

Módulo 1

1. Tecnología y creatividad

La tecnología convive con nosotros diariamente. Está presente en nuestras vidas desde las rudimentarias herramientas desarrolladas por los primeros humanos para cazar y cocinar hasta los celulares con conectividad a internet. La palabra tecnología viene del griego *τέχνη* y quiere decir “*arte, oficio o destreza*”. La tecnología implica un proceso, una capacidad para transformar lo que nos rodea en algo nuevo u otorgarle una nueva función.

Podemos pensar que la tecnología nos brinda herramientas que nos permiten aumentar la productividad, por ejemplo, para la fabricación masiva y estandarizada de materiales textiles, pinturas y accesorios. Pero la ciencia y el desarrollo también se presentan como una posibilidad para crear nuevas cosas. Fomentar espacios donde la tecnología y la creatividad se unen da lugar a producciones novedosas y de alto impacto.

A continuación, algunos ejemplos de nuevas tecnologías aplicadas a producciones culturales:

- Música creada a partir de inteligencia artificial

[Conocé a Pierre Barreau, el experto detrás del algoritmo que crea música con Inteligencia Artificial \(IA\)](#)

- Digitalización del arte y los museos

[Tres museos argentinos suman su colección a la plataforma Google Arts and Culture](#)

[Recorrido virtual Museo de La Plata](#)

- Impresión 3D y arte

[Cordobés del Año: Irene Presti, de la Cámara Argentina de Impresión 3D](#)

[Mariel Lluch, llenando Argentina de impresión 3D!](#)

2. Robótica: conceptos y evolución

Veamos el siguiente ejemplo de tecnología aplicada a la escenografía teatral:

[Detrás de escena | Efectos Escénicos](#)

¿Qué se nos viene a la mente cuando pensamos en un robot?

Seguramente la mayoría de nosotros pensamos en máquinas metálicas, articuladas con forma humanoide, que pueden realizar diversas tareas y hasta en algunos casos comunicarse con las personas.



Izq. Robot humanoide fabricado por Toyota. Der. El personaje de Disney Wall-e.

Lo cierto es que no existe una definición universal sobre qué es un robot. El concepto fue evolucionando con el paso de los años y los avances de la tecnología.

El término robot proviene del vocablo checo *robota* que significa “trabajo forzado”. Fue utilizado por primera vez por el escritor checo Karel Capek en 1921 en el estreno de una obra de teatro.

Al día de la fecha se puede considerar robot a cualquier dispositivo que muestra un comportamiento inteligente para realizar una tarea. No necesariamente se asemejan a la figura humana ni son programados para reemplazar a las personas en tareas repetitivas, forzadas o de precisión.

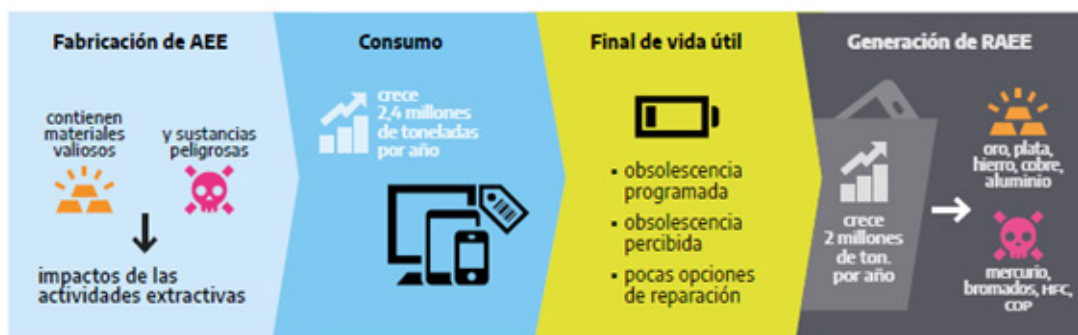
A lo largo de los módulos de este curso vamos a ver las partes que componen un sistema automatizado o robótico, así como también conceptos básicos que nos introducen al uso de estas tecnologías en contextos artísticos.

3. Uso responsable y sustentable de los artefactos electrónicos

La aplicación de tecnologías como los sistemas de control implica el uso de artefactos electrónicos (instrumentos de monitoreo, pantallas, luces, computadoras, etc.).

En los últimos años, los residuos de aparatos eléctricos y electrónicos (RAEE) representan la fracción de residuos con mayor crecimiento a nivel mundial. Argentina no es una excepción. En el país se generaron 465.000 toneladas anuales de RAEE, casi un 25% más que en los últimos dos años.

La fabricación del aparato (AEE) así como su descarte producen una serie de impactos sobre la naturaleza y la salud de las personas. Los AEE emplean materiales obtenidos a partir de fuentes no renovables y pueden contener sustancias peligrosas.



Ciclo de vida de un artículo eléctrico o electrónico, desde su fabricación hasta la generación de los residuos. Los valores corresponden a cifras mundiales.

¿Qué podemos hacer para minimizar este impacto?

La gestión integral de los RAEE permite reducir los riesgos asociados a la liberación de sustancias peligrosas y recuperar materiales para su reinserción en la industria. De esta manera, se reduciría el uso de materias primas no renovables y se evitaría la contaminación del ambiente.

Esta gestión integral abarca diversas medidas y regulaciones que van desde la generación de los residuos y su valorización hasta la disposición final. Para que estas acciones sean exitosas deben estar involucrados todos los actores: fabricantes, distribuidores y Estados; como también los consumidores de los productos electrónicos, entre otros.



Representación de la jerarquía en la gestión integral de residuos.

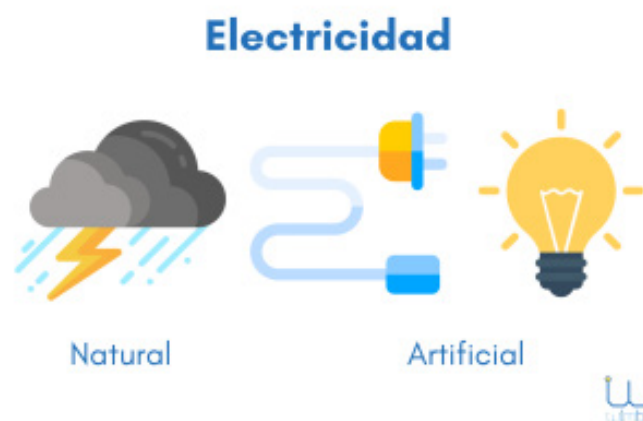
Como se ve en la representación, el primer paso y el de mayor relevancia es la prevención. Esto implica que la ciudadanía tiene un rol activo como consumidora de tecnología. Es importante priorizar la reparación en lugar de la adquisición de nuevos equipos. También disponerlos de forma adecuada, separando correctamente los residuos y llevándolos a las estaciones o sectores preparados para su recepción.

Módulo 2

Bienvenidos y bienvenidas al segundo módulo del curso. En este espacio abordaremos contenidos básicos de electricidad y electrónica. En nuestros hogares y trabajamos empleamos artefactos eléctricos a diario pero ¿Qué es la electricidad? ¿Qué magnitudes la representan y cómo podemos medirlas?. Los circuitos electrónicos por su parte son el corazón de la tecnología de automatización, los cuales permiten integrar la información de entrada y salida para llevar a cabo las acciones programadas.

1. Electricidad

La electricidad es una forma de energía que depende de la carga eléctrica de los cuerpos. Se trata de una propiedad fundamental de la materia y se genera por la atracción o la repulsión de sus partes.



La electricidad puede generarse desde fuentes naturales como las tormentas eléctricas o por dispositivos artificiales

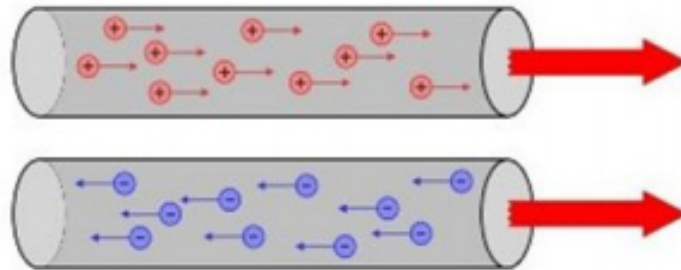
Todos los cuerpos poseen una carga. Las mismas pueden ser positivas o negativas, y son capaces de pasar de un cuerpo a otro. Un cuerpo estará cargado positivamente cuando tenga más carga positiva que negativa. Por el contrario, un cuerpo estará cargado negativamente cuando tenga más carga negativa que positiva. También existe la posibilidad de un cuerpo neutro, que es aquel que tiene la misma cantidad de carga positiva que negativa.

La carga positiva se llama protones y la carga negativa recibe el nombre de electrones, luego:

- Carga Positiva = Más protones que electrones.
- Carga Negativa = Más electrones que protones.
- Carga Neutra = La misma proporción de Protones y Electrones.

1.1 Corriente eléctrica

La corriente eléctrica es la cantidad de carga eléctrica que fluye a través de un conductor en una unidad de tiempo. En la práctica, podemos entenderla como la cantidad de electrones que se trasladan a través de un cable en un determinado tiempo. La intensidad de la corriente eléctrica se mide en Amperios.



