Electrónica y sistemas embebidos.

Una visión a nivel técnico y tecnológico

Luis Alberto Freire Sánchez

CIDE
EDITORIAL

Electrónica y sistemas embebidos

Una visión a nivel técnico y tecnológico

Electrónica y sistemas embebidos

Una visión a nivel técnico y tecnológico

Autor

Luis Alberto Freire Sánchez

Electrónica y sistemas embebidos. Una visión a nivel técnico y tecnológico

Centro de Investigación y Desarrollo Ecuador
Tel.: + (593) 04 2037524
http.:/www.cidecuador.org

ISBN: 978-9942-636-22-5

https://doi.org/10.33996/cide.ecuador.ES2636225

Impreso y hecho en Ecuador

Dirección editorial: Lic. Pedro Misacc Naranjo, Msc.
Coordinación técnica: Lic. María J. Delgado
Diseño gráfico: Lic. Danissa Colmenares
Diagramación: Lic. Alba Gil
Fecha de publicación: junio, 2023

Guayaquil - Ecuador

La presente obra fue evaluada por pares académicos experimentados en el área.

Catalogación en la Fuente

Electrónica y sistemas embebidos. Una visión a nivel técnico y tecnológico. Luis Alberto Freire Sánchez .--Ecuador: Editorial CIDE, 2023.

164 p.: incluye tablas, figuras; 14,8 x 21 cm.

ISBN 978-9942-636-22-5

1.- Ingenieria Eléctrica

Semblanza del Autor

LUIS ALBERTO FREIRE SÁNCHEZ

Ingeniero en Electrónica y Computación, Master en Ingeniería de Software y Sistemas Informáticos, Director de Investigación Instituto Superior Tecnológico San Gabriel.

luis_freire@sangabrielriobamba.edu.ec
https://orcid.org/0000-0003-3849-7326

Dedicatoria

Dedico este trabajo a:

Dios, por darnos fortaleza y ser luz que guía nuestro camino.

Mi querida esposa Magy y a nuestra hija Hannita, ahora que está iniciando sus pasos Dios la bendiga con amor y sabiduría para que nunca deje de aprender.

Mi padre José, por ser ejemplo de esfuerzo y sabiduría.

Mi querida madre Rosa (+) y mi amada hija Emily (+), con la certeza que estarán disfrutando del infinito amor de nuestro Padre celestial.

Agradecimiento

Al Instituto Superior Tecnológico San Gabriel por haber otorgado las facilidades para el desarrollo de estas páginas, además de ser el principal colaborador financiero.

Los libros son, entre mis consejeros,
los que más me agradan,
porque ni el temor ni la
esperanza les impide decirme
lo que debo hacer.
Alfonso V el Magnánimo.

Contenido

Capítulo 1
Fundamentos de electrónica

Capítulo 2
Electricidad

Capítulo 3
Sistemas embebidos

Índice de Tablas

Índice de Figuras

Capítulo 1

Fundamentos de electrónica

Capítulo 1
Fundamentos de electrónica

1

1.1. Introducción

Pocos siglos atrás el desarrollo de las naciones era lento debido a que los únicos medios de transporte fueron el barco y la carroza, además que la comunicación se limitaba a fuentes orales y escritas.

Sin embargo, en la actualidad la humanidad ha sufrido un drástico cambio con el desarrollo tecnológico propiciado por los avances en electrónica e informática, permitiendo diversificar notablemente muchos aspectos. La radio, la televisión, el *smartphone* y

la red mundial internet constituyen claros ejemplos de desarrollo tecnológico comunicacional, de este modo las noticias y los conocimientos científicos pueden difundirse por todo el mundo en cuestión de segundos, incluso toma unas cuantas horas recibir datos de otros planetas proveniente de satélites artificiales controladas por computadoras que envían a la Tierra información sobre sus viajes, tal es el caso de las famosas sondas Voyager 1 lanzada el 5 de septiembre de 1977 a los confines del universo y que se encuentra unas tres veces la distancia a Plutón.

1.2. Historia

El origen de la electrónica puede ubicarse hacia 1883, cuando el inventor estadounidense Thomas Alva Edison descubrió la emisión

termoiónica en los filamentos de las lámparas incandescentes. Observó que en su lámpara incandescente había un punto sobre la superficie del vidrio que se calentaba más que otras zonas. En este punto colocó en el interior de la lámpara, una pequeña placa de metal unida a un cable conductor y luego se le ocurrió conectar éste al polo positivo de la batería; finalmente observó que a través del cable circulaba una corriente.

A este fenómeno le llamó emisión termoiónica porque creía que por efectos del calor se producían iones negativos (electrones) que eran atraídos hacia la placa positiva.

En 1944, el investigador inglés John Ambrose Fleming aplicó el efecto termoiónico en sus experimentos, dando origen a un tubo de

vacío llamado diodo. Este dispositivo estaba formado por tres elementos: un filamento que generaba calor, un cátodo revestido de un material que permitía una mayor producción de electrones y una placa. El diodo dejaba fluir la corriente eléctrica desde el cátodo hacia la placa, pero nunca en sentido opuesto. Esta fue la base de la electrónica.

1.3. Definiciones

1.3.1. Electrónica

Etimológicamente la palabra electrónica proviene de las palabras griegas *elektron* que se significa ámbar y el sufijo *–iko* que significa relativo a, es decir estudia el comportamiento de los electrones y la aplicación de sus principios en diferentes contextos. (Ávarez-Láinez, Martínez-

Tejada & Jaramillo, 2019)(Arboledas, 2014). A la vez la electrónica como ciencia forma parte de la física y la ingeniería aplicada al diseño de dispositivos electrónicos cuyo funcionamiento depende del flujo de electrones para la generación, transmisión, recepción y almacenamiento de información. Esta se encuentra conformada por una serie de conocimientos técnicos, prácticos y teóricos.

1.3.2. Átomos

La corriente eléctrica es el paso de electrones por un conductor. Dichos electrones forman parte constitutiva de todos los objetos pero arraigados a la estructura de un átomo constituyente de un elemento químico. (Seippel, 2021).

Como un ejemplo para aclarar el tema, digamos que todos los cuerpos están formados por elementos químicos (el agua, por ejemplo, está formada por los elementos químicos hidrógeno y oxígeno), y que un átomo es la parte más pequeña a la que puede ser reducido un elemento químico.

Si se pudiera dividir el átomo de un elemento, tendríamos pequeñísimas partículas que son las que dan a los átomos sus características particulares, en el centro del átomo está el núcleo, que tiene dos clases de partícula: los protones y los neutrones; alrededor del núcleo giran los electrones, así como los planetas en torno al Sol.

Una característica importantísima de los protones es que tienen carga eléctrica, vale decir: tienen una energía intrínseca y natural, puesta de manifiesto por las fuerzas que pueden ejercer sobre otras partículas del mismo tipo y que originan fenómenos de atracción y repulsión entre partículas cargadas eléctricamente.

Se ha constatado que dos electrones o dos protones se repelen entre sí, y por tanto es indudable que las dos partículas tienen cargas eléctricas de distinto signo, a las que se denominó carga eléctrica positiva (+) al protón, y carga eléctrica negativa (-) al electrón.

1.3.3. Iones positivos y negativos

Cuando por cualquier circunstancia un átomo gana o pierde electrones, se dice que

dicho átomo se ha ionizado. Se denomina "Ion Positivo" cuando el átomo tiene más protones que electrones e "Ion negativo" cuando tiene más electrones que protones. (Bushong, 2022). Como cargas de distinto signo se atraen, cuando están cerca iones negativos y positivos éstos se unen, pero también puede ocurrir que solamente se desprendan los electrones que tiene de más el ión negativo y se dirijan hacia el ión positivo para neutralizar su carga. Cuando esto ocurre, se dice que: "el paso de electrones neutralizadores de carga constituye una corriente eléctrica."

1.3.4. Conductores

Existen materiales que permiten el paso de los electrones con mayor facilidad que otros. Se denomina conductor de corriente eléctrica a todo aquel material que ofrece muy poca

resistencia al paso de electrones y se pueden clasificarse en tres tipos: (Farina, 2010)

a) **Conductores de primer orden.** Son aquellos que poseen conductancia eléctrica, en los cuales los portadores de la carga son los electrones. Se caracterizan por tener una conducción sin transferencia substancial de masa. La mayoría de los metales, el grafito y algunos óxidos muestran este tipo de conducción. A veces, a estos materiales se les conoce como conductores metálicos y su conductividad decrece cuando aumenta la temperatura.
b) **Conductores de segundo orden**. Son aquellos que poseen conductancia eléctrica, en los cuales los portadores de carga son los iones. En este tipo de conductores se da una transferencia de masa asociada con la

conductividad. Las soluciones acuosas con sales disueltas, los suelos y las sales iónicas son algunos ejemplos de este tipo de conductores. Su conductividad aumenta cuando se incrementa la temperatura.

c) **Conductores de tercer orden**. Estos materiales son llamados también conductores mixtos o semiconductores, poseen tanto conductancia iónica como eléctrica. Por lo general predomina el carácter eléctrico. Generalmente su conductividad es demasiado baja, pero aumenta rápidamente con la temperatura, de modo que según como se los trate, se comportan como conductores o como aislantes. Dicho de otra manera, son materiales sobre los cuales se puede "regular" el paso de la corriente eléctrica. La mayoría de los óxidos metálicos (NiO, ZnO, etc.) y

algunos metales (Si, Ge, etc.) se agrupan dentro de esta categoría.

1.3.5. Aislantes

Un aislante de la corriente eléctrica es todo aquel material que ofrece una elevada resistencia al paso de los electrones, como ejemplos podemos citar la madera, caucho, plástico, vidrio, silicato, cerámica de óxidos, oxido de aluminio, etc.

1.3.6. Flujo de electrones

Se denomina corriente eléctrica al paso de electrones por un conductor eléctrico (o semiconductor). Su unidad es el ampere (A) y "mide" la cantidad de electrones que atraviesan a un elemento en una unidad de tiempo. Para que pueda establecerse una corriente eléctrica

tiene que existir algo que impulse a los electrones a circular de un lado a otro (Seippel, 2021).

1.3.7. Voltaje, tensión o diferencia de potencial

Como hemos dicho, para que se establezca una corriente eléctrica debe existir algo que impulse a los electrones. Así el voltaje, tensión o diferencia de potencial es la presión que ejerce una fuente de suministro de energía eléctrica o fuerza electromotriz (FEM) sobre las cargas eléctricas en un circuito eléctrico cerrado, para que se establezca el flujo de una corriente eléctrica (Fowler, 2007). A mayor diferencia de potencial o presión que ejerza una fuente de FEM sobre las cargas eléctricas o electrones contenidos en un conductor, mayor será el

voltaje o tensión existente en el circuito al que corresponda ese conductor. El voltaje se mide en volt y se representa por la letra (V).

1.4. Unidades básicas y prefijos del SI

A lo largo de la historia han existido muchas formas de medidas que se adoptaron en distintos lugares, por ejemplo, los egipcios usaban el codo para medir distancias (0, 523 metros), mientras que los romanos usaban el paso (1,481 metros), pero con el tiempo la mayoría de estas fueron cayendo en desuso debido principalmente a que no eran confiables y de uso general para todos los pueblos. Es así que en el año de 1960 la Conferencia General de Pesas y Medidas (CGPM) adoptó un sistema práctico de unidades con el nombre de Sistema Internacional de Unidades (abreviado SI),

estableciendo reglas para los prefijos, las unidades derivadas, antiguas unidades suplementarias y otras cuestiones (Lleó & lleó, 2011). Se define así una reglamentación exhaustiva para las unidades de medida, misma que ha sido actualizada, reconocida y usada en todos los países del mundo excepto Estados Unidos, Birmania y Liberia. En la actualidad se unifico con las normas ISO para instaurar el Sistema Internacional de Magnitudes ISO/IEC 80000, con las siglas ISQn. En la siguiente tabla se muestran las unidades básicas establecidas por el SI.

Tabla 1.

Unidades básicas del SI

Magnitudes básicas		Unidades SI básicas	
Nombre	Símbolo	Nombre	Símbolo
Longitud	l, x, r, etc.	metro	m
Masa	m	kilogramo	kg
Tiempo, duración	t	segundo	s
Corriente eléctrica	I,i	amperio	A
Temperatura termodinámica	T	kelvin	K
Cantidad de sustancia	n	mol	mol
Intensidad luminosa	I_v	candela	cd

Nota. Adaptado del Sistema internacional de unidades SI. Copyright 2008 por la Oficina Internacional de Pesas y Medidas y la Organización Intergubernamental de la Convención del Metro.

