UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

SEDE VICTOR LEVI SASSO

FOLLETO DE

INGENIERÍA DE SISTEMAS ROBÓTICOS

INCLUYE PRUEBAS SUMATIVAS Y PRESENTACIONES DEL CONTENIDO

ELABORADO POR:

DR. CARLOS A. ROVETTO R.

2021

CONTENIDO

Módulo I: INTRODUCCIÓN A LA ROBÓTICA

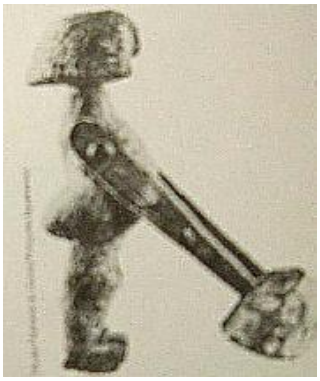
1. CAPÍTULO 1: FUNDAMENTOS DE ROBÓTICA

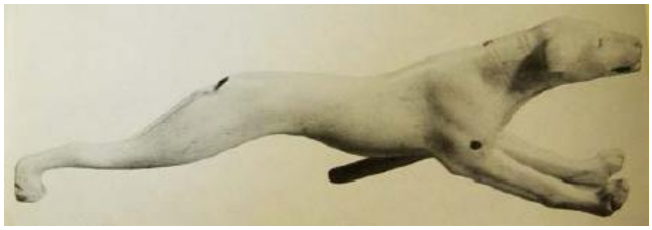
1.1. Antecedentes históricos

Aunque la ciencia de la robótica solo surgió en el siglo XX, la historia de los robots y la automatización inventada por humanos tiene un pasado mucho más largo. De hecho, el antiguo ingeniero griego Hero de Alejandría, produjo dos textos, Pneumatica y Automata, que atestiguan la existencia de cientos de diferentes tipos de máquinas “maravillosas” capaces de movimientos automatizados. Los mecanismos animados de Herón de Alejandría (85 d.C.) se movían a través de dispositivos hidráulicos, poleas y palancas y tenían fines eminentemente lúdicos.

Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses. Estos brazos fueron operados por sacerdotes, quienes clamaban que el movimiento de estos era inspiración de sus dioses.

Los griegos construyeron estatuas que operaban con sistemas hidráulicos, los cuales se utilizaban para fascinar a los adoradores de los templos.





Esculturas animadas

egipcias (2000 a.C)

Muchas fuentes dan fe de la popularidad de los autómatas en la antigüedad y la Edad Media. Los antiguos griegos y romanos desarrollaron autómatas simples para usar como herramientas, juguetes y como parte de ceremonias religiosas.

En la Edad Media, tanto en Europa como en Oriente Medio, los autómatas eran populares como parte de los relojes y el culto religioso. La polímata árabe Al-Jazari (1136-1206) dejó textos que describían e ilustraban sus diversos dispositivos mecánicos, incluido un gran reloj de elefante que se movía y sonaba a la hora, una banda musical de robots y un autómata de camarera que servía bebidas.

En Europa, existe un monje autómata que besa la cruz en sus manos. Se crearon muchos otros autómatas que mostraban animales en movimiento y figuras humanoides que operaban en sistemas de levas simples, pero en el siglo XVIII, los autómatas se entendían lo suficientemente bien y la tecnología avanzaba hasta el punto en que se podían hacer piezas mucho más complejas.

En 1805 Henry Maillardet construyó una muñeca mecánica capaz de hacer dibujos. Una serie de levas se utilizaban como el programa para el dispositivo en el proceso de escribir y dibujar.

Al ingeniero francés Jacques de Vaucanson se le atribuye la creación del primer autómata biomecánico exitoso, una figura humana que toca una flauta. Los autómatas fueron tan populares que viajaron por Europa entreteniendo a jefes de estado como Federico el Grande y Napoleón Bonaparte.

La Revolución Industrial y el mayor enfoque en las matemáticas, la ingeniería y la ciencia en Inglaterra en la época victoriana contribuyeron al impulso hacia la robótica real. Charles Babbage (1791-1871) trabajó para desarrollar los fundamentos de la informática a principios y mediados del siglo XIX, siendo sus proyectos más exitosos el motor de diferencia y el motor analítico. Aunque nunca se completó debido a la falta de fondos, estas dos máquinas establecieron los conceptos básicos para los cálculos mecánicos. Otros, como Ada Lovelace, reconocieron la posibilidad futura de que las computadoras creen imágenes o reproduzcan música.

Los autómatas continuaron brindando entretenimiento durante el siglo XIX, pero coincidió con este período el desarrollo de máquinas y motores a vapor que ayudaron a hacer que la fabricación fuera mucho más eficiente y rápida. Las fábricas comenzaron a emplear máquinas para aumentar las cargas de trabajo o la precisión en la producción de muchos productos.

En 1920 Karel Capek que introdujo la palabra "robot" en su obra R.U.R. (Universal Robots de Rossum) una sátira donde los robots eran seres biológicos fabricados para que realizasen todo el trabajo manual desagradable. El término robot deriva de la palabra checa ***robota*** que significa "trabajo forzado" o "siervo".

Se considera Isacc Asimov como el máximo impulsor de la palabra robot. La palabra robótica apareció por primera vez en la historia de ciencia ficción de Isaac Asimov Runaround (1942). Junto con las historias de robots posteriores de Asimov, estableció un nuevo estándar de plausibilidad sobre la probable dificultad de desarrollar robots inteligentes, estableció las leyes de la robótica y los problemas

técnicos y sociales que podrían resultar. En octubre de 1945, Asimov publicó en la revista Galaxy Science Fiction una historia en la que por primera vez enunció sus tres leyes de la robótica. En la novela Robots e Imperio, publicada en 1985, Asimov incorporó una cuarta ley, conocida como ley cero. Esta ley, de mayor prioridad que la primera, antepone el bien comunitario al individual.

Estas son las cuatro leyes de la robótica establecidas por Asimov:

- **Ley Cero:** Un robot no puede dañar a la humanidad, o a través de su inacción, permitir que se dañe a la humanidad.

- **Primera Ley:** Un robot no puede dañar a un ser humano, o a través de su inacción, permitir que se dañe a un ser humano.

- **Segunda Ley:** Un robot debe obedecer las órdenes dadas por los seres humanos, excepto cuando tales órdenes estén en contra de la Primera Ley.

- **Tercera Ley:** Un robot debe proteger su propia existencia, siempre y cuando esta protección no entre en conflicto con la Primera y la Segunda Ley.

En la década de 1950, George Devol diseñó el Unimate, un dispositivo de brazo robótico que transportaba piezas fundidas en una planta de General Motors en Nueva Jersey, que comenzó a funcionar en 1961. Unimation, la compañía Devol fundada con el empresario robótico Joseph Engelberger, fue el primer fabricante de robots. Los robots se extendieron a Japón, Corea del Sur y muchas partes de Europa.

Los robots han encontrado un lugar en otras esferas, como juguetes y entretenimiento, armas militares, asistentes de búsqueda y rescate, y muchos otros trabajos. Esencialmente, a medida que la programación y la tecnología mejoran, los robots encuentran su camino hacia muchos trabajos que en el pasado han sido demasiado peligrosos, aburridos o imposibles de lograr para los humanos. De

hecho, se están lanzando robots al espacio para completar etapas de la investigación.

1.2. Origen y desarrollo de la robótica

Luego de los primeros autómatas los progenitores más directos de los robots fueron los manipuladores teleoperados. Con el objetivo de manejar elementos radioactivos sin riesgo para el operador se desarrolló el primer sistema de telemanipulación en 1948 por R. C. Goertz del Argonne National Laboratory. A partir de este sistema, en los años setenta, otras industrias comenzaron a utilizar telemanipuladores como lo fueron la industria submarina y la industria espacial. La sustitución del operador que controlaba los movimientos del manipulador por un programa de computadoras condujo al concepto de robot.

George C. Devol estableció las bases del robot industrial moderno. En 1954 Devol concibió la idea de un dispositivo de transferencia de artículos programada, la cual presentó en 1956 a Joseph F. Engelberge, director de ingeniería de la división aeroespacial de la empresa Manning Max-well y Moore en Stanford. Juntos, Devol y Engelberger comienzan a trabajar en la utilización industrial de sus máquinas, fundando la Consolidated Controls Corporation, que más tarde se convierte en Unimation (Universal Automation). En 1961 instalaron su primera máquina Unimate (Máquina de Transferencia Universal) en la fábrica de General Motors. En 1968 J. F. Engelberger visitó Japón y poco más tarde se firmaron acuerdos con Kawasaki para la construcción de robots tipo Unimate.

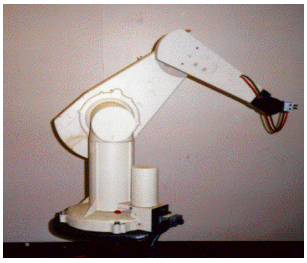
En 1972 se forma la primera asociación robótica del mundo gracias a Nissan, la Asociación de Robótica Industrial de Japón (JIRA). En 1974 se formó el Instituto de Robótica de América (RIA), el cual en 1984 cambió su nombre por Asociación de Industrias Robóticas, manteniendo las mismas siglas (RIA).

A finales de la década de los sesenta se establecen las bases de la investigación en robótica en las universidades. En 1969 se establecen las bases de los diseños

actuales de brazos manipuladores, principalmente por Victor Scheinman, con el diseño del brazo de Stanford, el cual fue el primer brazo robótico manipulador controlado por computador y con accionamiento eléctrico. Este brazo conduciría a desarrollar en 1975 el robot PUMA. Este robot era capaz de mover un objeto y colocarlo en cualquier orientación en un lugar deseado que estuviera a su alcance. El concepto básico multiarticulado del PUMA es la base de la mayoría de los robots actuales.

