Proyecto SEMELI

TIC y Jóvenes

andròmines
Creant oportunitats

SEMELI
Socially excluded youth of Media literacy

Primeros pasos....

- Rellenar la Documentación del Proyecto SEMELI

¿Qué es la robótica?

La robótica es la rama de la ingeniería mecánica, de la ingeniería electrónica y de las ciencias de la computación que se ocupa del diseño, construcción, operación, estructura, manufactura y aplicación de los robots.

La robótica combina diferentes disciplinas: la mecánica, la electrónica, la informática, la ingeniería de control, la física, la inteligencia artificial, etc.

El término “robot” se popularizó con el éxito de la obra titulada R.U.R (Robots Universales Rossum) escrita por Karel Capek en el año 1920. Cuando esta obra fue traducida al inglés, la palabra checa *robota*, que significa trabajos forzados o trabajador, fue traducida como robot.

Clasificación: según su cronología

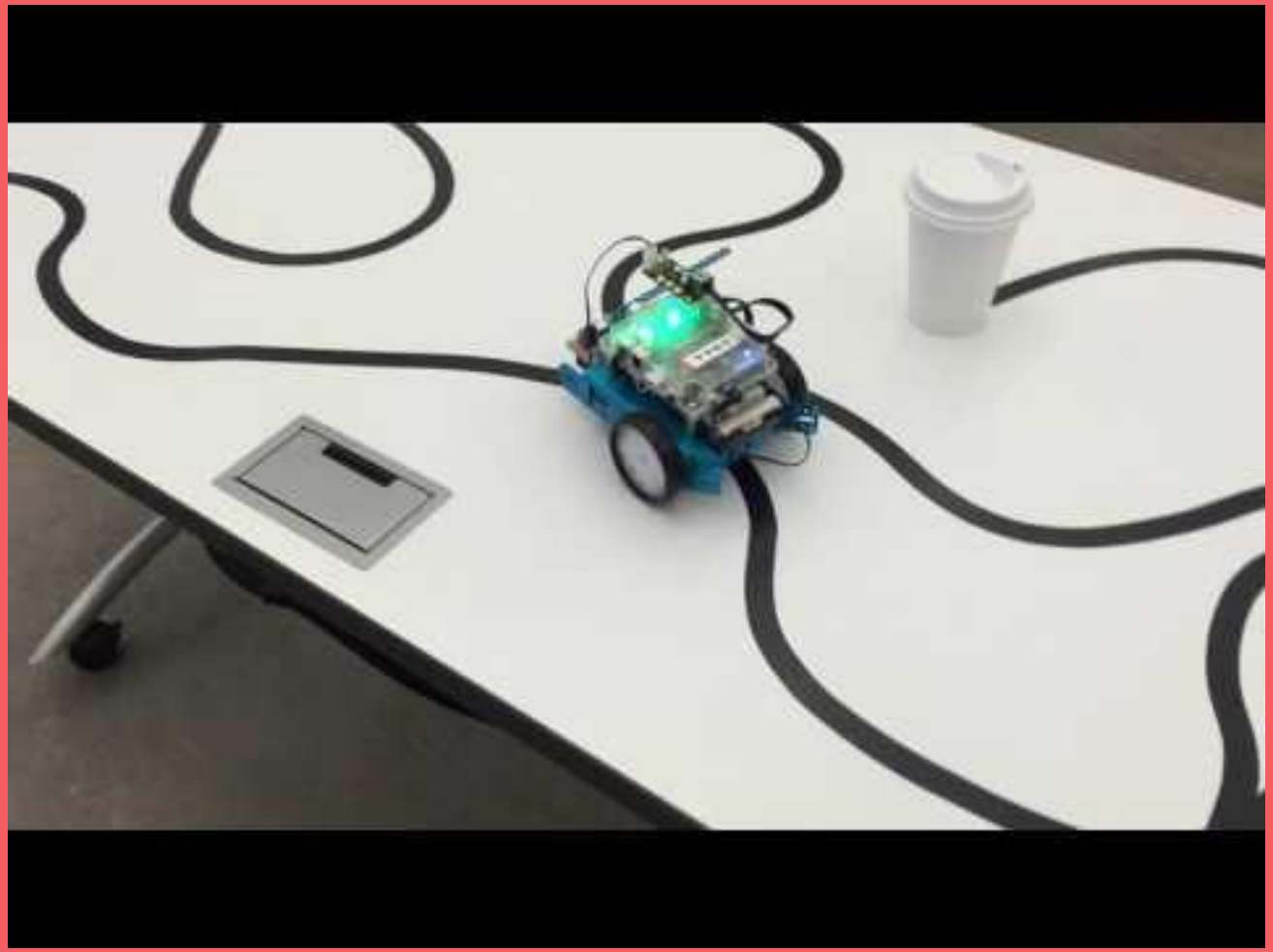
1a generación: Robots manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control: manual, de secuencia fija o de secuencia variable.

2n generación: robots de aprendizaje. Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

3r generación: robots con control sensorizado. El controlador es un ordenador que ejecuta las órdenes de un programa y las envía al manipulador o robot para que realice los movimientos necesarios.

4t generación: robots inteligentes. Estos están dotados de cierta inteligencia que poseen sensores mucho más sofisticados capaces de enviar la información recopilada a sus computadoras internas, procesarla en tiempo real mediante procesos complejos, y actuar en consecuencia.


ATLAS
ATLAS




ROBOTS JUEGAN INCLUSO
MEJOR QUE LOS HUMANOS
RT

Clasificación: según su estructura

- **Poliarticulados:** robots de diversas formas y configuración. Su característica en común es la de ser sedentarios o efectuar desplazamientos limitados. Están estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas, y con un número limitado de grados de libertad. En este grupo se encuentran los robots manipuladores, los robots industriales y los robots cartesianos.
- **Móviles:** tienen gran capacidad de desplazamiento, basado en carros o plataformas y dotados de un sistema locomotor rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación.

Clasificación: según su estructura

- **Androides:** son los robots que intentan reproducir, de forma parcial o total, la forma y el comportamiento del ser humano. en la actualidad los androides son dispositivos poco evolucionados.
- **Zoomórficos:** sus sistemas de locomoción imitando a los seres vivos. se pueden dividir en dos categorías principales: caminadores y no caminadores. El grupo de robots zoomórficos no caminadores están poco evolucionados. Los experimentos están basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. Los robots zoomórficos caminadores multípedos son muy numerosos. El desarrollo de estos robots llevará a vehículos terrenos, pilotados o autónomos, capaces de evolucionar en superficies muy accidentadas. La aplicación de estos robots será interesante en el campo de la exploración espacial y en el estudio de los volcanes.
- **Híbridos:** estos corresponden a aquellos de difícil clasificación. su estructura se sitúa en combinación con alguna de las anteriores ya expuestas, ya sea por conjunción o por yuxtaposición.


RIVUS

Instalación de Puglins para trabajar con UBUNTU

-Primer paso descargar mlink

https://mblock.makeblock.com/en-us/download/mlink/

-Segundo paso instalar mlink

sudo dpkg -i mLink-1.2.0-amd64.deb

(ejecutar esta linea de codigo en la carpeta donde esta descargado el archivo)

-Tercer paso iniciar el servicio mlink

sudo mblock-mlink start

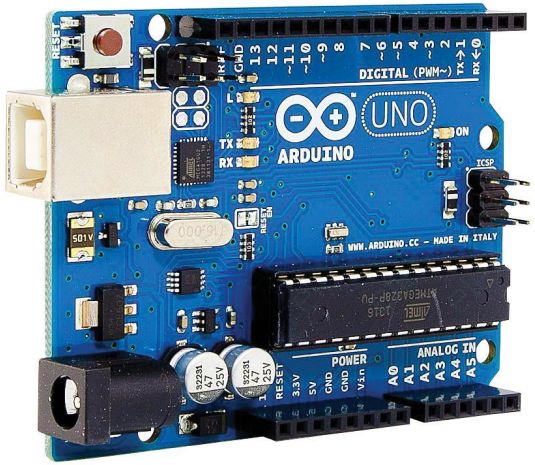
se tiene que iniciar el servicio cada vez que se enciende el ordenador

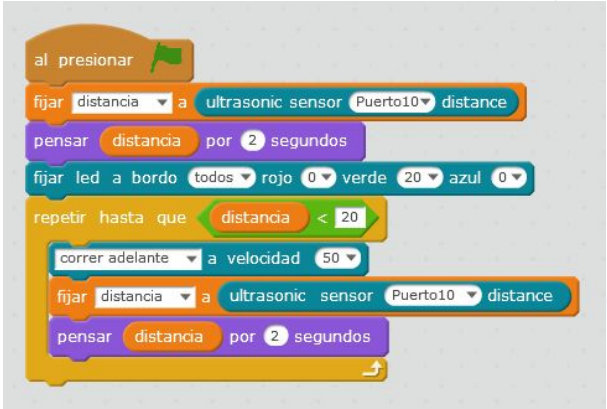
se puede finalizar el servicio con el siguiente comando: **sudo mblock-mlink stop**

¿Qué robotica vamos a trabajar?

En este curso trabajaremos con tres herramientas:

- Mbot
- Arduino
- Lenguaje informático.





andròmines
Creant oportunitats

SEMELI

MBot

MBot es un robot educativo de la empresa Makeblock, que persigue los siguientes objetivos:

- El objetivo principal es desarrollar el pensamiento computacional
- Basado en hardware libre y software libre

El hardware libre se basa en la placa ARDUINO, que lo han personalizado con más sensores y conexiones rápidas. El software libre en este robot está en el programa mBlock que está basado en el software de programación Scracth.

Componentes de un Mb

Componentes exteriores importantes a la hora de programar:

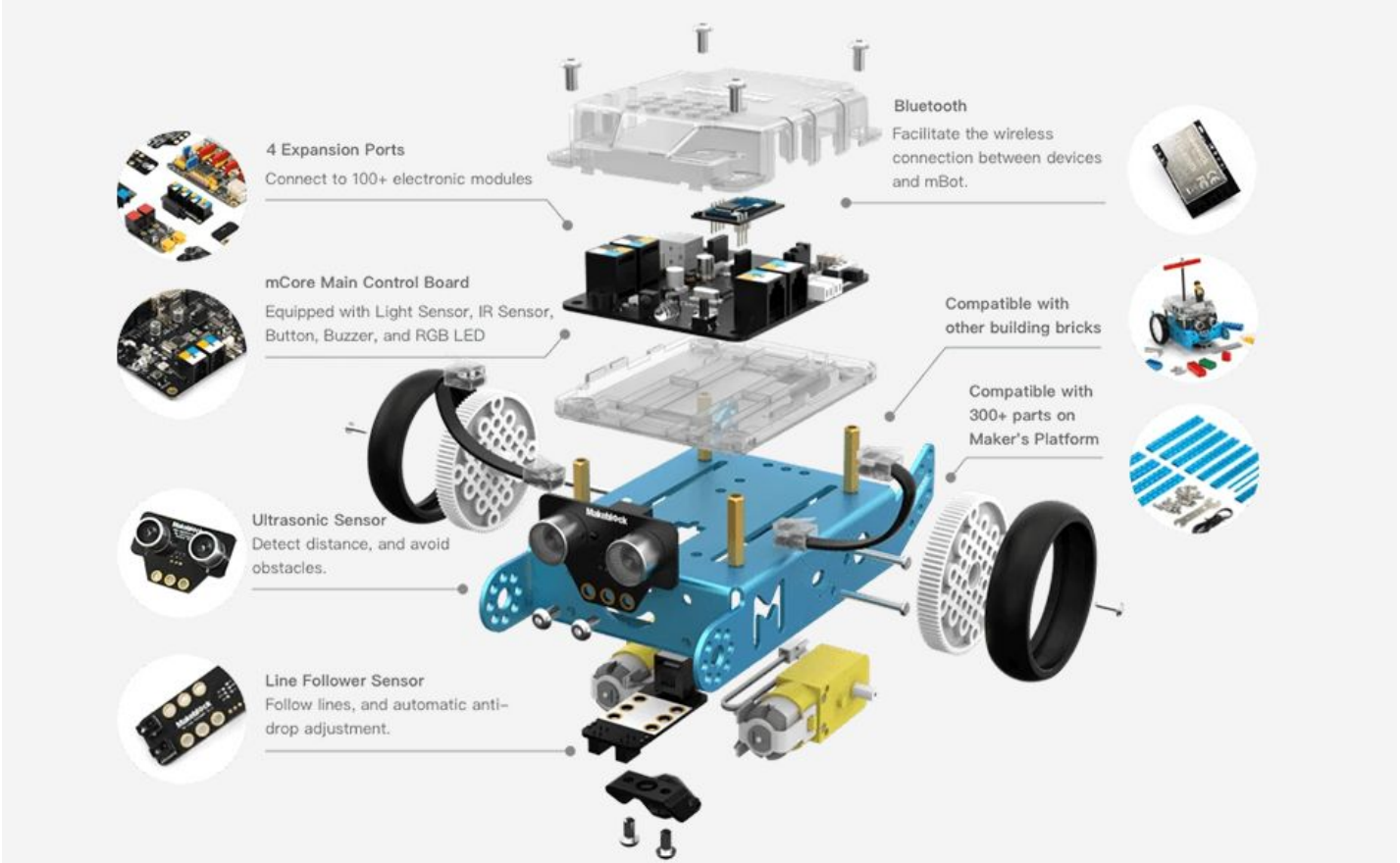
- Sensor de Línea para utilizarlo por ejemplo como sigue líneas.
- Sensor de distancia por ultrasonidos para utilizarlo por ejemplo como evita-obstáculos.
- El sensor de sonido está destinado a detectar la intensidad de sonido de su entorno. Puedes utilizarlo para realizar proyectos que tengan que interactuar con la voz.