En cuanto a una nomenclatura que facilite la escritura de números pequeños y grandes, el SI adoptó una serie de nombres y símbolos de prefijos para formar múltiplos y submúltiplos

decimales desde 10^{-24} hasta 10^{24}, la siguiente tabla muestra los prefijos aprobados.

Tabla 2.

Prefijos del SI

Factor	Nombre	Símbolo	Factor	Nombre	Símbolo
10^{1}	Deca	da	10^{-1}	deci	d
10^{2}	Hecto	h	10^{-2}	centi	c
10^{3}	Kilo	k	10^{-3}	Mili	m
10^{6}	Mega	M	10^{-6}	micro	μ
10^{9}	Giga	G	10^{-9}	nano	n
10^{12}	Tera	T	10^{-12}	Pico	p
10^{15}	peta	P	10^{-15}	femto	f
10^{18}	Exa	E	10^{-18}	Atto	a
10^{21}	Zetta	Z	10^{-21}	Zepto	z
10^{24}	Yotta	Y	10^{-24}	Yocto	Y

Nota. Adaptado del Sistema internacional de unidades SI. Copyright 2008 por la Oficina Internacional de Pesas y Medidas y la Organización Intergubernamental de la Convención del Metro.

En el ámbito científico existe una extensa lista de unidades derivadas expresadas en función de las unidades básicas SI. Por la magnitud se listarán aquellos de utilidad en este libro.

Tabla 3.

Ejemplos de unidades SI derivadas

Magnitud derivada		Unidad SI derivada coherente	
Nombre	Símbolo	Nombre	Símbolo
Densidad de corriente	j	Amperio por metro cuadrado	A/m^2
Campo magnético	H	Amperio por metro	A/m
Resistencia eléctrica	Ω (ohm)	Voltio por amperio	V/A
Conductancia eléctrica	S (siemens)	Amperio por Voltio	A/V
Carga eléctrica	C (coulomb)	Amperio por segundo	A.s
Capacitancia eléctrica	F (faraday)	Coulomb por Voltio	C/V
Flujo magnético	Wb (weber)	Voltio por segundo	V.s
Inductancia	H (henrio)	Voltio, segundo por amperio	V.s/A

1.5. El circuito eléctrico y sus elementos básicos

Se puede definir al circuito eléctrico como

Un camino cerrado cuyo fin es generar, transportar y utilizar energía eléctrica desde unos elementos conectados entre sí hasta otros elementos con la finalidad de transformarla en otro tipo de energía. Por ejemplo, energía calorífica (estufa), energía lumínica (bombilla) o energía mecánica (motor).

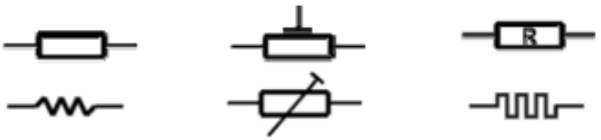
En dicho circuito podemos encontrar una gran variedad de componentes, los cuales difieren en funciones y características específicas. En el siguiente apartado vamos a realizar una breve descripción de ellos.

1.5.1. Resistencias

Una resistencia es un elemento que por sus propiedades físicas dificulta el paso de

electrones a través de ella, su unidad de medida son los ohmios (Ω) (Perez, 2018). Comercialmente estos elementos pueden adquirirse en una gran gama de medidas, algunas de estas tienen su medida impresa directamente en su empaque, y otras usan un código de colores para representar su valor. Estos son algunos de los símbolos normalizados.

Figura 1.
Simbología para resistencias



Resistencia | Resistencia ajustable | Resistencia no reactiva

A continuación, se presentan algunos tipos:

Figura 2.
Tipos de resistencias

Resistencia de alta potencia

Resistencia de alambre de alta corriente

Resistencia de carbón depositado

Resistencia con recubrimiento de óxido de metal

Resistencia de potencia de película gruesa

Resistencia SMD para circuitos impresos

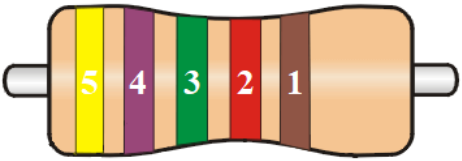
Resistencia de recubrimiento de carbón

Resistencia con recubrimiento de metal

En el código de colores de cada resistencia se puede distinguir dos parámetros. El primero es el valor de la resistencia expresado en Ohmios (Ω), y representa su resistividad al paso de electrones. El segundo parámetro es la tolerancia expresado en porcentaje, y permite estimar el

margen de error de su valor resistivo. Por ejemplo, una resistencia de 100Ω con una tolerancia del ±5% implica que este dispositivo podría trabajar entre 95 Ω a 105 Ω, y esto depende de las condiciones ambientales o físicas a las que se encuentre sometido. Para poder traducir estos valores se siguen las siguientes indicaciones:

Figura 3.
Código de colores en resistencias



De derecha a izquierda el primer color representa la tolerancia, el segundo valor representa el factor de multiplicación de los

colores descritos en la 3ra, 4ta y 5ta posición. El valor correspondiente a cada color se presenta a continuación en la siguiente tabla.

Tabla 4.

Código de colores para resistencias

Color	5ta, 4ta y 3ra posición (valor de la resistencia)	2da posición (factor de multiplicación)	1ra posición (tolerancia estimada)
Negro	0	x 1	--
Marrón	1	x 10	--
Rojo	2	x 100	--
Naranja	3	x 1000	--
Amarillo	4	x 10000	--
Verde	5	x 100000	--
Azul	6	x 1000000	--
Violeta	7	x 10000000	--
Gris	8	--	--
Blanco	9	--	--
Dorado	--	0,1	±5%
Plateado	--	0,01	±10%
Sin color	--	--	±20%

1.5.2. Potenciómetros

Son dispositivos resistores de capacidad variable y su unidad de medida es la misma que las resistencias es decir Ohmios (Beiroa, 2018). Su funcionamiento puede ser digital o mecánico, siendo este último el más común. Estos son algunos de los símbolos normalizados.

Figura 4.
Simbología de potenciómetros

Potenciómetro | Potenciómetro de ajuste predeterminado | Potenciómetro de contacto móvil

A continuación, se muestran algunos de ellos:

Figura 5.
Tipos de potenciómetro

Potenciómetro digital

Potenciómetro de palanca

Potenciómetro lineal

Potenciómetro relativo

Potenciómetro de precisión

Potenciómetro múltiple

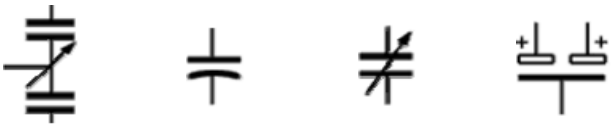
1.5.3. Condensadores

Un condensador eléctrico o capacitor, es un dispositivo pasivo capaz de almacenar energía sustentando un campo eléctrico y su unidad de medida es el Faradio (F). Estos dispositivos están formados por un par de superficies conductoras, generalmente en forma de láminas o placas separadas por un material dieléctrico(Cerdá,

2014). Estos son algunos de los símbolos normalizados.

Figura 6.
Simbología de condensadores

Condensador no polarizado | Condensador electrolítico | Condensador ajustable | Condensador diferencial



Condensador variable de doble armadura | Condensador no polarizado | Condensador variable | Condensador electrolítico múltiple

A continuación, se presentan algunos tipos de condensadores disponibles comercialmente:

Figura 7.

Tipos de capacitores

Capacitores cerámicos

Capacitores de aire

Capacitores variables

Capacitores de papel

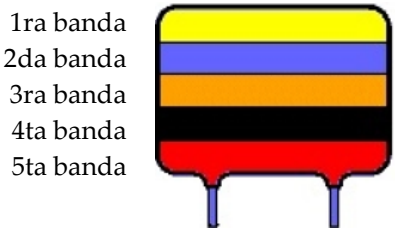
Capacitores electrolíticos

Aunque la mayoría condensadores llevan su correspondiente valor inscrito en el mismo dispositivo (Ver Figura 8), existen algunos tipos de condensadores cerámicos que llevan un código de colores similar a las resistencias.

Figura 8.
Inscripción en capacitores

A continuación, aprenderemos como interpretar dicho código.

Figura 9.
Color de banda para capacitores



Los colores de la primera y segunda banda representar el valor numérico del capacitor, la

tercera banda representa el factor de multiplicación, la cuarta banda representa la tolerancia de trabajo y finalmente la quinta banda representa la tensión de trabajo.

La tabla 5, presenta los valores correspondientes de cada color.

Tabla 5.

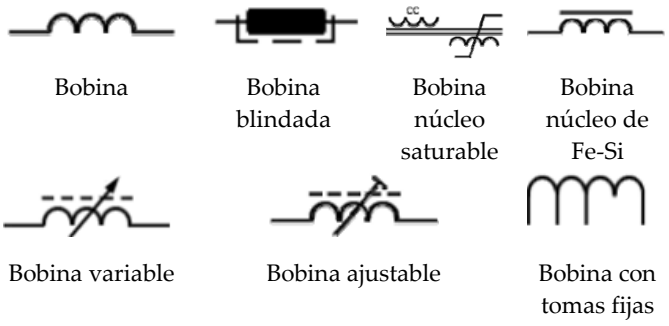
Código de colores para capacitores

Color de la banda	1ra banda (valor del capacitor)	2da banda (valor del capacitor)	3ra banda (factor de multiplicación)	4ta banda (tolerancia)		5ta banda (tención de trabajo)
				C>20 pF	C<20 pF	
Negro	--	0	x 1	±20 pF	±1 pF	--
Marrón	1	1	x 10	±1 pF	±0,1 pF	100 V
Rojo	2	2	x 100	±2 pF	±0,25 pF	250 V
Naranja	3	3	x 1000	--	--	--
Amarillo	4	4	x 10.000	--	--	400 V
Verde	5	5	x 100.000	±5 pF	±0,5 pF	--
Azul	6	6	x 1000.000	--	--	630 V
Violeta	7	7	--	--	--	--
Gris	8	8	--	--	--	--
Blanco	9	9	--	±10 pF	±1 pF	--

1.5.4. La bobina

Este tipo de componentes pasivos permite almacenar energía en forma de campo magnético, consta de dos o más terminales de hilo conductor arrollando sobre un núcleo de material ferromagnético o al aire, mismo que generan un flujo magnético cuando circula corriente eléctrica por ella. Su unidad de medida es el Henrio (H) y estos son algunos de sus símbolos normalizados (Alcalde, 2016).

Figura 10.
Simbología de las bobinas



Aquí presentamos algunos tipos de bobinas comerciales.

Figura 11.
Bobinas comerciales

Bobina núcleo de aire

Bobina núcleo de ferrita

Bobina blindada

Bobina núcleo toroidal

Bobina núcleo saturable

Bobina ajustable

1.5.5. El diodo

Este elemento permite la circulación de corriente eléctrica en un solo sentido, y consta de dos materiales semiconductores juntos (Schuler,

2021), el primero contiene portadores de carga negativa y el segundo portadores de carga positiva con uniones a los extremos de modo que solo pueda pasar corriente en un sentido. Este elemento no tiene unidad de medida.

Existes varios tipos de diodos pendiendo su utilidad. A continuación, hablaremos de algunos:

Diodo Varactor (Varicap o de capacidad variable). Este tipo obtiene una capacidad que depende de la tensión inversa a él aplicada, Es usado frecuentemente en circuitos sintonizadores de televisión y receptores de radio en FM.

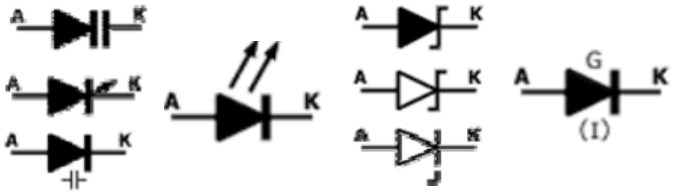
Diodos emisores de luz (LED). Este tipo de elemento tiene la particularidad de emitir luz como si fuese un pequeño foco. Los pequeños foquitos que vemos cuando activamos alguna función de nuestra impresora, microondas, refrigeradora, etc. son implementados a través de estos dispositivos. Otro ejemplo son los focos led que otorgan mayor luminosidad y menor consumo de energía en nuestros hogares con un menor consumo de energía.