En 1973, ASEA construyó el IRb6, el primer robot con accionamiento totalmente eléctrico.

En 1982, el profesor Makinode la Universidad Yamanashi de Japón, desarrolla el concepto de robot SCARA (SelectiveCompliance Assembly Robot Arm) que busca un robot con un número reducido de grados de libertad (3 o 4), un coste limitado y una configuración orientada al ensamblado de piezas.



En 1975 el ingeniero mecánico Victor Scheinman desarrolló un manipulador polivalente realmente flexible conocido como Brazo Manipulador Universal Programable (PUMA, siglas en inglés). El PUMA era capaz de mover un objeto y colocarlo en cualquier orientación en un lugar deseado que estuviera a su alcance. El concepto básico multiarticulado del PUMA es la base de la mayoría de los robots actuales.

En 1987 se crea el E1 que es un prototipo que andaba en un paso estático a 0,25 Km/h con una cierta distinción entre el movimiento de las dos piernas.

En 1989 se crea el Genghis, el primer robot andante, es desarrollado en el Massachusetts Institute of Technology.

En 1996 se crea el P2 fue el primer robot humanoide bípedo autoregulable del mundo. Su torso contiene un computador, motores, la batería, una radio inalámbrica y otros controles necesarios para permitir el control inalámbrico. El P2 es capaz de andar de manera autónoma, subir y bajar escaleras y empujar carretillas, todo esto realizado de una manera independiente, sin cables.

En 1999 Sony Lanza el Perro mascota Robótico AIBO.

En el 2000 Honda crea el primer robot humanoide llamado ASIMO, el nombre proviene de Advanced Step in Innovative MObility. Este robot es el resultado de catorce años de investigación por parte de Honda en la mecánica de la locomoción bípeda.

A partir del 2002 se introduce de forma masiva la robótica en los hogares a través de una aspiradora autónoma, denominada Roomba, y desarrollada por la empresa iRobot.

Una de las tendencias futuras en los desarrollos de la robótica va dirigida a aumentar la movilidad, destreza y autonomía de las acciones de los robots, así como también aumentar su capacidad de mantener una elevada interacción con los humanos.

1.3. Definición del Robot

Existen muchas definiciones de robots en la literatura debido al conjunto de sistemas que son entendidos hoy en día por robot. Por tal motivo se dará, en primera

instancia una definición general de robot y luego se definirá a los robots añadiéndolo un adjetivo al término robot el cual nos permite acotar con mayor detalle sus características o campo de aplicación. Por ejemplo, se definirán los robots manipuladores, robots humanoides, robots domésticos, robots aéreos y submarinos, robots caminantes, telerobots, etc.

Definición general de robot: un robot es una máquina programable que es capaz de moverse en la realización de una tarea.

Definición de Robot Industrial Manipulador (ISO 8373): manipulador de 3 o más ejes, con control automático, reprogramable, multiaplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia).

Existen otros tipos de robots que se utilizan en la actualidad para realizar otras tareas distintas a la de fabricar piezas o bienes mediante procesos de manufactura. A estos se les denomina, de manera genérica, con el término de robots de servicio. La Federación Internacional de Robótica (IFR), los define como:

Robots de servicio (IFR): un robot que opera de manera semi o totalmente autónoma para realizar servicios útiles a los humanos y equipos, excluidas las operaciones de manu- factura. La IFR complementa la definición anterior con las siguientes aclaraciones:

Los robots manipuladores industriales, pueden entrar también en la categoría de robots de servicio cuando estén dedicados a tareas no manufactureras. Los robots de servicio pue-den estar equipados o no con un brazo manipulador como los industriales. A menudo, pero no siempre, son robots móviles. En algunos casos, los robots de servicio consisten en una plataforma móvil con uno o varios brazos que son controlados de la misma manera que los robots industriales.

Actualmente, dentro de los robots de servicio se incluyen los robots domésticos o personales. Éstos podrían definirse como:

Robot doméstico: aquel robot destinado a ser usado por humanos sin formación técnica específica, al objeto de servirle como ayudante o colaborador en sus quehaceres o actividades diarias

Otras definiciones muy importantes de robots son:
Robot móvil (ISO 8373): robot que contiene todo lo necesario para su pilotaje y movimiento (potencia, control y sistema de navegación).

Robot Teleoperado (ISO 8373): un robot que puede ser controlado remotamente por un operador humano, extendiendo las capacidades sensoriales y motoras de éste a localizaciones remotas.

1.4. Clasificación de los Robots

Un robot puede ser clasificado atendiendo a diferentes criterios o características o a la aplicación o tarea a que se destinan. A continuación, se presentan algunas de las clasificaciones más utilizadas.

Clasificación de los Robots según su generación o cronología

1ª Generación -- Manipuladores.
Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

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

3ª Generación -- Robots con control sensorizado.

El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

4ª Generación -- Robots inteligentes.

Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

Clasificación de los Robots según su arquitectura

Poliarticulados

Son robots cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo. Entre ellos están los manipuladores, los Robots industriales, los Robots cartesianos.

Móviles

Estos robots tienen una gran capacidad de desplazamiento. Están basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores.

Androides

Estos robots intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Uno de los aspectos más complejos de estos robots es el de la locomoción bípeda, en donde, el principal problema es controlar la dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del robot.

Zoomórficos

Constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. Se pueden agrupar en dos categorías principales: caminadores y no caminadores. El grupo de los robots zoomórficos no caminadores está muy poco evolucionado.

Híbridos

Estos robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo uno de los atributos de los robots móviles y de los robots zoomórficos. También pueden considerarse híbridos algunos robots formados por la yuxtaposición de un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales.

1.5. Morfología del robot

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de potencia y control, y elementos terminales. En las siguientes secciones se explicarán cada uno de estos elementos.

1.5.1. Estructura Mecánica de un Robot

Un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. Este movimiento es producido por los actuadores. El movimiento de cada articulación puede ser de desplazamiento, de giro o una combinación de ambos. De este modo son posibles seis tipos diferentes de articulaciones, aunque en la práctica, en los robots sólo se emplean la de rotación y la prismática.

Tipos de Articulación:

- **Articulación de rotación (1GDL):** suministra un grado de libertad consistente en una rotación alrededor del eje de la articulación. Está articulación es una de la más empleada.
- **Articulación prismática(1GDL):** el grado de libertad consiste en una traslación a lo largo del eje de la articulación.
- **Articulación cilíndrica (2 GDL):** existen dos grados de libertad: una rotación y una traslación.
- **Articulación planar (2 GDL):** está caracterizada por el movimiento de desplazamiento en un plano existiendo, por lo tanto, dos grados de libertad.
- **Articulación esférica o rótula (3 GDL):** combina tres giros en tres direcciones perpendiculares en el espacio.
- **Articulación de tornillo (1 GDL):** se define por consistir en un movimiento rotacional que traslada verticalmente, similar a la articulación cilíndrica.

Se denomina grado de libertad (GDL) a los movimientos independientes que puede realizar cada articulación. Podríamos dar una definición sencilla de grado de libertad como la capacidad de moverse a lo largo de un eje (movimiento lineal) o de rotar a lo largo de un eje (movimiento rotacional). De esta forma, los grados de libertad equivalen al número de parámetros independientes que fijan la situación del elemento terminal. El número de GDL viene dado por la suma de los GDL de las articulaciones que lo componen.

Los elementos terminales son los encargados de interaccionar directamente con el entorno del robot. Estos pueden ser tanto elementos de aprehensión como herramientas y normalmente son diseñados específicamente para cada tipo de trabajo.

1.5.2. Transmisiones y reductores

Las **transmisiones** son elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones, especialmente aquellas situadas en el extremo del robot, para reducir al máximo el momento de inercia del robot debido a que el mismo mueve sus extremos con aceleraciones elevadas.

De igual forma, los pares estáticos dependen directamente de la distancia de las masas al actuador. Es por ello, que se procura que los actuadores, sobre todo los más pesados, estén lo más cerca posible de la base del robot. También, cuando sea necesario, las transmisiones pueden ser utilizadas para convertir movimiento circular en lineal o viceversa.

Los **reductores o engranajes** son elementos encargados de adaptar el par y la velocidad de la salida del actuador a los valores adecuados para el movimiento de los elementos del robot. Generalmente se reduce la velocidad del actuador (de ahí el nombre). Por motivos de diseño, los reductores, tienen una velocidad máxima de entrada admisible, la cual como regla general aumenta a medida que disminuye su capacidad de transmitir par.

Debido a las altas prestaciones que se le piden al robot en cuanto a precisión y velocidad de posicionamiento a los reductores utilizados en robótica se les exige unas condiciones de funcionamiento muy restrictivas. Generalmente, se buscan reductores con las siguientes características: reducido tamaño, bajo rozamiento y peso, pero que, al mismo tiempo sean capaces de realizar una reducción elevada de velocidad en un único paso. También, se tiende a minimizar su momento de

inercia, de negativa influencia en el funcionamiento del motor, especialmente crítico en el caso de motores de baja inercia.

1.5.3. Actuadores

Los actuadores generan el movimiento de los elementos del robot siguiendo las órdenes recibidas de la unidad de control. La mayoría de los actuadores simples controlan únicamente 1 grado de libertad (GDL), el cual puede ser: izquierda-derecha o arriba-abajo. Los primeros robots industriales utilizaban actuadores hidráulicos para mover sus elementos.