Sensor de línea

Sensor ultrasonido

Sensor de sonido

Más componentes y ubicación



andròmines
Creant oportunitats

SEMELI
Socially Excluded youth of Media Literacy

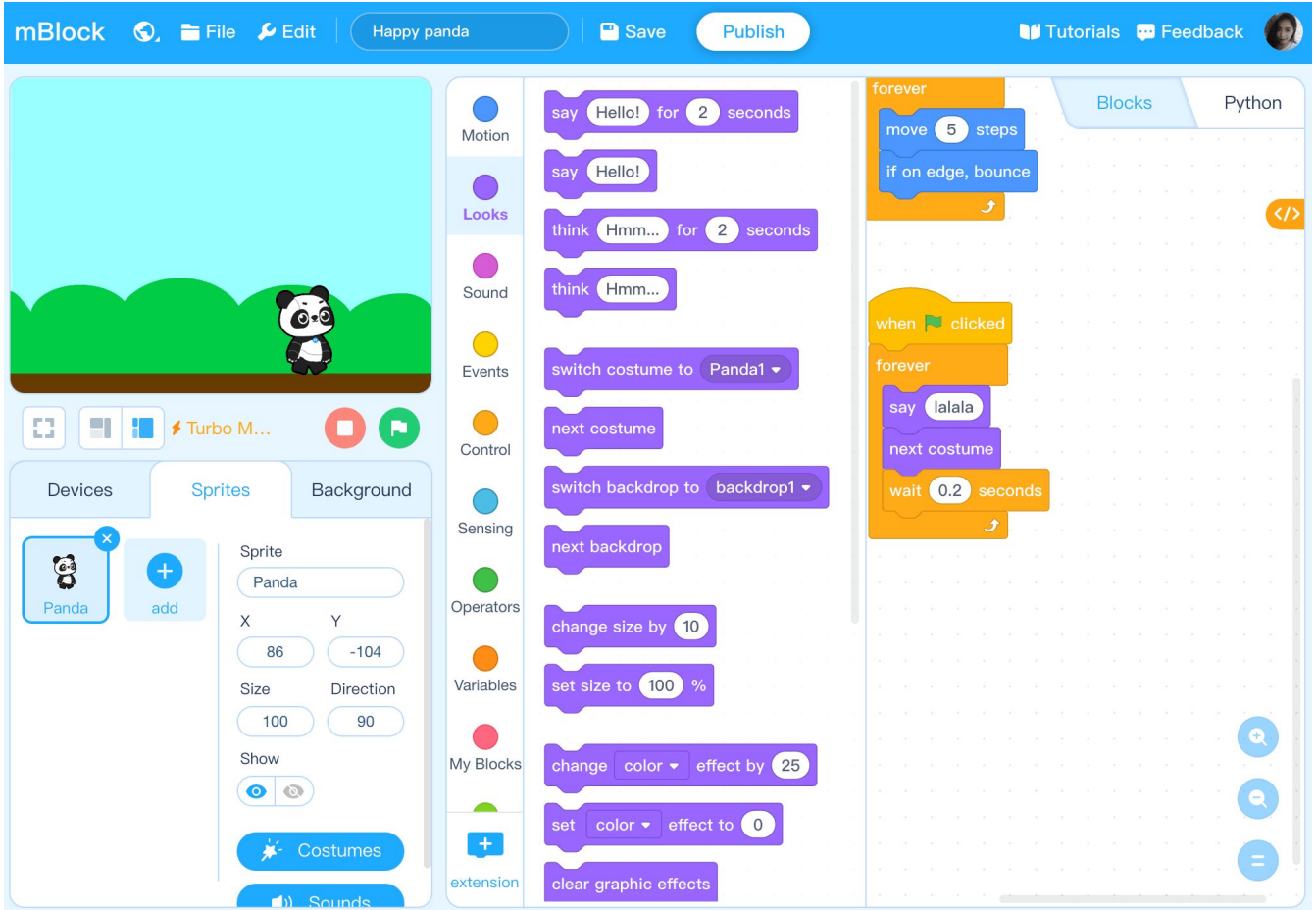
¿Cómo vamos a programar el MBot?

Utilizaremos la programación por bloques. En la actualidad hay muchas plataformas donde practicar la programación por bloques.

Utilizaremos el programa Mblock: su versión online (mBlock - One-Stop Coding Platform for Teaching and Learning) o su versión descargada.

Mblock es un entorno gráfico de programación basado en el conocido editor Scratch.

Esta programación está basada en bloques. Cada bloque tiene una orden diferente y acabará formando una línea de comandas.



andròmines
Creant oportunitats

SEMELI

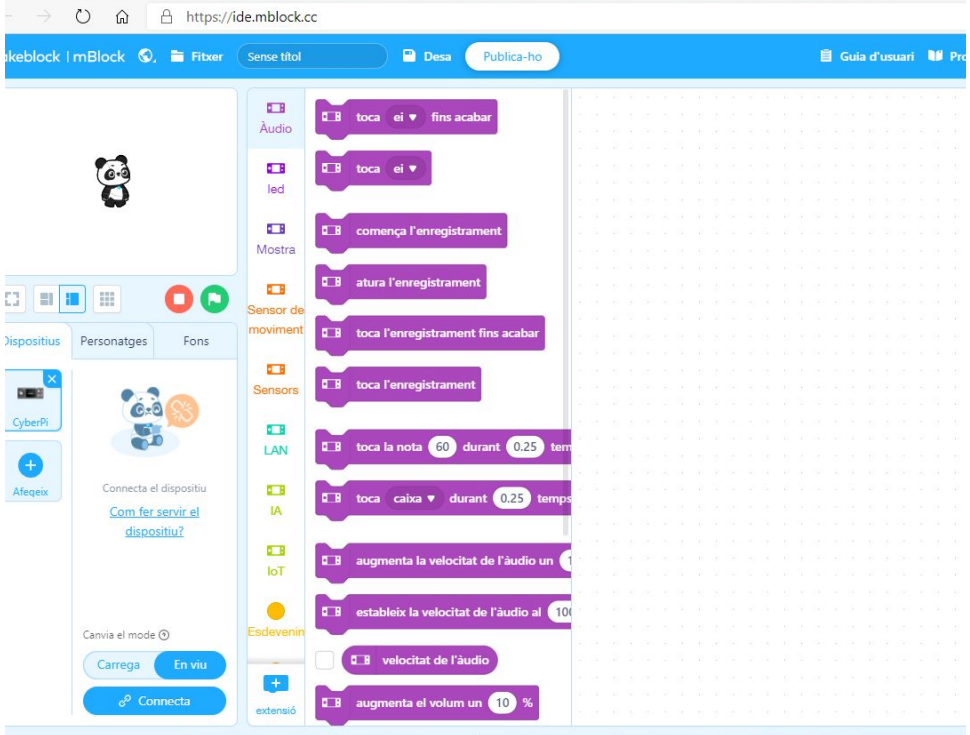
Interface de Mblock

Rojo: Escenario, donde observamos el resultado de nuestro programa cuando es en tiempo real.

Verde: Es el conjunto de instrucciones, pulsando en cada una de ellas se despliega el catálogo de instrucciones.

Azul: Es lugar donde estructuramos nuestro programa que vamos realizando en bloques de la zona verde.

Amarillo: Personajes, fondos, dispositivos, que van interviniendo en nuestro programa.



andròmines
Creant oportunitats

SEMELI

Ejercicio: ¡Identifica!

- Abre Mblock en tu ordenador e identifica las diferentes áreas de trabajo de la interfaz
- Presiona las diferentes áreas de trabajo y experimenta con ellas.

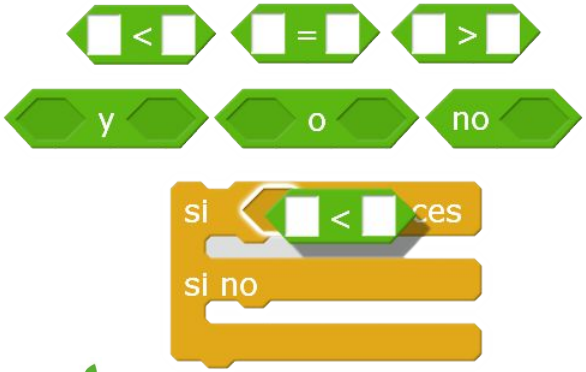
Tipología de bloques

Bloques de inicio: Es un bloque que inicia el script y siempre se coloca encima de otros bloques. Cada bloque se activa mediante un método específico, de modo que se pueden iniciar distintos scripts en diferentes momentos.

Bloques de acciones: Es un bloque que ejecutan los comandos principales y constituyen la mayoría de todos los bloques de programación. Es un bloque rectangular que puede caber encima o debajo, excepto en los bloques de inicio, donde solo se pueden colocar debajo

Tipología de Bloques

Bloques de valores: Es un bloque que contiene un valor, ya sea números o cadenas. Pueden caber donde sea que se necesite un valor, entre sí mismo, pero no pueden estar solos.



Bloques booleanos: Es un bloque que contiene una condición, que puede ser “verdadera” o “falsa”, es un hexágono que encajan en las ranuras hexagonales correspondientes en otros bloques, por lo que tampoco pueden estar solos.

Tipología de Bloques

Bloques de condicionales: Es un bloque que aplica un condicionante o finaliza un script o proyecto. Se coloca de forma que engloba a los otros bloques.

Ejercicio: Mis primeros bloques

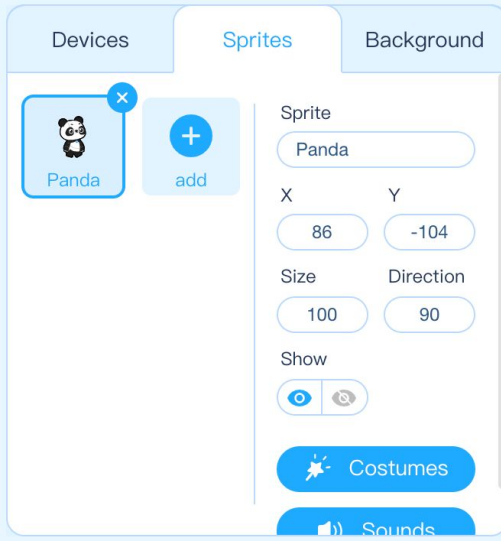
Realiza tus primeras programaciones:

- Cambia el fondo con un intervalo de 1 segundos entre fondo y fondo
- Añade un bucle

Solución


cuando clic en
para siempre
cambia al fondo Bedroom1
espera 1 segundos
cambia al fondo Bedroom3
espera 1 segundos

Como conectar el Mbot al Mblock



+add : En esta opción elegiremos el dispositivo que queremos conectar: Arduino, Mblock, etc.

+extensión: En esta opción elegiremos los bloques necesarios para el dispositivo que acabamos de conectar.

andròmines
Creant oportunitats

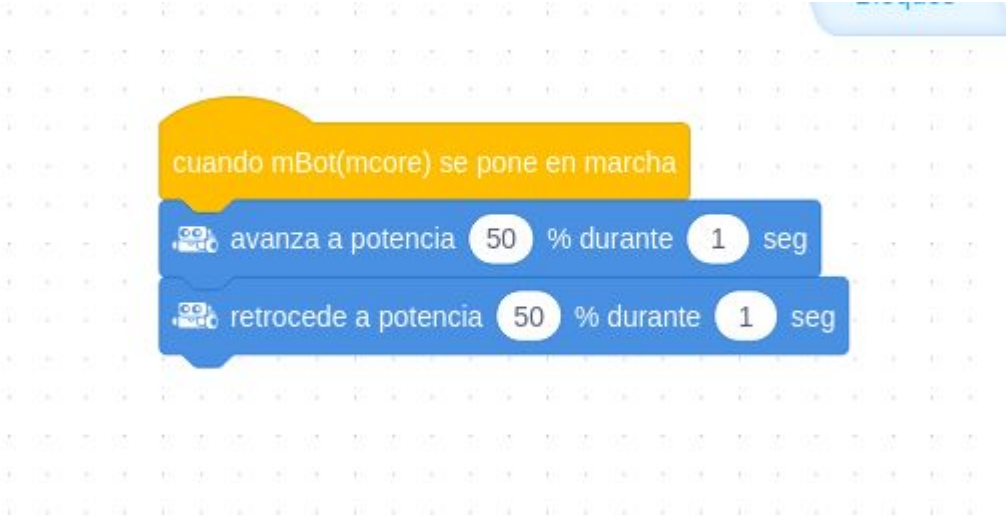
SEMELI

Programación 1: movimientos básicos

- Haz que el Mbot vaya hacia delante y hacia atrás

Siempre cargaremos la configuración en el Mbot. No trabajamos nunca en vivo.

Solución

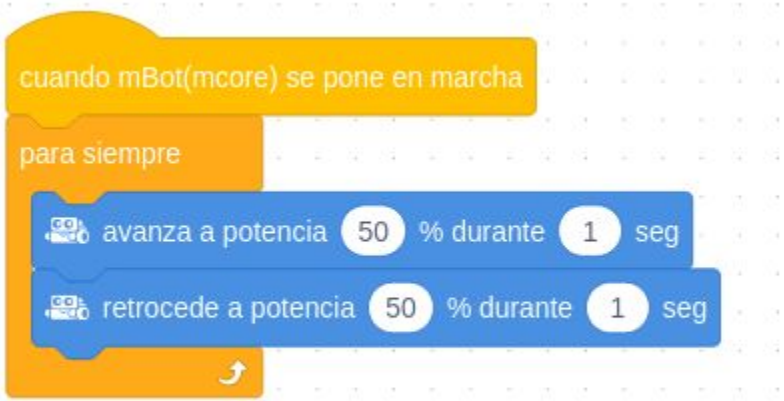

cuando mBot(mcore) se pone en marcha
avanza a potencia 50 % durante 1 seg
retrocede a potencia 50 % durante 1 seg

Modificaciones de programación

Realiza las siguientes modificaciones en la configuración:

- Haz que el Mbot realiza el mismo movimiento (hacia delante y hacia atrás) en buckle
- Haz que el Mbot siga el siguiente comando:
  - alante, pausa, atras, paus en buckle
- Modifica las potencias y los segundo de los movimientos

Solución



Configuración con el movimiento en Buckle

Solución

Configuración con el movimiento en buckle y pausas

Programación 2: movimientos con giros

- Realiza las siguientes secuencias de órdenes:
  - Alante + giro a la derecha (90 grados)
  - Alante + giro a la izquierda (90 grados)
  - Alante + giro de 360 grados
  - Atrás + giro de 120 grados
  - Atrás + giro (90 grados) + atrás + giro de 360 grados

Solución

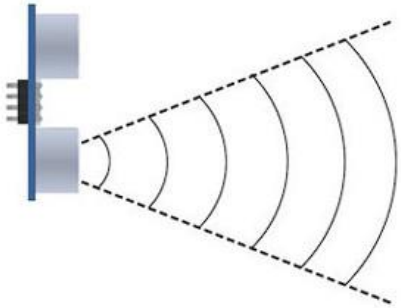
es un ejemplo de todos los que hay que hacer

Programación 3: sensor de ultrasonido

El sensor de ultrasonidos o distancia son detectores de proximidad que detectan objetos a distancias que van desde pocos centímetros hasta varios metros. El sensor emite un sonido y mide el tiempo que la señal tarda en regresar. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración.