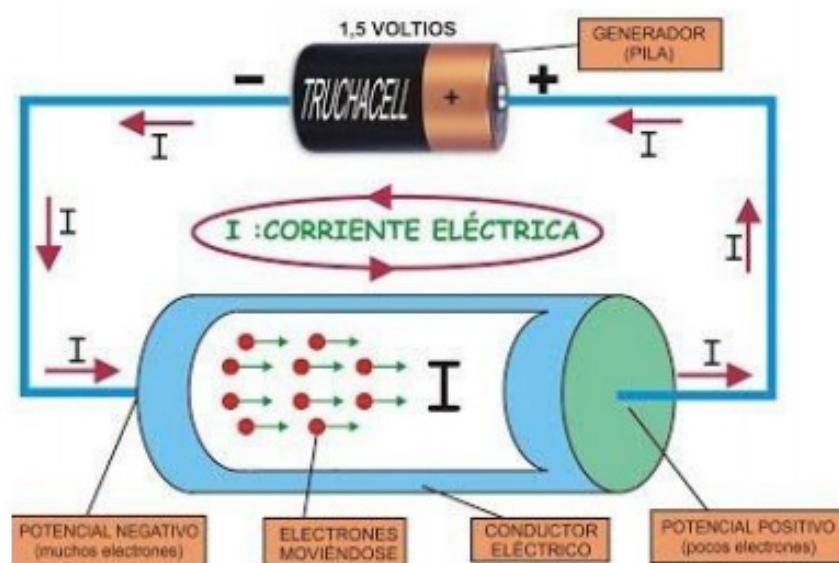
Representación de la corriente como movimiento de cargas positivas o negativas.

- Magnitud: Propiedad de los cuerpos que se puede medir y representar numéricamente, por ejemplo: masa, temperatura, volumen, entre otros.
- Conductor: Materiales que no oponen resistencia al paso de la corriente eléctrica, por ejemplo: oro, cobre, plata, hierro, grafito, soluciones salinas, entre otros.

El transporte de la electricidad puede ser de dos tipos:

- Corrientes de Cargas Positivas: Circulan Protones.
- Corriente de Cargas Negativas: Circulan Electrones.

Entonces, la corriente eléctrica es el movimiento de las cargas a través de un material conductor, la cual debe circular por un circuito cerrado.



Esquema de un circuito eléctrico. Con la letra I se identifica la corriente eléctrica y su sentido de circulación por el material conductor.

Pero... ¿Qué es el circuito eléctrico? Se define como una especie de “camino” por el que pasa la corriente. El inicio del camino será la fuente, mientras que el final del camino será la carga o aquello que necesite consumir electricidad. Pueden ser objetos de lo más cotidianos, por ejemplo, una lámpara.

2. Conceptos fundamentales de la electrónica

2.1 Magnitudes eléctricas

Para comprender el funcionamiento de circuitos eléctricos y electrónicos necesitamos conocer las magnitudes eléctricas que los caracterizan y saber cómo medirlas (por ej. con un polímetro). Las magnitudes eléctricas que veremos son: voltaje, resistencia e intensidad.

Voltaje

Es la fuerza eléctrica con que son empujados los electrones a través de un conductor. El voltaje se mide en Voltios (V).

La pila o batería brinda la energía necesaria para que las cargas eléctricas circulen por un circuito. Cuando hablamos de diferencia de potencia nos referimos a la diferencia de energía por unidad de carga entre dos puntos de un circuito.

Resistencia

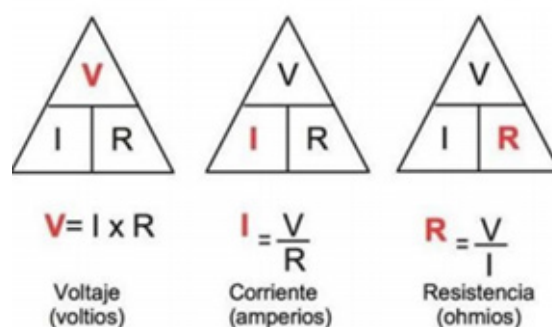
La resistencia eléctrica (R) indica la oposición que presentan los conductores al paso de la corriente, se mide en Ohmios (Ω).

Intensidad

La intensidad de la corriente (I) es la cantidad de electricidad o carga eléctrica (Q) que circula por un circuito en la unidad de tiempo (t). Para denominar la Intensidad se utiliza la letra I y su unidad es el Amperio (A).

2.2 Ley de Ohm

La intensidad de corriente que atraviesa un circuito es directamente proporcional al voltaje o tensión del mismo e inversamente proporcional a la resistencia que presenta. En forma de fracción se expresa de la siguiente manera:



3. Componentes electrónicos de un circuito

Un componente electrónico es un dispositivo que forma parte de un circuito electrónico. Se suelen encapsular, generalmente en un material cerámico, metálico o plástico, y terminar en dos o más terminales o patillas metálicas.

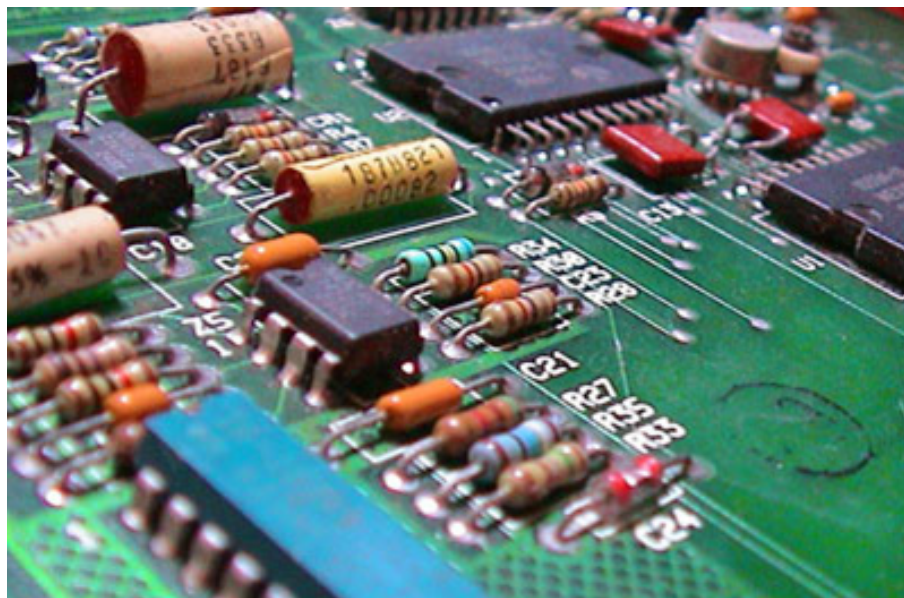


Foto de un circuito electrónico. Se observan resistencias, microprocesadores (piezas prismáticas) y capacitores (cilíndricos).

Los elementos se traducen en símbolos para representarlos de manera simplificada en planos o esquemas.

Interruptor			Conmutador unipolar		
Pulsadores NA - NC			Conmutador bipolar		
Microinterruptor			Relé		

Tabla demostrativa con componentes y sus símbolos 1



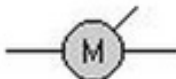
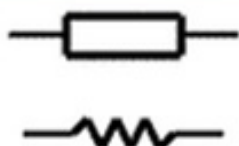






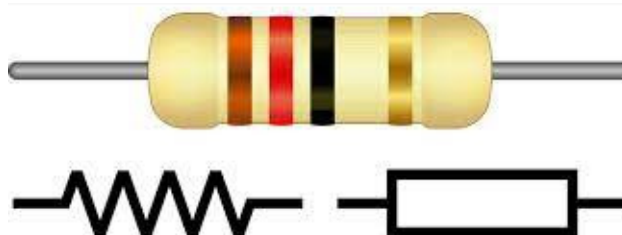
Lámpara	Diodo LED	Zumbador	Motor	Resistencia	Altavoz
					
					

Tabla demostrativa con componentes y sus símbolos 2

A continuación se detallan los elementos más relevantes para el desarrollo del curso.

Resistencia eléctrica

Se denomina resistencia o resistor al componente electrónico diseñado para introducir una resistencia eléctrica determinada entre dos puntos de un circuito eléctrico. Es un material formado por carbón y otros elementos resistivos que producen una oposición parcial al paso de la corriente.



Representaciones de una resistencia

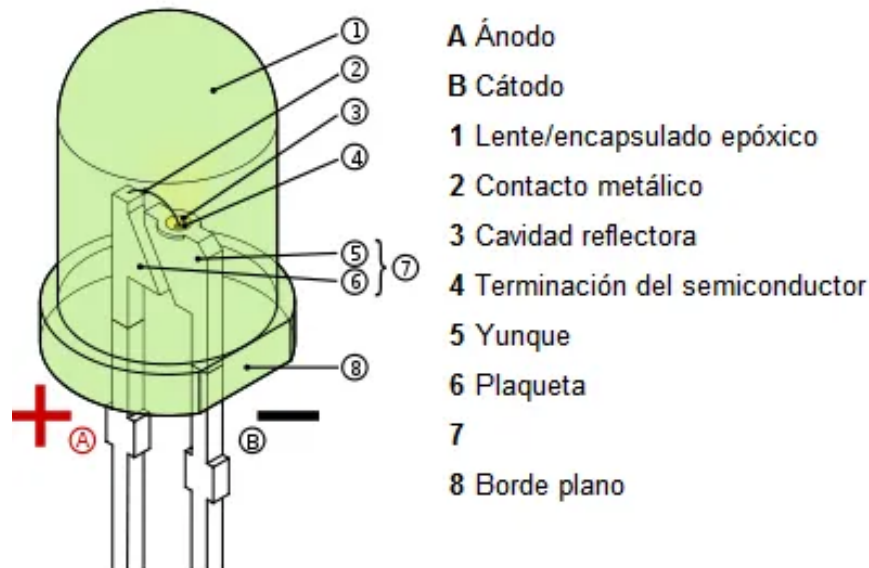
Diodo LED

La palabra LED viene del inglés Light Emitting Diode que traducido al español es Diodo Emisor de Luz.