Clasificación de los actuadores dependiendo del tipo de energía utilizada por los ejes principales del robot

Cada uno de estos sistemas presenta características diferentes, entre ellas: potencia, controlabilidad, peso y volumen, precisión, velocidad, mantenimiento y coste. A continuación, se explican con más detalle estos sistemas.

Actuadores Neumáticos

La fuente de energía es el aire a presión entre 5 y 10 bar. Los actuadores neumáticos se pueden utilizar para producir tanto movimiento de rotación como movimiento lineal. El diseño básico incluye un cilindro, un pistón y algunos componentes de soporte como válvulas, resortes, tapones, etc. El aire comprimido o gas presurizado se llena en el cilindro y el aire comprimido intenta expandirse para alcanzar la presión atmosférica. Esta expansión se fuerza hacia un pistón o cualquier otro dispositivo mecánico que hace que un objeto adjunto se mueva.

Los actuadores neumáticos se utilizan principalmente para sistemas que requieren una respuesta rápida y precisa. Estos actuadores son limpios, hacen menos ruido y son relativamente compactos en su diseño. Son muy seguros y robustos.

Existen dos tipos de actuadores neumáticos:

- Cilindros neumáticos

- Motores neumáticos

Actuadores Hidráulicos

Utilizan aceites minerales a una presión comprendida normalmente entre los 50 y 10 bar, en ocasiones supera los 300 bar. Son útiles para levantar grandes cargas. Algunos de los problemas que presentan estos actuadores corresponde a que son: complejos, peligrosos (inflamables), difícil mantenimiento (fugas).

Existen dos tipos de actuadores hidráulicos:

- Cilindros
- Motores de aletas y pistones

Los actuadores hidráulicos se utilizan principalmente para sistemas que requieren una fuerza muy grande, pero no muy restrictiva en el posicionamiento y la precisión.

Actuadores Eléctricos

Son los más adecuados para robots que no requieren gran velocidad o potencia, pero que sí requieren precisión y repetitividad, como es el caso de la robótica industrial. Su uso en este sector es especialmente interesante por su sencilla instalación, facilidad de control y fiabilidad.

Existen tres tipos de actuadores eléctricos:

- Motores de corriente continua (DC)
    - Controlados por inducido
    - Controlados por excitación

- Motores de corriente alterna (AC)
    - Síncronos
    - Asíncronos

- Motores paso a paso

1.5.4. Sensores Internos

Los sensores son los dispositivos que permiten a un robot percibir su entorno. Estos permiten a los robots comprender y medir las propiedades geométricas y físicas de los objetos en su entorno circundante, como posición, orientación, velocidad, aceleración, distancia, tamaño, fuerza, momento, temperatura, luminancia, peso, etc.

Los sensores generalmente se clasifican en dos grupos: sensores internos y sensores externos. Los sensores internos, como su sensor de posición, sensor de velocidad, sensores de aceleración, sensor de par motor, etc., obtienen la información sobre el propio robot, mientras que los sensores externos como cámaras, sensores de rango (sensor IR, telémetro láser y sensor ultrasónico) contactan y los sensores de proximidad (fotodiodo, detector de infrarrojos, RFID, táctil, etc.) y los sensores de fuerza recopilan la información del entorno circundante.

Los sensores se definen por varias propiedades que describen sus capacidades:

- Sensibilidad (cambio de salida y cambio de entrada)
- Linealidad (constancia de salida y entrada)
- Tiempo de respuesta (tiempo necesario para que un cambio en la entrada fuerce un cambio en la salida)
- Medida / rango dinámico (diferencia entre mínimo y máximo)
- Precisión (diferencia entre medido y real)
- Repetibilidad (diferencia entre medidas repetidas)
- Resolución (incremento observable más pequeño)
- Ancho de banda (resultado de alta resolución o tiempo de ciclo)

Los sensores se utilizan en robots para una variedad de tareas. Por tanto, un mayor uso de sensores es fundamental para evitar la incertidumbre y lograr una mayor productividad.

1.5.5. Elementos Terminales

También llamados efectores finales, es el dispositivo o herramienta al final del brazo robótico. Tiene como objetivo interactuar con el entorno laboral. Los efectores finales varían y dependen de la aplicación del robot. Los accesorios utilizados pueden ser para agarrar, levantar, soldar, pintar y muchos más. Esto significa que el robot estándar puede llevar a cabo una amplia gama de aplicaciones diferentes dependiendo del efector final que se le instale. Los pies de un robot humanoide o las ruedas de un robot móvil no se consideran efectores finales; se consideran parte de la movilidad del robot.

Tipos de efectores finales

- **Herramientas:** se utilizan en operaciones como la soldadura, pintar o perforar. Sus formas y los tipos son numerosos y variados.

- **Pinzas o dedos mecánicos:** son el mecanismo de agarre de los robots. La pinza puede ser de dos dedos, tres dedos o incluso cinco dedos. En la robótica industrial se utilizan dos pinzas de dedos los cuales pueden ser reemplazados debido al uso gradual. La forma de la superficie de agarre en los dedos se puede elegir de acuerdo con la forma del objeto que será levantado por las pinzas.

  Categorías básicas de pinzas

  - **Pinza mecánica:** se utiliza como efector final en un robot para agarrar los objetos con sus dedos operados mecánicamente.

  - **Pinza neumática:** es un tipo específico de actuador neumático que normalmente involucra movimiento paralelo o angular de superficies.

  - **Pinzas magnéticas:** se utilizan con mayor frecuencia en un robot como efector final para agarrar materiales ferrosos.

1.6. Programación de robots

A través de la programación le indicamos al robot la secuencia de acciones que debe llevar a cabo durante la realización de su tarea. Estas instrucciones contenidas en los algoritmos se traducirán después en un lenguaje de programación inteligible por el sistema de control del robot.

Por medio de la programación el usuario para acceder a las diversas prestaciones del robot, entre ellas, el sistema de control cinemático y dinámico del robot que están encargados de dar la señal de mando a los accionamientos del robot a partir de las especificaciones del movimiento que se les proporciona. También con las entradas-salidas del sistema, consiguiéndose así la sincronización del robot con el resto de las máquinas y elementos que componen su entorno.

1.6.1. Métodos de programación de robots

Existen muchos criterios para clasificar los métodos de programación de robots. Esto se debe a la falta de estandarización de los métodos de programación para robots ya que la mayoría de los fabricantes de hardware de robot también proporcionan su propio software. Una de las clasificaciones más utilizada hace referencia al sistema empleado para indicar la secuencia de acciones a realizar, este lo clasifica de la siguiente forma:

Programación guiada o directa

En este tipo de programación el operario interviene guiando manualmente el brazo del robot, se describen los movimientos y se trazan las trayectorias necesarias para cumplir la función del robot. Para conseguir que el robot funcione de forma autónoma, sin intervención humana, se almacena en la memoria del robot cada uno de los movimientos realizados, de esta forma, se logra que los movimientos puedan ser repetidos posteriormente.

Programación textual o indirecta

Las instrucciones se encuentran escritas en un lenguaje de programación específico, luego este programa se graba en la memoria del robot para que el mismo

realice las acciones indicadas en el mismo. Entre las ventajas que este tipo de programación ofrece están que nos permite realizar operaciones más complejas y con mayor grado de precisión y que es posible establecer relaciones entre el robot y su entorno. Para lograr que el robot actúe en consonancia con su entorno, se debe introducir en el programa los datos procedentes de los sensores.

La programación textual puede ser de dos tipos: explícita y especificativa.

La programación textual explícita: el conjunto de acciones que debe realizar el robot para llevar a cabo su tarea se programa de forma secuenciada y estructurada. También pueden introducirse en las instrucciones las características del medio. Existe una correspondencia entre este tipo de programación y los llamados lenguajes estructurados.

La programación textual especificativa: el programa gira en torno a los elementos manipulados por el robot y las acciones que ha de realizar con ellos, teniendo en cuenta el ámbito en el que se desarrollan dichas acciones. Este tipo de programación está más en consonancia con los lenguajes de programación orientados a objetos.

La programación textual puede ser clasificada en tres niveles: robot, objeto y tarea dependiendo del tipo de órdenes que recibirá el robot, es decir, si se refieren a los movimientos que debe realizar el robot, al estado en que deben ir quedando los objetos manipulados o al objetivo que se pretende conseguir.

Niveles de la programación textual

- **Nivel Robot:** se deben especificar cada uno de los movimientos que realizará el robot, así como también otros valores como lo son: la velocidad, direcciones de aproximación, salida y apertura y cierre de la pinza.

- **Nivel Objeto:** en este tipo de programación las instrucciones se dan en función de los objetos a manejar.
- **Nivel tarea:** en este tipo de programación se especifica qué es lo que debe hacer el robot en lugar de como hacerlo, esto reduce el programa a una única sentencia.

1.6.2. Requerimientos de un sistema de programación de robots

Actualmente, existen una serie de necesidades comunes en los métodos de programación de los robots que nos permite establecer una serie de características generales que se manifiestan en los elementos de programación que éstos contienen. Estos requerimientos generales para un sistema de programación de robots son los siguientes:

Entorno de programación: está relacionado con la programación de las acciones que debe realizar el robot. Dada la interacción que tiene el robot con el entorno ésta resulta una tarea muy tediosa, por lo que, la mayoría de los sistemas de programación de robots optan por una programación interpretada. En este tipo de programación se puede realizar el seguimiento paso a paso, en cada momento, de lo programado.

Modelado del entorno: es la representación que tiene el robot de los objetos con los que interacciona. Generalmente, la posición y orientación de los objetos, y en algunas ocasiones, la forma, dimensiones, peso, etc.