Diodo Zener o diodo regulador de tensión. Este elemento polarizado en forma directa funciona de forma normal, pero cuando es polarizado en forma inversa con una corriente inversa superior a un determinado valor, presenta una tensión de valor constante. Es usado muy comúnmente como dispositivo regulador de tensión.

Diodo Gunn. Se trata de un generador de microondas, formado por un semiconductor de dos terminales que utiliza el llamado efecto Gunn, La emisión de microondas se produce cuando las zonas de campo eléctrico elevado se desplazan del ánodo al cátodo y del cátodo al ánodo en un constante viaje rapidísimo entre ambas zonas, lo que determina la frecuencia en los impulsos. Son usados en aplicaciones como pistolas de radar de velocidad y transmisores de relé de microondas.

A continuación, se presentan algunos de los símbolos normalizados:

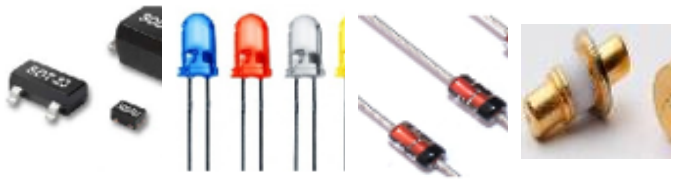
Figura 12.
Simbología del diodo



Diodo varicap | Diodo led | Diodo zener | Diodo gunn impatt

También presentamos algunas imágenes de sus correspondientes diodos comerciales.

Figura 13.
Diodos comerciales



Diodo varicap | Diodo led | Diodo zener | Diodo gunn impatt

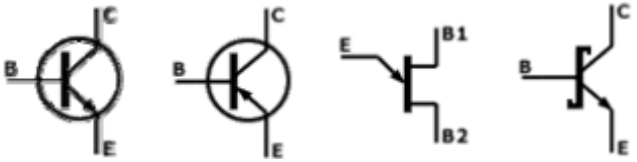
1.5.6. El transistor

En electrónica este es el dispositivo más importante, es un semiconductor utilizado para entregar una señal de salida en respuesta a una señal de entrada. Cumple funciones de amplificador, oscilador, conmutador o rectificador.

Actualmente son utilizados para la construcción de circuitos integrados que se encuentran dentro de todo tipo de aparatos electrónicos como radios, televisores, reproductores de audio y video, computadoras, teléfonos celulares, etc., (Rengel, 2020).

Al igual que el diodo, este dispositivo únicamente controla el paso de corriente por sus terminales de modo que no tiene unidades de medida. A continuación, algunos de los símbolos normalizados.

Figura 14.
Simbología del Transistor



Transistor NPN | Transistor PNP | Transistor UJT | Transistor Schottky

También mostramos algunos transistores comerciales.

Figura 15.
Transistores comerciales

Transistor NPN | Transistor PNP | Transistor UJT | Transistor Schottky

1.5.7. Circuitos integrados (C.I.)

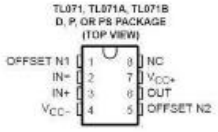
Conocidos también como chip o microchip, son elementos que tienen una estructura de pequeñas dimensiones de material semiconductor (normalmente silicio) de algunos milímetros cuadrados de superficie, sobre la que se fabrican circuitos electrónicos generalmente mediante fotolitografía y que está protegida dentro de un encapsulado de plástico o de cerámica.

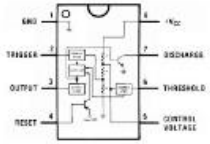
El encapsulado posee conductores metálicos apropiados para hacer conexión entre el Circuito Integrado y un circuito impreso. Se diseñan para ser conectados entre ellos mediante soldadura a un circuito impreso (De la Rosa, 2021). Las señales que entrega pueden ser analógico (ondas sinusoidales) o digitales

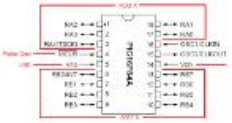
(valores de "0" o "1"). A continuación, se presentan algunos ejemplos con su correspondiente simbología:

Figura 16.

Algunos circuitos integrados comerciales







Capítulo 2

Electricidad

Capítulo 2
Electricidad

2

2.1. Introducción

El estudio de circuitos eléctricos permite tener un conocimiento general sobre las diferentes formas en que se realizan las conexiones eléctricas en un circuito cerrado, en su forma básica pueden interactuar elementos como resistencias, condensadores, fuentes de voltaje, entre otros, y en su forma compleja pueden intervenir elementos como interruptores, transistores, circuitos integrados, motores, entre otros. En nuestros hogares cuando conectamos algún dispositivo como una

televisión, estufa eléctrica, refrigeradora, computadora, etc. En realidad, estamos cerrando un circuito eléctrico, y tener un conocimiento general de las variables que actúan permite tener un mejor entendimiento sobre su funcionamiento para en lo posterior realizar un análisis de cargas que asegure el sistema.

2.2. Fuerza electromotriz (FEM)

La FEM de una celda se mide en volts y está definida como la sumatoria de las diferencias de potencial producidas por los componentes de un circuito al cual está conectado, incluyendo la diferencia de potencial requerida para impulsar la corriente a través de la misma celda (Zapata, 2022).

La FEM de una celda en volts se define entonces como el trabajo total efectuado en joules por los coulombs de electricidad transportados en un circuito en el que la celda está conectada (Pérez, 2018).

2.3. Ley de Ohm

La ley de Ohm fue establecida por el profesor de física Georg Simon Ohm en el año de 1826 como resultado de varios experimentos donde se investigaba la relación entre la corriente que pasa por un alambre y la diferencia de potencial establecida entre los extremos del mismo, concluyendo que "La intensidad de la corriente eléctrica que pasa por un conductor en un circuito es directamente proporcional a la diferencia de potencial aplicada a sus extremos e inversamente proporcional a la resistencia del

conductor" (Pérez, 2019). Así el conductor que siga esta relación obedecerá la siguiente ley:

$$\Delta V = K.I$$

Donde, ΔV es la diferencia de potencial entre los extremos del circuito, K es la constante de proporcionalidad e I es la corriente eléctrica que circula por el circuito. Mediante experimentación se determinó que el valor de K es alto cuando el valor de la corriente es pequeño y viceversa, pudiendo representar la medida de la resistencia R del alambre. De modo que:

$$\Delta V = R.I$$

Por tanto, la ley de Ohm, quedaría de la siguiente manera:

Voltaje (V)= Corriente (I) x Resistencia (R)

Despejando R tenemos que:

$$R = V / I \qquad \text{(Ecuación 1)}$$

En otras palabras, la resistencia de un conductor es directamente proporcional al voltaje, e inversamente proporcional a la corriente que fluye a través de él. Donde R es la resistencia eléctrica expresada en ohmios y se define como: ″la resistencia de un conductor dado, cuando se aplica una diferencia de potencial de 1 volt en sus extremos y una corriente de 1 ampere fluye por él″ (Seippel, 2021)(Portis, 1985).

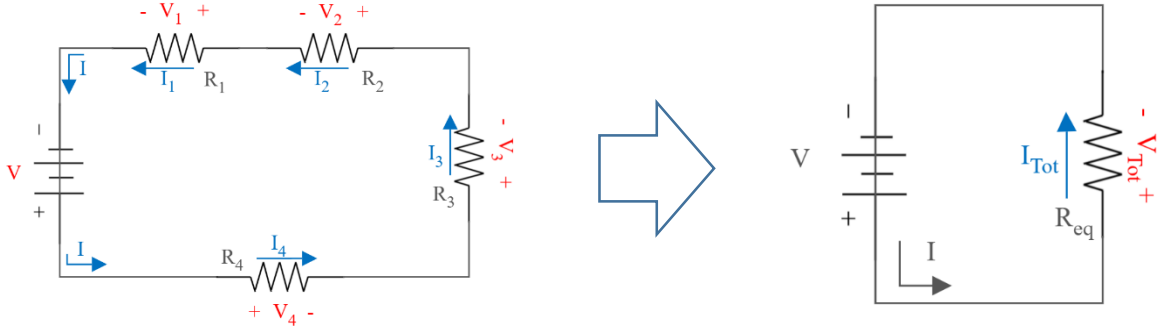
También se sabe que la resistencia de un metal puro aumenta con la temperatura, pero la

resistencia de otros materiales conductores, como el carbón, por ejemplo, decrece con la temperatura. En otros elementos, como los semiconductores (germanio, silicio y selenio) la resistencia también disminuye cuando aumenta la temperatura, esto por las disoluciones iónicas que contienen las sales y los suelos.

2.4. Cálculo de resistores en serie

Para un número de resistores, (R_1, R_2, R_3, …, Rn) conectados consecutivamente extremo con extremo, se sabe que la corriente I (en amperes) que fluye a través de cada resistencia es la misma, mientras que el voltaje depende del valor de cada resistor. Por tanto, de acuerdo con la Figura 17, se puede saber que:

Figura 17.
Resistores en serie



Cálculo de valores individuales. - R_1; R_2; $R_{3;\ ...;}$ R_n. Se calcula mediante la Ecuación 1, mientras que I_1, I_2, I_3, …, I_n y V_1, V_2, V_3, …, V_n se calculan individualmente en cada variable, Despejando V y R de la Ecuación 1 tenemos.

$$V=I.R \qquad \text{(Ecuación 2)}$$

$$I=V/R \qquad \text{(Ecuación 3)}$$

Cálculo de valores equivalentes en función de valores individuales. La resistividad total del circuito es igual a la suma total de todos sus resistores individuales, para el voltaje se procede de la misma manera, mientras que la corriente al ser la misma en cualquier parte del circuito simplemente se iguala.

$$R_{eq} = R_1 + R_2 + R_3 + \ldots + R_n \qquad \text{(Ecuación 4)}$$

$$V_{Tot} = V_1 + V_2 + V_3 + \ldots + V_n \qquad \text{(Ecuación 5)}$$

$$I_{Tot} = I_1 = I_2 = I_n \qquad \text{(Ecuación 6)}$$

Cálculo de valores equivalentes en función de valores totales. - Para este cálculo usamos la Ecuación 1:

$$R_{eq} = V_{Tot} / I_{Tot} \quad \text{(Ecuación 7)}$$

$$V_{Tot} = I_{Tot} \cdot R_{eq} \quad \text{(Ecuación 8)}$$

$$I_{Tot} = V_{Tot} / R_{eq} \quad \text{(Ecuación 9)}$$

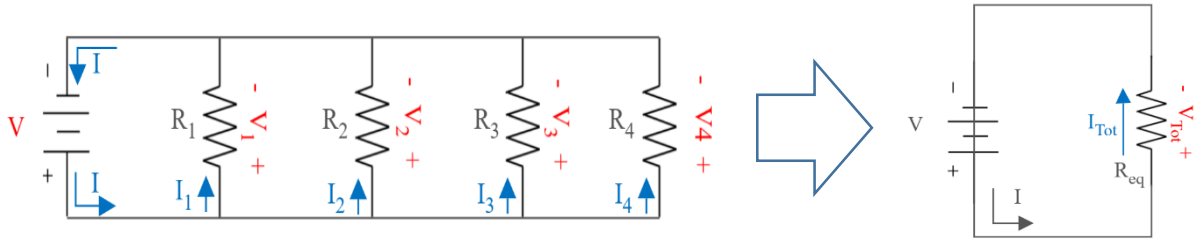
2.5. Cálculo de resistores en paralelo

Se dice que los resistores están en paralelo cuando son colocados uno al lado del otro y sus extremos permanecen unidos como se muestra en la Figura 18. En esta configuración, la corriente eléctrica del circuito se divide para cada resistor, mientras que el voltaje permanece igual en todo el circuito.

Cálculo de valores individuales. - Al igual que en la configuración serie, R_1; R_2; $R_{3;..;}$ R_n. Se calcula mediante la Ecuación 1, mientras que I_1; I_2; $I_{3;...;}$ I_n y V se calculan en cada variable. Mire las ecuaciones 3 y 2 respectivamente.

Figura 18.