Los diodos son componentes electrónicos que permiten el paso de la corriente en un solo sentido (como si fuera un interruptor abierto). Cuando la corriente pasa por el diodo, este emite luz. Cuando se conecta un diodo en el sentido que permite el paso de la corriente, se dice que está polarizado directamente.

Entonces la definición correcta sería: un LED es un diodo que cuando está polarizado directamente emite luz.

Los LEDs tienen dos terminales de conexión, siendo uno más largo llamado ánodo (positivo) y otro más corto, llamado cátodo (negativo). Para que pase la corriente y emita luz se deben conectar el terminal largo al polo positivo y el corto al negativo. En caso contrario la corriente no pasará y no emitirá luz.



Detalle de un diodo LED

Interruptor

Un interruptor eléctrico es un dispositivo que permite desviar o interrumpir el curso de una corriente eléctrica. En el mundo moderno sus tipos y aplicaciones son innumerables, desde un simple interruptor que apaga o enciende un foco de una casa, hasta un complicado selector de transferencia automático de múltiples capas, controlado por computadora.



Imagen de un interruptor

Inductores

Los inductores son un componente que almacena energía a través de campo magnético. Concretamente, induce un campo magnético cuando es atravesado por una corriente. También se les llama bobinas o solenoides.

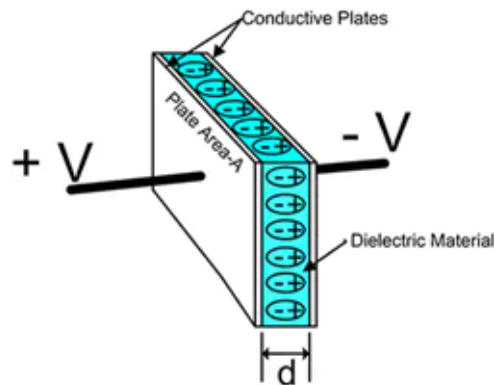
En principio, cualquier conductor podría usarse para construir una bobina. Se elabora enrollando alambre conductor en círculos, dando varias vueltas, de modo de formar un helicoide. Para evitar que el alambre enrollado entre en cortocircuito al hacer contacto consigo mismo al enrollarse, en la confección del inductor se emplea alambre esmaltado. Cada vuelta que el alambre efectúa se llama espira.



Bobina inductora

Capacitores

Los capacitores son dispositivos de almacenamiento de energía en forma de campo eléctrico que son esenciales para los circuitos electrónicos tanto analógicos como digitales. Se utilizan en la sincronización para la creación y formación de formas de onda, el bloqueo de la corriente continua y el acoplamiento de señales de corriente alterna; el filtrado y el suavizado y, por supuesto, el almacenamiento de energía.



Representación de un capacitor. Está conformado por placas conductoras separadas por un material dieléctrico. Se induce una diferencia de potencial cuando se conecta a una fuente.

Protoboard

Una placa de pruebas o placa de inserción (en inglés protoboard o breadboard) es un tablero con orificios que se encuentran conectados eléctricamente entre sí de manera interna, habitualmente siguiendo patrones de líneas, en el cual se pueden insertar componentes electrónicos y cables para el armado y prototipado de circuitos electrónicos y sistemas similares.

¿Cómo funciona una protoboard?

Una protoboard o placa de pruebas es una herramienta que permite el ensamblaje de circuitos electrónicos sin usar soldador. Fue creada con la idea de permitir practicar prototipos sin tener que fijar los componentes. Por tanto, facilita llevar a cabo el primer montaje y la puesta en funcionamiento de los prototipos de diseños dentro de la electrónica. El uso principal es crear, probar y depurar rápidamente circuitos electrónicos.

Son básicamente una tableta o placa hecha en resina o plástico resistentes al calor con numerosas perforaciones. Internamente posee láminas metálicas para sujetar y conectar los hilos de conexión con los terminales de los componentes. La conexión de las láminas así como su disposición dependen del tipo y tamaño de la protoboard. En cuanto a su estructura, los agujeros que la componen se conectan internamente. Por eso es posible interconectar diversos elementos con tan solo pinchar la placa. Se divide en filas y columnas lo que facilita su uso, siempre y cuando se conozca cómo conectarlas.

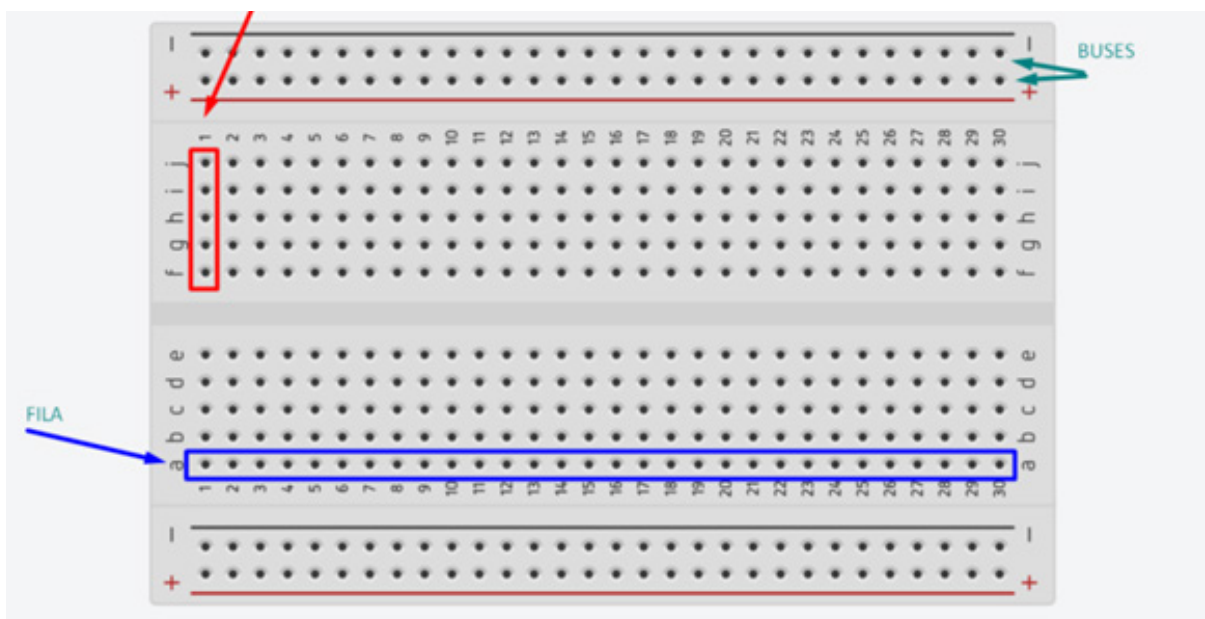


Imagen de una placa de prueba. Se señalan las filas, buses y columnas

La manera como funciona una protoboard tiene que ver con las partes que la componen, las cuales sin importar la cantidad de orificios son:

- Buses: están en los extremos. Los buses de voltaje o positivos son los rojos y los buses de tierra o negativos son los azules. Entre ellos no existe conexión. Sirven para conectar las fuentes de poder.
- Canal central: es la que se usa para ubicar los circuitos integrados.
- Pistas: se encuentran en el centro de la placa, se conducen y representan de acuerdo a las líneas rosadas.

Fuentes de alimentación

Una fuente de alimentación es un dispositivo utilizado para alimentar los circuitos de los aparatos electrónicos.

Transforma la corriente alterna en corriente continua y regula o cambia la tensión de salida a unos valores determinados. Por ejemplo, una fuente de alimentación puede conectarse en la entrada a 220V en corriente alterna (enchufe normal de una vivienda) y la transforma en corriente continua de 9V a la salida.

Para más detalles les recomendamos ver los videos de la videoteca correspondientes a este módulo.

Módulo 3

Sensores y actuadores

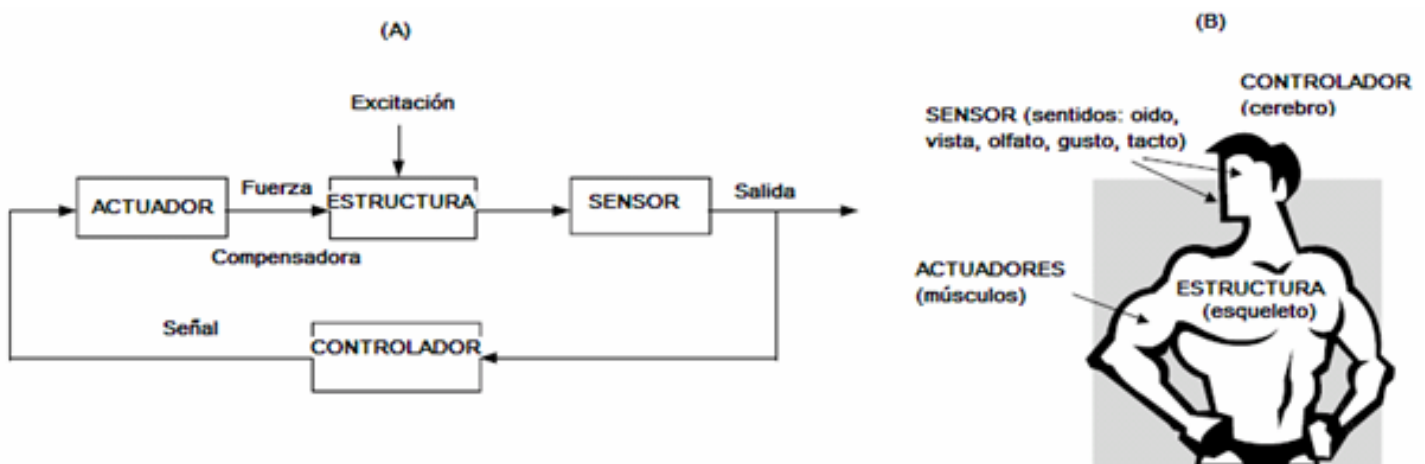
Hasta el momento conocimos y observamos diferentes tipos de circuitos eléctricos y sus componentes, los cuales se integran para lograr ciertos objetivos.