Tipo de datos: además de los tipos de datos usados en los lenguajes de programación, enteros, reales, booleanos, etc., los robots cuentan con otros tipos de datos específicamente destinados a definir las operaciones de interacción con el entorno tales como aquellos que especifican la posición y orientación de los puntos y objetos a los que debe acceder el robot.

Manejo de entradas/salidas: estás le permiten al robot comunicarse con otras máquinas o procesos que cooperan con él. El manejo de las entradas/salidas se puede llevar a cabo mediante señales binarias, digitales o analógicas. Para lograr la integración y sincronización del robot con los procesos de fabricación este manejo es fundamental.

Comunicaciones: necesarias para que el robot pueda comunicarse con los otros robots o máquinas que estén participando de la producción.

Control de movimiento: especifica el movimiento del robot a su destino incluyendo: la trayectoria que puede ser punto a punto, coordinadas o trayectoria continua (línea recta, interpolación circular, etc.), la velocidad indicada como tanto por ciento de una velocidad base definida de manera independientemente, entre otros.

Control del flujo de ejecución del programa: en donde se utilizan las clásicas estructuras de control utilizadas en los lenguajes de programación para especificar el flujo de ejecución de las operaciones que debe realizar el robot. En este entorno de trabajo resulta de gran importancia para controlar mediante las órdenes de un solo programa a varios robots trabajando conjuntamente o un solo robot en una celda de trabajo con equipos controlados por el programa del robot, contar con la capacidad de procesamiento en paralelo. Para lograr este objetivo, se emplean señales de sincronismo basadas en semáforos y ejecución de tareas en paralelo.

1.6.3. Estandarización

Tener una estandarización en los métodos de programación de los robots no ha sido una tarea fácil. A través de los años han existido diferentes intentos para establecer una estandarización que sea aceptada a nivel mundial. Una de las propuestas que ha ganado cierta aceptación, siendo norma DIN desde el año 1996 [DIN-96] y que fue sido incluido como modo de programación en algunos simuladores como COSIMIR de FESTO es la presentada por el Industrial Robot Language (IRL).

Esta propuesta permite programar los movimientos del robot y otras funciones auxiliares a través de la definición para un lenguaje textual de la sintaxis y la semántica. Para dicho lenguaje la sintaxis puede ser traducida a un código intermedio que sirve de interfase común entre lenguajes o procedimientos de programación genéricos. Además, el lenguaje de partida (preproceso) genera de manera automática el código intermedio, el cual, luego es traducido al lenguaje de programación del robot (postproceso).

1.7. Criterios de implantación de un robot industrial

Generalmente, un robot industrial forma parte de un proceso de fabricación que incluye muchos otros equipos, a los cuales se les denomina en su conjunto, como célula de trabajo robotizada. En dicho proceso, el robot forma parte de una estructura de fabricación superior que interactúa con otras máquinas.

El diseño de la célula es uno de los primeros aspectos a considerar si deseamos alcanzar las máximas prestaciones a un robot industrial durante su utilización. Otros aspectos que se deben tomar en cuenta son: seleccionar correctamente el tipo de robot en función a la aplicación que el mismo realizará, la arquitectura de control que se utilizará y el lay-out o plano de implantación del sistema que incluye el esquema de disposición de equipos, las máquinas y otros elementos de la planta. A continuación, se desarrollarán cada uno de estos aspectos con más profundidad.

1.7.1. Diseño de control de una célula de trabajo

El correcto diseño de una célula de trabajo debe definir los distintos elementos con los que contará la célula de trabajo, tales como:

- elementos periféricos pasivos: mesas, alimentadores, utillajes, etc.
- elementos periféricos activos: manipuladores secuenciales, máquinas de Control Numérico, etc.
- seleccionar el robot más adecuada para la aplicación que se va a desarrollar
- definir y seleccionar la arquitectura de control, hardware y software, que todo sistema flexible de fabricación debe incluir

Como resultado de este proceso iterativo debemos obtener la especificación, diseño y ubicación física en el sistema de los elementos periféricos, así como también el tipo y número de robots a utilizar. Es muy importante que el equipo técnico responsable del diseño posea experiencia. También facilita la tarea contar con:

- herramientas informáticas, tales como, sistemas CAD
- simuladores de sistemas de fabricación flexible que nos permite ensayar diferentes estrategias de control de la célula con el objetivo de optimizar su funcionamiento, dimensionar adecuadamente la célula brindando, a la vez, información relacionada con el rendimiento, productividad, y comportamiento que puede presentar frente a los cambios de la demanda o situaciones imprevistas, entre ellas, cambios en el producto, averías, etc.
- simuladores específicos para robots con los que se puede evaluar distintas ubicaciones físicas de los robots en la célula y las alternativas de los robots a utilizar

1.7.2. Características a considerar en la selección de un robot

Al momento de seleccionar un robot existen muchas características que se deben considerar, una de la más importante es evaluar para qué tipo de aplicaciones y de procesos va a realizar el robot. En función de esta primera evaluación, se debe considerar:

- la carga máxima que el robot puede soportar en su trabajo teniendo en cuenta si el robot cargará la pieza de trabajo objetivo de una posición a otra. También el peso de la pieza de trabajo y de la pinza o garras del robot debe adicionarse a dicha carga y que la misma está relacionada con la distancia de la carga final.
- el número de ejes que serán configurados para el robot. Este está directamente relacionado con su grado de libertad, por ejemplo, en una operación como pasar de una cinta trasportadora a otra, un simple robot de 4 ejes es suficiente, mientras que, si la aplicación está en un espacio de

trabajo pequeño y el robot necesita un alto grado de flexión y torsión un robot de 6 o 7 ejes será una buena elección

- las características del controlador del robot, entre ellas, el tamaño compacto y peso ligero; velocidad de procesamiento rápida; capacidad de expansión modular para adaptarse a equipos periféricos adicionales sin tener que comprar un nuevo controlador; facilidad de integración con un sistema de visión, PLC u otros dispositivos; y facilidad de mantenimiento
- debe tener un software de programación fuera de línea asequible
- debe tener un bajo consumo energético. Es importante verificar que los brazos robóticos sean de diseño eficiente, delgados y livianos porque requieren de menos energía debido a que sus motores consumen menos corriente eléctrica y esto puede resultar en importantes ahorros de costos a largo plazo
- que el robot cumpla todos los códigos de seguridad actuales para proteger a los empleados y limitar la responsabilidad de la empresa
- que el robot tenga un momento de inercia máximo admisible elevado. Busque un robot con un momento de inercia máximo permitido alto, una medida de cuánta fuerza puede ejercer. Cuanto mayor sea el momento de inercia máximo permitido, más fácilmente el robot puede levantar y mover un tamaño de carga útil determinado, lo que ejerce menos tensión sobre los motores del robot y aumenta la vida útil

1.7.3. Seguridad en instalaciones robotizadas

La seguridad es un aspecto crítico durante el desarrollo y explotación de una célula robotizada. Esto se debe a que el robot posee un mayor índice de riesgo a un accidente que otra máquina de características similares. Es por ello, que en los sistemas robotizados podemos dividir la seguridad en dos grandes áreas: una de ellas está directamente relacionada con el robot y que involucra la supervisión del sistema de control, velocidad máxima limitada, frenos mecánicos adicionales, comprobación de señales de autodiagnóstico, paradas de emergencia, entre otros, cuya responsabilidad recae en el fabricante.

La otra área está relacionada con el diseño e implantación de la célula robotizada, su programación y mantenimiento. Esta debe considerar distintas medidas de seguridad entre las que se pueden destacar el uso de barreras de acceso a la célula y protecciones en general. Estas medidas básicas intentan minimizar el riesgo de aparición de un accidente. Además de las medidas básicas antes mencionadas, podemos mencionar las siguientes:

- Contar con movimientos condicionados, en donde se programará al robot para que no efectúe ningún movimiento cuando el operario deba entrar al campo de acción del robot, por ejemplo, para alimentar de nuevas piezas al mismo.
- Prever la existencia de zonas de reparación y mantenimiento, dentro del campo de acción del robot, en donde se garantice que el robot no realizará movimientos durante la realización de estos trabajos.
- Contar con condiciones adecuadas en la instalación auxiliar, los cuales abarcan desde aislamientos, sistema eléctrico con protecciones, hasta sistemas neumáticos o hidráulicos correctos.
- Disponer de dispositivos de intercambio de piezas, los cuales serán utilizados cuando el operador deba poner o recoger las piezas situadas dentro del área de trabajo del robot.

1.7.4. Justificación económica

Se deben considerar distintos factores antes de realizar un proyecto de instalación de una célula robotizada. Para realizar el análisis económico es necesario conocer, además de los beneficios resultantes del proyecto relacionados con el aumento de la productividad debido a la utilización de un sistema robotizado y que es consecuencia directa de la realización del trabajo de forma más rápida y con menores interrupciones, lo siguiente:

- el tipo de instalación que se va a desarrollar
- el coste del robot
- coste de inversión que incluye el coste de herramientas y equipos especiales

- costes de instalación, en donde se deben incluir los costes derivados del estudio de planificación y diseño, así como también, los costes específicos de la instalación del sistema y de desarrollo para la utilización del robot.

1.7.5. Mercado de Robots

En 2018 los cinco mercados principales representaron el 74% de los robots industriales instalados a nivel global: China, Japón, República de Corea, Estados Unidos y Alemania.

Asia sigue siendo el mercado de robots industriales que lidera el ranking a nivel mundial. En 2018, las unidades instaladas en China y República de Corea disminuyeron, mientras que en Japón aumentaron considerablemente. En total, Asia creció un 1%.