Resistores en paralelo



Cálculo de valores totales en función de valores individuales. - La resistencia equivalente del circuito es igual a la sumatoria inversa de todos sus resistores individuales, La corriente equivalente es igual a la suma de sus valores individuales y el voltaje es el mismo en todo el circuito.

$$R_{eq} = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} + \cdots + \frac{1}{Rn}} \qquad \text{(Ecuación 10)}$$

$$I_{Tot} = I_1 + I_2 + I_3 + I_n \qquad \text{(Ecuación 11)}$$

$$V_{Tot} = V_1 = V_2 = V_n \qquad \text{(Ecuación 12)}$$

Cálculo de valores totales en función de valores totales. - Para este cálculo usamos las ecuaciones 6, 7 y 8 del apartado anterior.

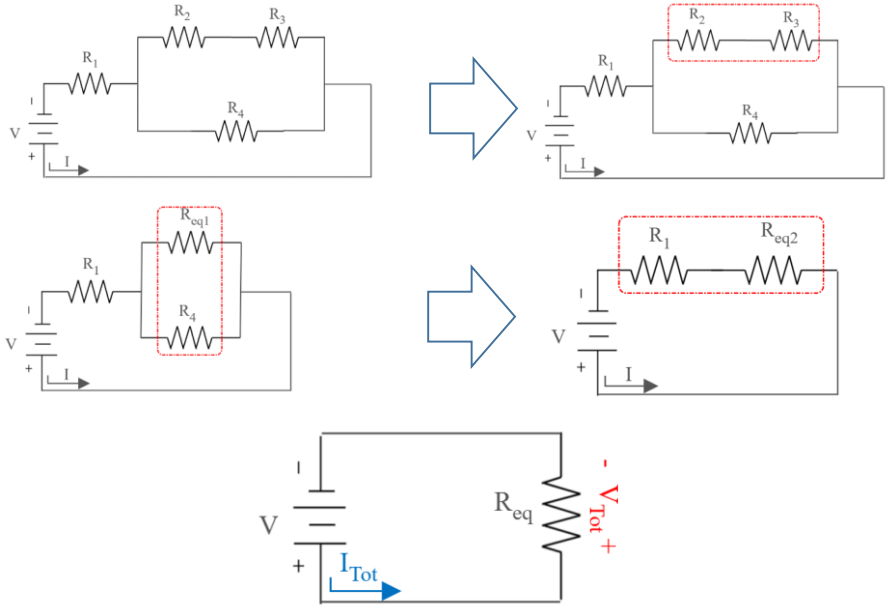
2.6. Cálculo de circuitos mixtos

En los apartados anteriores aprendimos a calcular las distribuciones en serie y paralelo,

pero en la vida real los circuitos normalmente están compuestos por varias secciones, algunas conectadas en serie y en otras conectadas en paralelo. A este tipo de distribución se lo conoce como circuitos mixtos, y por tanto se resuelven por partes. Aquí un ejemplo gráfico paso a paso.

Figura 19.

Solución de circuitos mixtos



Como se observa en la figura 19, el procedimiento es sencillo y únicamente requiere un poco de práctica en la solución de circuitos serie y paralelo vista en las secciones anteriores.

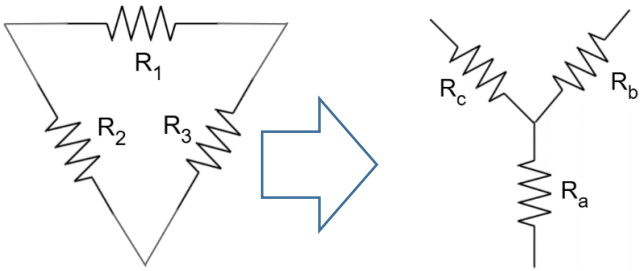
2.7. Configuraciones especiales

En algunas ocasiones sucede que la distribución de los elementos del circuito no coincide con las configuraciones serie o paralelo, en esos casos conviene ordenar los elementos de una forma diferente, pero sin que el funcionamiento general de este cambie.

Configuración de resistencias delta-estrella

Cuando tres elementos resistivos están unidos en forma de un triángulo invertido, entonces teóricamente se puede transformar en una configuración tipo Y tal como se muestra en la figura 20.

Figura 20.
Transformación delta-estrella



Para realizar esta transformación usamos las siguientes ecuaciones:

$$R_a = \frac{R_2.R_3}{R_1 + R_2 + R_3} \qquad \text{(Ecuación 13)}$$

$$R_b = \frac{R_1.R_3}{R_1 + R_2 + R_3} \qquad \text{(Ecuación 14)}$$

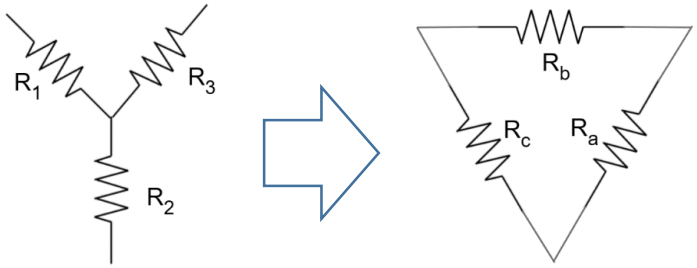
$$R_c = \frac{R_1.R_2}{R_1 + R_2 + R_3} \qquad \text{(Ecuación 15)}$$

Configuración de resistencias estrella-delta

Cuando tres elementos resistivos están unidos en un solo nodo en forma de Y, entonces teóricamente se puede transformar en una

configuración tipo triangulo invertido tal como se muestra en la Figura 21.

Figura 21.
Transformación estrella- delta



Para realizar esta transformación usamos las siguientes ecuaciones:

$$R_a = \frac{(R_1.R_2)+(R_1.R_3)+(R_2.R_3)}{R_1} \qquad \text{(Ecuación 16)}$$

$$R_b = \frac{(R_1.R_2)+(R_1.R_3)+(R_2.R_3)}{R_2} \qquad \text{(Ecuación 17)}$$

$$R_c = \frac{(R_1.R_2)+(R_1.R_3)+(R_2.R_3)}{R_3} \qquad \text{(Ecuación 18)}$$

2.8. Cálculo de celdas o capacitores en serie y paralelo

A diferencia de los resistores que disipan energía, las celdas o capacitores son elementos capaces de absorber y almacenar energía eléctrica para luego devolverla al circuito como si fuese una batería. La carga y la diferencia de potencial de un capacitor está relacionada por la siguiente ecuación:

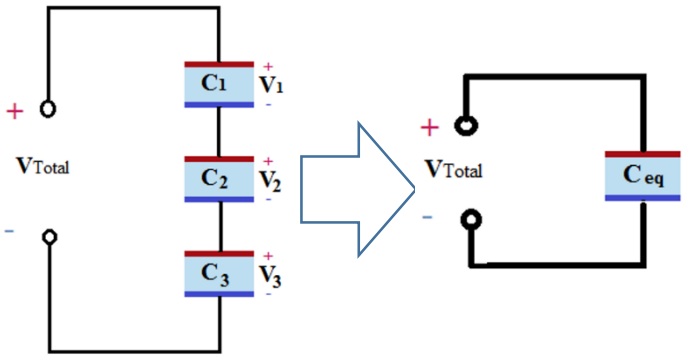
$$q = C.V \qquad \text{(Ecuación 19)}$$

Donde q es la carga del dispositivo medida en coulomb (C), C es la capacitancia medida en faradios (F), V es la diferencia de potencial medida en voltios (V).

Para un número determinado de celdas, (C_1, C_2, C_3, …, Cn) conectados consecutivamente

extremo con extremo, se sabe que la corriente I (en amperes) que fluya a través de cada celda una es la misma. Por tanto, de acurdo la figura 22 se puede saber que:

Figura 22.
Celdas en Serie

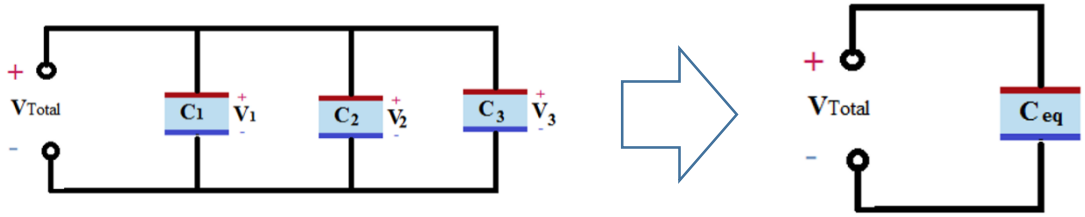


La capacitancia total es igual a la sumatoria de las inversas, y su cálculo obedece a la siguiente fórmula:

$$C_{Total} = \frac{1}{\frac{1}{C1} + \frac{1}{C2} + \frac{1}{C3} + \cdots + \frac{1}{Cn}} \qquad \text{(Ecuación 20)}$$

Por el contrario, para una conexión en paralelo, es decir cuando un número determinado de celdas, (C_1, C_2, C_3, …, Cn) se encuentran conectados uno al lado del otro unidos por sus extremos como se muestra a continuación:

Figura 23.
Celdas en paralelo



Donde la capacitancia equivalente es igual a la sumatoria de las capacitancias individuales:

$$C_{eq} = C_1 + C_2 + C_3 + .. + C_n \qquad \text{(Ecuación 21)}$$

Capítulo 3

Sistemas embebidos

Capítulo 3
Sistemas embebidos

3

3.1. Introducción

Un sistema embebido o empotrado es un sistema electrónico creado para realizar funciones dedicadas, a diferencia con los dispositivos de propósito general como una computadora que están diseñados para cubrir un amplio rango de necesidades (Pérez y Mejía, 2021)(Chetto y Queudet, 2020).

En este tipo de sistemas todos sus componentes se encuentran soldados en misma placa. Algunos ejemplos de sistemas embebidos podrían ser dispositivos como un control de

humedad, un sistema de alerta contra incendios, etc.

Los sistemas embebidos se pueden programar directamente en el lenguaje ensamblador del microcontrolador, lenguaje C o también, utilizando compiladores específicos; en algunos casos, cuando el tiempo de respuesta de la aplicación no es un factor crítico, también pueden usarse lenguajes Orientados a Objetos como JAVA, pero estos no son muy comunes puesto que los programas de sistemas embebidos generalmente se enfrentan a tareas de procesamiento en tiempo real.

Existen varios proyectos desarrollados por distintos fabricantes que proporcionan herramientas para el desarrollo y diseño de aplicaciones y prototipos, algunos ejemplos de

estás son: Arduino, mbed, Raspberry Pi, BeagleBone, etc. En este capítulo enseñaremos a montar y programar módulos Arduino por ser una de las plataformas de mayor popularidad en la actualidad.

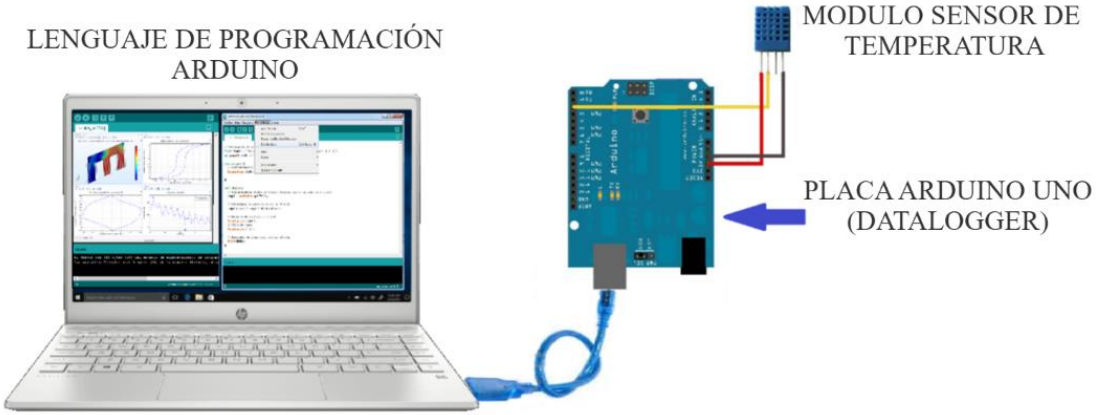
3.2. Introducción Arduino

Arduino es una compañía de desarrollo de software y hardware, así como una comunidad internacional que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan censar diferentes tipos de variables (Corona, Abarca y Mares, 2019). Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Los productos que vende la compañía son

distribuidos como Hardware y Software Libre, permitiendo la manufacturación de las placas Arduino y distribución del software por cualquier individuo.