Es esperable que estos objetivos estén puestos al servicio de las personas o procesos industriales, por lo que necesariamente deben estar en contacto con el mundo exterior. Esa “conexión” con el mundo exterior en la electrónica se realiza a través de sensores y actuadores, los cuales absorben los datos del medio ambiente y luego actúan sobre él.

Según Edgar Morín, un sistema es “una unidad global organizada de interrelaciones entre elementos, acciones o individuos”.

Este concepto se puede aplicar en diferentes áreas. Es indudable que el ser humano es una unidad organizada ya que tiene su propia disposición de interrelaciones entre sus elementos. Todo sistema necesita nutrirse de elementos del exterior y proveer mecanismos para actuar en respuesta a ellos.

En los sistemas de control automatizados los sensores son los sentidos del sistema: le proporcionan información sobre lo que está ocurriendo. Los actuadores por su parte son las manos del sistema de control: le permiten modificar lo que ocurre en el entorno.



(A) Esquema básico de un sistema de control activo.

(B) Analogía con el cuerpo humano (De La Cruz, 2003).

Sensores

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas del mundo 'real' y entregarlas al sistema de control para que las 'entienda', pueda procesarlas y tomar decisiones.

En este sentido los sensores nos permiten conocer el valor de las variables físicas que participan en el proceso y convertirlas en señales eléctricas.

Existe gran cantidad de sensores para medidas de todo tipo y por tanto, se pueden clasificar de muchas formas distintas:

Según el tipo de salida que proporcionan:

- **Analógicos:** entregan una salida de nivel variable en función del parámetro que midan. Por ejemplo, un sensor de temperatura de -20° a $+50^{\circ}$ con salida 0-10V. Funciona de manera similar a la de un termómetro: en la medida que aumenta la temperatura, el termómetro dilata más el mercurio y hace subir el líquido, indicando de manera indirecta la temperatura. Del mismo modo, en la medida que aumente la temperatura, estos dispositivos generarán una mayor salida de voltaje, de manera proporcional.



Sensor analógico de temperatura TMP36.

- **Binarios:** entregan un nivel 'todo' o 'nada' (1/0). Como por ejemplo el estado de una puerta, la cual puede estar abierta o cerrada. Solo tienen dos posibles valores de salida.

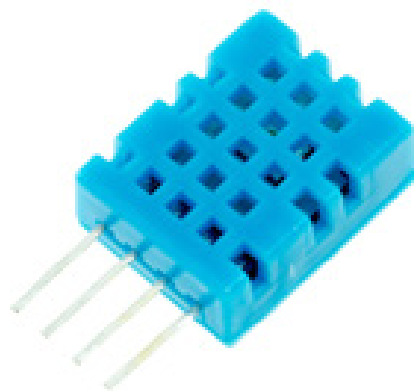
En el siguiente ejemplo, mientras que la puerta se encuentre cerrada el sensor tendrá un valor 0 (cerrado) y cuando se abra cambiará a 1. No importa si la puerta está solo "un poquito" abierta o totalmente, apenas deje de estar cerrada se activará el sensor.



Sensor binario

- **Digitales:** dan la información relativa a la medida con un protocolo de comunicaciones específico que el fabricante facilita: por ejemplo el sensor de temperatura y humedad DHT-11.

A diferencia de los analógicos no entregan una salida proporcional de energía de acuerdo a lo que mida, sino que directamente entregan el valor numérico de la medición (por ejemplo: 25° centígrados). La conversión de lo sensado se hace internamente en el dispositivo, lo cual asume una cierta “inteligencia” adicional.



Sensor de temperatura y humedad DHT-11

Según su estructura interna, los sensores pueden ser:

- **Pasivos:** son aquellos que generan señales representativas de las magnitudes a medir por intermedio de una fuente auxiliar. Ejemplo: sensores de parámetros variables (de resistencia variable, de capacidad variable, de inductancia variable).



Sensor infrarrojo de alarma

Según su estructura interna, los sensores pueden ser:

- **Activos:** son aquellos que generan señales representativas de las magnitudes a medir en forma autónoma, sin requerir de fuente alguna de alimentación. Ejemplo: sensores piezoeléctricos, fotovoltaicos, termoelectrónicos, electroquímicos, magnetoeléctricos. Por ejemplo: los sensores fotovoltaicos, reaccionan a la luz con un valor proporcional al que reciben. Su salida se genera independientemente de alguna otra fuente externa, solo se nutre del sol.



Sensor fotovoltaico Solar-Log™

Sensores activos o generadores de señal:

Son aquellos que generan señales representativas de las magnitudes a medir en forma autónoma, sin requerir de fuente alguna de alimentación. Ejemplo: sensores piezoeléctricos, fotovoltaicos, termoeléctricos, electroquímicos y magnetoeléctricos.

Según el tipo de parámetros que son capaces de detectar:

- **Mecánicos:** detectan parámetros relacionados con acciones mecánicas, contactos, aceleración, etc.



Sensor de precisión. Mide la distancia entre dos extremos.

- **Ambientales:** miden temperatura, humedad, pluviometría, velocidad del viento, etc.



Sensor de pluviometría

- **Químicos:** miden niveles de CO₂, de oxígeno, contaminación en el aire, azúcar en sangre, etc.



Sensor continuo de glucosa en sangre

Actuadores

Los actuadores son los dispositivos que permiten al sistema de control ‘actuar’ sobre el ‘mundo real’ para realizar las acciones deseadas.

A diferencia de los sensores, los actuadores o periféricos son los encargados de transformar la energía eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un elemento externo. Lo análogo en el cuerpo humano son los músculos y extremidades, que se encargan de realizar las acciones que el cerebro le indica.

Un porcentaje muy elevado de actuadores solo tienen dos estados: marcha y paro, o abrir y cerrar. Estos actuadores se manejan mediante señales binarias 0 / 1.



Actuadores binarios

Otros actuadores requieren valores analógicos para activar su salida. Esto quiere decir que necesitan la graduación de su activación (se pueden “activar” poco, mucho o en su totalidad)

Un ejemplo es la velocidad de rotación de un motor, donde en función de la presión en un pedal, esta se transforma en una mayor velocidad de giro.



Actuadores analógicos

Los **periféricos** se podrían considerar como un caso particular de actuadores. Periférico es la denominación genérica para designar al aparato o dispositivo auxiliar e independiente conectado a la unidad central de procesamiento, o en este caso a Arduino. Se consideran periféricos a las unidades o dispositivos de hardware a través de los cuales Arduino se comunica con el exterior, y también a los sistemas que almacenan o archivan la información, sirviendo de memoria auxiliar de la memoria principal.

Por ejemplo:

- Pantallas LCD
- Teclados
- Memorias externas
- Cámaras
- Micrófonos
- Impresoras
- Pantalla táctil
- Displays numéricos.
- Zumbadores.
- Indicadores luminosos

Ejemplo de Actuadores y periféricos:

- [Sacar por TV datos de Arduino, librería TV OUT](#)

Links de referencia

- https://bookdown.org/alberto_brunete/intro_automatica/sensoractuador.html
- http://www.micronica.es/files/pdfs/SIHD/SIHD_Sens_Actu_EC.pdf
- <https://aprendiendoarduino.wordpress.com/2016/12/18/sensores-y-actuadores/>

Referencia Wiring/Arduino/Reduino

Referencias de programación utilizando Wiring/Arduino/Reduino

Agradecimientos

Este documento se construyó utilizando las referencias oficiales y repositorios de código abiertos de los proyectos Wiring, Arduino y Reduino. Un especial agradecimiento para autores y colaboradores de las respectivas comunidades, gracias a sus contribuciones la robótica libre está al alcance de más personas que nunca.