En Europa, el segundo mercado más grande del mundo, las instalaciones de robots aumentaron un 14% y alcanzaron un nuevo pico por sexto año consecutivo.

En las Américas, la tasa de crecimiento subió un 20%, lo que también constituye un nuevo récord por sexto año consecutivo.

Para 2022 se espera que el segmento de robots industriales abarque un 72% del mercado y siga predominando, según los resultados publicados por la IDC, la principal firma de inteligencia de mercado, servicios de consultoría y conferencias para el sector de Tecnologías de la Información y Telecomunicaciones, en la Worldwide Semiannual Robotics Spending Guide. Los casos de uso más relevantes de los robots industriales son: ensamble, soldadura, mezclado, empaque de productos, inspección y embotellado.

De igual forma, los robots para producción de alimentos están ganando terreno debido a la alta demanda de mano de obra para el trabajo dentro de esta actividad.

Con relación a los robots de servicios, los casos de uso más importantes son: empacado, inspección de tuberías, producción agrícola y seguridad. En materia de agricultura, los casos de uso están principalmente en la recolección de frutas y verduras.

1.8. Aplicaciones de los robots

En la actualidad podemos ver una gran variedad de robots utilizados para diferentes aplicaciones en el mundo y, a medida que avanza la tecnología, es probable que los robots desempeñen un papel cada vez más importante en nuestras vidas. Los diferentes tipos de robots y sus aplicaciones varían mucho, desde robots industriales hasta robots de conserjería en edificios comerciales. A continuación, se presentan las distintas aplicaciones para los robots industriales y los de servicio.

1.8.1. Aplicaciones industriales de los robots

Los robots industriales se utilizan para automatizar aplicaciones en toda su línea de producción para ahorrar tiempo y dinero. Los robots industriales reducen los residuos y producen productos de mayor calidad con precisión continua. Los robots también pueden manejar las aplicaciones de fabricación más tediosas y peligrosas para mantener a sus trabajadores seguros, saludables y motivados.

Los tipos de robots industriales en servicio hoy en día varían mucho, pero generalmente son brazos articulados programables que son estacionarios pero capaces de moverse en varios ejes. Se utilizan en una serie de aplicaciones en la fabricación de bienes, como soldadura, pintura y montaje. Los robots industriales son extremadamente rápidos y precisos y ofrecen a los fabricantes grandes beneficios de ahorro de costes. Los robots industriales también se utilizan en el espacio exterior para la construcción y otras tareas.

Los robots han revolucionado el lugar de trabajo industrial en todas las industrias desde su introducción al panorama de la fabricación. Estas son las tareas específicas para las que están diseñados los robots industriales tradicionales:

Soldadura por arco: la soldadura por arco, o soldadura por robot, se convirtió en algo común en la década de 1980. Una de las fuerzas impulsoras para cambiar a la soldadura robotizada es mejorar la seguridad de los trabajadores frente a las quemaduras por arco y la inhalación de humos peligrosos.

Soldadura por puntos: la soldadura por puntos une dos superficies metálicas en contacto dirigiendo una gran corriente a través del punto, que derrite el metal y forma la soldadura entregada al punto en muy poco tiempo (aproximadamente diez milisegundos).

Manipulación de materiales: los robots de manipulación de materiales se utilizan para mover, embalar y seleccionar productos. También pueden automatizar funciones involucradas en la transferencia de piezas de un equipo a otro. Se reducen los costos de mano de obra directa y se eliminan muchas de las actividades tediosas y peligrosas que tradicionalmente realiza la mano de obra humana.

Cuidado de máquinas: la automatización robótica para el cuidado de máquinas es el proceso de carga y descarga de materias primas en la maquinaria para procesar y supervisar la máquina mientras realiza un trabajo.

Pintura: la pintura robótica se utiliza en la producción de automóviles y en muchas otras industrias, ya que aumenta la calidad y consistencia del producto. Los ahorros de costos también se obtienen mediante menos reprocesos.

Embalaje y Paletizado: la mayoría de los productos se manipulan varias veces antes del envío final. La preparación y el embalaje robóticos aumentan la velocidad y la precisión, además de reducir los costos de producción.

Montaje: los robots ensamblan productos de manera rutinaria, eliminando tareas tediosas y tediosas. Los robots aumentan la producción y reducen los costos operativos.

Corte, rectificado, desbarbado y pulido mecánicos: incorporar destreza a los robots proporciona una opción de fabricación que, de otro modo, sería muy difícil de automatizar. Un ejemplo de esto es la producción de implantes ortopédicos, como articulaciones de rodilla y cadera. Pulir y pulir una articulación de la cadera a mano normalmente puede llevar entre 45 y 90 minutos, mientras que un robot puede realizar la misma función en solo unos minutos.

Materiales de pegado, sellado adhesivo y pulverización: los robots selladores están construidos con numerosas configuraciones de brazos robóticos que permiten al robot aplicar adhesivos a cualquier tipo de producto. El principal beneficio de esta aplicación es una mayor calidad, velocidad y consistencia del producto final.

Otros procesos: estos incluyen robots de inspección, corte por chorro de agua y soldadura.

1.8.2. Aplicaciones de los robots de servicio

La mayoría de los robots de servicios profesionales son robots semiautónomos o totalmente autónomos que tienen cierta movilidad e interactúan con las personas, generalmente en un entorno minorista, hotelero, sanitario, de almacén o de cumplimiento, mientras que otros se utilizan en entornos más exigentes, como en el espacio y la defensa, aplicaciones agrícolas y trabajo policial.

A diferencia de los robots industriales, la robótica de servicios está diseñada para darles más flexibilidad a la hora de reaccionar ante diferentes situaciones y entornos en los que prestan el servicio a los humanos. En este sentido, la interacción de un robot con humanos se vuelve más complicada y desafiante.

Existen diferentes tipos de robots de servicio. Estos realizan múltiples tipos de tareas para los humanos. Los desarrollos en la industria de la robótica brindan robots de servicios que son más manejables e inteligentes, más inteligentes y mucho menos costosos. Los beneficios de los robots de servicio son su mayor flexibilidad. Ahora, los robots de servicio se pueden utilizar tanto para servicios personales como profesionales. Al igual que todos los demás tipos de automatización robótica, un robot de servicio a menudo mejora la recopilación y el análisis de datos, que son útiles para una mayor operación y optimización comercial.

Los robots de servicio actualmente tienen aplicaciones en múltiples áreas y campos algunos ejemplos de ellos son:

Robots de servicio minorista: pueden descubrir cualquier error en las etiquetas de los estantes y evitar mostrar un precio diferente al que se muestra en la etiqueta del estante. Estos robots de servicio generalmente se mueven de un pasillo a otro para descubrir cualquier información errónea en las etiquetas de los estantes utilizando un sistema de cámara incorporado.

Robots de servicio doméstico: se encargan de las tareas habituales que los humanos realizarían normalmente fuera del entorno industrial. Estos servicios pueden incluir cosas como tareas de limpieza del hogar, corte de césped y mantenimiento de piscinas. Estos servicios son especialmente importantes para las personas mayores y las personas con discapacidades. Muy pronto, estos ejemplos de robots de servicios harían posible que estas clases de personas vivieran de forma independiente.

Robots de servicio científico: llevan a cabo tareas recurrentes comúnmente involucradas en la investigación, como la recopilación de datos, el análisis y la formación de hipótesis. Algunos ejemplos son los robots de servicio de investigación en aguas profundas y el espacio ultraterrestre.

Robots de servicio de eventos: se utilizan para atender a clientes comerciales y visitantes en eventos. Estos robots de servicio ofrecen a los asistentes al evento una gran interacción.

Robots de servicio agrícola: se están incorporando cada día más a todas las facetas de la actividad agrícola. Usando sensores avanzados, estos robots de servicio rocían para el control de malezas, cosechan cultivos, plantan semillas y podan plantas y árboles existentes.

Robots médicos: son robots de servicio profesional que se utilizan dentro y fuera de los entornos hospitalarios para mejorar el nivel de atención al paciente. Estos robots reducen la carga de trabajo del personal médico, lo que les permite dedicar más tiempo al cuidado directo de los pacientes.

Robots de servicios médicos: incluyen robots de telepresencia para cuidados remotos, robots desinfectantes para reducir las infecciones adquiridas en el hospital, robots que pueden extraer sangre de manera precisa y eficiente y exoesqueletos robóticos que brindan apoyo externo y entrenamiento muscular para la rehabilitación. Los robots médicos móviles se utilizan para la entrega de medicamentos y otros materiales sensibles en

un hospital. Hay bots de chat que diagnostican a los pacientes a través de mensajes de texto, utilizando inteligencia artificial.

Robots de servicios logísticos: son vehículos móviles guiados automatizados en almacenes e instalaciones de almacenamiento para transportar mercancías. Estos robots siguen rutas predefinidas, moviendo productos para su envío y almacenamiento durante todo el día. Desempeñan un papel importante en la reducción del costo logístico.

Módulo II: LA ROBÓTICA CON ARDUINO: HARDWARE Y SOFTWARE ARDUINO

2. CAPÍTULO 2. INTRODUCCIÓN A LA ROBÓTICA CON ARDUINO.

2.1. Arduino como Plataforma Electrónica de Código Abierto

Arduino nació en el Ivrea Interaction Design Institute como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación.

Tan pronto se popularizó, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde placas simples de 8 bits hasta productos para aplicaciones de IoT, impresión 3D y entornos integrados.

Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto.

Arduino es una plataforma prototipo basada en un hardware y software fáciles de usar. Consiste en una placa de circuito, que se puede programar (denominada microcontrolador) y un software listo para usar llamado Arduino IDE (Entorno de desarrollo integrado), que se utiliza para escribir y cargar el código de la computadora en la placa física.