Para implementar proyectos, necesitamos como mínimo dos elementos, el primero es modulo sensor que permitirá transformar variables físicas en analógicas (por ejemplo, un sensor de temperatura), y el segundo es la placa Datalogger que recibe las señales del sensor y las procesa para que puedan ser entendibles al ser humano como se muestra en la Figura 24.

Figura 24.
Conexión Arduino-PC



A continuación, vamos a describir algunos de los módulos sensores y placas comerciales de Arduino.

3.3. Sensores para Arduino

Estos son solo algunos módulos, de una gran variedad.

Tabla 6.
Sensores Arduino.

	**Medidor de distancias por ultrasonidos. -** Los sensores de ultrasonidos se usan principalmente para medir distancias o superar obstáculos. Envía una señal ultrasónica inaudible y mide el tiempo que tarda en regresar al ser reflejado por algún objeto.
	**Sensor de temperatura y humedad relativa (RH).** El sensor posee una interfaz serial propietaria, que solo requiere de un pin para comunicarse con un microcontrolador.

Sensor PIR detección de movimiento. - Este módulo permite detectar personas y animales grandes a través de un sensor PIR (Passive Infrared). Los sensores PIR se utilizan ampliamente en aplicaciones como sistemas de alarma, puertas automáticas, luces automáticas, etc.

Sensor de calidad de aire amoniaco benceno.- Permite detectar algunos gases peligrosos como Amoniaco, Dióxido de Nitrógeno, Alcohol, Benzeno, Dioxido y Monoxido de carbono.

Caudalímetro (sensor de flujo) .- Permite medir el flujo de líquidos o gases a través de una hélice interna.

Sensor de superficie muscular electrodos.- La electromiografía es una técnica de electro-diagnóstico para evaluar y registrar la actividad eléctrica producida por los músculos esqueléticos.

	**Sensor de Corriente.-** Es un módulo de ladrillo electrónico que se puede usar para probar el bucle de corriente electrónico que circula por un circuito.
	**Electronic brick sensor UV ultra violeta.-** Este sensor permite medir la radiación ultravioleta producida por el sol.

Nota. Adaptado de store.arduino.cc

3.4. Placas Arduino

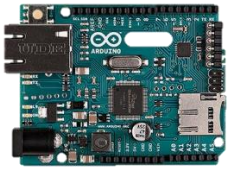
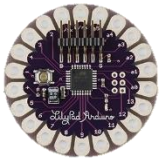
A continuación, algunos módulos más utilizados.

Tabla 7.

Tarjetas Arduino.

Arduino Uno. - Es la placa estándar y la más conocida y documentada. Salió a la luz en septiembre de 2010 sustituyendo su predecesor Duemilanove con varias mejoras de hardware que consisten básicamente en el uso de un USB HID propio en lugar de utilizar un conversor FTDI para la conexión USB.

Arduino Mega. - Es con mucha diferencia el más potente de las placas con microcontrolador de 8 bits y el que más pines i/o tiene, apto para trabajos ya algo más complejos, aunque tengamos que sacrificar un poco el espacio. Cuenta con el microcontrolador Atmega2560 con más memoria para el programa, más RAM y más pines que el resto de los modelos.

	**Arduino Ethernet**.- Incorpora un puerto ethernet, está basado en el Arduino Uno y nos permite conectarnos a una red o a Internet mediante su puerto de red.
	**Arduino Mini. -** Versión miniaturizada de la placa Arduino UNO basado en el ATMega328. Mide tan sólo 30x18mm y permite ahorrar espacio en los proyectos que lo requieran. Las funcionalidades son las misma que Arduino UNO.
	**Arduino Lilypad. -** Diseñado para dispositivos "wearables" y e-textiles. Para coser con hilo conductor e instalarlo sobre prendas.

Nota. Adaptado de store.arduino.cc

Indistintamente de las funcionalidades que tengan cada tarjeta Arduino, todas se comportan de la siguiente manera:

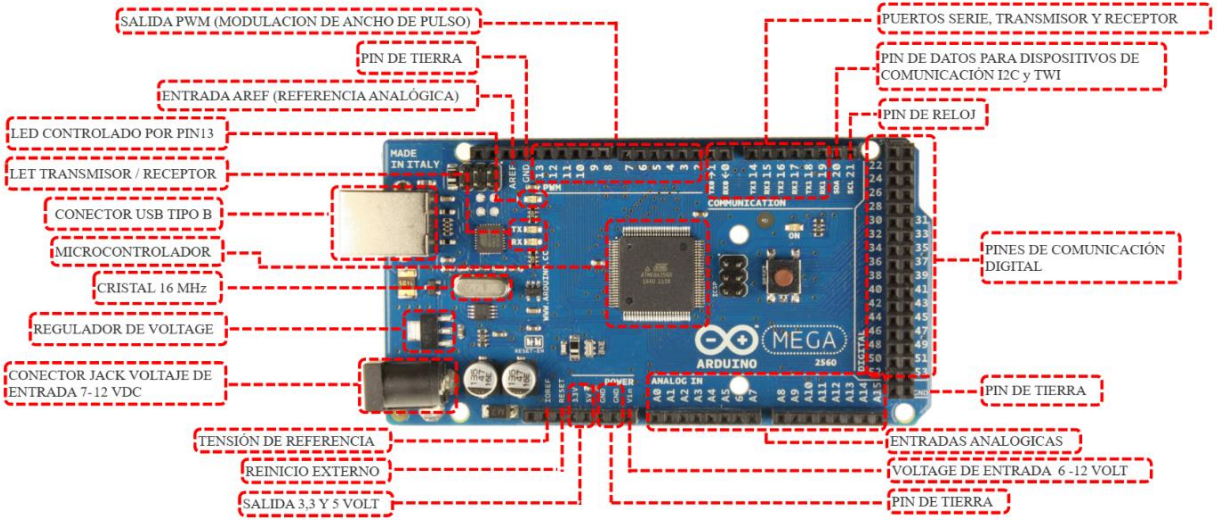
Figura 25.
Comportamiento de las tarjetas Arduino

Los pines de entrada permiten recibir datos de sensores ya sean analógicos o digitales conectados a la tarjeta, posterior estos datos se procesan según el código que el usuario programe dentro de la tarjeta, finalmente, se pueden enviar señales al exterior para que por ejemplo se mueva un motor, encienda un foco, active una alarma, etc.

En la Figura 26 se puede observar las partes de una placa Arduino mega 2560.

Figura 26.

Partes de la tarjeta Arduino mega



3.4.1. Alimentación

Las placas Arduino se pueden obtener energía de tres formas:

La primera es a través de la conexión del puerto USB tipo B, este cable también se usa para conectar la tarjeta con el pc al momento de programar. Esta conexión tiene la desventaja de que su potencia es medianamente baja (5V a 500mA). Generalmente esta conexión se usa únicamente para programar el Arduino y hacer pruebas en tiempo real, pero una vez concluido el proyecto lo mejor es buscar otra alternativa de alimentación.

El segundo método usa el conector Jack, este admite una alimentación entre 7 a 12 Voltios. Un voltaje menor a 7V provocara que los puertos de salida a 5V del Arduino entreguen una tensión insuficiente provocando errores y un voltaje superior a 12 V provocara un

sobrecalentamiento de la placa desgastándola o dañándola a corto plazo.

Y la tercera forma es utilizar los pines Vin para conectar el cable positivo de la fuente de energía y GND para conectar el cable negativo de la tarjeta Arduino. Al igual que el conector Jack este admite alimentación entre 7 a 12 Voltios.

3.4.2. Conexión USB

Como se mencionó anteriormente, el puerto USB de Arduino se usan para conectar la tarjeta con el computador y de esta forma acceder a la programación a través de la instalación en Windows del software Arduino, los archivos de instalación se pueden descargar directamente de la página web de Arduino https://www.arduino.cc/en/software.

3.4.3. Entradas y salida digitales

Una de las principales funcionalidades de Arduino es precisamente la iteración con el mundo físico mediante la lectura de diferentes parámetros como la apertura o cierre de una ventana, la medición de nivel de agua, el censado de colores, entre otros.

Con estos estímulos de entrada se pueden tomar decisiones y controlar reacciones de salida como el encendido de un motor, el sellado de una válvula, la activación de alarmas, etc.

Esta tarea se lleva a cabo mediante el uso de los pines de comunicación digital, estos se pueden configurar para leer datos (entrada) o para entregar datos (salida) y la configuración se la realiza a través del comando pinMode(). Las

señales digitales usan variaciones de voltaje –Vcc a +Vcc sin pasar por valores intermedios, por tanto, una señal digital dispone únicamente de dos estados, el primero es un estado bajo '0' y el otro es un estado alto '1', estos generalmente se representan por las palabras inglesas LOW y HIGH respectivamente.

3.4.4. Entradas y salida analógicas

Si bien es cierto que los sensores digitales son muy útiles, existe una gran variedad de parámetros físicos que se miden de forma analógica (es decir su comportamiento tienen forma de onda). Para entender la diferencia entre sensores digitales y analógicos tomemos el siguiente ejemplo. Un sensor de luz permite saber si es día o noche, en este caso solo tenemos dos estados (día o noche) por tanto el uso de la

tecnología digital en este tipo de sensor es ideal porque como se vio anteriormente las señales digitales tienen solo dos estados 0 o 1.

Ahora, imagine otro ejemplo. Un sensor de temperatura ambiente que tome lecturas una vez cada minuto durante todo el día podría generar 1440 medidas diferentes, en estos casos lo que se hace es utilizar un sensor analógico. Y bueno, desde la escuela nos han enseñado que los sistemas electrónicos trabajan en ceros y unos (es decir digitalmente), entonces como pueden interpretar estas señales analógicas las tarjetas Arduino, pues la respuesta es sencilla.

Todas las tarjetas cuentan con un pequeño chip de conversión analógico-digital para convertir una lectura analógica en un código digital, este procesamiento es tan rápido que no

es perceptible para el ser humano. Los sensores analógicos son muy comunes porque la mayoría de variables físicas son analógicas (temperatura, humedad, presión, distancia, ubicación geográfica, calidad de aire).

3.4.5. Pines de alimentación

Las tarjetas Arduino cuentan con una serie de conectores para alimentar de energía a los diferentes dispositivos que se conecten a esta. Estos pines se encuentran señalados como 3.3V, 5V GND que significa conexión a tierra y el conector Vin que puede ser utilizado de dos formas, la primera es como alimentación de la tarjeta Arduino y la segunda es para alimentar algún dispositivo con la misma tensión de entrada, es decir, si el Arduino en su puerto Jack tienen 8V, la salida Vin entregará también 8V.

3.4.6. Comunicación serial

La comunicación serial en las placas Arduino es una forma específica de comunicación en la que la información se recibe o se envía bit a bit a través de un canal de comunicación.

Un ejemplo es la transmisión de datos por medio inalámbrico, en este caso el pin TX se conecta al módulo de transmisión para que este a su vez envíe esos datos uno a la vez hasta llegar a un módulo receptor que se conecta al pin RX de otra tarjeta Arduino.

Algunas tarjetas como Arduino Mega tienen varios puertos de TX y RX para conectar varios dispositivos. Un aspecto importante al usar la comunicación serial es la sincronización

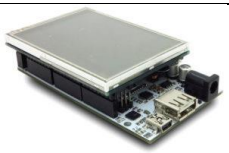
entre dispositivos, en este caso se debe configurar la misma velocidad de transmisión en el módulo emisor como en el receptor. La ventaja de este tipo de comunicación es que se pueden enviar datos a grandes distancias y la desventaja es que la información al ser enviada bit a bit es más lenta.

3.5. Módulos que agregan funcionalidades

Estos son algunos módulos que permiten agregar funcionalidades según las necesidades de cada proyecto.

Tabla 8.

Módulos Arduino.

	**Arduino pantalla TFT Táctil.-** Este componente se encarga de convertir las señales eléctricas de la placa en información visual fácilmente entendible por los seres humanos.
	**Módulo Wifi.-** Con este módulo podemos establecer una comunicación WIFI de manera sencilla y barata.
	**Modulo bluetooth. -** Este módulo está basado en el chip CC2540 de Texas Instruments. Tanto en Google Play como en Apple Store puedes encontrar una aplicación llamada "BLE Arduino" que es capaz de comunicarse directamente con este módulo.