Programación 3: sensor de ultrasonido

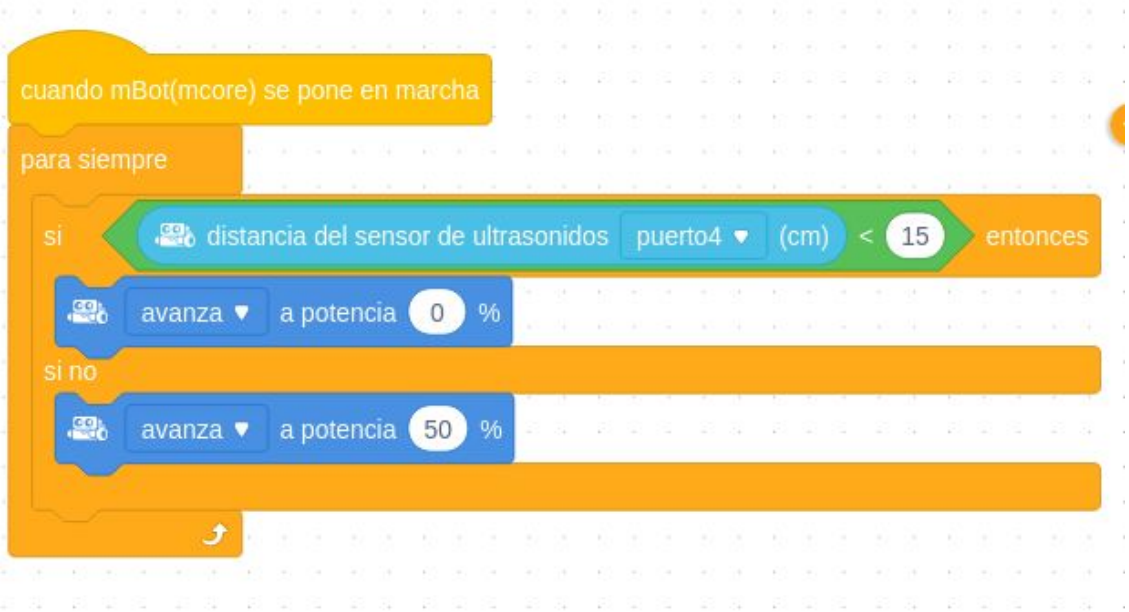
El funcionamiento del sensor es muy sencillo; se utiliza un emisor para emitir una onda de sonido de alta frecuencia (ultrasonido) y un receptor para escuchar la onda reflejada. En mBlock solamente tenemos que utilizar el bloque que nos devuelve la longitud a la cual se encuentra de un obstáculo. A partir de esa distancia podemos actuar según nuestro objetivo.



Programación 3: sensor de ultrasonido

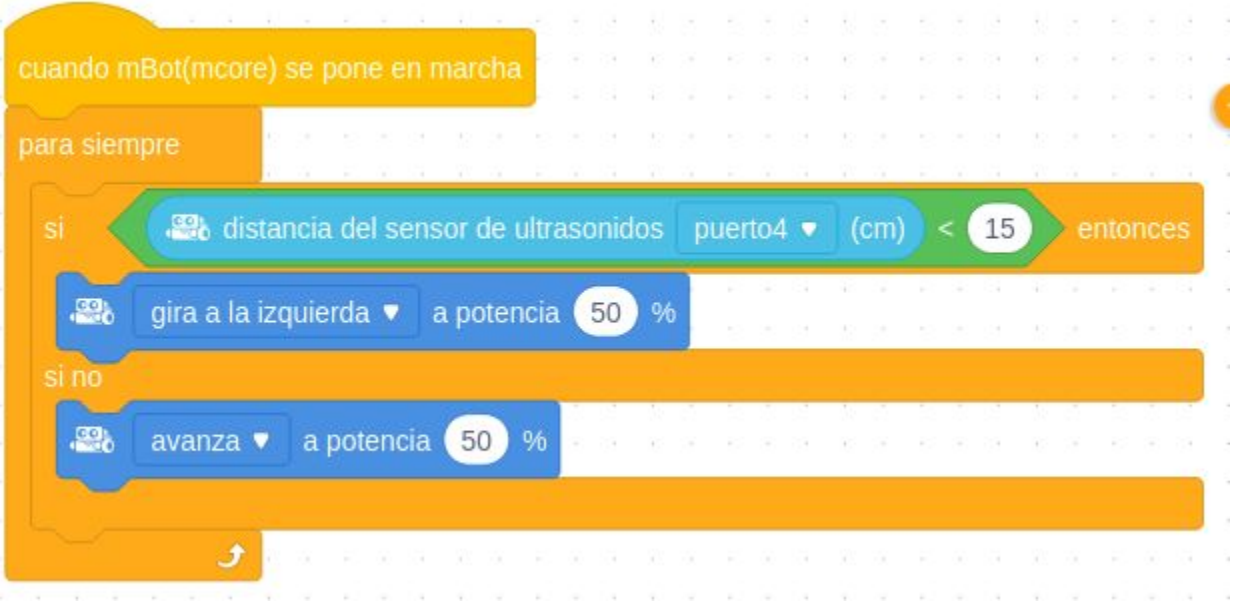
- Haz que cuando se encuentre con un obstáculo, el Mbot se pare
- Haz que cuando se encuentre con un obstáculo, el Mbot gire y pare
- Cuando se encuentre un obstáculo: gire y continúe su marcha

Solución



Avanza y se para cuando se encuentra un obstáculo

Solución



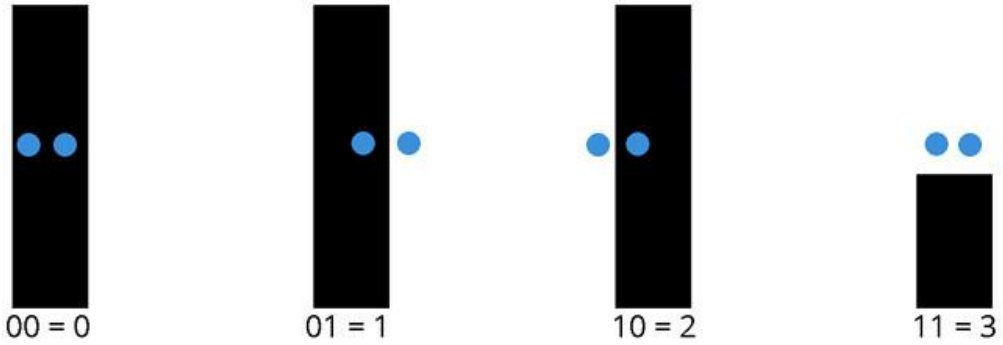
Avanza y gira cuando se encuentra un obstáculo

Programación 4: siguelineas

El sigue-líneas o sensor de infrarrojos (IR) es un sensor de medición de distancia que se basa en un sistema de emisión y recepción de radiación lumínica en el espectro de los infrarrojos. Dicho de otro modo, es un sensor con una fuente de luz (diodo emisor) y detector (fototransistor) integrados en un mismo encapsulado. La detección del objeto se consigue por la reflexión (o no) del haz infrarrojo sobre la superficie del objeto. Este sensor se utiliza habitualmente para detectar el color de un objeto, en nuestro caso, distinguir blanco y negro en aplicaciones para el seguimiento de línea.

Programación 4: siguelineas

El funcionamiento del sensor es el siguiente; si el sensor del robot mBot está sobre la línea negra, los reflejos del sensor son absorbidos por la misma con lo cual el receptor del sensor no recibe ningún valor, o lo que es lo mismo, dando el valor 0. Del mismo modo, si uno de los dos sensores del robot queda fuera de la línea, se producirá el reflejo indicando los valores 1 o 2 respectivamente. Por último, si ambos sensores quedan fuera de la línea se mostrará el valor 3.



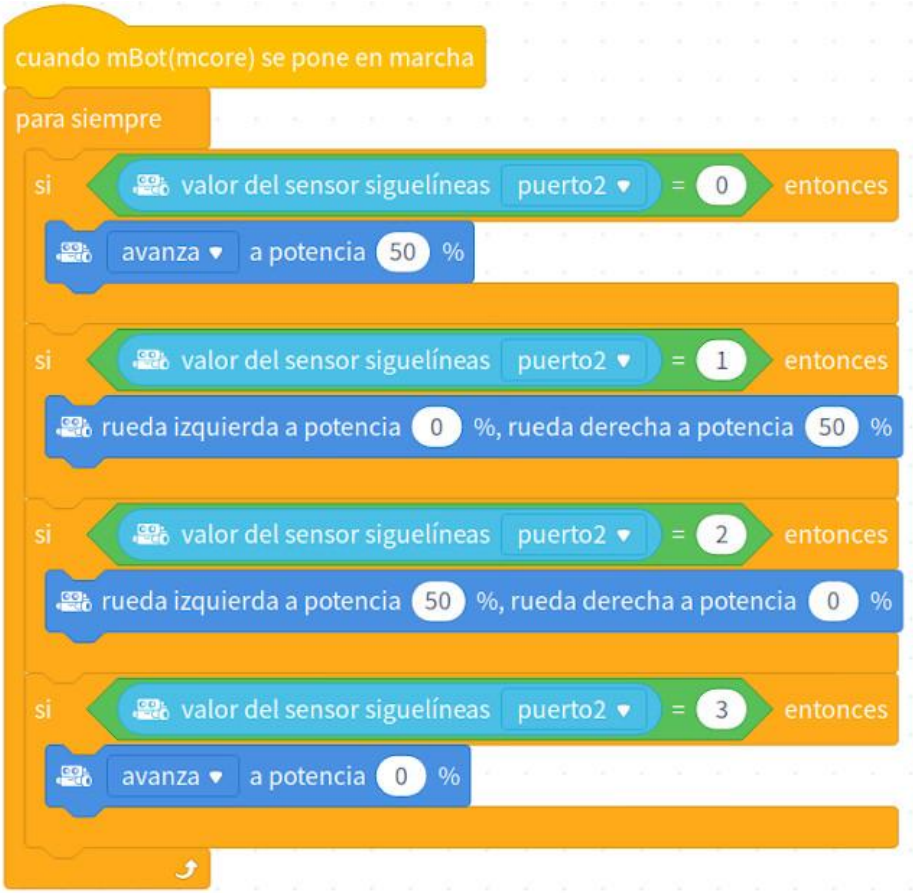
Programación 4: siguelineas

- Valor 0: mBot tiene los dos detectores IR sobre la línea.
- Valor 1: mBot tiene el detector IR de la derecha fuera de la línea.
- Valor 2: mBot tiene el detector IR de la izquierda fuera de la línea.
- Valor 3: mBot tiene ambos detectores IR fuera de la línea.

Programación 4: siguelineas

- Haz que el Mbot siga las líneas marcadas en el mapa
- Haz que el Mbot siga las líneas marcadas en el mapa y que se pare si hay un obstáculo
- Haz que el Mbot siga las líneas marcadas en el mapa y esquive los obstáculos.

Solución: sigue las líneas marcadas



¡Sube de nivel!
Intenta modificar las velocidades de tu Mbot según esté en una zona del circuito o en otra.

andròmines
Creant oportunitats

SEMELI

Programación 4: siguelineas

- Haz que el Mbot siga las líneas marcadas en el mapa y que se pare si hay un obstáculo
- Haz que el Mbot siga las líneas marcadas en el mapa y esquive los obstáculos.

El juego del rey:

- Programa el Mbot en secreto. Los demás deberán de adivinar la programación que has hecho.

Otros retos

- Programación de LEDs
  - Todos los leds
  - Solo los del lado derecho - Solo los del lado izquierdo
  - Cambia el color de los leds
- Programación del panel de LEDs
  - Haz que aparezcan diferentes Caras en la pantalla LEd

Reto Final:

- Realiza la programación necesaria para que el MBot realice el siguiente circuito:
  - Debe realizar el circuito siguiendo las líneas
  - Si se encuentra un obstáculo debe pararse o esquivarlo
  - LEDs
    - En las curvas debe haber un colo
    - En las rectas otro color
    - Cuando se para otro color
  - Panel de LEd
    - Cuando va recto debe salir un dibujo
    - Cuando hace las curvas otro
  - Sensor de sonido
    - Cuando el Mbot se para, debe emitir un sonido

Arduino

Filosofía Maker

Arduino es una plataforma de Hardware y Software libre que va muy ligada a la filosofía maker

¿Que es esta filosofía?

La filosofía Maker aboga por el aprendizaje en acción. Se inspira en el DIY (Do It Yourself): Una invitación a personas que no son profesionales o expertas en una técnica a la construcción de sus propios dispositivos e ideas, y en el DIWO (Do IT With Others): hacerlo en comunidad.

Pero la filosofía maker no es una introducción de la tecnología en el aprendizaje, sino que El desarrollo de la mentalidad maker desplaza por completo el foco de la experiencia educativa al estudiante. Es una metodología activa, de instrucción no directa y centrada en el aprendiz, que se enmarca dentro de la teoría constructivista. Es decir, el estudiante es el protagonista de su propio aprendizaje, que sucede derivado su acción (learning by doing). El alumno aprende porque hace.

La labor del docente en la cultura maker es facilitar un espacio de exploración, más que el acompañamiento estructurado del proceso de enseñanza. Permite tener en consideración, potenciar y enriquecer el Entorno Personal de Aprendizaje del estudiante

¿Qué aporta la Filosofía Maker?

- El aprendizaje maker contribuye a trabajar las denominadas habilidades del siglo XXI: creatividad, colaboración, pensamiento crítico, iniciativa, etc. (Taylor, 2016).
- Propicia el aprendizaje cooperativo, en el que los estudiantes aprenden de la interacción con otros compañeros y adultos (Montanero Fernández 2019). En el movimiento maker esto se produce a dos escalas: la primera se refiere al entorno directo con sus compañeros de proyecto, profesores, etc. Y la segunda con la comunidad maker, cuya filosofía se basa en el conocimiento libre y el código abierto. Los estudiantes entonces aprenden cooperando con su entorno y también enseñando a sus compañeros y a su comunidad (Gartner, 1971).
- Potencia el pensamiento crítico. La tecnología suele ser una 'black box', un complicado dispositivo que sabemos usar, pero no cómo funciona realmente (Resnick, Berg, Einsemberg 2000). La cultura maker invita a construir nuestra propia tecnología, creando ciudadanos que no son meros consumidores (Taylor, 2016).
- Facilita la motivación intrínseca y anima a los estudiantes a creer que pueden aprender a hacer cualquier cosa (Dougherty, 2013, p.10).
- Reconoce el aprendizaje interdisciplinar y durante toda la vida.