Algunas de las principales características que han hecho popular al Arduino son:

- Código abierto y extensible.

- Uso sencillo para principiantes y a la vez flexible para los usuarios más avanzados.

- Se ejecuta en Mac, Windows y Linux.

- Bajo coste.

El software Arduino se publica como herramienta de código abierto, disponible para su extensión por programadores experimentados.

El lenguaje se puede expandir a través de bibliotecas C++, y las personas que deseen comprender los detalles técnicos pueden trabajar directamente con el lenguaje de programación AVR C en el que se basa.

Del mismo modo, se puede agregar código AVR-C directamente en sus programas Arduino si desea.

El software Arduino (IDE) se ejecuta en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los sistemas de microcontroladores están limitados a Windows.

2.2. Medidas de Seguridad con la Plataforma Electrónica Arduino

Se debe tener en cuenta que la plataforma es electrónica y aunque sea de baja potencia, se puede dañar tanto la placa como el ordenador donde se tiene conectada.

Algunas medidas que se deben tener presentes al trabajar con esta plataforma para garantizar la seguridad son:

- No trabajar sobre una mesa metálica o cualquier otro material conductor de electricidad.

- Se debe desconectar la placa cada vez que se deba cambiar o mover los cables, quitar o poner un led, etc.

- No conectar dispositivos que requieran más de 40mA de manera directa.

- Utilizar fuentes de alimentación estables y seguras.
- Tener un área de trabajo limpia y evitar tener vasos o bebidas cerca que se puedan derramar.

- Se puede medir con un multímetro y múltiples conexiones, la conductividad entre tierra y los pines de potencia para asegurar que no haya ningún corto.

También es recomendable revisar:

- la conexión de todas partes antes de colocar la energía
- el consumo de corriente que requiere tu dispositivo
- que el sensor o dispositivo a conectar no entregue voltajes mayores de 5V
- la polaridad de los cables
- que los cables estén en buen estado

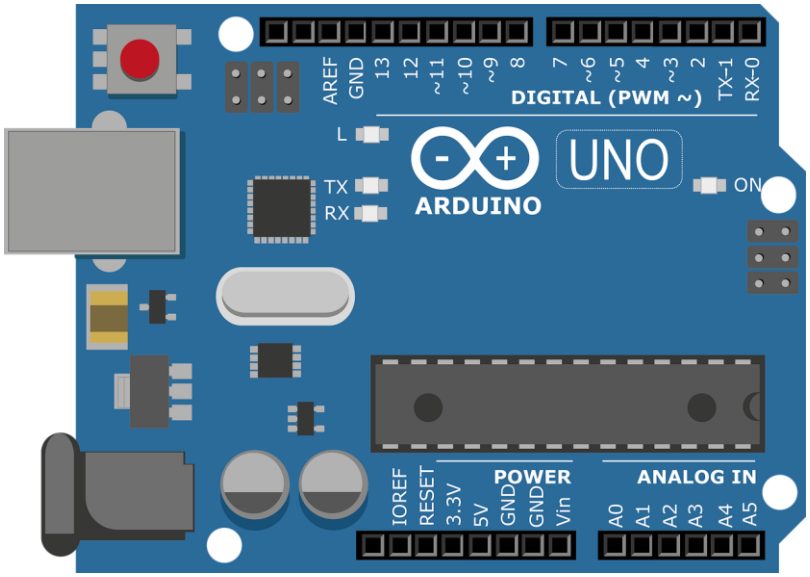
2.3. Uso de Arduino en la Robótica

Esta plataforma de código abierto ha sido de gran ayuda tanto para principiantes como para expertos. Miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos se han realizado con Arduino. Las contribuciones de la comunidad que la utiliza se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda para los que trabajan con ella.

Actualmente, lo utilizan mucho en el ámbito educativo los docentes y los estudiantes para iniciarse en la electrónica, la programación y la robótica. Se desarrollan muchos proyectos tecnológicos destinados a la educación basados en Arduino. Esto les permite a los estudiantes incursionar en la robótica y desarrollar nuevos proyectos relacionados con las impresoras en 3D o los drones gracias a que esta plataforma es de software y hardware de código abierto.

Además de que muchos programadores, aficionados, diseñadores y educadores han introducido numerosos desarrollos que le han permitido ampliar el tipo de software utilizado para programar nuevo hardware.

2.4. Hardware Arduino



El hardware de Arduino está formado por distintas partes como son entradas, salidas, alimentación, comunicación y shields. Cada una de ellas se explicarán a continuación.

2.4.1. Placas Arduino de Entrada / Salida

Hay varios tipos de placas Arduino disponibles en función de los diferentes microcontroladores utilizados. Todas las placas Arduino están programadas a través del IDE de Arduino.

Las diferencias entre las placas se basan en la cantidad de entradas y salidas lo que incide en la cantidad de sensores, LED y botones que puede usar en una sola placa, velocidad, voltaje de funcionamiento, etc. Algunas placas están diseñadas para integrarse y no tienen una interfaz de programación (hardware), la cual es necesario comprar por separado. Algunas pueden funcionar directamente con una batería de 3,7 V, mientras que otras, necesitan al menos 5 V.

Las placas Arduino están basadas en el microcontrolador que poseen y se pueden clasificar de la siguiente forma:

- Placas Arduino basadas en microcontrolador ATMEGA328

Arduino Uno R3
Arduino Uno R3 SMD
Red Board
Arduino mini 05

- Placas Arduino basadas en microcontrolador ATMEGA32u4

Arduino Leonardo
Pro micro 5V/16MHz
Pro micro 3.3V/8MHz
LilyPad Arduino USB

- Placas Arduino basadas en microcontrolador ATMEGA2560

Arduino Mega 2560 R3
Mega Pro 3.3V
Mega Pro 5V

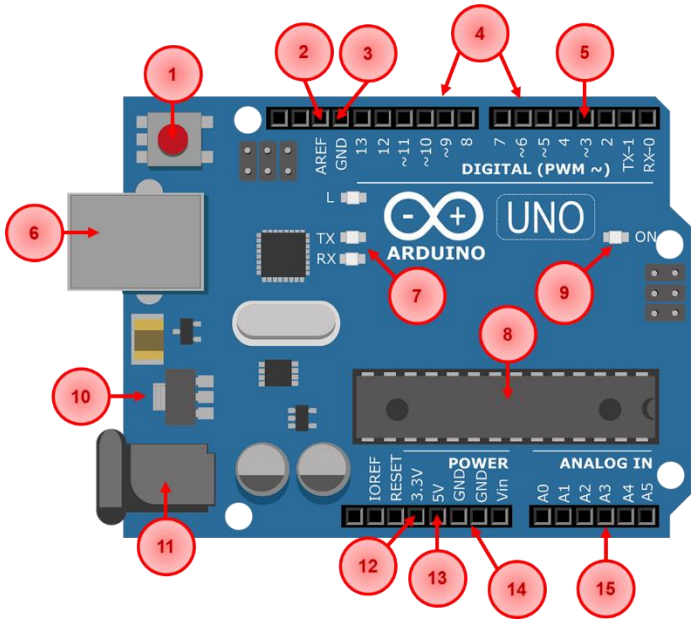
2.4.2. Entradas/Salidas y esquema de pines de la placa Arduino

En las siguientes dos subsecciones de explican los tipos de entradas y salidas de la placa Arduino y su ubicación en la misma.

2.4.2.1. Entradas y salidas de la placa Arduino

1. **Botón de reinicio:** esto reiniciará cualquier código que se cargue en la placa Arduino.
2. **AREF:** Significa "Referencia analógica" y se utiliza para establecer una referencia de voltaje externo.
3. **Pin de tierra:** hay algunos pines de tierra en el Arduino y todos funcionan igual.
4. **Entrada / salida digital:** los pines 0-13 se pueden usar para entrada o salida digital.
5. **PWM:** The pins marked with the (~) symbol can simulate analog output.
6. **Conexión USB:** se utiliza para encender su Arduino y cargar proyectos.
7. **TX / RX:** LED de indicación de transmisión y recepción de datos.
8. **Microcontrolador ATmega:** este es el cerebro y es donde se almacenan los programas.
9. **Indicador LED de alimentación:** este LED se enciende cada vez que la placa está conectada a una fuente de alimentación.
10. **Regulador de voltaje:** controla la cantidad de voltaje que ingresa a la placa Arduino.
11. **Conector de barril de alimentación de CC:** se utiliza para alimentar su Arduino con una fuente de alimentación.
12. **Pin de 3,3 V:** este pin proporciona 3,3 voltios de potencia a sus proyectos.
13. **Pin de 5V:** este pin suministra 5 voltios de potencia a sus proyectos.
14. **Pines de tierra:** hay algunos pines de tierra en el Arduino y todos funcionan igual.
15. **Pines analógicos:** estos pines pueden leer la señal de un sensor analógico y convertirla en digital.

2.4.2.2. Esquema y pines de la placa Arduino



2.4.3. Placa Arduino

Dependiendo de nuestras necesidades Arduino dispone de una amplia variedad de placas y shields para usar. Un shield es una placa compatible que se puede colocar en la parte superior de los arduinos y permite extender las capacidades del arduino. A continuación, se explica con mayor detalle las características que poseen las placas de Arduino.

2.4.3.1. Características técnicas y físicas de la placa Arduino

Algunas de las características que posee la placa Arduino son:

- Las placas Arduino pueden leer señales de entrada analógicas o digitales de diferentes sensores y convertirlas en una salida, como activar un motor, encender / apagar LED, conectarse a la nube y muchas otras acciones.

