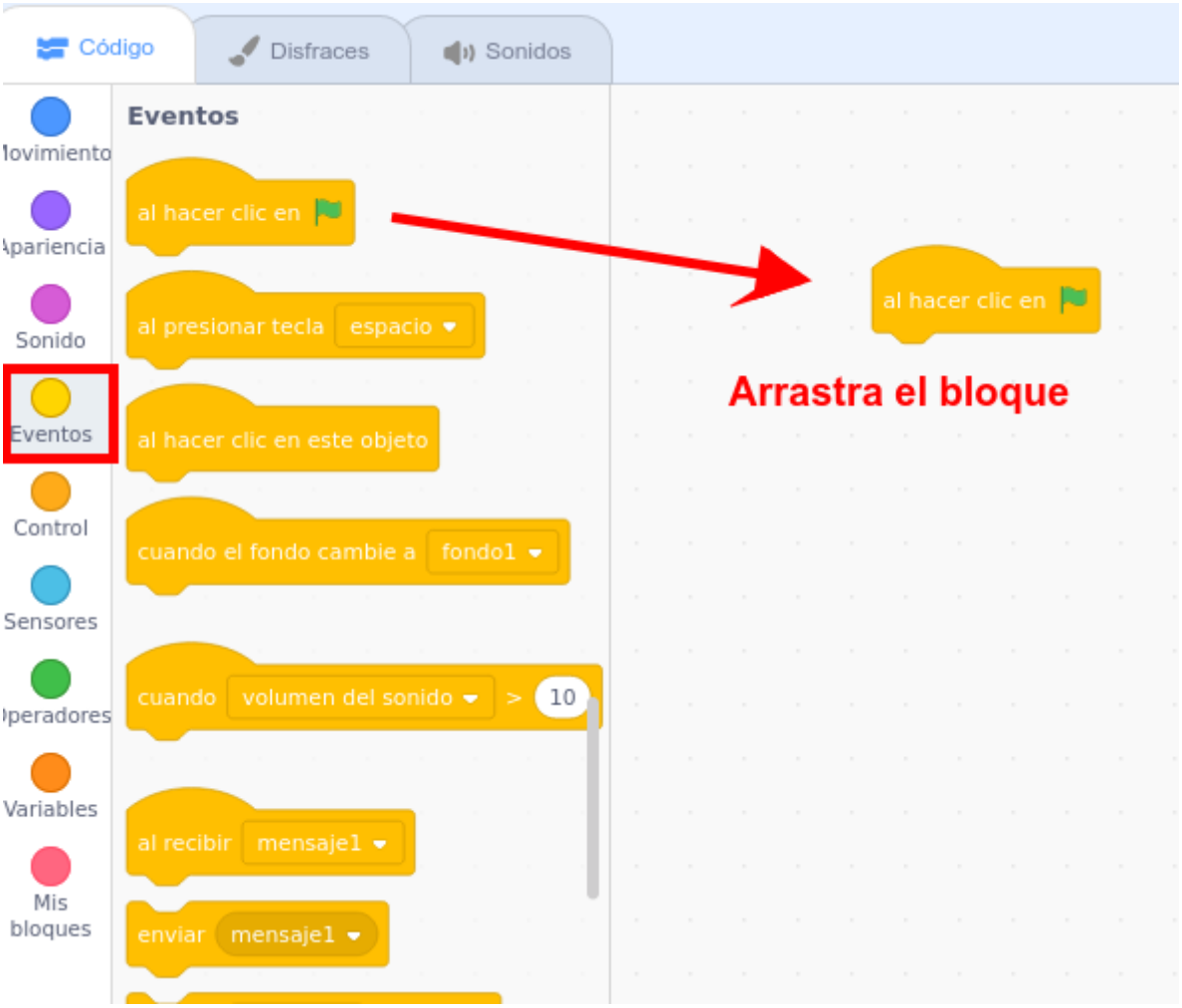
¡Ya funciona!

Actividad:

Vamos a hacer nuestro primer programa en Scratch:

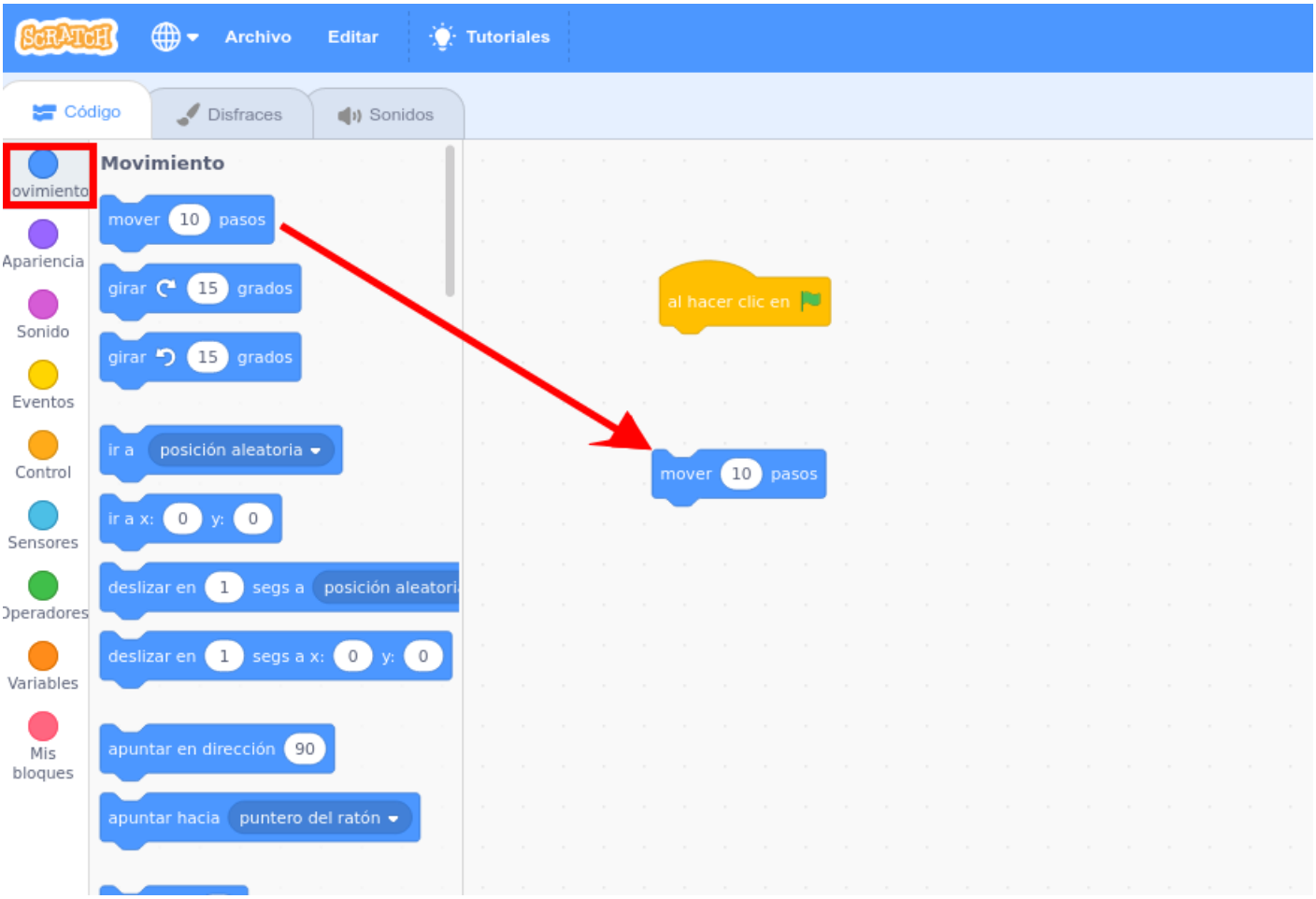
1. Haz clic en la pestaña Código, en el menú **Eventos**

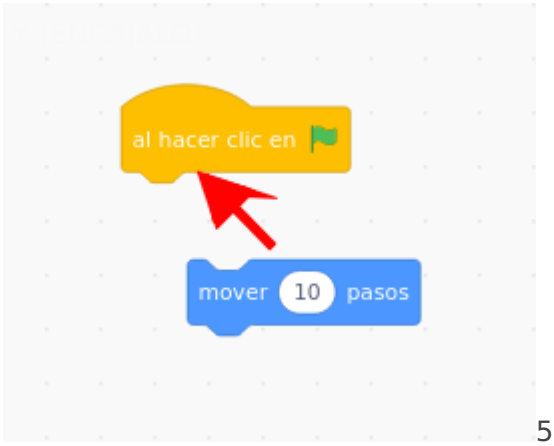
2. Arrastra el bloque "Al hacer clic en bandera":



3. Haz clic ahora en el menú **Movimientos**.

4. Arrastra el bloque "**mover 10 pasos**" a la zona derecha y encájalo arrastrándolo justo debajo del bloque "Al hacer clic en bandera":





5 Haz clic en la bandera verde para iniciar el programa:

¡Verás que el gato Scratch se mueve un poco hacia la derecha! Se mueve 10 pasos, correspondientes a 10 píxeles.

El programa termina sólo porque ha terminado de ejecutar todas las instrucciones. Por ello no hay que darle al botón Stop rojo (al lado de la bandera verde) para que finalice. El botón Stop servirá cuando nuestro programa tenga bloques de control que impliquen repetición continua de ejecución (esto se verá más adelante).