Arduino y los Makers

La filosofía maker, normalmente, se relaciona estrechamente con la impresión 3D, pero Arduino también es una herramienta excepcional para adentrarnos en esta filosofía de cooperación y aprender robótica de una forma amena.

Nos permite trabajar por proyectos, aprender mientras hacemos, trabajar de forma colectiva y experimentar, es fácil de usar, sencilla y barata. Y sobre todo, lo más importante, existe una gran comunidad a su alrededor donde siempre podremos encontrar ayuda o entablar debates constructivos.

El presente curso se trata de: experimentar, aprender, intercambiar conocimientos, trabajar en equipo y disfrutar .

¿Qué es Arduino?

Arduino es una plataforma de creación electrónica de código abierto basada en el Hardware y el Software

- El código abierto es un modelo de desarrollo de software basado en la colaboración abierta entre diferentes personas o agentes.
- El Hardware libre hace referencia a aquellos dispositivos que sus especificaciones y diagramas son de acceso público, ya sea pagando o de forma gratuita.
- El software libre es aquel que su código fuente puede ser estudiado, modificado y utilizada libremente con cualquier finalidad y distribuir los cambios o mejoras realizadas.

Hablamos de una herramienta de carácter libre en todos los sentidos, y que todo el mundo puede acceder y utilizar.

¿Para qué sirve Arduino?

Arduino es una placa que permite conectar elementos periféricos a las entradas y salidas de un microcontrolador. Con Arduino es posible automatizar cualquier cosa para hacer agentes autónomos

Circuito integrado donde se pueden grabar instrucciones

- Trabajando con Arduino, se manejan conceptos de diferentes tecnologías que a priori no tienen nada que ver entre ellos pero que los unifica: electronica digital y analogica, electricidad, programación, microcontroladores, tratamiento de señales, protocolos de comunicación, arquitectura de procesadores, mecánica, motores, diseño de placas electrónicas etc…
- Al ser un entorno Open hardware y ser una gran comunidad, se han diseñado muchos tipos de placas Nosotros nos centraremos en la Placa **Arduino UNO** que es una placa standard y nos permite hacer pruebas de una manera sencilla.

Algunos ejemplos de proyectos con Arduino

- Reloj de ajedrez: https://www.instructables.com/Simple-Arduino-Chess-Clock/
- Robot que pinta: https://www.instructables.com/Arduino-Powered-Painting-Robot/
- Piano: https://www.instructables.com/Kit-Ciencia-Y-Arte-Piano/
- CNC: https://www.instructables.com/Arduino-CNC/
- Planta flotante: https://www.instructables.com/Arduino-Air-Bonsai-Levitation/
- Estación artisitca de arena: https://www.instructables.com/Arduino-Sand-Art-Display/
- Jardín controlado por arduino: https://www.instructables.com/Garduino-the-Smart-Garden-With-Arduino/

Morfología de Arduino

Entradas y Salidas

- Cada uno de los 14 pines digitales del Arduino puede ser usado como entrada o salida digital.
- Operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia interna "pull-up" (desconectada por defecto) de 20-50 KOhms.

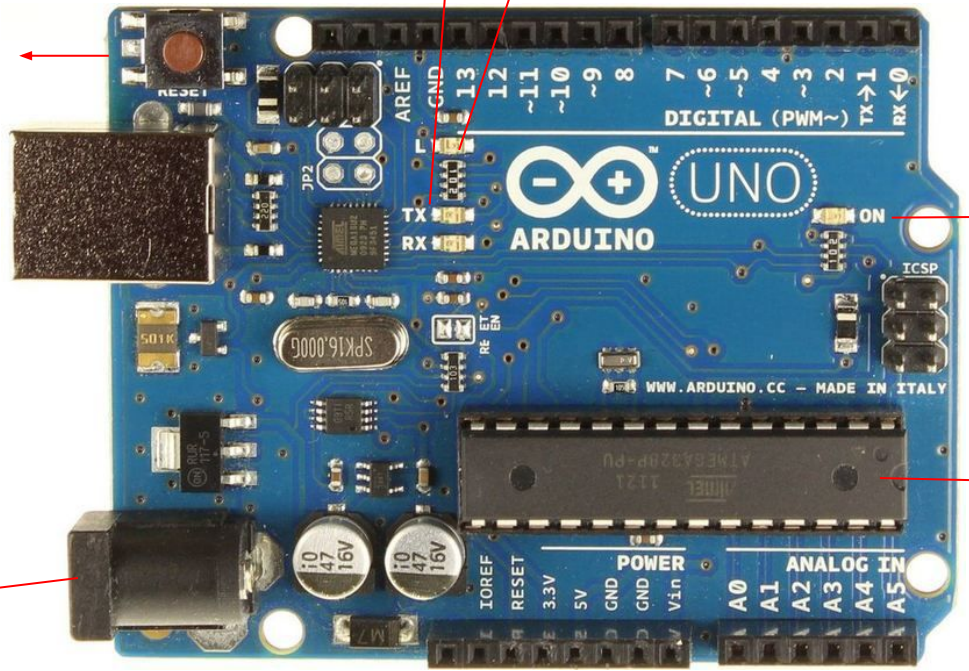
Elementos principales

LEDs TX y RX LEDs: Estos indican que hay comunicación entre tu Arduino y tu Ordenador. Puedes esperar a que parpadeen cuando cargues tu boceto, o durante la comunicación en serie. Útil para depurar.

Clavija 13 LED: El único activador que viene por defecto en tu Arduino Uno.

Botón de Reinicio: Reinicia el microcontrolador ATmega

Puerto de USB: Se usa para dar corriente al Arduino, cargar boceto a tu Arduino, y para conectar con tu boceto Arduino



Led de Corriente: Indica que el Arduino està recibiendo corriente. Útil para depurar.

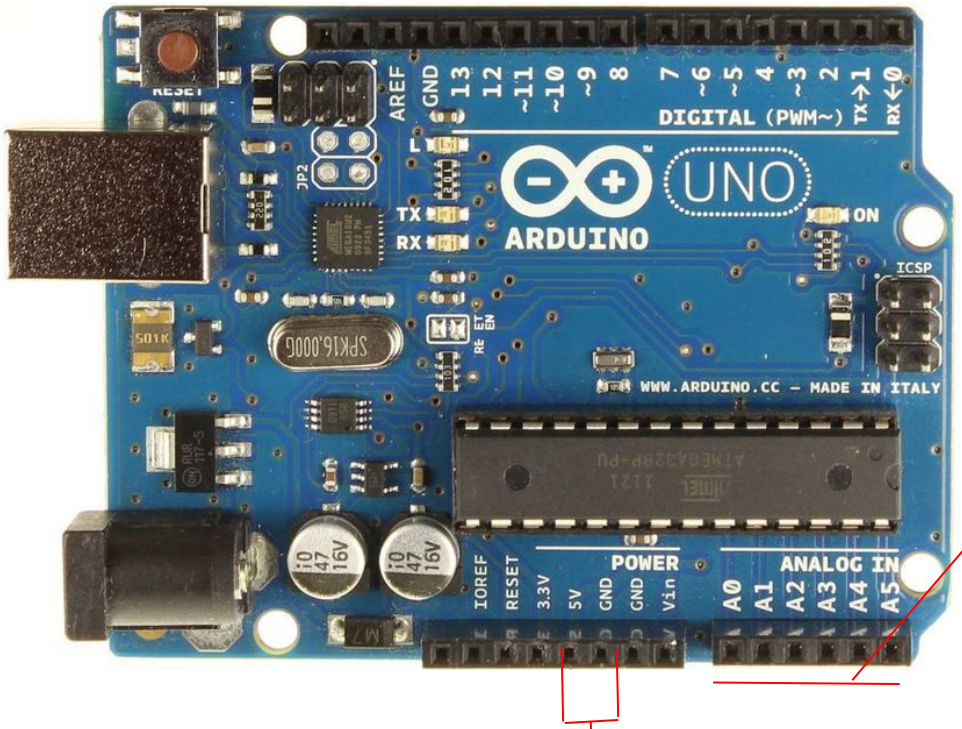
Microcontrolador ATmega: Es el corazón del Arduino

Conector de Corriente: Así es como le das corriente al arduino cuando no está conectado al Puerto USB. Puede aceptar voltajes entre 7-12V

andròmines
Creant oportunitats

SEMELI

Clavijas Digitales: Usa estas Clavijas para *digitalRead()*, *digitalWrite()*, *analogWrite()*.
analogWrite() funciona solo con las clavijas que tienen el símbolo PWM



Entrada Analógica:
Usa estas clavijas con *analogRead()*

Clavijas GND y 5V: Usa estas clavijas para darle corriente de +5V y toma de tierra a tus circuitos

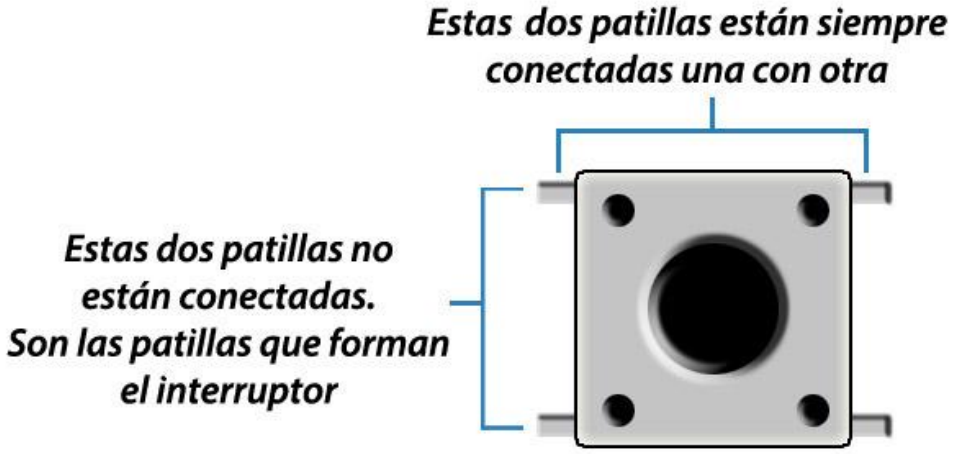
andròmines
Creant oportunitats

SEMELI

Practica 1: pulsador + LED

Material necesario :

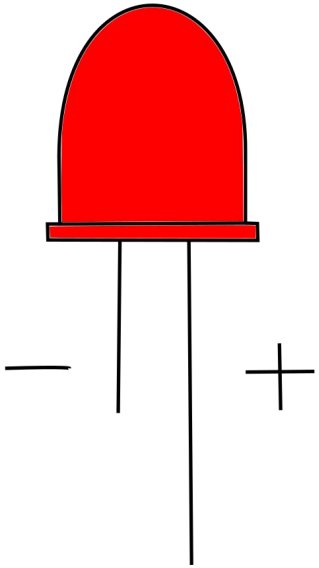
Interruptor o pulsador: Interruptores momentáneos que cierran un circuito al ser pulsados. Encaja con facilidad en la protoboards. Son útiles para detectar el encendido/apagado de señales.



andròmines
Creant oportunitats

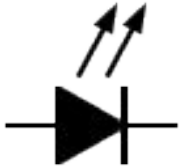
SEMELI

Práctica 1



Los LEDs *(Light Emitting Diode):* Es un tipo de Diodo emisor de Luz. En este caso será el elemento central de nuestro circuito.
Como todos los diodos la electricidad solo pasa en una dirección.

Podemos observar que las "patitas" de nuestro Led son diferentes. La "patita" corta índica el lado **negativo** (cátodo) del led y la "patita" larga indica la parte **positiva** (ánodo) del led

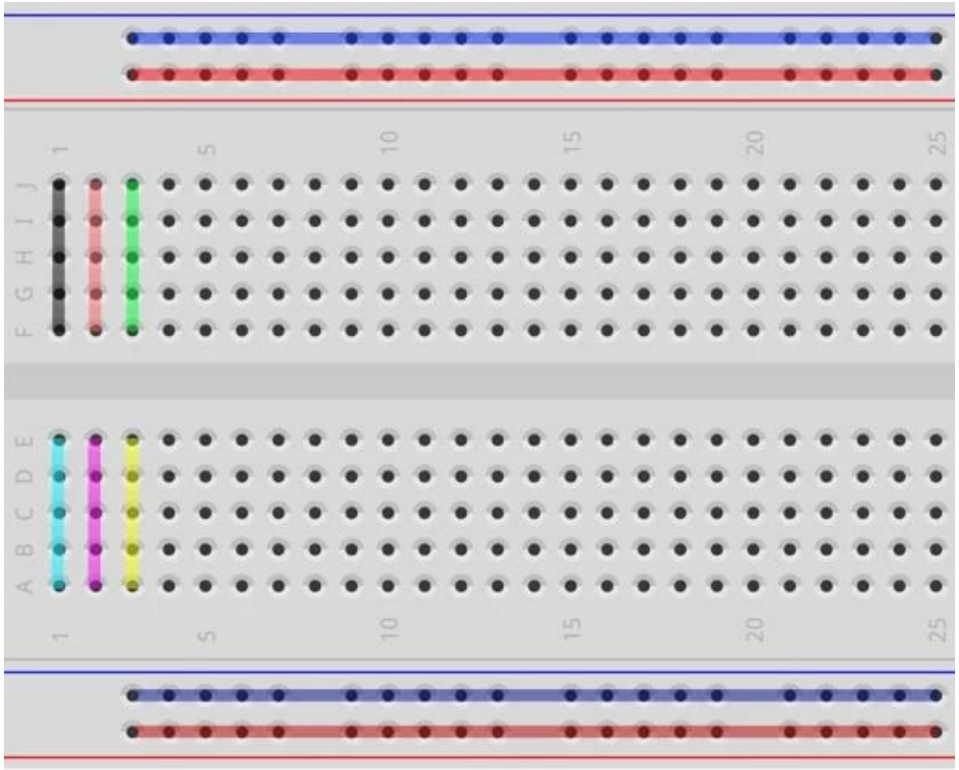


Símbolo del Led dentro del esquema del circuito

Protoboard

Será nuestra mesa de trabajo y experimentación. Es donde montaremos los circuitos. Podemos observar dos zonas claramente diferenciadas. Las franjas horizontales superiores e inferiores son aquellas donde encontraremos los polos positivos y negativos. Y las franjas verticales de la parte central de la Protoboard es donde montaremos el circuito (la conexión en esta zona va de forma vertical)