- Puede controlar las funciones de su placa enviando un conjunto de instrucciones al microcontrolador en la placa a través de Arduino IDE (conocido como software de carga).

- A diferencia de la mayoría de las placas de circuitos programables anteriores, Arduino no necesita una pieza adicional de hardware (llamada programador) para cargar un nuevo código en la placa. Simplemente puede usar un cable USB.

- El IDE de Arduino utiliza una versión simplificada de C ++, lo que facilita el aprendizaje de la programación.

- Arduino proporciona un factor de forma estándar que divide las funciones del microcontrolador en un paquete más accesible.

- Para trabajar con Arduino es importante una placa de pruebas sin soldadura. Este dispositivo le permite crear un prototipo de su proyecto Arduino sin tener que soldar permanentemente el circuito. El uso de una placa (breadboard) le permite crear prototipos temporales y experimentar con diferentes diseños de circuitos. Dentro de los orificios (puntos de unión) de la carcasa de plástico, hay clips de metal que están conectados entre sí mediante tiras de material conductor.

- Se utilizan cables para formar los circuitos conectando resistencias, interruptores y otros componentes juntos.

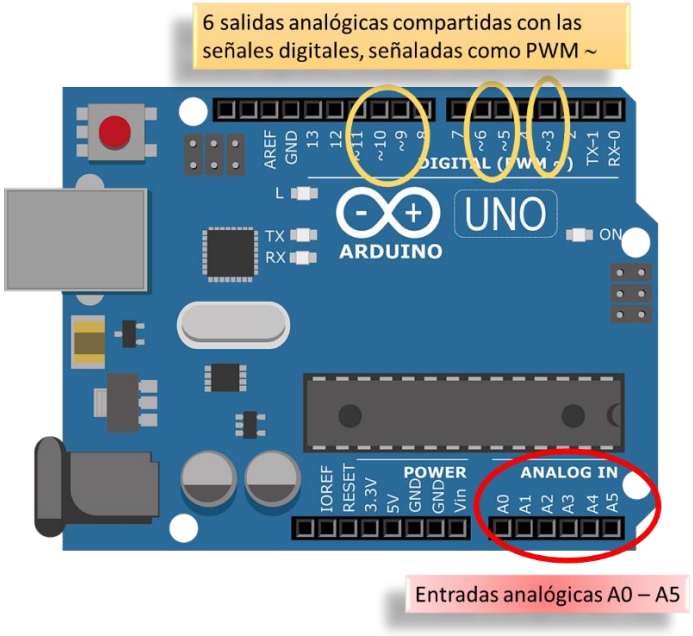
2.4.3.2. Entrada y Salida de la placa Arduino

Existen dos tipos de entradas y salidas de la placa Arduino las analógicas y las digitales.

Cada una de ellas se describen a continuación:

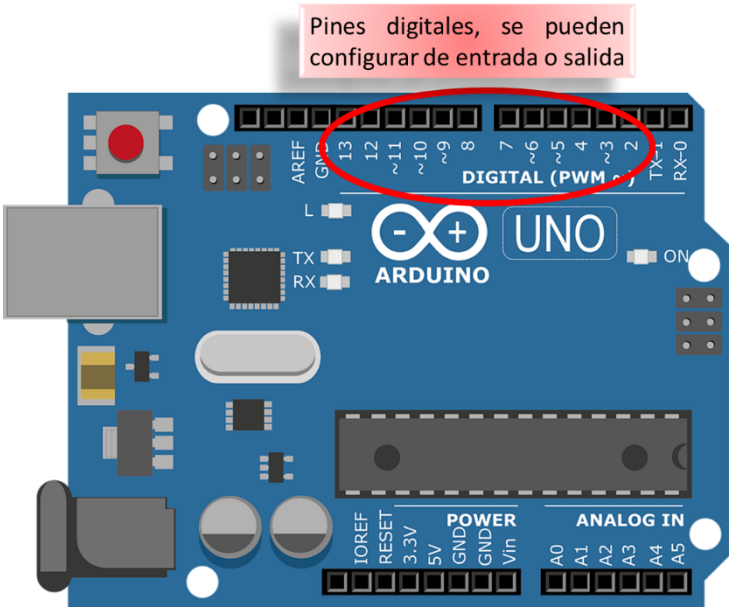
- **Entradas/salidas analógicas**

Las entradas analógicas van marcadas con una A delante. La cantidad de entradas que tenga cada placa depende del modelo, por ejemplo, el Arduino UNO posee 6 entradas analógicas (A0-A5), mientras que otros modelos como el NANO posee 8 entradas y el MEGA, 16 entradas. Aunque estas pueden ser utilizadas como salida, su uso más común es la lectura de datos de dispositivos analógicos. Una de las restricciones que tiene estas entradas es que no se pueden aplicar voltajes superiores a 5 voltios.



- **Entradas/salidas digitales**

Son pines que pueden ser activado, o sea poner tensión, o desactivados, es decir quitarle la tensión. Este proceso es similar a escribir 0 que equivale a 0 V y 1 para 5V.



2.4.3.3. Forma de alimentación de la placa Arduino

Debido a que Arduino puede suministrar solo alrededor de 40ma de corriente a través de cualquiera de sus pines de salida, es severamente limitado a lo que puede alimentar por sí mismo. Un LED rojo típico de 5 mm requiere aproximadamente 30ma de corriente, por lo que el Arduino no tiene problemas para encenderlo hasta el 100%, pero cualquier otro dispositivo que requiera más corriente tendrá problemas para activarlo. Usar el Arduino para controlar un dispositivo de alta potencia requiere el uso de un "amplificador". También llamado "búfer de señal", un amplificador simplemente reproduce una señal de entrada de baja potencia, con una potencia de salida mucho mayor para conducir una carga.

El Arduino Uno necesita una fuente de alimentación para que funcione y se puede alimentar de varias formas. Puede conectar la placa directamente a su computadora mediante un cable USB. Si desea que su proyecto sea móvil, considere usar una batería de 9V para darle energía. El último método sería utilizar una fuente de alimentación de 9 V CA.

2.4.3.4. Forma de comunicación de la placa Arduino

Se han definido muchos protocolos de comunicación para lograr el intercambio de datos. Cada protocolo se puede clasificar en una de las dos categorías: paralelo o en serie.

- **Comunicación paralela**

La conexión en paralelo entre el Arduino y los periféricos a través de puertos de entrada / salida es la solución ideal para distancias más cortas de hasta varios metros. Sin embargo, en otros casos cuando es necesario establecer comunicación entre dos dispositivos para distancias más largas, no es posible utilizar una conexión en paralelo. Las interfaces paralelas transfieren varios bits al mismo tiempo. Por lo general, requieren buses de datos, que se transmiten a través de ocho, dieciséis o más cables. Los datos se transfieren en oleadas enormes y estrepitosas de 1 y 0.

- **Módulos de comunicación en serie**

Hoy en día, la mayoría de las placas Arduino están construidas con varios sistemas diferentes para la comunicación en serie como equipo estándar.

La decisión de cuál sistema utilizar depende de los siguientes factores:

- ¿Con cuántos dispositivos tiene que intercambiar datos el microcontrolador?
- ¿Qué tan rápido debe ser el intercambio de datos?
- ¿Cuál es la distancia entre estos dispositivos?
- ¿Es necesario enviar y recibir datos simultáneamente?

Una de las cosas más importantes en relación con la comunicación en serie es el Protocolo, que debe observarse estrictamente. Este es un conjunto de reglas, que deben aplicarse de manera que los dispositivos puedan interpretar correctamente los datos que intercambian mutuamente. Afortunadamente, Arduino se encarga

automáticamente de esto, de modo que el trabajo del programador o usuario se reduce a una simple escritura (datos a enviar) y lectura (datos recibidos).

2.4.3.5. Forma de programación de la placa Arduino

Una vez que se haya creado el circuito en la placa de pruebas, deberá cargar el programa, conocido como sketch (boceto), en Arduino. El sketch es un conjunto de instrucciones que le dice al tablero qué funciones necesita realizar. Una placa Arduino solo puede contener y realizar un sketch a la vez. El software utilizado para crear sketches de Arduino se llama IDE, que significa Entorno de Desarrollo Integrado.

El código que se escriba es "legible por humanos" y estará organizado para que lo siga un humano. Parte del trabajo del IDE es tomar el código legible por humanos y traducirlo en código legible por máquina para ser ejecutado por Arduino. Este proceso se llama compilación.

Sintaxis

- Un punto y coma debe seguir cada declaración escrita en el lenguaje de programación Arduino.

- Se utiliza doble barra inclinada (//) para agregar comentarios al código.

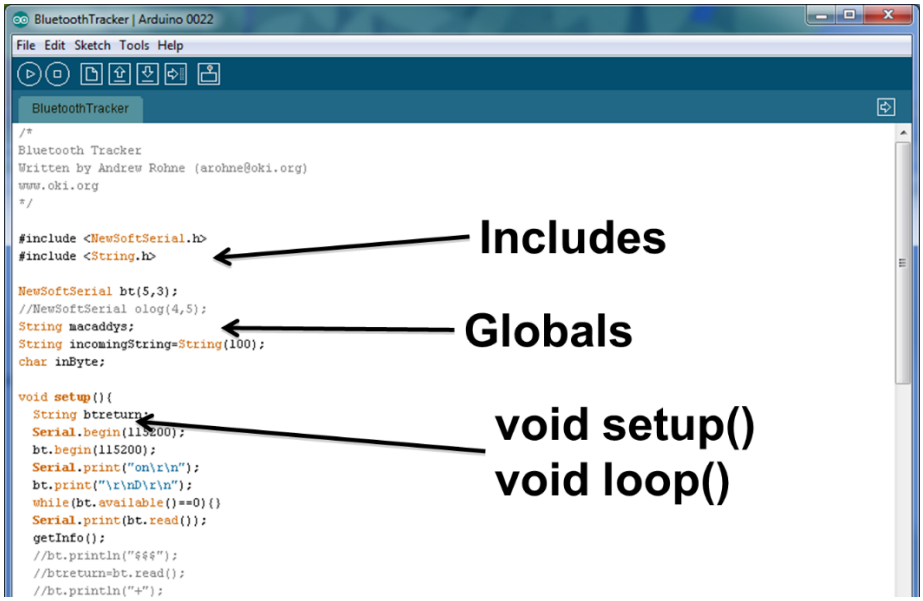
- Las llaves se usan para encerrar instrucciones adicionales realizadas por una función. Siempre hay una llave de apertura y una llave de cierre. Si olvida cerrar una llave, el compilador enviará un código de error.