Ref: <http://wiring.org.co/hardware/es/compare.html>

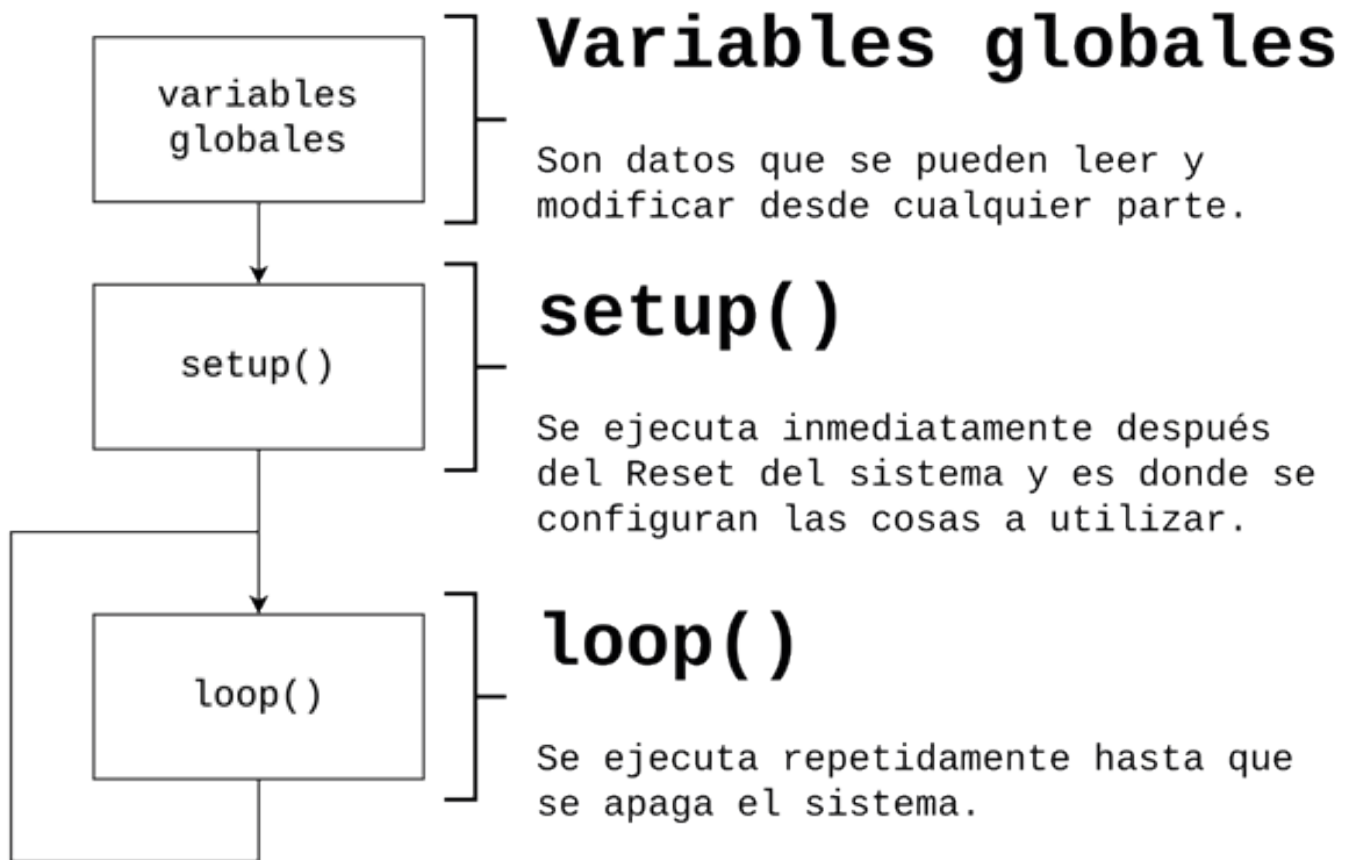
Ref: <http://wiring.org.co/reference/es/>

¡Hola Mundo!

Siempre que queramos aprender una nueva plataforma programable, ya sea porque vamos a utilizar un nuevo lenguaje o placa programable, o porque actualizamos algún componente de software o hardware, es recomendable correr un simple programa de principio a fin para asegurarnos que todo está funcionando armoniosamente.

Comúnmente este programa se denomina “Hola Mundo”, y puede consistir en escribir la frase Hola Mundo en un display directamente o a través de algún puerto de comunicación. En particular las placas “open Hardware” utilizadas para robótica educativa. incluyen una luz led para cumplir este propósito y entonces el hola mundo se concreta al encender o hacer parpadear dicha luz led.

Estructura de todo programa



Datos y variables Programando variables

Se almacenan en la memoria RAM en bloques de 8 bits, pueden tener signo o ser no signadas.

Algunos tipos de variables son:

`int32_t`: -2147483648 a +2147483647
`uint32_t`: 0 a 4294967295

la `//` da comienzo a una línea de comentarios

```
// esto es una variable con valor inicial 1
uint32_t contador = 1;
```

```
// esto es una variable con valor inicial -5
int32_t temperatura = -5;
```

Estructura

Nombre	loop()
Ejemplos	<pre>void setup() { pinMode(0, OUTPUT); } void loop() { digitalWrite(0, HIGH); }</pre>
Descripción	Continuamente ejecuta las líneas de código dentro, bloqueándose hasta que el programa es detenido. La función loop() es usada en conjunción con setup() . El número de veces que loop() se ejecuta por segundo puede ser controlada con las funciones delay() y delayMicroseconds() .
Sintaxis	<pre>loop() { declaraciones }</pre>
Parámetros	<div> <div>declaraciones</div> <div>Una secuencia de declaraciones para ser ejecutados una y otra vez</div> </div>
Retorna	Ninguno

Nombre	setup()
--------	----------

Ejemplos	<pre>void setup() { pinMode(0, OUTPUT); Serial.begin(9600); } void loop() { Serial.print(":"); delay(1000); }</pre>
Descripción	<p>Función llamada una vez cuando el programa empieza a ser ejecutado. Suele definir el ambiente inicial tal como el estado de los pines (INPUT o OUTPUT), inicializar el puerto serial, etc. antes que loop() empiece a ejecutarse. Las variables declaradas en setup() no son accesibles desde loop().</p>
Sintaxis	<pre>void setup() { declaraciones }</pre>
Parámetros	<div>declaraciones</div> <div>Cualquier declaración válida</div>

Nombre	= (asignación)
Ejemplos	<pre>int a; a = 30; // Asigna el valor 30 a la variable "a" a = a + 40; // Asigna el valor 70 a la variable "a"</pre>

Descripción	Asigna un valor a una variable. El signo “=” no significa “igual a”, pero es usado para colocar un dato en una variable. El operador “=” es formalmente llamado operador de asignación. Existen diferentes tipos de variables (int, floats, chars, etc.) y el operador de asignación puede solamente asignar valores que son del mismo tipo de la variable a la que le asignan. Por ejemplo, si la variable es de tipo int , el valor debe ser también un int .
Sintaxis	var = valor
Parámetros	<p>var Cualquier nombre de variable válido</p> <p>valor Cualquier valor del mismo tipo que la variable. Por ejemplo, si la variable es de tipo “int”, el valor debe ser int</p>

Nombre	, (comma)
Ejemplos	<pre>// Coma usada para separar una lista de declaraciones de variables int a=20, b=30, c=80; // Coma usada para separar una lista de valores asignados a un arreglo int d[] = { 20, 60, 80 }; // Coma usada para separar una lista de // parámetros pasados a una función digitalWrite(a, LOW); analogWrite(4, 255);</pre>
Descripción	Separa los parámetros en la llamada a una función o elementos durante una asignación.

Sintaxis	valor1, ..., valorN
Parámetros	valor1, ..., valorN cualquiera int, float, byte, boolean, char, etc.

Nombre	// (comentario)
Ejemplos	// Inicializa el pin 0 como OUTPUT // Asigna el valor HIGH al pin 0 pinMode(0, OUTPUT); // Inicializa el pin 0 como OUTPUT digitalWrite(0, HIGH); // Asigna el valor HIGH al pin 0
Descripción	Notas explicativas dentro del código. Los comentarios son usados para recordarse a sí mismo y para informar a los demás acerca de una función en su programa. Comentarios de una sola línea son marcados con los caracteres //. Los comentarios son ignorados por el compilador.
Sintaxis	// comentario
Parámetros	comentario Cualquier secuencia de caracteres

Nombre	{ } (llaves)
Ejemplos	<code>int a[] = { 5, 20, 25, 45, 70 };</code>
Descripción	Define el inicio y fin del bloque de una función y de bloques de instrucciones de estructuras como el for() y el if() . Los corchetes son también usados para definir los valores iniciales en la declaración de un arreglo.
Sintaxis	<code>{ declaración }</code> <code>{ ele0, ..., eleN }</code>
Parámetros	<code>declaración</code> Cualquier secuencia de instrucciones válida <code>ele0 ... eleN</code> Lista de elementos separados por coma

Nombre	<code>#define</code>
Ejemplos	<code>// reemplaza COUNT con el número 1000</code> <code>// no usa memoria para una variable</code> <code>#define COUNT 1000</code> <code>int i = 0;</code> <code>void setup()</code>

	<pre> { Serial.begin(9600); } void loop() { if (i < COUNT) { Serial.print("i = "); Serial.println(i); } i = i+1; // i har&aacute; overflow a -32768 entialmente // imprimir&aacute; nuevamente cuando pase } // reemplaza MYLED con el n&uacute;mero 8 // no usa memoria para una variable #define MYLED 8 void setup() { pinMode(MYLED, OUTPUT); } void loop() { digitalWrite(MYLED, HIGH); delay(100); digitalWrite(MYLED, LOW); delay(100); } </pre>
Descripción	<p>La directiva #define direcciona al preprocesador para reemplazar todas las ocurrencias de un identificador con los correspondientes tokens de reemplazo. Existen macros como objetos y macros como funciones. La definición de una macro como objeto reemplaza un identificador sencillo con los tokens de reemplazo. La siguiente definición de macro como objeto causa que el preprocesador reemplace todas las apariciones subsecuentes del identificador COUNT con el token 1000 como se muestra en el primer ejemplo.</p> <p>#define COUNT 1000. Nota IMPORTANTE: La directiva #define hace reemplazo directo del identificador en el código del programa, este no es una variable. Los macros más complejos como funciones van más allá de este tutorial. Para mayor información sobre el tema refiérase a “function-like macro definitions in C/C++”.</p>

Sintaxis	<code>#define</code> identifier replacement
Parámetros	<p>identifier el token a ser reemplazado</p> <p>replacement el valor con el que se reemplaza</p>

Nombre	<code>#include</code>
Ejemplos	<pre>#include "LiquidCrystal.h" // incluye la librería LiquidCrystal en el programa LiquidCrystal myDisplay = LiquidCrystal(8,9,10,2); int a = 0; void setup() { } void loop() { myDisplay.clear(); myDisplay.home(); myDisplay.print("Variable a es: "); myDisplay.setCursor(16, 0); myDisplay.print(a); a = a + 1; delay(200); }</pre>
Descripción	<p>La directiva <code>#include</code> es usada para importar una librería externa en un programa. Una directiva <code>#include</code> es agregada automáticamente a un programa después de seleccionar una librería con el menú Sketch/Import Library. Note que la directiva tiene una sintaxis especial, a diferencia de otros comandos de Wiring esta no finaliza con un punto y coma. En el ambiente de Wiring es posible usar <code>" "</code> o <code>< ></code> para incluir un archivo de librería.</p>