Franjas del negativo y positivo (sentido de la corriente en horizontal)



Franjas donde conectar elementos (sentido de la corriente en vertical)

andròmines
Creant oportunitats

SEMELI

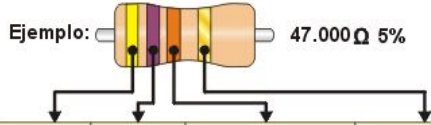
Cables de puente: Se usan para conectar componentes uno a otro en el protoboard, y con Arduino

Resistencias: Resisten el flujo de energía eléctrica en un circuito, cambiando el voltaje y la corriente.
Los valores de la resistencia se miden en ohmios.
Las bandas de colores en los lados de las resistencias indican su valor

Ω

los ohmios son representados por la letra griega omega

Código de colores para resistencias con 4 bandas



COLOR	BANDA 1	BANDA 2	MULTIPLICADOR	TOLERANCIA
NEGRO	0	0	x 1Ω	
MARRON	1	1	x 10Ω	±1%
ROJO	2	2	x 100Ω	±2%
NARANJA	3	3	x 1KΩ	
AMARILLO	4	4	x 10KΩ	
VERDE	5	5	x 100KΩ	
AZUL	6	6	x 1MΩ	
VIOLETA	7	7		
GRIS	8	8		
BLANCO	9	9		
DORADO			x 0,1Ω	±5%
PLATEADO			x 0,01Ω	±10%
			SIN BANDA →	±20%

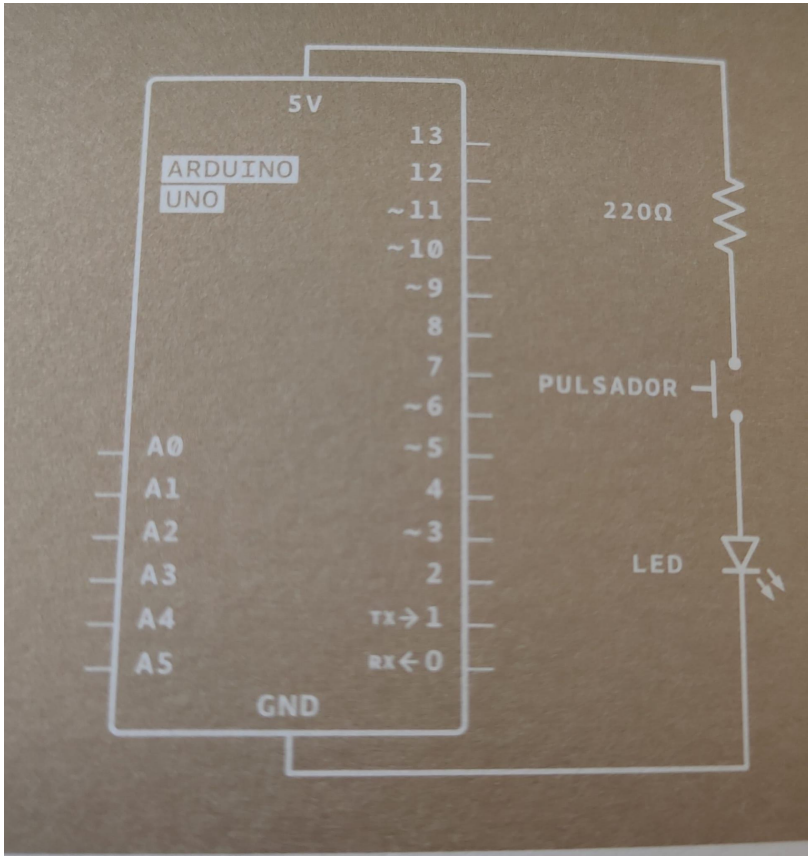
WWW.INVENTABLE.EU

En la pràctica 1 utilizaremos una resistència de 220 ohms

andròmines
Creant oportunitats

SEMELI

Practica 1: LED + pulsador

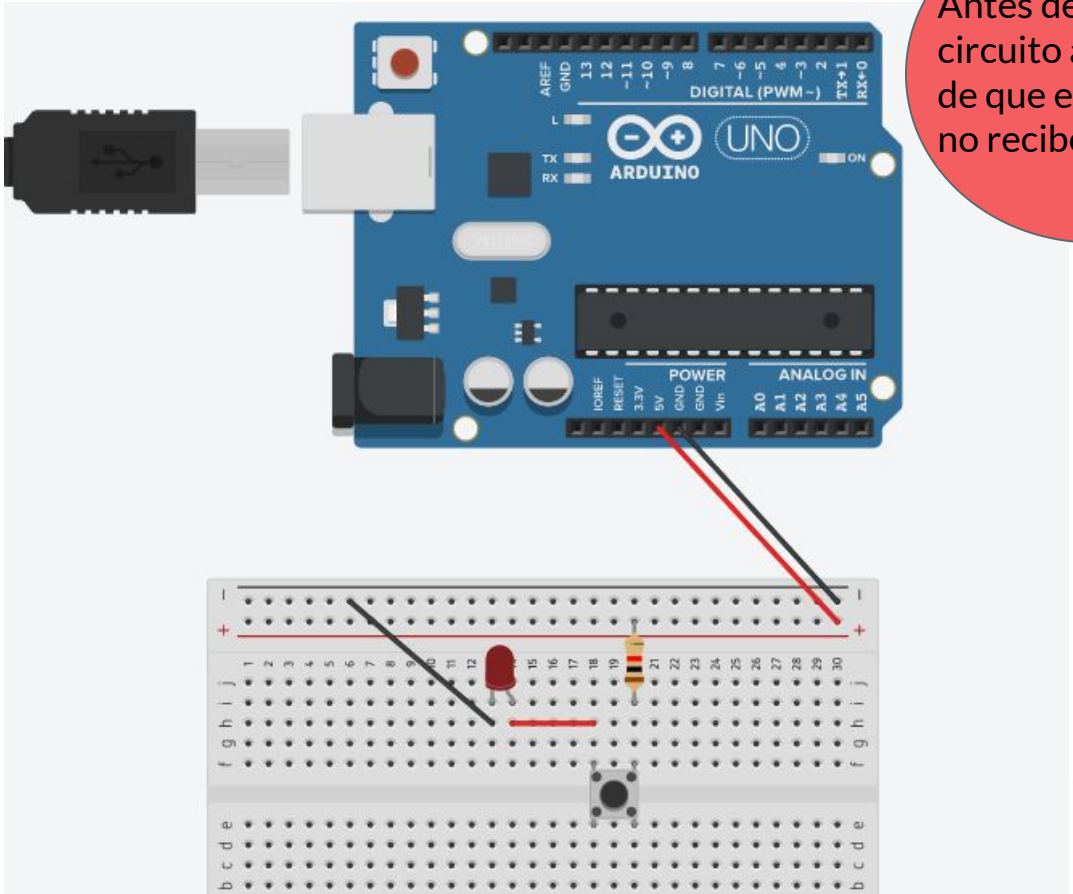


andròmines
Creant oportunitats

SEMELI
Socially Excluded youth of Mòdra literacy

Practica 1: Diagrama

- Primero haz llegar la potencia a la protoboard: cable rojo de los 5V a la franja postiviva y el cable negro del GND (tierra) a la franja negativa.
- Coloca el interruptor en el centro del tablero. La curvatura de las patitas debe quedar hacia dentro.
- Une la resistencia con el interruptor
- El interruptor con el ánodo del LED
- Une el cátodo del LED con la tierra
- Conecta el Arduino al USB



Antes de montar el circuito asegúrate de que el Arduino no recibe corriente

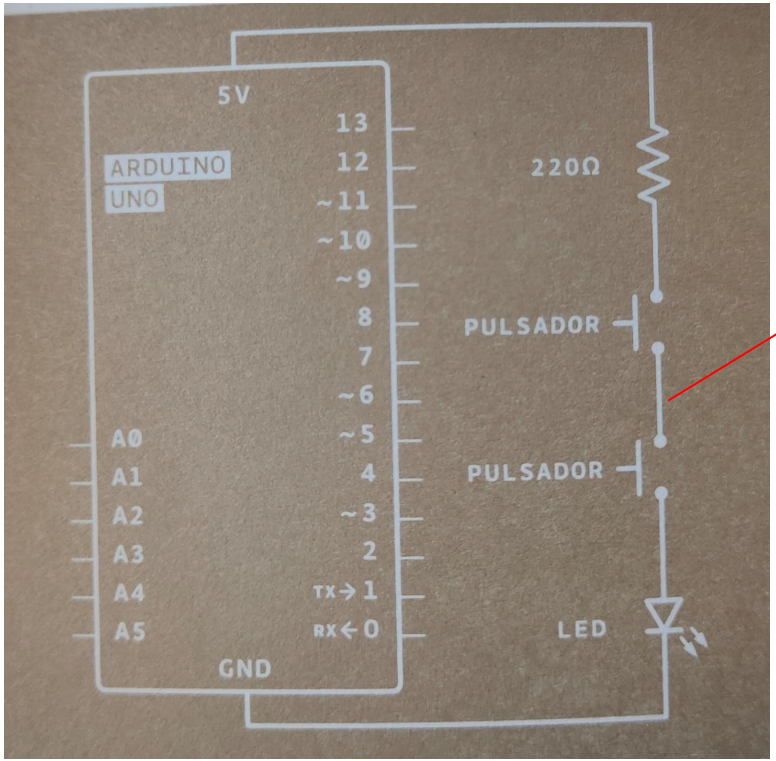
andròmines
Creant oportunitats

SEMELI

Practica 2: Circuito en serie

Circuito en serie: Los componentes se colocan uno detrás de otro.

Los dos interruptores están en serie. Es decir, por ellos fluye la misma corriente eléctrica



Indica que es un circuito en serie

¿Cómo deben estar los dos interruptores para que el LED se encienda?

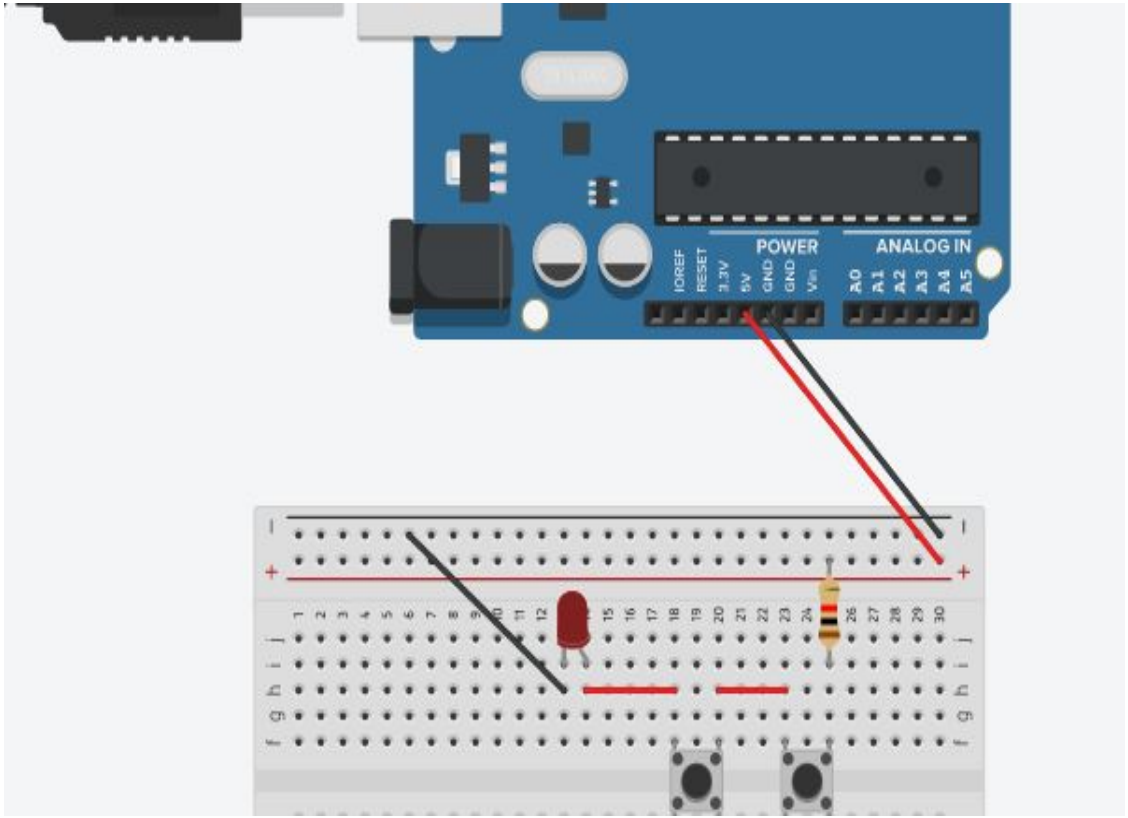
andròmines
Creant oportunitats

SEMELI

Practica 2: circuito en serie

- Al circuito anterior le añadiremos un interruptor más
- Cuando apretemos ambos interruptores, el LED se encenderà

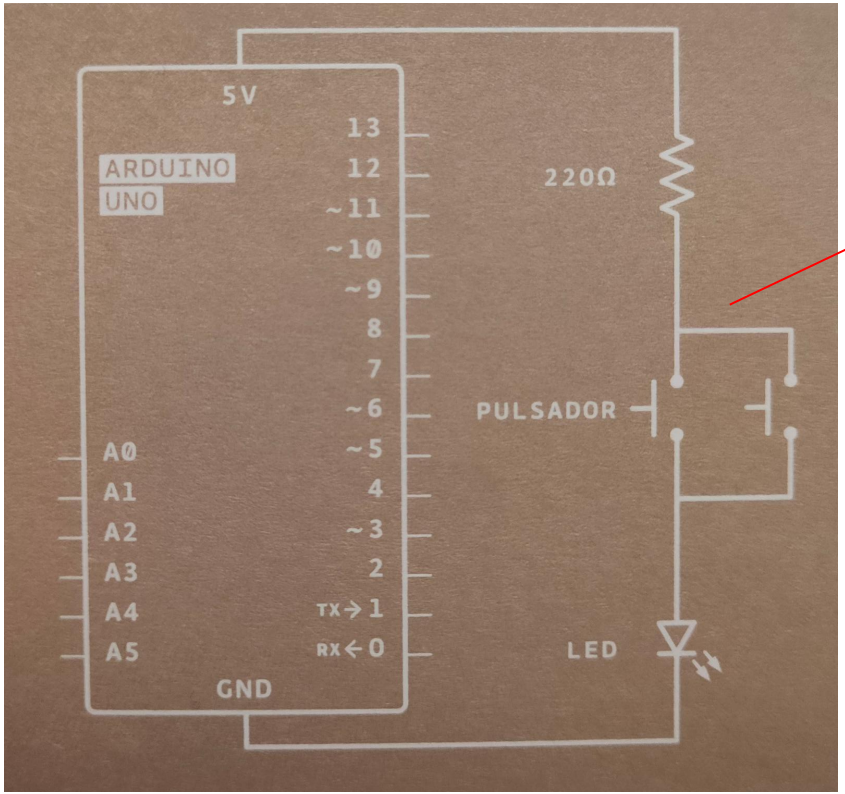
Práctica 2: Solución



Practica 3: Circuito en Paralelo

Los componentes trabajan de forma paralela.