	**Amplificador de audio.-** Con éste módulo puedes recibir una señal de audio vía bluetooth y amplificarla para alimentar directamente dos altavoces de 5W. Incluye conexión para micrófono y pines de control para subir y bajar el volumen, play, siguiente y anterior.
	**Tarjeta Micro SD. -** Este módulo permite almacenar grandes cantidades de información de manera sencilla en una tarjeta de memoria SD. Puedes usarla, por ejemplo, para datalogging, almacenamiento de archivos (mp3, por ejemplo), etc. Soporta los sistemas FAT16 y FAT32.

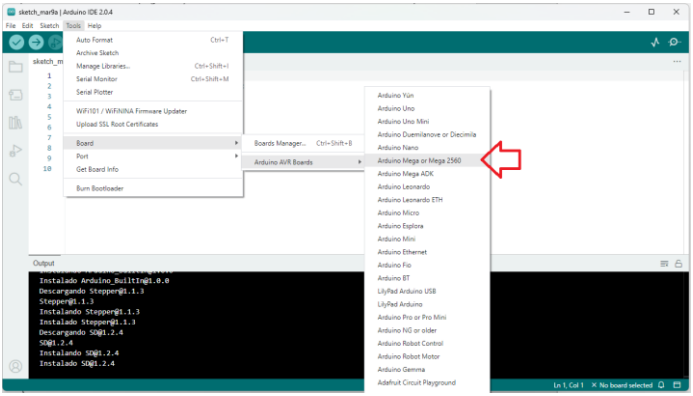
Nota. Adaptado de store.arduino.cc

3.6. Programación en Arduino

Una vez descargado e instalado el software Arduino en nuestro computador y lo tengamos abierto, lo primero que haremos es conectar la

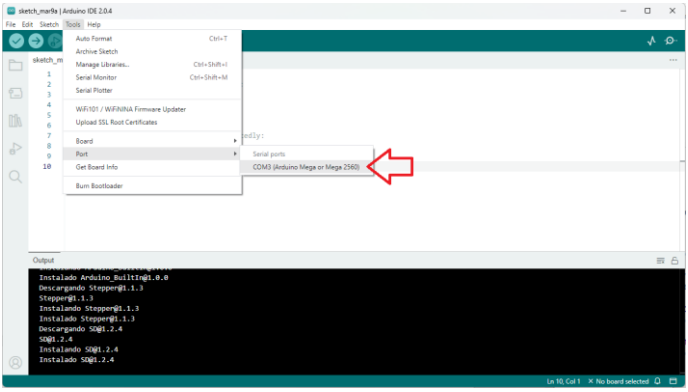
tarjeta Arduino con un cable serial al computador (Ver figura 24). El siguiente paso es configurar el tipo de tarjeta que vamos a utilizar, para ello abrimos el menú Herramientas -> Placa -> placas Arduino (Ver figura 27)

Figura 27.
Configuración del tipo de placa



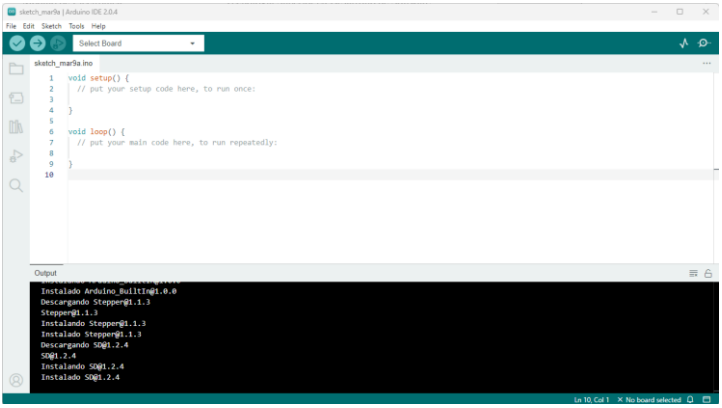
El siguiente paso es configurar el puerto USB de nuestro computador usado para conectar nuestra tarjeta Arduino con el computador. Para ello vamos al menú Herramientas -> Puerto y a seleccionamos aquel puerto que tiene nuestra placa (Ver figura 28)

Figura 28.
Configuración del puerto USB



Ahora ya tenemos todo listo para programar.

Figura 29.
Entorno de programación Arduino



3.6.1. Estructura básica

La estructura de un programa Arduino es bastante simple, divide la ejecución en dos partes: void setup() y void loop().

La función Setup() incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa, hay que tomar en

cuenta que esta función solo se ejecuta una sola vez (cuando se enciende el Arduino) y es empleada para configurar el pinMode (p. ej. si un determinado pin digital va a recibir o enviar datos) e inicializar la comunicación serie.

Por otra parte la función loop() incluye el código a ser ejecutado continuamente durante toda la vida del programa como si fuera un bucle infinito y se utiliza para leer datos de sensores permanentemente, procesarlos y transmitir señales de respuesta.

Ejemplo:

```
void setup(){
        pinMode(13, OUTPUT);    // Definimos al pin 13 como salida digital
}
void loop(){
        digitalWrite(13, HIGH); /* Enciende el LED de la tarjeta Arduino
                                   controlado por el pin13 (ver figura 25) */
        delay(1000);            // Pausa de 1000 ms, es decir 1 segundo
        digitalWrite(13, LOW);  // Apaga el LED
        delay(1000);
}
```

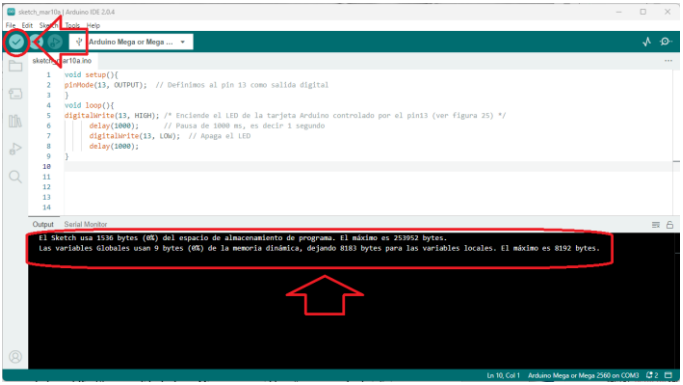
Como la función void setup() solo se ejecuta una vez en toda la vida del programa, esta se usa para declarar al Pin 13 como salida de datos. Mientras que void loop() por ser un bucle infinito permite encender y apagar el diodo led incorporado en la tarjeta Arduino cada 5 segundos de forma indefinida.

Aprovecharemos este ejemplo para identificar algunos elementos de la programación Arduino. Antes que nada, toda línea de código siempre termina con un punto y coma (;). En los programas se pueden agregar comentarios, es decir palabras que no forman parte de la programación pero que son de mucha utilidad para los programadores puesto que allí que escriben frases que nos ayudan a recordar que hace determinada sección del programa. Esto es de especial utilidad en programas extensos. Para agregar varias líneas se usa los símbolos /* para iniciar y */ para terminar el

comentario. Si se desea agregar un comentario pequeño de una sola línea, es suficiente con agregar al inicio del texto slash doble (//).

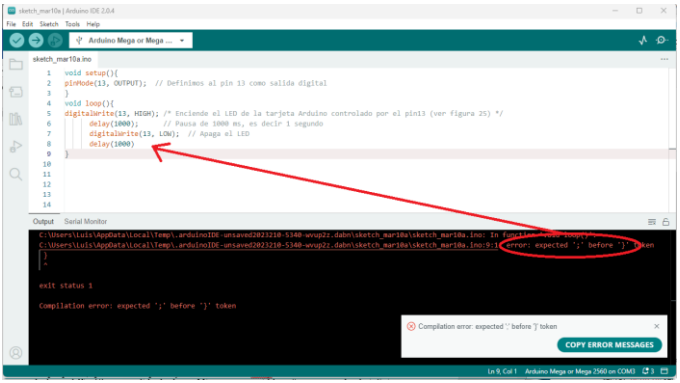
Una vez copiado el código en nuestro computador, podemos verificar errores de sintaxis dando clic en el botón “verificar”, si el programa esta correcto en la parte inferior aparecerá en color blanco información del programa.

Figura 30.
Verificación de código



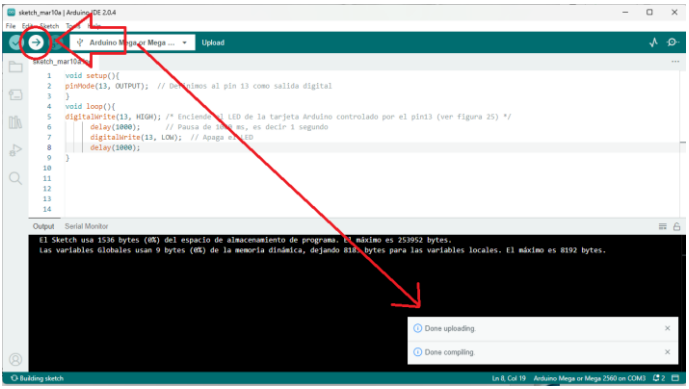
Si nuestro programa tuviera errores, aparecerá en color rojo información de la línea y el tipo de error. A propósito, quitaremos el punto y coma del último delay.

Figura 31.
Error de coMpilación



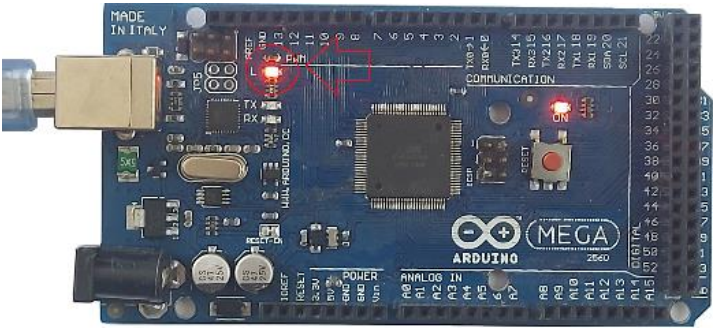
Una vez verificado o corregido nuestro programa, el siguiente paso será subirlo a la memoria de nuestra tarjeta Arduino.

Figura 32.
Carga de código en Arduino



Y ahora si podemos ver como titila el foco de nuestra tarjeta Arduino.

Figura 33.
Led Arduino



3.6.2. Variables y tipos de datos

Una variable es una forma de almacenar un valor o carácter para ser usado en lo posterior dentro del programa. No es obligatorio que los programas tengan variables, pero su uso en todo programa es casi inevitable.

Formato:

```
tipo_dato nombre_variable = valor;
```

Ejemplo:

```
int puertoutilizado = 13;
```

La primera parte *tipo_dato* es una palabra reservada propia del lenguaje de programación (es decir estos nombres no pueden ser usados como nombres de variables) y como su nombre lo indica especifica el tipo de dato que vamos almacenar en la variable, en el ejemplo anterior como 13 es un número entero el tipo de datos

correspondiente es int. De esta forma presentamos una tabla con los tipos de datos aceptados en el lenguaje de programación Arduino.

Tabla 9.

Tipos de datos Arduino

TIPOS DE DATO	MEMORIA QUE OCUPA	RANGO DE VALORES
byte	8 bits	Número sin signo entre 0 y 255
unsignedchar	8 bits	Lo mismo que 'byte'
word	16 bits	Número sin signo entre, 0 y 65535
Unsignedint	16 bits	Lo mismo que 'word'
int	16 bits	Número con signo, entre -32768 y 32767
long	32 bits	Número con signo, entre -2,147,483,648 y 2,147,483,647
float	32 bits	Número con signo, entre 3.4028235E38 y 3.4028235E38
unsignedlong	32 bits	Número sin signo entre 0 y 4294967295
boolean	8 bits	Verdadero/falso
char	8 bits	Número con signo, entre -128 y 127
void	---	Usado en la declaración de funciones que no devuelven ningún valor.

Existen dos formas de declarar una variable, de forma global o localmente. Las variables de globales se apuntan al inicio del programa antes de la función void setup(), mientras que las variables locales se declaran dentro de la función void loop().

La diferencia entre los dos tipos de declaraciones radica en que las variables globales se las puede llamar (es decir utilizar) en cualquier parte del programa incluso si el programa tuviera otras funciones definidas por el usuario (leer siguiente sección), mientras que las variables locales solo se pueden llamar dentro de la función donde fue declarada, y si intentamos usar esta variable fuera de la función de origen obtendremos un error en la compilación.