Sintaxis	#include
----------	----------

Estructura de todo programa

Nombre	Serial
Ejemplos	<pre>int val; void setup() { Serial.begin(9600); // Inicializa el puerto serial en 9600 baud } void loop() { if (Serial.available() > 0) // Si los datos est&aacute;n disponibles para leer { val = Serial.read(); // los lee y los almacena en 'val' } analogWrite(0, val); }</pre>
Descripción	<p>El puerto serial Serial de Wiring permite fácilmente leer o escribir datos hacia y desde un dispositivo externo. Permite comunicar dos máquinas y da la flexibilidad de hacer sus propios dispositivos y usarlos como entrada/salida de Wiring. El puerto serial es un puerto de nueve pines que solía existir en la mayoría de PCs y puede ser emulado a través de una adaptador serial USB. El hardware de Wiring tiene dos puertos seriales internos (hardware) llamados Serial y Serial1. El puerto Serial está disponible a través del conector USB en la tarjeta Wiring. El puerto Serial1 está disponible en los pines 2 (Rx) y 3 (Tx). Es posible acceder al puerto serial Serial a través de los pines. En las tarjetas Wiring v1 está disponible en los pines 32(Rx) y 33(Tx). En la tarjeta Wiring S está disponible en los pines 0(Rx) y 1(Tx). Las velocidades típicas son: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 y 115200 baudios.</p> <p>Uso del puerto serial Todas las plataformas: Si está usando un programa para revisar si el puerto serial está funcionando, este no estará disponible en Wiring. Eso significa que si está usando HyperTerminal o cualquier otro software para ver si su dispositivo serial está funcionando, es necesario salir de la aplicación antes de usar el puerto con Wiring, también si el puerto está en uso con Processing. Los puertos seriales solo pueden estar en uso de manera simultánea por una sola aplicación. El menú Tools » Serial Ports muestra los puertos disponibles.</p>

Sintaxis	Serial Serial1 Serial2 Serial3
Métodos	<p>begin() Abre el puerto serial para leer o escribir.</p> <p>read() Retorna un número entre 0 y 255 para el siguiente byte esperado por el buffer. Retorna -1 si no hay ningún byte, aunque debería estar en la primera búsqueda available() para ver si los datos están disponibles.</p> <p>write() Escribe un byte en el puerto serial.</p> <p>print() Escribe datos (int, float, byte, char, char[], números en base (DEC, BIN, OCT or HEX) o Strings en puerto serial.</p> <p>println() Funciona como el método print pero imprime un carácter new line cada vez que es llamada la función.</p> <p>available() Retorna el número de bytes disponibles.</p> <p>peek() Examina el siguiente byte que se encuentra disponible en el buffer del puerto serial. Este método no saca el byte del buffer. Retorna -1 si no hay datos disponibles.</p> <p>flush() Vacía el buffer del puerto serial. Posteriormente se llama read() y available() las cuales retornarán datos recibidos después del uso del comando flush().</p> <p>end() Cierra el puerto serial.</p>

Pin digital de Entrada/Salida

Nombre	pinMode()
Ejemplos	<pre> int inpin = 8; int outpin = 9; int val = 0; void setup() { pinMode(inpin, INPUT); pinMode(outpin, OUTPUT); } void loop() { val = digitalRead(inpin, HIGH); if (val == HIGH) { digitalWrite(outpin, HIGH); } else { digitalWrite(outpin, LOW); } } </pre>
Descripción	El método pinMode() asigna pin digital I/O especificado como INPUT o OUTPUT. Un pin I/O digital o binario puede tener dos posibles valores: HIGH o LOW. Es posible asignar o leer el valor de un pin digital I/O usando los métodos digitalWrite() y digitalRead() .
Sintaxis	pinMode (pin,valor)
Parámetros	<div>pin</div> <div>valor</div> <div>Número del pin</div> <div>INPUT o OUTPUT</div>
Retorna	Ninguno

Nombre	<code>digitalRead()</code>
Ejemplos	<pre> int inpin = 8; int outpin = 9; int val = 0; void setup() { pinMode(inpin, INPUT); pinMode(outpin, OUTPUT); } void loop() { val = digitalRead(inpin); if (val == HIGH) { digitalWrite(outpin, HIGH); } else { digitalWrite(outpin, LOW); } } </pre>
Descripción	El comando digitalRead() lee el estado (o valor) de un pin digital.
Sintaxis	<code>digitalRead(pin)</code>
Parámetros	pin int

Nombre	<code>digitalWrite()</code>
Ejemplos	<pre> int outpin = 0; void setup() { pinMode(outpin, OUTPUT); } void loop() { digitalWrite(outpin, HIGH); } </pre>

Descripción	El comando digitalWrite() escribe un valor a (o pone en un estado) un pin digital. Los valores o estados posibles son HIGH o LOW.
Sintaxis	digitalWrite (pin,valor)
Parámetros	pin int: El número del pin valor HIGH o LOW

Pin análogo de Entrada

Nombre	analogRead()
Ejemplos	<pre> int inpin = 0; int val = 0; void setup() { Serial.begin(9600); } void loop() { val = analogRead(inpin); // lee el valor de entrada del pin analogRead();logo 0 Serial.println(val); // escribe el valor al puerto serial } </pre>
Descripción	El comando analogRead() lee el valor de un pin análogo. Los valores posibles están en el rango 0-1023, donde 0 es 0 voltios y 1023 es 5 voltios.
Sintaxis	analogRead (pin)

Parámetros	pin int: número del pin análogo que se quiere leer
Retorna	int: el valor leído del pin análogo.

Pin PWM (Analógico) de Salida

Nombre	<code>analogWrite()</code>
Ejemplos	<pre>int outpin = 29; int val = 0; void setup() { } void loop() { analogWrite(outpin, val); // escribe un valor en el pin 29 PWM val = (val + 10) % 255; // incrementa el valor y lo mantiene en // el rango 0-255. }</pre>
Descripción	<p>El comando analogWrite() asigna el valor de un pin de salida PWM. Los valores posibles por defecto están en el rango 0-255 (ver el comando <code>setPWMResolution()</code>). En las tarjetas Wiring v1 los pines con capacidad PWM son: 29, 30, 31, 35, 36 y 37. En la tarjeta Wiring S los pines con capacidad PWM son: 4, 5, 6, 7, 19 y 20. Nota: <code>analogWrite</code> es un alias para el comando <code>PWMWrite()</code>. Usar analogWrite() en un pin sin capacidad PWM hace que el pin sea colocado en HIGH sin ningún otro efecto.</p>
Sintaxis	<code>analogWrite(pin,valor)</code>
Parámetros	pin int: El número del pin de salida PWM valor int: Un valor en el rango 0-255
Retorna	Ninguno

Tiempo

Nombre	delay()
Ejemplos	<pre> int pin = 0; int ledpin = 1; void setup() { pinMode(pin, INPUT); pinMode(ledpin, OUTPUT); } void loop() { if (pin == HIGH) { digitalWrite(ledpin, HIGH); } else { digitalWrite(ledpin, LOW); } delay(250); // Detiene el programa por 250 milisegundos } </pre>
Descripción	Obliga al programa a detenerse por un tiempo especificado. El tiempo de demora es especificado en milésimas de segundo. La llamada a la función delay(3000) detendrá el programa por tres segundos. Alias de delayMilliseconds() .
Sintaxis	delay(milisegundos)
Parámetros	<div> <div>milisegundos int:</div> <div>Especificado en milisegundos (hay 1000 milisegundos en 1 segundo)</div> </div>
Retorna	Ninguno

Nombre	millis()
--------	----------

Ejemplos	<pre> bool ledState=LOW; void setup() { Serial.begin(9600); pinMode(13,OUTPUT); } void loop() { currentTime=millis(); if(ledState){ interval=interval2; }else{ interval=interval1; } if((currentTime-previousTime)>interval){ previousTime=currentTime; ledState=!ledState; digitalWrite(13,!ledState); Serial.print(F("LED State : "));Serial.println(ledState); } } </pre>
Ref Web	https://www.luisllamas.es/multitarea-en-arduino-blink-sin-delay/
Descripción	Retorna el número de milisegundos(milésimas de segundo) desde que empieza la aplicación. Esta información es a menudo usada para secuencias de tiempo.
Sintaxis	<code>millis()</code>
Retorna	long

Lectura de pulsos (bloqueante)

Nombre	<code>pulseIn()</code>
Ejemplos	<pre> int inpin = 8; int val = 0; </pre>

	<pre>void setup() { pinMode(inpin, INPUT); Serial.begin(9600); } void loop() { val = pulseIn(inpin, HIGH); Serial.println(val); delay(100); }</pre>
Descripción	<p>El método pulseIn() retorna la longitud en microsegundos en un pin de entrada digital por un periodo de tiempo desde 10 milisegundos hasta 3 minutos. Puede ser usado para leer sensores que retornan una serie de pulsos como algunos acelerómetros o medidores de rango. Los usuarios pueden especificar el conteo en las transiciones de HIGH o LOW. El comando pulseIn detiene toda actividad mientras se ejecuta. Un descanso de 1 segundo (1000000 microsegundos) previene que el comando espere por siempre. Un descanso específico en microsegundos puede también ser determinado.</p>
Sintaxis	<pre>pulseIn(pin,transición) pulseIn(pin,transición,descanso)</pre>
Parámetros	<p>pin int: El pin usado para leer el pulso</p> <p>transición HIGH o LOW</p> <p>descanso int: Un valor de descanso en microsegundos</p>
Retorna	int