- Vamos a eliminar la conexión entre los dos interruptores
- Conectamos ambos interruptores a la resistència y al LED



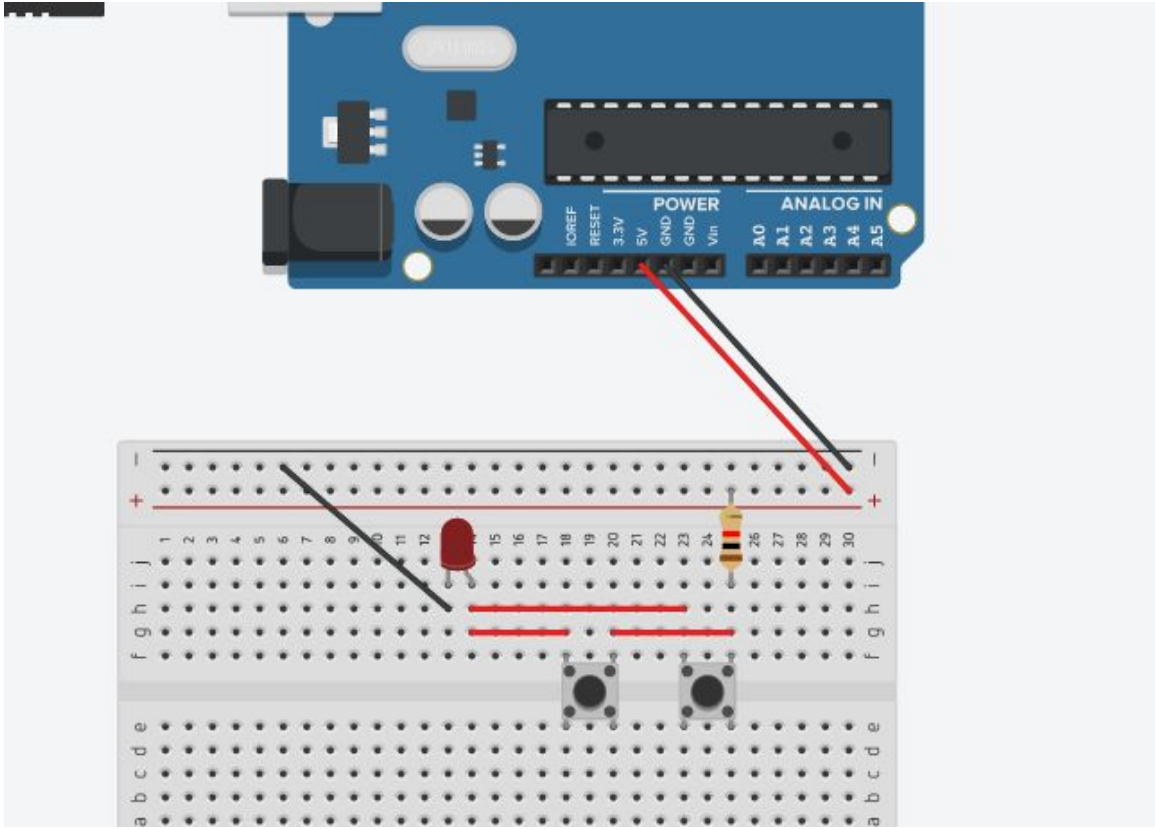
¿Qué pasará con esta configuración del circuito?

Indica que es un circuito paralelo

andròmines
Creant oportunitats

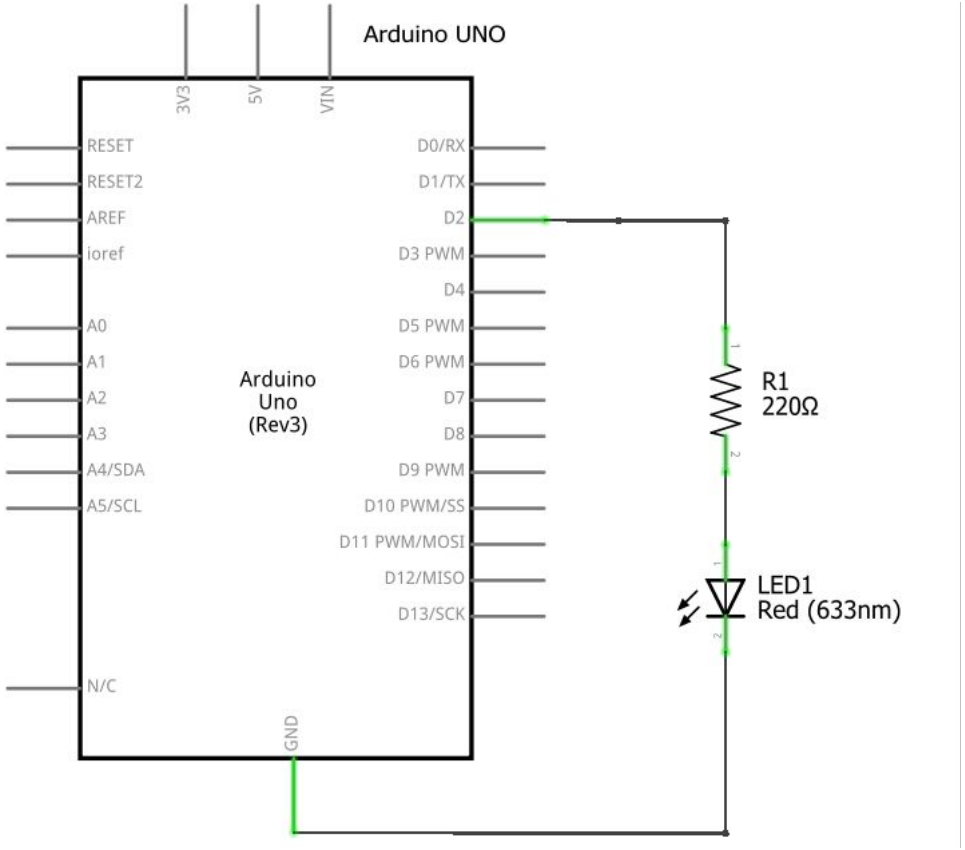
SEMELI

Practica 3: Solución



Los interruptores están en paralelo. Esto significa que la corriente eléctrica está repartida entre ellos. Si se pulsa cualquiera de ellos, el LED se encenderá

Practica 4: entrada y salida digital



andròmines
Creant oportunitats

SEMELI

Práctica 4: diagrama

Antes de montar el circuito asegúrate de que el Arduino no recibe corriente

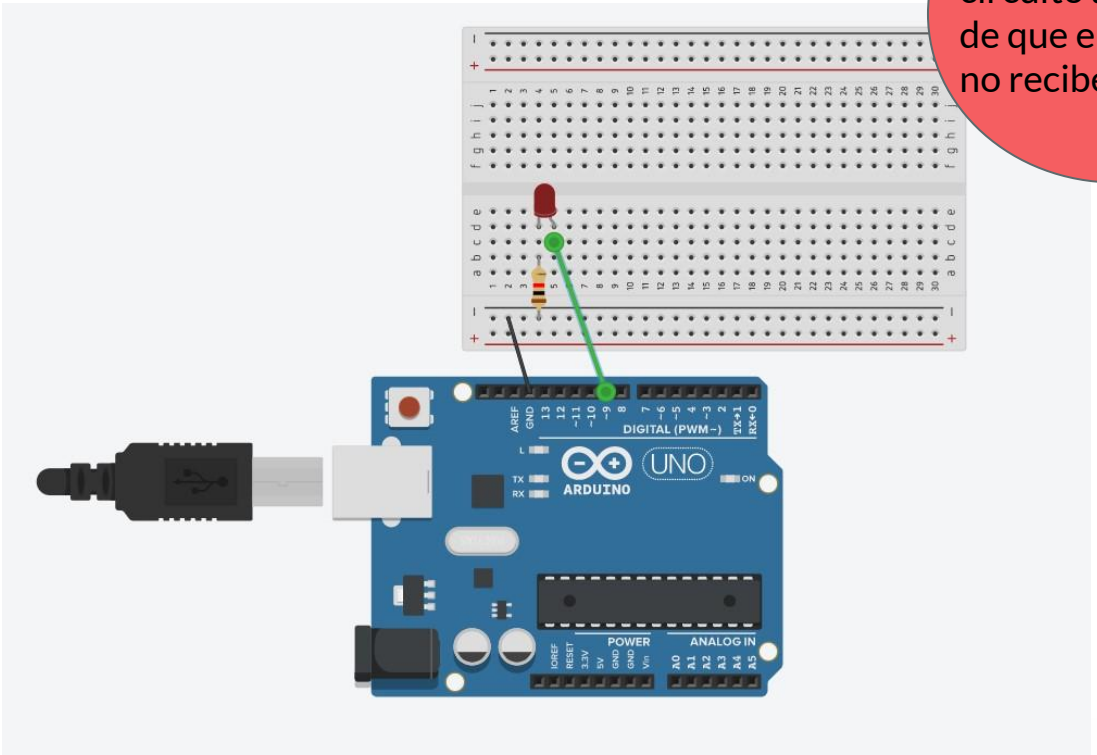
El primer circuito se configura de la siguiente manera:

-Pondremos una resistencia que va del lado negativo del Led a la parte negativa de la protoboard.

-Un cable que va de la parte negativa de la Protoboard al GND (tierra) del Arduino

-Del lado positivo del Led, saldrá un cable que nos llevará al Pin 9

(sin la programación, el Led no se enciende)



andròmines
Creant oportunitats

SEMELI

Practica 4: Código

Pasamos a realizar el código con el programa de Arduino

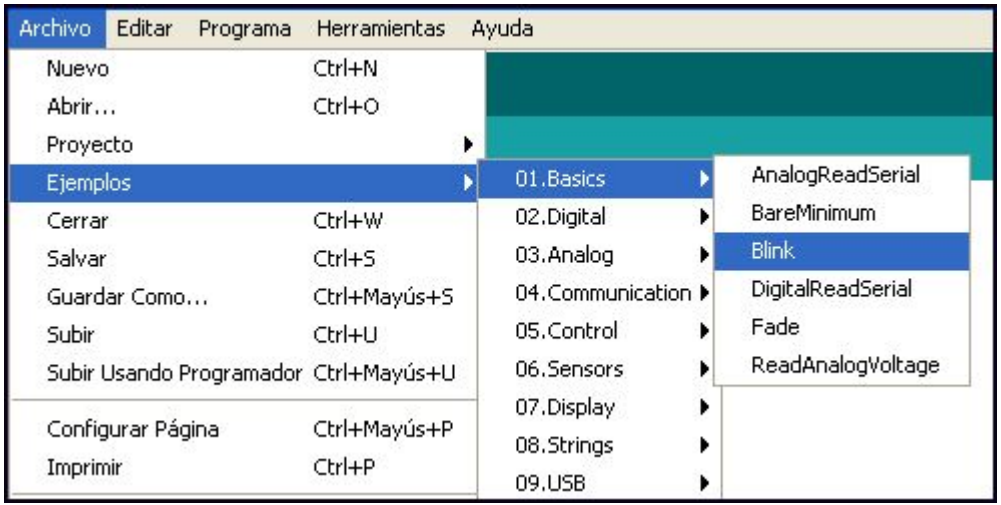
Ya no haremos código por bloques, sino que **escribiremos** el Código

Practica 4: Configuración de Arduino IDE

Podemos cambiar el idioma del Programa en:
Archivo > Preferencias>Editor de Idiomas
(se debe reiniciar el programa)

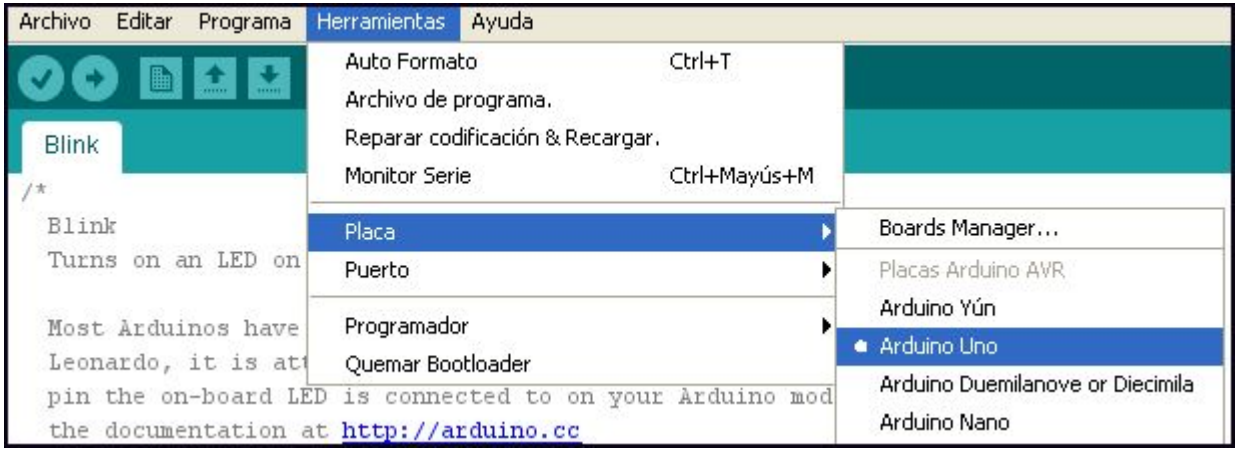
Vamos a comprobar que el Arduino se conecta bien. Abriremos un Sketch (es como se llama a los programas de Arduino) de ejemplo:

- Archivo>Ejemplo>Basic>Blink



Practica 4: Configuración de Arduino IDE

1-Se abrirá una nueva ventana con un texto dentro.