Ejemplo de variable global

```
int puerto utilizado = 13;
void setup(){
        pinMode(puerto utilizado,
OUTPUT);
}
void loop(){
        digitalWrite(puerto utilizado, HIGH);

        delay(1000);
        digitalWrite(puerto utilizado, LOW);

        delay(1000);
}
```

Ejemplo de variable local

```
void setup(){
        pinMode(13, OUTPUT);
}
void loop(){
        int Puerto utilizado = 13;
        digitalWrite(puerto utilizado, HIGH);

        delay(1000);
        digitalWrite(puerto utilizado, LOW);

        delay(1000);
}
```

Compare estos ejemplos con los ejemplos de las secciones 3.6.1 y 3.6.2.

3.6.3. Operadores

Operadores aritméticos. Todo lenguaje de programación requiere símbolos para realizar operaciones aritméticas como suma (+), resta (-), multiplicación (*), división (/), módulo (%) y asignación (=). Cada operador debe ser usado en correspondencia a un tipo de dato para obtener un resultado óptimo. Por ejemplo, si como resultado de una división se quiere obtener un resultado con decimales habría que trabajar con variables tipo float.

Operadores especiales

Estos operadores aparecen como herencia del lenguaje C (lenguaje nativo de Arduino) y se usan para realizar algunas operaciones de comparación o simplificar algunas operaciones aritméticas básicas. A continuación, la descripción de cada una.

Tabla 10.

Operadores en Arduino

Tipo	Sintaxis	Descripción
Compuestos	++	Incremento
	--	Decremento
	+=	Suma compuesta
	- =	Resta compuesta
	*=	Multiplicación compuesta
	/=	División compuesta
	%=	Modulo compuesto
	&=	and bit a bit compuesto
	|=	or bit a bit compuesto
Incremento /Decremento	x++;	Incrementa x una unidad y devuelve el antiguo valor de x

Tipo	Sintaxis	Descripción
	++x;	Incrementa x una unidad y devuelve el nuevo valor de x
	x--;	Decrementa x una unidad y devuelve el antiguo valor de x
	--x;	Decrementa x una unidad y devuelve el nuevo valor de x
	x += y;	Equivalente a la expresión x = x + y;
	x -= y;	Equivalente a la expresión x = x - y;
	x *= y;	Equivalente a la expresión x = x * y;
	x %= y;	Equivalente a la expresión x = x % y;
	x /= y;	Equivalente a la expresión x = x / y;
Lógicos o booleanos	&&	Equivalente a la función booleana AND
	||	Equivalente a la función booleana OR
	!	Equivalente a la función booleana NOT
Comparación	>	Mayor que
	<	Menor que
	>=	Mayor o igual que
	<=	Menor o igual que
	!=	No es igual que
	==	Es igual que

3.6.4. Estructuras de control

GOTO

Transfiere el flujo del programa de un punto a otro del programa.

Formato

goto nombre;

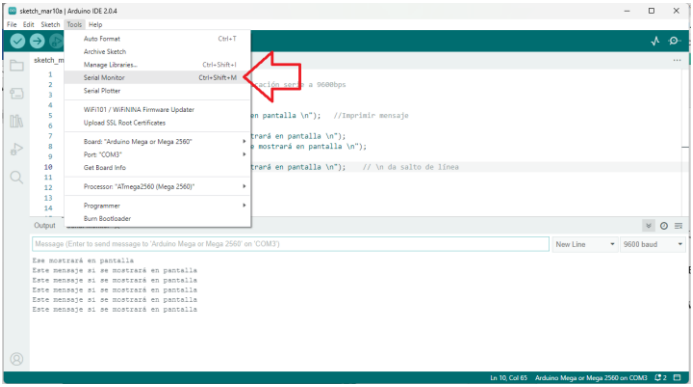
Ejemplo

```
void setup(){
    Serial.begin(9600); // Inicia la comunicación serie a 9600bps
}
void loop(){
    Serial.print("Este mensaje si se mostrará en pantalla \n"); //Imprimir mensaje
    goto saltar;
    Serial.print("Este mensaje no se mostrará en pantalla \n ");
```

```
        Serial.print("Este mensaje tampoco
    se mostrará en pantalla \n");
        saltar:
        Serial.print("Este mensaje si se
    mostrará en pantalla \n ");   // \n da salto
    de línea
        delay(5000);
    }
```

Serial.begin(9600) se usa para indicar que el Arduino inicia una comunicación serial con el computador a 9600bps. En este ejemplo se realiza esta declaración para poder imprimir datos en pantalla del computador a través del puerto serie. Una vez que el programa ha sido cargado en el Arduino podemos abril el monitor dando clic en el menú Herramientas -> monitor serie.

Figura 34.
Monitor serie



BREAK

Se usa dentro de una estructura de comparación o repetición para romper o salir del bucle de forma inesperada.

Formato

Break;

Ejemplo

Ver ejemplo abajo en la estructura CASE

IF-ELSE

En Arduino al igual que todo lenguaje de programación existes estructuras para realizar comparaciones entre variables.

Formato

if (comparación) {

Lista de instrucciones que ejecutará si la condición evaluada es verdadera

} else {

Lista de instrucciones que ejecutará si la condición evaluada es falsa

}

Ejemplo

```
int a;
int b;
void setup(){
        Serial.begin(9600);
}
void loop(){
    a = random(1,100);      //random genera un número aleatorio entre 1 a 100
    b = random(1,100);
    if (a>=b) {
     Serial.print("La variable "); Serial.print(a); Serial.print(" es >= a ");Serial.println(b);
   } else {
```

```
        Serial.print("La variable ");
    Serial.print(a); Serial.print(" es < que
    ");Serial.println(b);
      }
      delay(5000);
    }
```

SWICH-CASE

La sentencia Swich permite seleccionar una opción de entre una lista de valores.

Formato

```
witch (variable) {
        case valor_1:
```

Lista de instrucciones que ejecutará si la condición evaluada es verdadera

```
                break;
        case valor_2:
```

Lista de instrucciones que ejecutará si la condición evaluada es verdadera

break;

case valor_n:

Lista de instrucciones que ejecutará si la condición evaluada es verdadera

break;

default:

Lista de instrucciones que ejecutará si la condición evaluada no coincide

con ninguna anterior

break;

}

Ejemplo

```
int op=1;
void setup(){
```

```
        Serial.begin(9600);
}
void loop(){
   switch (op) {
            case 1:
                    Serial.print("La    variable
op es igual a 1 \n");
                    break;
            case 2:
                    Serial.print("La    variable
op es igual a 2 \n");
                    break;
             case 3:
                    Serial.print("La    variable
op es igual a 3 \n");
                    break;
             default:
```

```
                    Serial.print("La variable
        op es diferente a 1, 2 y 3 \n");
                    break;
          }
          delay(5000);
          op = random(1,4);       //random genera
        un numero aleatorio entre 1 y 4
}
```

FOR

Esta es FOR permite ejecutar repetidamente n veces un grupo de instrucciones.

Formato

```
for (int variable = valor_inicial ; variable
comparación valor_final ; incremento) {
        Lista de instrucciones a ejecutar
}
```

Ejemplo

```
void setup(){
        Serial.begin(9600);
}
void loop(){
        for (int x = 1 ; x <= 12 ; x++) {
        // Imprime la table del 2
                Serial.print("2 x ");
                Serial.print(x);
                Serial.print(" = ");
                Serial.print(2*x);
                 Serial.println();
        // da un salto de línea
        }
        delay(10000);
}
```

WHILE

Es igual que la anterior con la diferencia que esta realiza las repeticiones mientras se cumpla una determinada condición.

Formato

```
while (condición) {
        Lista de instrucciones a ejecutar
}
```

Ejemplo

```
int x=1;
void setup(){
        Serial.begin(9600);
}
void loop(){
        while (x <= 20){
                Serial.print(x);
                Serial.println(" Es menor de
20");
```

```
            x=x+1;
        }
        Serial.print(x);
        Serial.println(" Es mayor que 20");
        delay(5000);
}
```

3.6.5. Funciones definidas por el usuario. Alternativamente a las funciones void setup() y void loop() que son obligatorias en todo programa, se pueden crear otras funciones que ayuden a reducir el tamaño de los programas. En realidad, el uso de funciones es uno de los elementos más importantes dentro de todo lenguaje de programación porque permite crear como pequeños programas que realicen una tarea específica y pueden ser llamados en cualquier parte del void loop() cuantas veces

queramos, de esta forma se reduce enormemente la lectura y escritura del código. En la mayoría de casos las funciones optimizan el uso de memoria, lo cual es muy importante en el ámbito computacional.

Las funciones definidas por el usuario generalmente se agregan al final del programa, después de la función void loop().

Existen funciones que no piden ni devuelven valores. Se los declara con la palabra reservada void.

Formato:

```
void nombrefuncion(){
        Lista de instrucciones a ejecutar
}
```

Ejemplo:

```
void mensaje1 ();              //Las
funciones se deben declarar antes de ser usadas
void mensaje2 ();

void setup(){
        Serial.begin(9600);
}

void loop(){
        mensaje1();
        goto saltar;
        mensaje2();
        saltar:
        mensaje1();
        delay(10000);
}
```

```
void mensaje1 (){
        Serial.print("Este mensaje si se
mostrará en pantalla \n");
}
void mensaje2 (){
        Serial.print("Este mensaje no se
mostrará en pantalla \n");
}
```

Existen funciones que piden valores de entrada, pero no devuelven ningún resultado. Se los declara con la palabra reservada void.

Formato:

```
void nombrefuncion(tipo_dato variable){
    Lista de instrucciones a ejecutar
}
```

Ejemplo

```
void tabla (int num);

void setup(){
        Serial.begin(9600);
}

void loop(){
        int x=2;
        tabla(x);
 }

void tabla (int num){
        for (int j = 1 ; j <= 12 ; j++) {
// Imprime la table del 2
                Serial.print("2 x ");
                Serial.print(j);
                Serial.print(" = ");
```

```
            Serial.println(2*num);
        }
        delay (10000);
    }
```

Existen también funciones que piden valores de entrada y que también devuelven valores de salida. La devolución del valor deseado se lo realiza con la palabra reservada return.

Formato:

```
Tipo_dato nombrefuncion(tipo_dato variable){
        Lista de instrucciones a ejecutar
}
```

Ejemplo

```
int multiplica (int num1, int num2);
void setup(){
        Serial.begin(9600);
}
void loop(){
        int x=2;
        int y=3;
        Serial.println(multiplica(x,y));
        delay(1000);
}
int multiplica (int num1, int num2) {
        return num1*num2;
}
```

3.6.6. Interfaz de Programación de Aplicaciones (API) de Arduino

Arduino ofrece un conjunto de funciones, rutinas y subrutinas que ayudan a realizar tareas de programación, por ejemplo, si de un conjunto de datos necesitamos encontrar el mayor valor no es necesario crear un función definida por el usuario que realice esta tarea porque Arduino ya trae consigo la función max() para realizar esta tarea. A continuación, presentamos las bibliotecas con las funciones más utilizadas en Arduino.

Tabla 11.

Funciones de entrada y salida

Función / Sintaxis	Descripción
pinMode(numeropin, modo)	Configura un pin digital para trabajar como receptor de datos (INPUT) o emisor de datos (OUTPUT).

Función / Sintaxis	Descripción
digitalRead(numeropin)	Se usa para realiza la lectura de un pin digital que previamente haya sido configurado como INPUT.
digitalWrite(numeropin, valor)	Se usa la escritura de un dato en un pin que previamente haya sido configurado como OUTPUT.

Tabla 12.

Funciones de entrada y salida Analógica

Función / Sintaxis	Descripción
analogRead(numeropin)	Permite realizar la lectura de un pin analógico de entrada.
analogReadResolution(bits)	Configura el tamaño en bits del valor devuelto por analogRead(). Su valor predeterminado es 10 bits (devuelve valores entre 0 y 1023).
analogWrite(numeropin, valor)	Permite realizar la escritura de un pin analógico de salida.
analogWriteResolution (bit)	Configura el tamaño en bits del valor a enviar por analogWrite(). Su valor predeterminado es 10 bits.

Tabla 13.