Referencia del lenguaje “se” 2.0.0 (se-v2r2b)

En esta versión se incorpora la librería “se.h” con funciones en castellano, destacase la función `hacerDesdeCada(ESPERA_DESDE, REPITE_CADA)` que posibilita hacer un guiño sin bloqueo (sin delay/esperar)

Referencia: <https://gitlab.com/ReDuino/SE/-/tree/2.0.0>

Valores

ENCENDER / ENCENDIDO / VERDADERO

APAGAR / APAGADO / FALSO

ACTUADOR / ACTUA

SENSOR

Variables

`declaroBoleano(nombre, VALOR_INICIAL);`

`alternarBoleano(nombre);`

`declaroEstados(nombre, val1, ...); / declararEstados(nombre, val1, ...)`

`declaroTexto(nombre, TEXTO); / declararTexto(nombre, TEXTO);`

`declaroNumero(nombre, VALOR_INICIAL); / declararNumero(nombre, VALOR_INICIAL);`

`declaroDecimal(nombre, VALOR_INICIAL); / declararDecimal(nombre, VALOR_INICIAL);`

`declaroEntero(nombre, VALOR_INICIAL); / declararEntero(nombre, VALOR_INICIAL);`

Actuador/Sensor digitales

`pataTipo(pata, tipo); / pataDigitalTipo(pata, tipo);`

`escribirPata(pata, valor); / escribirPataDigital(pata, valor);`

`leerPata(pata); / leerPataDigital(pata);`

Tiempo

`esperar(ESPERA_EN_MS);`

`hacerDesdeCada(ESPERA_DESDE, REPITE_CADA) { }`

Mensajes

pedirHablar();
encenderMensajeria();
enviarMensaje(MSG);
enviarEstado(estado);

Condicionales

si() { }
siNo() { }

Comparaciones

ES_IGUAL_A
ES_DISTINTO_A
ES_MAYOR_A
ES_MENOR_A
ES_MAYOR_IGUAL_A
ES_MENOR_IGUAL_A

Operaciones lógicas

Y
O

Laboratorio de Robótica Creativa

Materiales Kit de Robótica Básico

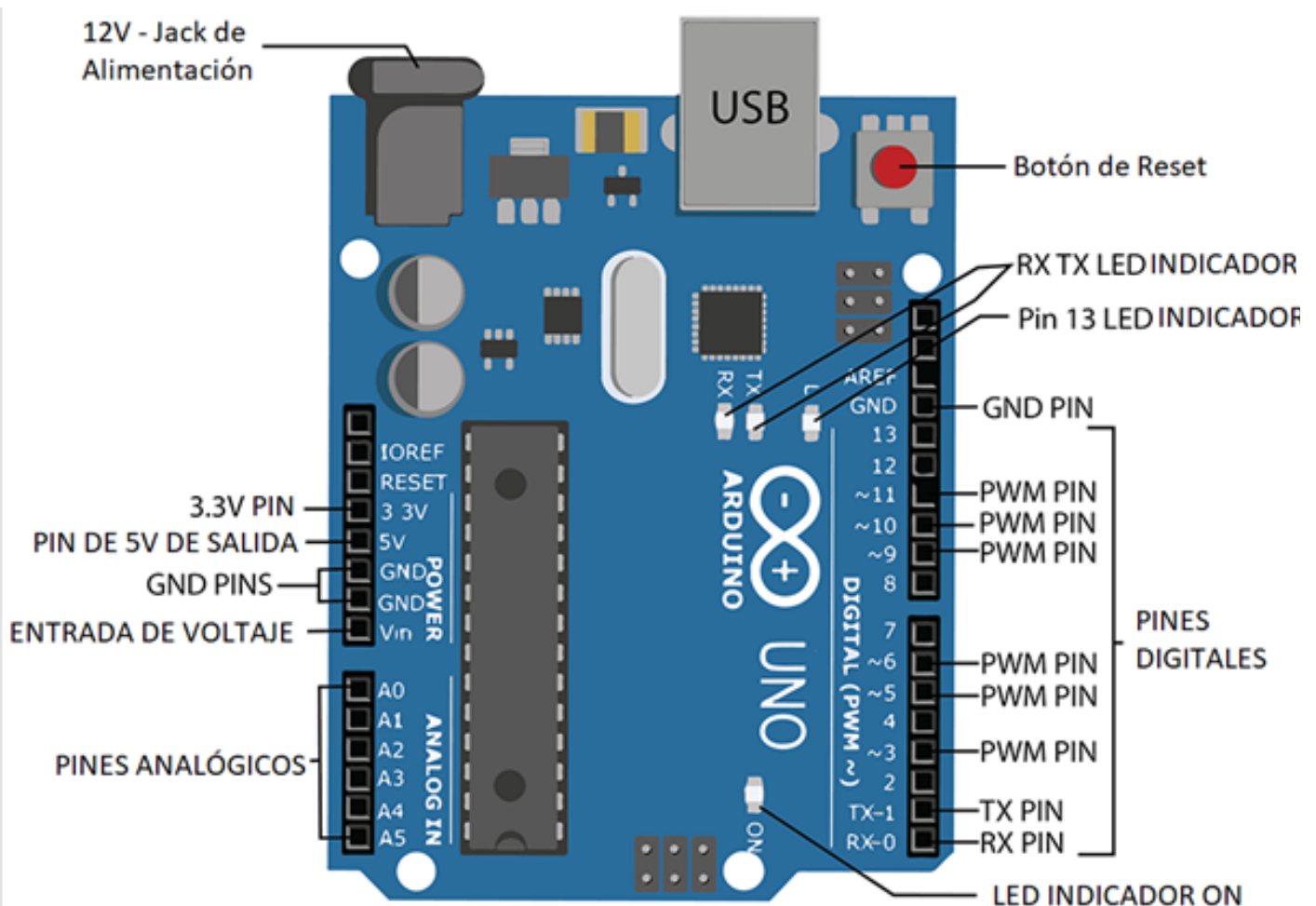
Referencias de materiales incluidos en el Kit de Robótica Básica

Listado general de materiales

El kit básico de robótica está compuesto por los siguientes materiales:

1. Placa Arduino UNO compatible
2. Actuador servomotor modelo SG90
3. Sensor de distancia HC-SR04
4. Accesorios varios como cables, protoboard, etc.

Plataforma de Hardware Libre Arduino



Variación de la velocidad de guiño de la baliza

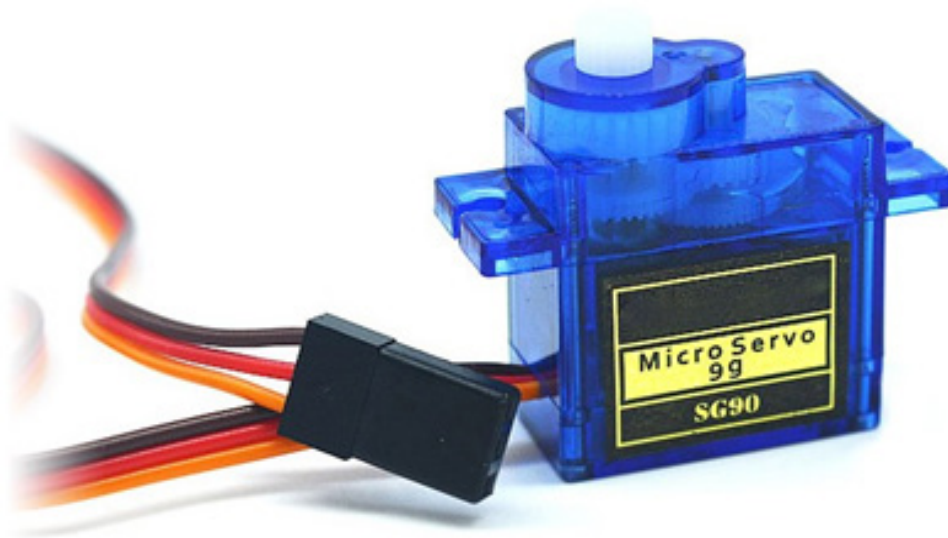
// <https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink>

// the setup once function runs when you press reset or power the board


```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Servomotor SG90



Verificación del servo

```
// https://docs.arduino.cc/learn/electronics/servo-motors
#include <Servo.h>
```

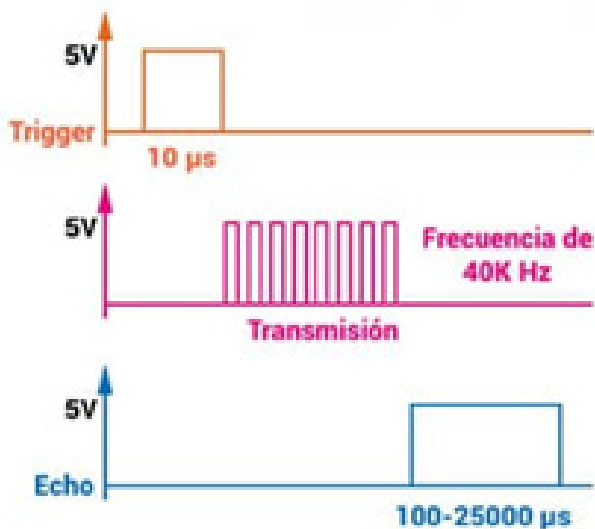
```
Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards
```