2- Dejamos el texto como esta

3- Abrimos:

Herramientas>Placa> Arduino Uno

andròmines
Creant oportunitats

SEMELI

Practica 4: Configuración de Arduino IDE

Escogemos puerto serie donde estará conectado Arduino:

- Herramienta>Puerto

Y le damos a subir para cargar la configuración en nuestra placa

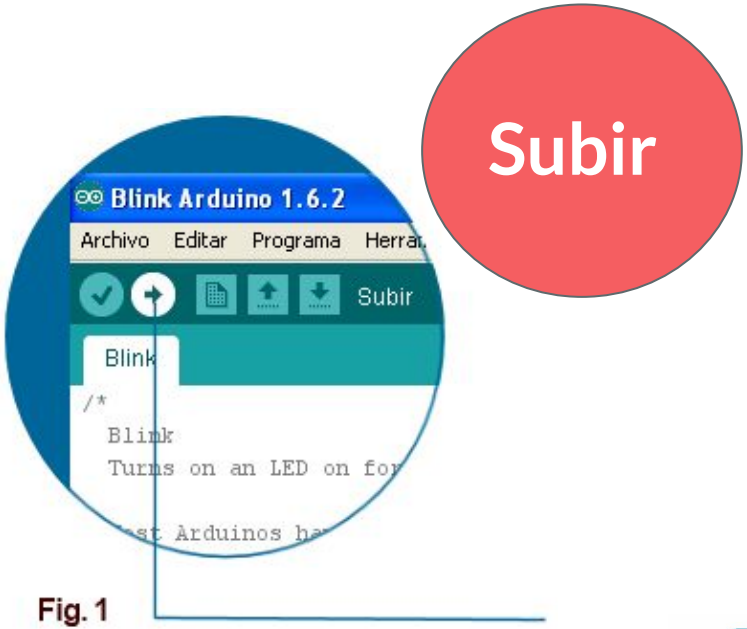


Fig. 1

Practica 4: Configuración de Arduino IDE

Debe de ver una barra indicando el progreso de carga del sketch cerca de la esquina inferior derecha del IDE de Arduino, y los diodos led de la placa Arduino con las etiquetas TX y RX estarán parpadeando en el momento de la carga. Si la carga se ha realizado correctamente, el IDE mostrará el mensaje SUBIDO en la esquina inferior izquierda.

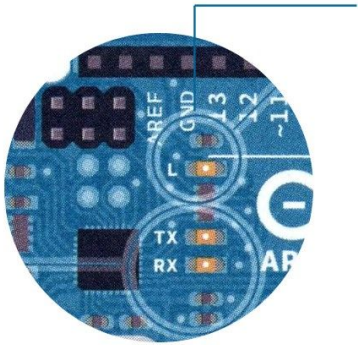


Fig. 2

A los pocos segundos de completar la carga del Sketh, debe de ver como el diodo led amarillo, con la etiqueta L cerca, comienza a parpadear.

A veces su nuevo Arduino está ya programado con el Sketch de parpadeo, así que no puede saber si realmente lo acaba de programar. En este caso, cambiar dentro de la instrucción "delay" el tiempo que aparece entre paréntesis a 100, y volver a subir de nuevo el sketch de parpadeo. Ahora el diodo led de la placa debe de parpadear más rápido.

Practica 4: Programación con MBlock

Esta es la programación básica para conseguir que el Led se encienda.

Modifica la programación para conseguir:

-Que el Led parpadee

Solucion

```
// Práctica encender y apagar un LED
const int LED=13;
void setup()
{
pinMode(LED,OUTPUT);
}
void loop()
{
digitalWrite(LED,HIGH);
delay(1000);
digitalWrite(LED,LOW);
delay(1000);
}
```

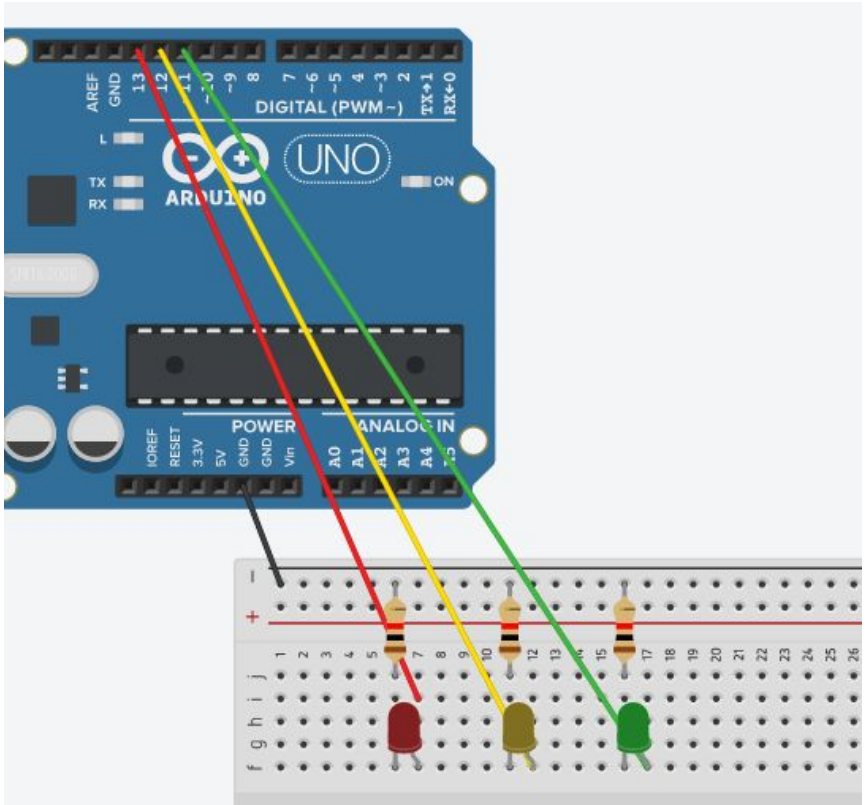
Practica 4.1: subimos de nivel

Basándonos en el circuito anterior:

- Crea un circuito con la misma estructura, pero en lugar de tener solo un LED, deberà tener:
  - 1LED rojo
  - 1 LED amarillo
  - 1LED verde

- Debes crear la programación con MBlock para que estos se vayan encendiendo igual que un semaforo.

Solución: diagrama



andròmines
Creant oportunitats

SEMELI

Solución: programación Mblock

Solución: Programación con Arduino IDE

```
1.  // Semáforo sencillo en Arduino
2.  int tiempoEspera = 5000;
3.
4.  void setup(){
5.    pinMode(10, OUTPUT);    // Rojo
6.    pinMode(9, OUTPUT);     // Amarillo
7.    pinMode(8, OUTPUT);     // Verde
8.  }
9.
10. void loop() {
11.   digitalWrite(10, HIGH);     // rojo encendido
12.   delay(tiempoEspera);
13.
14.   digitalWrite(8, HIGH);      // verde encendido
15.   digitalWrite(10, LOW);      // rojo apagado
16.   delay(tiempoEspera);
17.
18.   digitalWrite(9, HIGH);      // amarillo encendido
19.   digitalWrite(8, LOW);       // verde apagado
20.   delay(1000);
21.
22.   digitalWrite(9, LOW);       // amarillo apagado
23. }
```

Practica 4.2

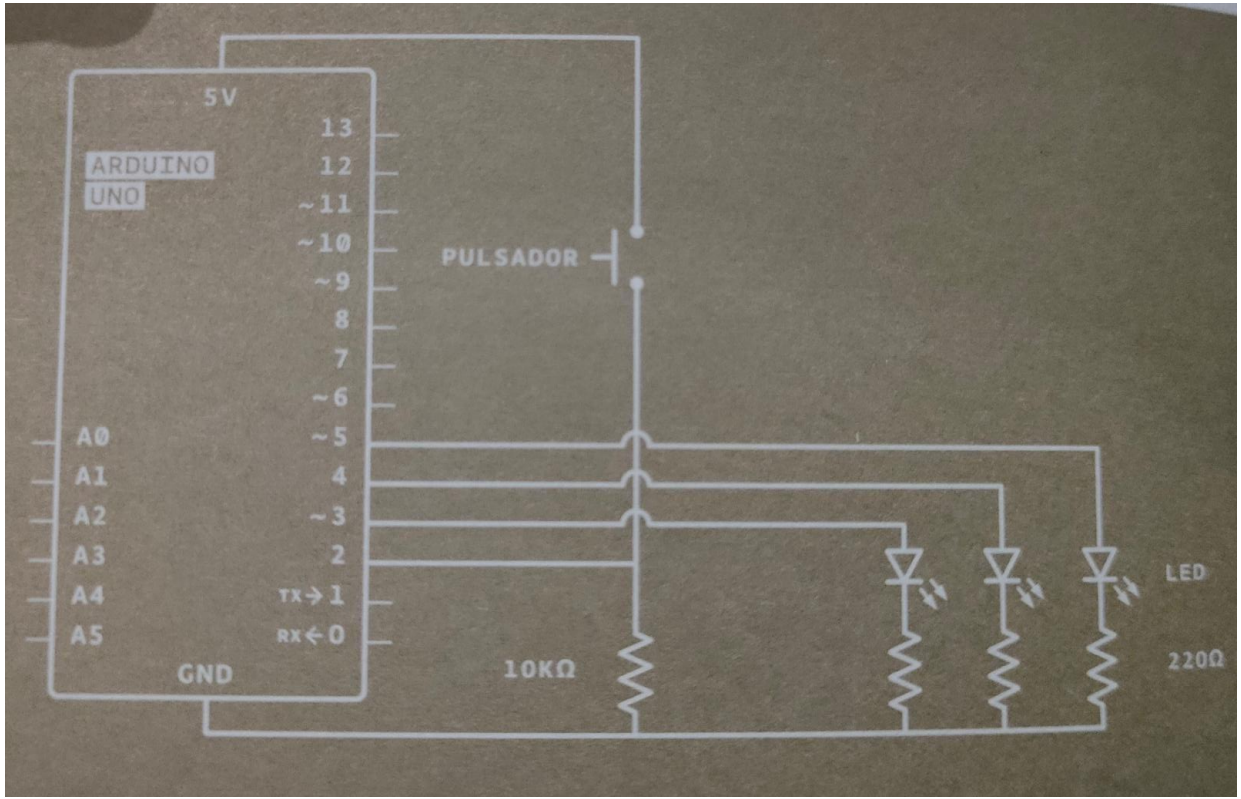
- Modifica el tiempo de espera
- Modifica el orden de las luces
- Haz que verde y rojo se enciendan a la vez, mientras que el amarilo está pagado

Practica 5: LEDS + Pulsador + Programación

Material necesario:

- 3 Leds: rojo, amarillo, verde
- 3 resistencias de 220 ohmios
- 1 pulsador
- 1 resistencia de 10k ohmios
- Cables Puente76

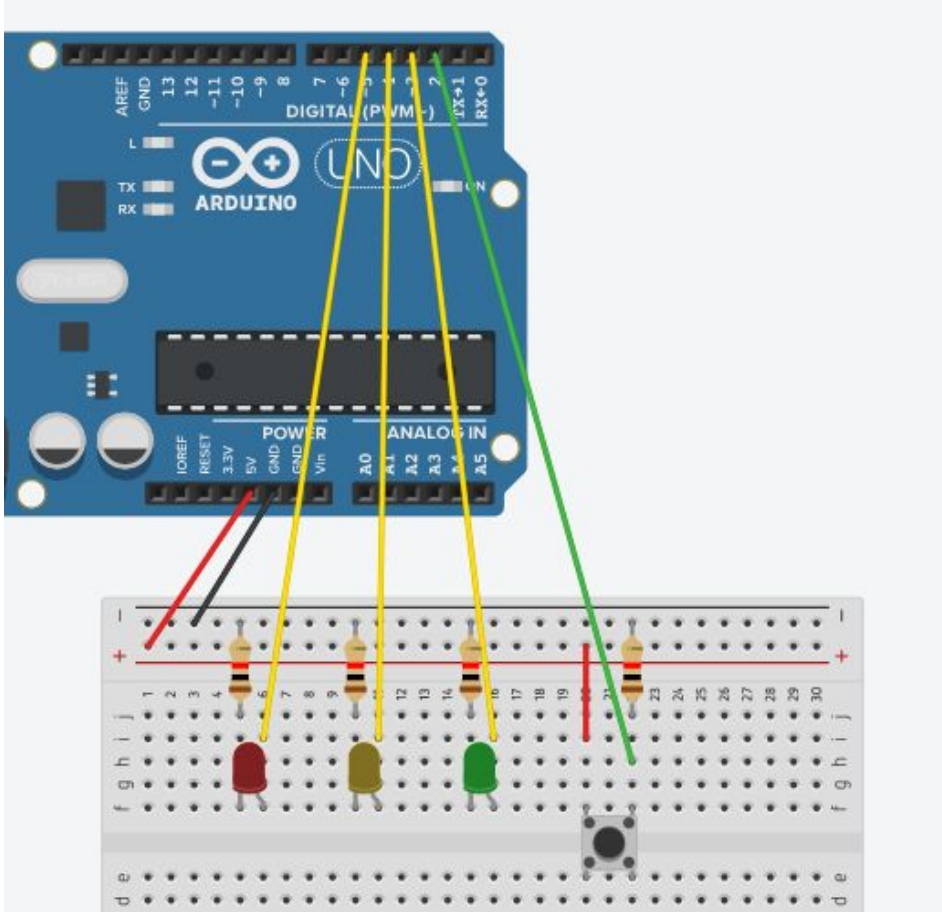
Practica 5: LEDS + Pulsador + Programación



andròmines
Creant oportunitats

SEMELI
Socially Excluded youth of Mòdra literacy

Practica 5: Diagrama



andròmines
Creant oportunitats

SEMELI

Practica 5: Programación

El objetivo de la programación és: Un diodo LED verde permanecerá encendido hasta que pulse el botón. Cuando Arduino reciba la señal del botón pulsado, la luz verde se apaga y se encienden otras dos luces que comienzan a parpadear.

Practica 5: Programación

Los terminales o pins digitales de Arduino solo pueden tener **dos estados**:

- cuando hay voltaje en un pin de entrada
- cuando no lo hay.

Este tipo de entrada es normalmente llamada digital (o algunas veces binaria, por tener dos estados). Estos estados se refieren comúnmente como **HIGH** (alto) y **LOW** (bajo).