Funciones avanzadas de entrada y salida

Función / Sintaxis	Descripción
tone()	Genera una onda cuadrada con un ciclo de trabajo del 50%. Se utiliza para reproducir sonidos.
noTone()	Para la ejecución de tone().

Tabla 14.

Funciones de gestión de tiempo

Función / Sintaxis	Descripción
delay(milisegundos)	Para la ejecución de un programa un tiempo en milisegundos.
delayMicroseconds (microsegundos)	Para la ejecución de un programa un tiempo en microsegundos.
millis()	Entrega el tiempo en milisegundos que un programa se encuentra ejecutándose en la placa Arduino desde que inició su ejecución.

Función / Sintaxis	Descripción
micros()	Igual que el anterior pero medido en microsegundos.

Tabla 15.

Funciones para comunicación serie

Función / Sintaxis	Descripción
Serial.begin(velocidad en bps)	Inicializa en canal serie para la comunicación entre la placa Arduino y el computador a una velocidad determinada.
Serial.end()	Desactiva la comunicación serie, lo que permite que los pines RX y TX se utilicen para la entrada y salida general

Tabla 16.

Funciones para envío de datos por puerto serie

Función / Sintaxis	Descripción
Serial.print(información)	Envía información parametrizada al canal serie
Serial.println(información)	Igual que el anterior, pero agrega un salto de línea al final.
Serial.write(valor)	Escribe datos en el puerto serie, estos datos se envían como un byte o serie de bytes.

Tabla 17. *Funciones de recepción de datos*

Función / Sintaxis	Descripción
Serial.available()	Devuelve el número de bytes (caracteres) disponibles para leer desde el puerto serie.
Serial.read()	Lee el primer byte del buffer. El byte es eliminado del buffer.
Serial.peek()	Lee el primer byte del buffer. El byte no es eliminado del buffer.

Función / Sintaxis	Descripción
Serial.find(cadena a buscar)	Lee los datos del buffer hasta encontrar una cadena indicada, si lo hace devuelve TRUE, de los contrario FALSE.
Serial.finduntil(cadena a buscar, cadena fin)	Igual que la anterior pero incorpora un parámetro para indicar una cadena de final de búsqueda.
Serial.readBytes(buffer, longitud)	Lee caracteres del puerto serie en un búfer.
Serial.readBytesUntil(carácter, buffer, longitud)	Lee el búfer recibido hasta que recibe un carácter de terminación.
Serial.readString()	Lee caracteres del búfer serie en una cadena.
Serial.setTimeout(tiempo)	Establece el máximo de milisegundos para esperar los datos serie.
Serial.parseFloat()	Devuelve el primer número de coma flotante válido del búfer serie

Función / Sintaxis	Descripción
Serial.parseInt()	Busca el primer dígito numérico válido en el búfer de recepción serie un byte a la vez.

Tabla 18.

Funciones para trabajar con cadenas de texto

Función / Sintaxis	Descripción
String.lenght(String)	Devuelve la longitud de caracteres que tienen una cadena.
String.comparateTo(String)	Realiza una comparación alfabética entre cadenas
String.equal(String)	Compara dos cadenas, distinguiendo entre mayúsculas y minúsculas.
String.equalsIgnoreCase(String)	Igual que la anterior pero no distingue entre mayúscula y minúscula
String.startsWith(String)	Permiten verificar con qué carácter o subcadena comienza otra cadena

Función / Sintaxis	Descripción
String.endsWith()	Permiten verificar con qué carácter o subcadena termina otra cadena

Tabla 19.

Funciones matemáticas

Función / Sintaxis	Descripción
Abs(valor)	Devuelve el valor absoluto de un número.
Min(valor1, valor2)	Devuelve el menor valor de dos números dados.
Max(valor1, valor2)	Devuelve el mayor valor de dos números dados.
Constrain(valor, rango_min, rango_max)	Ajusta el valor de un número para ajustarlo dentro de un rango.
Map(valor, lim_inf_rang_act, lim_sup_rang_act, lim_inf_rang_resultado, lim_sup_rang_resultado)	Reasigna un número de un rango a otro
pow(base, exponente)	Eleva un número a una potencia dada

Función / Sintaxis	Descripción
sq(valor)	Eleva al cuadrado un número
sqrt(valor)	Extrae la raíz cuadrada de un número

Tabla 20.

Funciones trigonométricas

Función / Sintaxis	Descripción
sin(ángulo)	Calcula el seno de una ángulo
con(ángulo)	Calcula el coseno de una ángulo
tan(ángulo)	Calcula el tangente de una ángulo

Tabla 21.

Funciones para aleatoriedad

Función / Sintaxis	Descripción
random(rango_inferior, rango_superior)	Devuelve un número aleatorio comprendido entre dos rangos.

3.6.7. Librerías

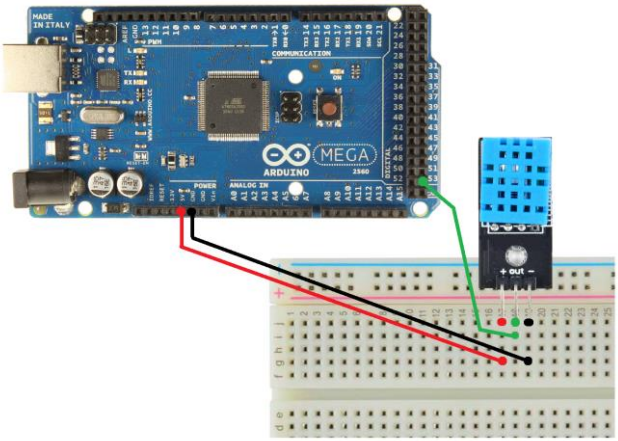
Las librerías son archivos o conjuntos de archivos que agregados al software Arduino le agrega funciones a la programación, normalmente cuando agregamos módulos a los nuestros proyectos debido que por ejemplo un sensor de temperatura tendrá su propia librería que agregue instrucciones para: inicializar el sensor, leer datos en grados, fahrenheit, kelvin, etc.

Para tener un mayor entendimiento vamos ejemplificar, imagine que su proyecto necesita un sensor de temperatura relativa, lo primero será realizar las conexiones con el Arduino. En este ejemplo usaremos un el sensor digital dht11 que mide humedad y temperatura ambiente y

una tarjeta Arduino mega que ya la conocemos.

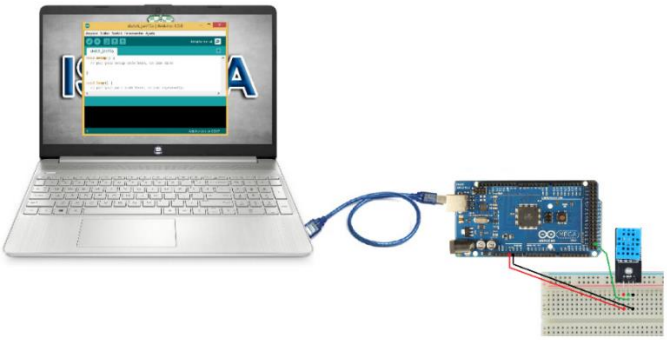
A continuación, el diagrama de conexiones.

Figura 35.
Diagrama de conexiones electrónicas



Conectamos el Arduino al computador a través del puerto USB.

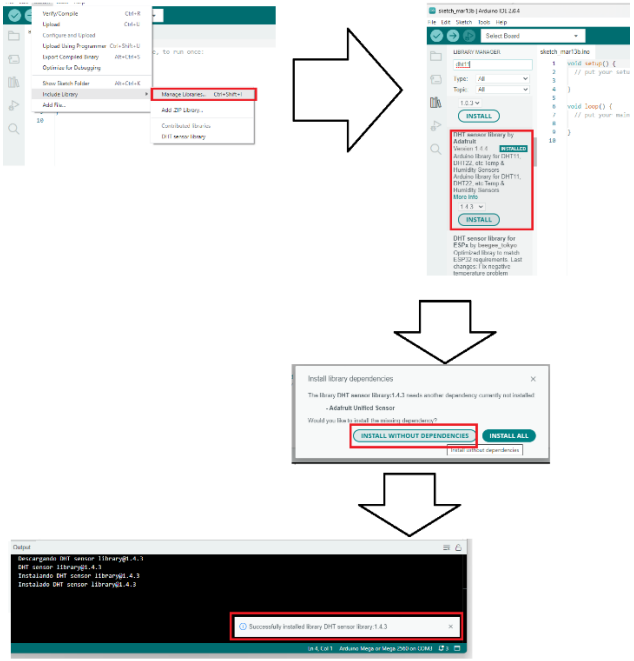
Figura 36.
Conexión con la PC



Buscamos y descargamos la librería del sensor dht11 de internet, https://cdn.shopify.com/s/files/1/0557/2945/files/DHT_11.zip?1491. A continuación, en nuestro programa vamos a Sketch -> Incluir librería -> Add.zip buscamos el archivo descargado y lo abrimos.

Figura 37.

Carga de librerías en Arduino



Una vez agregada la librería ya se puede ya se puede iniciar la programación:

```
#include <DHT.h>                    // Declaración
de librería
#include <DHT_U.h>                  // Declaración
de librería

DHT_Unified dht(53, DHT11);

        void setup() {
          Serial.begin(9600);
          dht.begin();                          //
        Iniciamos el sensor

          Serial.println("LECTURA DEL SENSOR
        DHT11");
        }
```

```
void loop() {
  // Obtener el valor del sensor e imprimir
en el puerto serie
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  Serial.print("Temperatura: ");
  Serial.println(event.temperature);

  dht.humidity().getEvent(&event);
  Serial.print("Humedad: ");
  Serial.println(event.relative_humidity);

delay(5000);
}
```

Corremos el programa, lo cargamos en la tarjeta Arduino y abrimos el monitor serie para visualizar los datos.

Figura 38.

Lectura de sensor DHT11

Referencias

Alcalde, P. (2016). *Electrónica aplicada. Madrid, España*: Ediciones Paraninfo.

Arboledas, D. (2014). *Electricidad básica*. Madrid, España: RA-MA

Ávarez-Láinez, M., Martínez-Tejada, H., & Jaramillo, F. (2019). *Nanotecnología fundamentos y aplicaciones*. Medellín, Colombia: Editorial Universidad de Antioquia.

Beiroa, R. (2018). Aprender Arduino, electrónica y programación con 100 ejercicios prácticos. España: Editorial Marcombo.

Bushong, S. (2022). *Manual de radiología para técnicos física, biología y protección adiológica*. Barcelona, España: Elsevier.

Cerdá, L. (2014). *Instalaciones eléctricas y automatismos*. Madrid, España: Ediciones Paraninfo.

Chetto, M., y Queudet, A. (2020). *Sistemas de tiempo real autónomos en energía*. Great Britain: ISTE Science Publishing.

Corona, L., Abarca, G., y Mares, J. (2019). *Sensores y actuadores, aplicaciones con arduino.* México D.F., México: Editorial Patria.

De la Rosa, J. (2021). *De la micro a nanoelectrónica*. Madrid, España: Editorial CSIC.

Farina, A. (2010). *Cables y conductores eléctricos*. Buenos Aires, Argentina: Alsina.

Fowler, R. (2007). *Electricity principles & applications* (7ma Ed). New York, EEUU: McGraw-Hill

Lleó, A., Lleó, L. (2011). *Gran manual de magnitudes físicas y sus unidades*. Barcelona, España: Ediciones Díaz de Santos.

Seippel, R. G. (2021). *Fundamentos de electricidad principios de electricidad, electrónica, control y ordenadores*. Barcelona, España: Reverté.

Pérez, H. (2018). *Física 1 para bachilleratos tecnológicos*. México D.F., México: Editorial Patria.

Pérez, H. (2019). *Temas selectos de física 2* (3ra ed). México D.F., México: Editorial Patria.

Pérez, M., y Mejía, K. (2021). *Tecnologías de la informática 2*. México D.F., México: Editorial Patria.

Portis, A. (1985). *Campos electromagnéticos.* Barcelona, España: Reverté.

Rengel, R. (2020). Fundamentos físicos de la informática. Salamanca, España: Ediciones Universidad de Salamanca.

Schuler, Ch. (2021). Electrónica, principio y aplicaciones. Barcelona, España: Editorial Reverté.

Zapata, A. (2022). *Química II temas selectos*. Iztapalapa, México: Soluciones educativas Klik.


ISBN: 978-9942-636-22-5
9789942636225