```
int pos = 0; // variable to store the servo position
```

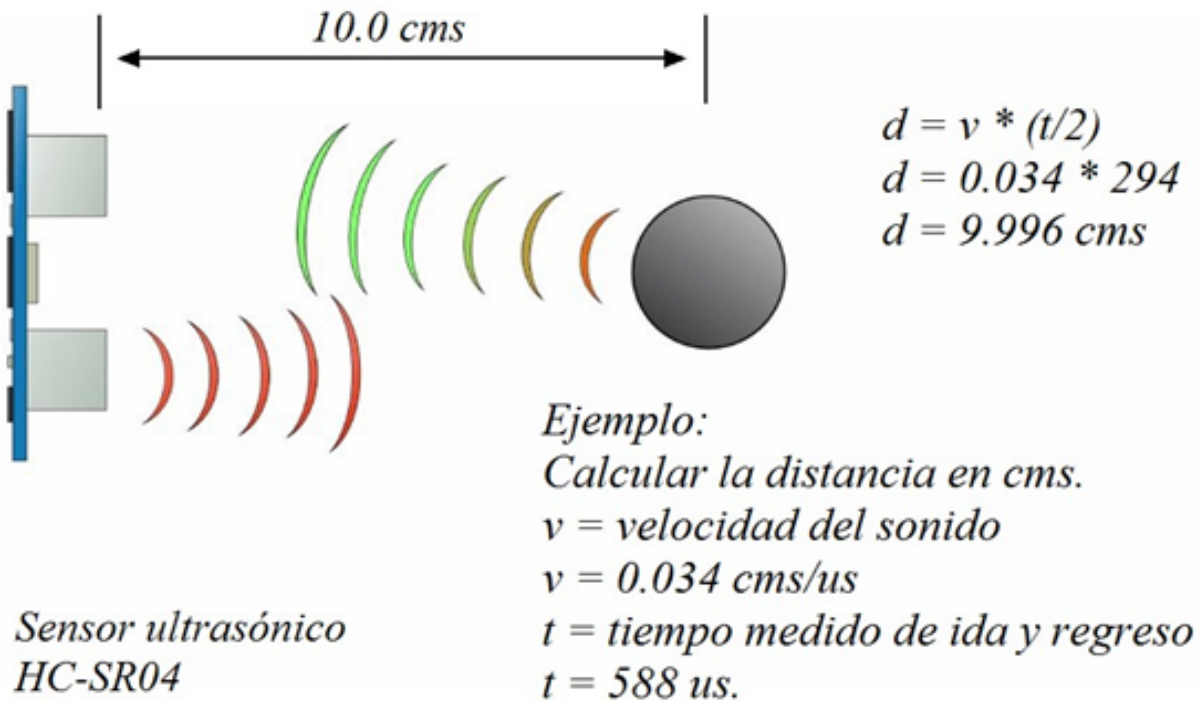
```
void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}
```

```
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);           // tell servo to go to position in variable 'pos'
    delay(15);                    // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);           // tell servo to go to position in variable 'pos'
    delay(15);                    // waits 15ms for the servo to reach the position
  }
}
```

Sensor ultrasonico de distancia HC-SR04



 **Vcc 5 V**
 **Trigger**
 **Echo**
 **GND**



Verificación del funcionamiento del sensor de distancia

// Using HC-SR04 Module

<https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6>

#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04

#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04

// defines variables

long duration; // variable for the duration of sound wave travel

int distance; // variable for the distance measurement

void setup() {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT

pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed

Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor

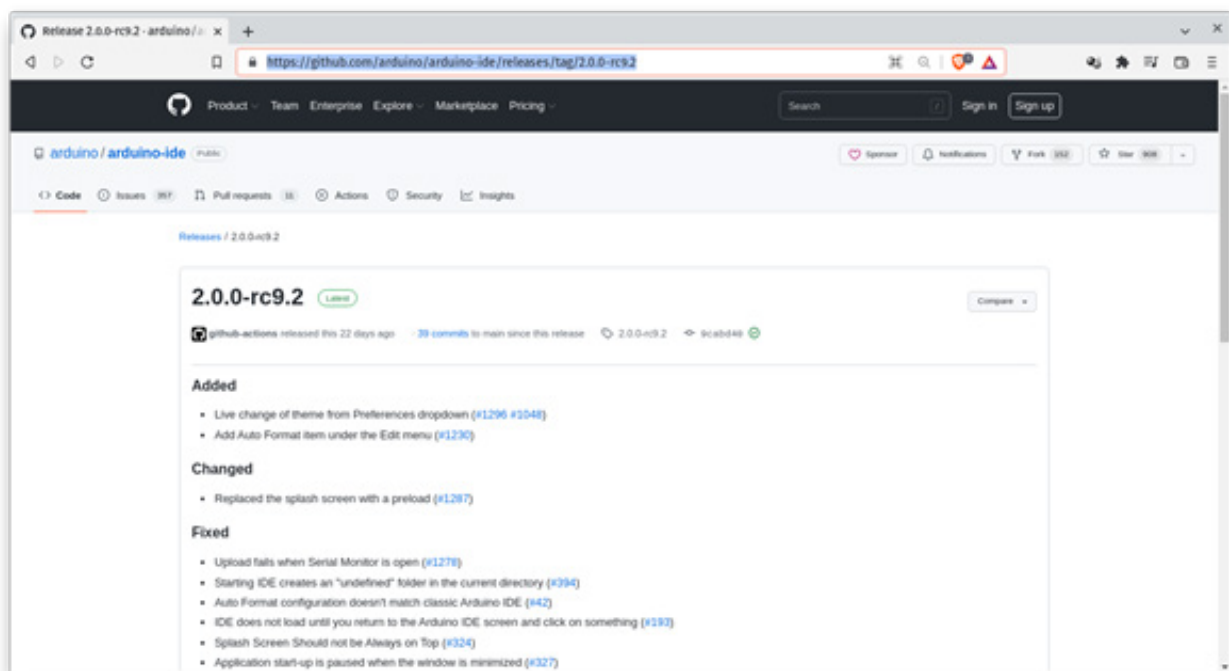
Serial.println("with Arduino UNO R3");

}

```
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distancia: ");
  Serial.print(distance);
  Serial.print(" cm, ");
  Serial.print("Tiempo: ");
  Serial.print(duration);
  Serial.println(" us");
  Serial delay(100); // espera 100ms
}
```

IDE de programación de Arduino en version 2.0-rc9.2

Ref: <https://github.com/arduino/arduino-ide/releases/tag/2.0.0-rc9.2>



Simulación

Ref: <https://wokwi.com/projects/341512499898614355?lang=es-ES>

Wokwi

sketch.js

Diagram

Serial Monitor

Library Manager

```

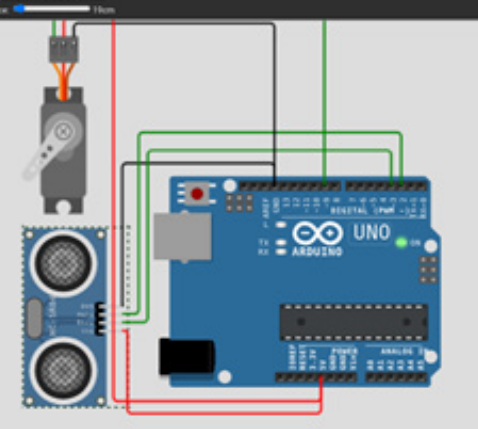
1 #include <Servo.h>
2 // https://wiki.arduino.cc/projects/shield-usb-17c455/arduino-sensor-hc-sr04-with-arduino-futurzel-321405
3 // https://wiki.arduino.cc/projects/shield-usb-17c455/arduino-sensor-hc-sr04-with-arduino-futurzel-321405
4
5 #define echoPin 2 // attach pin 2 of Arduino to pin Echo of HC-SR04
6 #define trigPin 3 //attach pin 3 of Arduino to pin Trig of HC-SR04
7
8 // Defines variables
9 long duration; // variable for the duration of sound wave travel
10 int distance; // variable for the distance measurement
11
12 Servo myservo; // create servo object to control a servo
13 // Twelve servo objects can be created on most boards
14
15 int pos = 0; // variable to store the servo position
16
17 void setup() {
18   pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
19   pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
20   Serial.begin(9600); // Serial Communication is starting with 9600 of baudrate speed
21   Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
22   Serial.println("With Arduino UNO R3");
23   myservo.attach(9); // attaches the servo on pin 9 to the servo object
24 }
25
26 void loop() {
27   // Clears the trigPin condition
28   digitalWrite(trigPin, LOW);
29   delayMicroseconds(2);
30   // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
31   digitalWrite(trigPin, HIGH);
32   delayMicroseconds(10);
33   // Reads the echoPin, returns the sound wave travel time in microseconds
34   duration = pulseIn(echoPin, HIGH, 20000);
35   // Calculating the distance
36   distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
37   // Displays the distance on the Serial Monitor
38   Serial.print("Distance: ");
39   Serial.print(distance);
40   Serial.print(" cm. ");
41   Serial.print(duration);
42   Serial.print(" us");
43   Serial.println();
44
45   for (pos = 0; pos <= 180; pos += 1) // goes from 0 degrees to 180 degrees
46     // in steps of 1 degree
47     myservo.write(pos); // tell servo to go to position in variable 'pos'
48     delay(150); // waits 150ms for the servo to reach the position
49
50   for (pos = 180; pos >= 0; pos -= 1) // goes from 180 degrees to 0 degrees
51     myservo.write(pos); // tell servo to go to position in variable 'pos'
52     delay(150); // waits 150ms for the servo to reach the position
53 }

```

Simulation

Editing Ultrasonic Distance Sensor

Distance: 18 cm



Distance: 18 cm, 1106 us

Distance: 18 cm, 1112 us

Distance: 18 cm, 1112 us

Distance: 18 cm, 1112 us

Distance: 18 cm, 1112 us

Distance: 18 cm, 1112 us

Distance: 18 cm, 1112 us