HIGH es lo mismo que decir “¡aquí hay tensión!” y LOW indica “¡no hay tensión en este pin!”. Cuando pone un pin de SALIDA (OUTPUT) en estado HIGH utilizando el comando llamado digitalWrite(), está activando-lo. Si mide el voltaje entre este pin y masa, obtendrá una tensión de 5 voltios. Cuando pone un pin de SALIDA (OUTPUT) en estado LOW, está apagando-lo.

Practica 5: Programación

Los pins digitales de Arduino pueden trabajar como:

- entradas
- salidas.

En su código, los configurará dependiendo de cual sea su función dentro del circuito.

- Cuando los pins se configuran como **salidas**, entonces podrá encender componentes como los diodos LEDs.
- Si se configuran como **entradas**, podrá verificar si un pulsador está siendo presionado o no. Ya que los pins 0 y 1 son usados para comunicación con el ordenador, es mejor comenzar con el pin 2.

Conceptos importantes

- **Sensores:** elementos electrónicos para sentir
- **Actuadores:** son elementos electrónicos que actuan; iluminan, mueven, hacer ruido

Todos los pines son entradas y salidas. Todos pueden actuar como ambos. En el **Pin Mode** tu le dices que funcion tiene.

Entradas: sensores. Por ejemplo: el interruptor es una entrada por que cuando tu lo pulsas envía la señal al pin (arduino). El arduino enciende el Led **(salida)**

- **La ortografía es MUY importante.** Un punto, mayúscula, punto y coma donde no toca, lo fastidia todo.

Practica 5: Programación

Notas Importantes antes de començar:

Cada programa de Arduino tiene dos funciones básicas principales. Estas dos funciones principales en un programa de Arduino son llamadas con **setup()** y **loop(),** la cuales necesitan **ser declaradas**, esto quiere decir que es necesario **indicarle a Arduino lo que estas funciones harán.**

Lo primero que hay que hacer antes de meterse en la parte principal del programa **es crear una variable**.

Las **variables** son nombres que se utilizan para **guardar información** dentro de la memoria de Arduino

El **nombre de las variables** deben de ser una **descripción de tipo de información que contienen.** Por ejemplo, una variable llamada SwitchState (estado del pulsador) le dice lo que está guardando: el estado de un pulsador.
Por otra parte, una variable llamada “x” no dice mucho acerca del tipo de información que guarda.

Practica 5: Configuración

Comenzamos el Programa:

Para crear una variable, es necesario declarar de que tipo se trata. Una variable de tipo **int** guardará un número entero (también llamado integer); eso significa que almacena cualquier número sin decimales. Cuando se declara una variable, normalmente también se le asigna a la vez un valor inicial. Las declaraciones de la variables siempre deben de **finalizar con un punto y coma (;).**

Practica 5: Configuración

Configurando como funciona una PIN:

La función **setup()** solo se ejecuta una vez, cuando Arduino recibe la alimentación para funcionar. Esta función define un bloque, el cual se abre con una llave **"{" y se cierra con otra llave "}",** en donde se escriben las instrucciones que configuran los pins digitales de Arduino como entradas o como salidas, usando para ello una función llamada **pindMode().** Los pins conectados a los LEDs serán **OUTPUTs** y el pin de un pulsador será una **INPUT.**

Para insertar las llaves: AltGr + {

Practica 5: Configuración

Crear una función Loop:

La función **loop()** se ejecuta continuamente después de que la función **setup()** se haya completado. A través de las instrucciones que se incluyen dentro del bloque después de la función **loop()** es donde se comprobará el voltaje de las entradas y si las salidas están activadas o desactivadas. Para verificar el nivel de voltaje de una entrada digital, se utiliza la función **digitalRead()** la cual comprueba el voltaje en el pin elegido. Para saber que pin debe de verificar **digitalRead()** espera un argumento.

Los argumentos son información que se le pasa a las funciones diciéndoles cómo deben de realizar su trabajo. Por ejemplo, **digitalRead()** necesita un argumento: que pin debe verificar. En el programa de Interface de nuestro circuito, **digitalRead()** va a verificar el estado del **pin 2** para después **almacenar el valor dentro de la variable switchState.**

Si hay voltaje en el pin 2 cuando digitalRead() es llamada, entonces la variable switchState **almacena un valor HIGH** (o 1). Si **no** hay voltaje en el pin 2, switchState almacenará el valor **LOW** (o 0).

```
void setup (){
}

void loop (){
}
```

{ Las llaves }
Cualquier código que escriba entre llaves será ejecutado cuando la función es llamada

```
1 int switchState = 0;
```

```
2 void setup (){
3   pinMode(3, OUTPUT);
4   pinMode(4, OUTPUT);
5   pinMode(5, OUTPUT);
6   pinMode(2, INPUT);
7 }
```

Mayúsculas y minúsculas

Poner atención a la hora de escribir en mayúsculas y minúsculas dentro del código. Por ejemplo, pinMode es el nombre de una instrucción, pero pinmode producirá un error.

Comentarios

Si alguna vez quiere usar el lenguaje natural dentro del programa, puede escribir un comentario.

Los comentarios son notas que se dejan para recordar lo hace el programa, además el microcontrolador ignora estos comentarios. Para añadir un comentario escribir antes dos líneas inclinadas // y a continuación escribir lo que se desee anotar. El microcontrolador ignorará cualquier texto escrito después de estas dos líneas.

```
8 void loop (){
9   switchState = digitalRead(2);
10  // Esto es un comentario
```

```
11    if (switchState == LOW) {
12    // el pulsador no está presionado
```

Indicador If

Se utiliza la palabra “if” para **verificar el estado** de algo (en este caso el estado del pulsador, en qué posición encuentra). Un estamento if() en programación compara dos cosas, y determina si la comparación es verdadera o falsa. Dependiendo de este resultado se lleva a cabo una acción u otra.

Cuando en programación se comparan dos cosas hay que usar dos signos de igual ==. Si solo se usa un signo de igual, simplemente se le asigna un valor a una variable en lugar de compararla con algo.

```
13    digitalWrite(3, HIGH); // LED verde
14    digitalWrite(4, LOW); // LED rojo
15    digitalWrite(5, LOW); // LED rojo
16  }
```

digitalWrite() es la instrucción que permite poner +5V o 0V en un pin de salida. **digitalWrite()** tiene dos argumentos: el número del pin a controlar y que valor se coloca en ese pin, HIGH o LOW.

Si quiere apagar los LEDs rojos y encender el LED verde dentro del estamento **if()** el código debería ser como el que se muestra en la siguiente página en la parte superior donde aparecen las tres instrucciones **digitalWrite()**.

```
17      else { // el pulsador está presionado
18          digitalWrite(3, LOW);
19          digitalWrite(4, LOW);
20          digitalWrite(5, HIGH);
21          delay(250); // en pausa un cuarto de segundo
22          // cambiar el estado de los LEDs
23          digitalWrite(4, HIGH);
24          digitalWrite(5, LOW);
25          delay(250); // en pausa un cuarto de segundo
26      }
27  } // Volver al comienzo de la instrucción loop
```

En las instrucciones anteriores se le indica a Arduino lo que tiene que hacer cuando el pulsador está abierto. También hay que indicarle qué hacer cuando el pulsador está cerrado. El estamento **if()** puede usar opcionalmente el componente else que permite hacer otra cosa si la condición inicial no se cumple. En este caso, como se acaba de comprobar a través del código **si el pulsador está LOW,** hay que escribir nuevas instrucciones para la condición HIGH dentro del bloque de la instrucción else.

```
17    else { // el pulsador está presionado
18        digitalWrite(3, LOW);
19        digitalWrite(4, LOW);
20        digitalWrite(5, HIGH);
21        delay(250); // en pausa un cuarto de segundo
22        // cambiar el estado de los LEDs
23        digitalWrite(4, HIGH);
24        digitalWrite(5, LOW);
25        delay(250); // en pausa un cuarto de segundo
26    }
27 } // Volver al comienzo de la instrucción loop
```

Para conseguir que los LEDs rojos **parpadeen** cuando el pulsador este presionado es necesario encenderlas y a pagarlas dentro del bloque de la instrucción else que se acaba de escribir.

Es necesario que Arduino realice una pausa antes de cambiarlos de nuevo a su estado anterior. Sino, no se verá el parpadeo. Dará la sensación que solo disminuyen su luminosidad. La instrucción **delay()** permite que Arduino deje de ejecutar cualquier cosa que esté haciendo durante un periodo de tiempo. Dentro del argumento de la instrucción **delay()** se establece el número de mili segundos que Arduino estará parado antes de ejecutar la siguiente parte del código. Hay 1000 mili segundos en un segundo. delay(250) producirá una pausa de un cuarto de segundo.

Practica 5: Resultado

Una vez que Arduino esta programado, el LED verde debe de verse encendido. Cuando presione el pulsador, los LEDs rojos comenzarán a parpadear, y el LED verde se apagará.

Intenta:

- Cambiar la velocidad del parpadeo **() delay**
- ¿Cómo podría conseguir que los diodos LEDs comiencen a parpadear cuando el programa comienza?

Practica 5: Resumen

En este proyecto, ha creado su primer programa para Arduino para controlar el comportamiento de algunos LEDs a través de un pulsador. Ha usado variables, una instrucción if()...else, y funciones para leer el estado de una entrada y controlar salidas.

¿Qué hemos usado?

If: "Si" esto esta asi….

delay: instrucciones de tiempo de espera o ciclos vacíos

digitalWrite: escribir en ese Pin. Le dice si es alto o bajo

switchState: pulsado. Es una variable

void loop: instrucciones que iran en buckle

else: "SI no" introducción de un cambio de estado

void setup: introducciones de cómo debe estar todo al empezar

int: Se indica antes del void setup

PinMode: detalles de cada PIN

Dato curioso del *delate*

Delate: ciclos que procesas sin hacer nada.

Arduino tiene 16 Mhz (megaherzios). Esto significa que hace 16.000 instrucciones por segundo.

Cuando pones un delate, es el número de operaciones o segundos que no va hacer. Por eso también se llama “ciclos vacíos”

Lo malo del delate es que no se puede interrumpir. Por este motivo, en programación avanzada, no se utiliza el delate y se utilizan los condicionales, que si se pueden interrumpir

Durante los ciclos de Delate, no lee sensores ni nada. Es como poner el freno de mano.

La máquina puede estar ardiendo que no lo lee

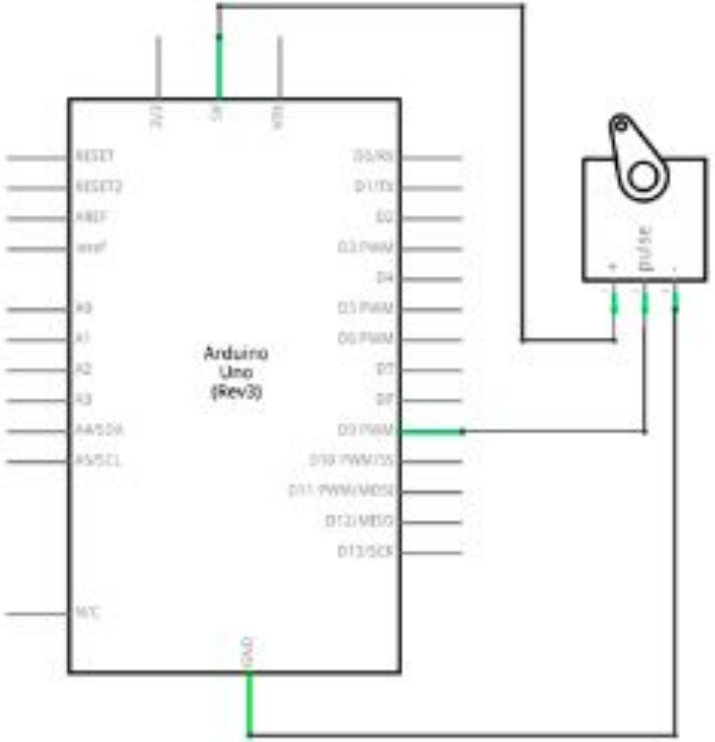
Practica 6: Servomotor

Un **servomotor** se mueve desde su posición origen a su posición final en ángulos de 10° repetidamente.

Movimiento:
Estos servomotores funcionan con un periodo de 20ms, es decir, podremos cambiar de posición cada 20ms como
mínimo.

Practica 6: esquema

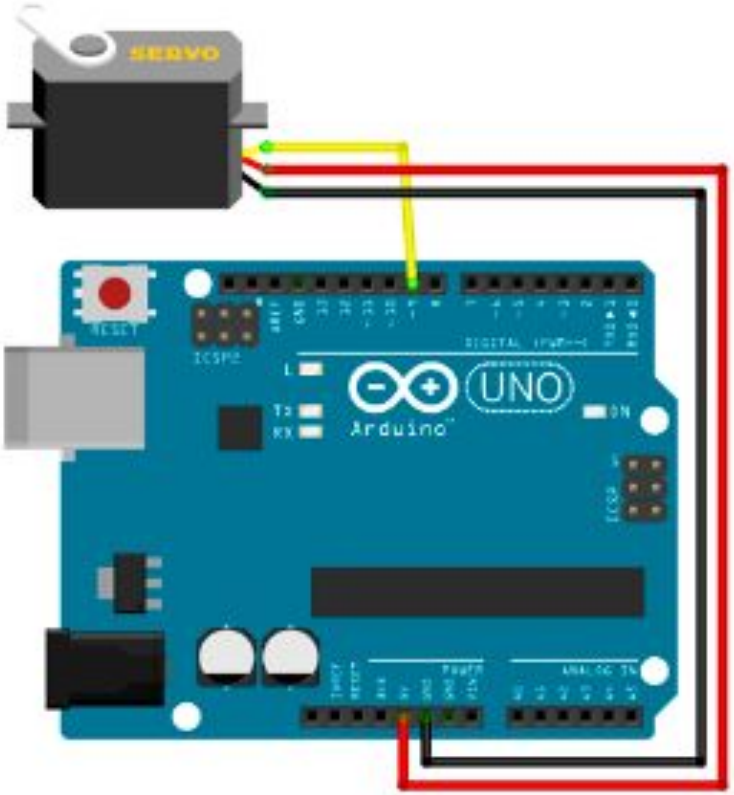
Los servomotores suelen tener los cables con los **colores marrón, rojo y naranja,** los cuales se corresponden con el GND, 5V y Pin de salida analógica respectivamente



andròmines
Creant oportunitats

SEMELI

Practica 6: esquema



andròmines
Creant oportunitats

SEMELI

Practica 6: Mblok

Practica 7: Detector de aparcamiento

Practica 5: Programación