  Ejemplo:

```
void loop() {   //this curly brace opens
//way cool program here
}    //this curly brace closes
```

- Las funciones son fragmentos de código que se utilizan con tanta frecuencia que se encapsulan en determinadas palabras clave para que pueda utilizarlas más fácilmente.

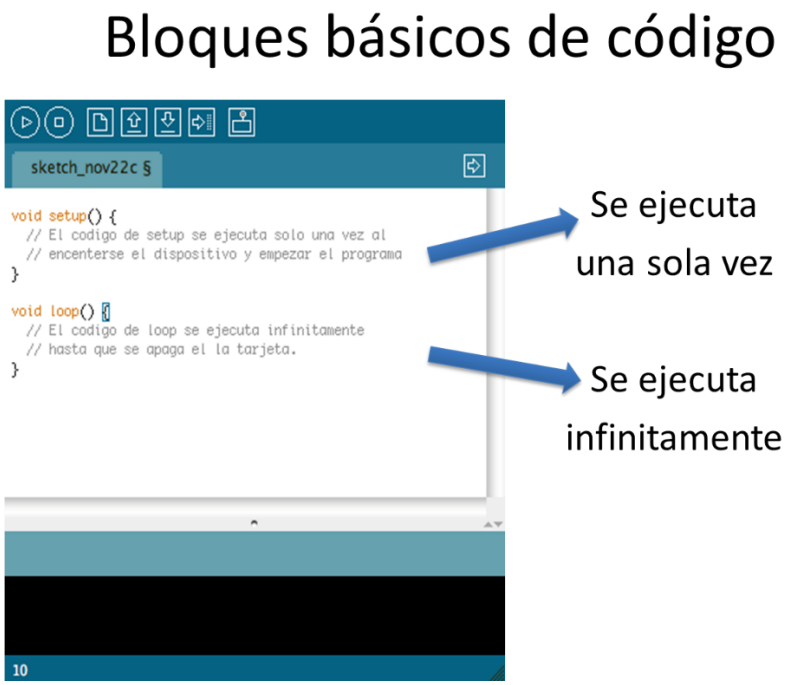
En Arduino, hay ciertas funciones que se utilizan con tanta frecuencia que se han integrado en el IDE. Cuando las escriba, el nombre de la función aparecerá en naranja.



Las principales funciones que posee el sketch son:

- **Void setup():** La función setup () se usa para configurar la placa Arduino. Esta debe ser la primera función en su boceto de Arduino y sólo se ejecuta una vez. También se debe recordar que el mismo debe tener llaves de apertura y cierre.

- **Void loop():** La función loop() es donde residirá el cuerpo de su programa. Al igual que con setup(), la función loop() no devuelve ningún valor, por lo que la palabra void la precede.



2.4.3.6. Memoria de la placa Arduino

La memoria de Arduino es la parte del sistema que almacena datos binarios en grandes cantidades. Las memorias semiconductoras están formadas por matrices de elementos de almacenamiento que pueden ser latches o condensadores.

Tipos de memoria:

- Memoria flash (espacio del programa), es donde se almacena el boceto de Arduino.

- SRAM (memoria estática de acceso aleatorio) es donde el boceto crea y manipula variables cuando se ejecuta.

- EEPROM es un espacio de memoria que los programadores pueden usar para almacenar información a largo plazo.

La memoria flash y la memoria EEPROM no son volátiles (la información persiste después de que se apaga la alimentación). SRAM es volátil y se perderá cuando se apague y vuelva a encender.

2.4.3.7. Protección de sobrecarga del USB de la placa Arduino

Aunque la mayoría de las computadoras ofrecen protección interna, la cual incorpora un fusible rearmable de intensidad máxima 500mA con la intención de proteger tanto la placa Arduino como el bus USB de sobrecargas y cortocircuitos. Si circula una intensidad mayor a 500mA por el bus USB (Intensidad máxima de funcionamiento), el fusible salta rompiendo la conexión de la alimentación.

2.4.3.8. Reseteo automático de la placa Arduino

Se puede restablecer la placa Arduino de dos formas. Primero, usando el botón de reinicio que se encuentra en la placa. En segundo lugar, puede conectar un botón de reinicio externo al pin Arduino etiquetado RESET.

2.5. Software Arduino.

El software Arduino o entorno de desarrollo integrado de Arduino (IDE), es donde se escriben los programas que luego se cargarán al hardware Arduino y que permite establecer la comunicación entre ellos.

El entorno de desarrollo integrado de Arduino (IDE) contiene un editor de texto para escribir código, un área de mensajes, una consola de texto, una barra de herramientas con botones para funciones comunes y una serie de menús.

2.5.1. Instalando el Software Arduino para Windows.

2.5.1.1. Obteniendo una placa Arduino y un cable USB

Actualmente, existen muchas variaciones de las placas Arduino que son las oficiales, mientras que, hay cientos más de competidores que ofrecen clones. A la hora de elegir una placa se debe considerar el tipo de proyecto que se va a realizar.

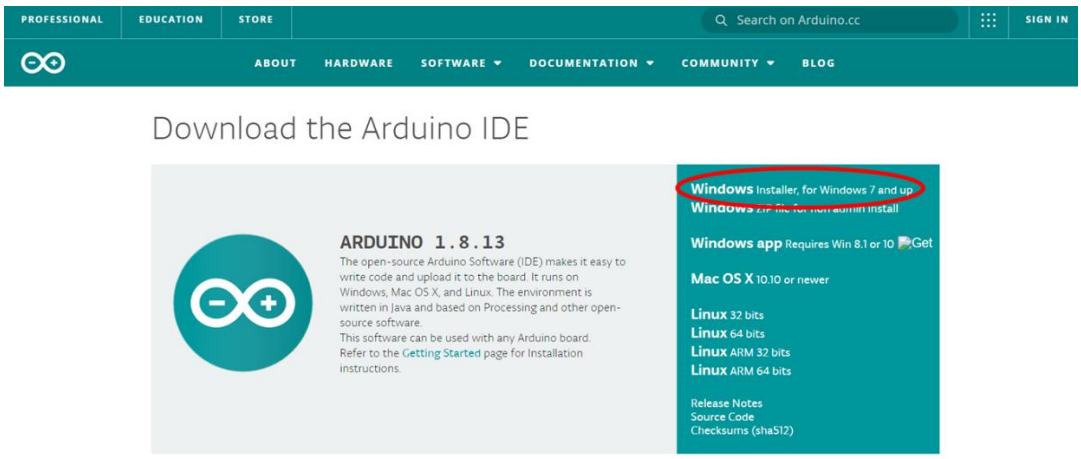
Las placas oficiales son aquellas placas con el nombre Arduino. Dentro de la familia de placas Arduino, la placa Arduino Uno es la más utilizada y documentada del mercado debido a sus excelentes funciones y facilidad de uso.

Una de las razones para comprar un clon es el hecho de que generalmente son menos costosos que su contraparte oficial. Por ejemplo, Adafruit y Sparkfun venden variaciones de las placas Arduino que cuestan menos, pero tienen la misma calidad que las originales.

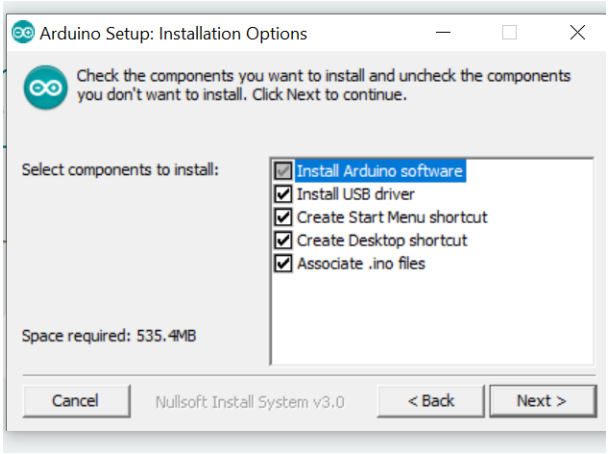
Además de la placa Arduino, se necesita un cable USB estándar (conector A a conector B), del tipo que conectaría a una impresora USB. Este cable es necesario ya sea para cualquiera de las placas Arduino oficiales: Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, Diecimila, etc, o para aquellas que son clones.

2.5.1.2. Descargando e instalando el entorno Arduino para Windows.

El IDE de Arduino se puede descargar desde la página en el sitio web oficial de Arduino https://www.arduino.cc/en/Main/Software. Debe seleccionar el software, que sea compatible con su sistema operativo (Windows, IOS o Linux). Para Windows puede elegir instalar o descargar comprimida (zip) la aplicación. Con la primera opción, se descarga e instala automáticamente la aplicación. Con la segunda opción, usted se debe realizar manualmente la instalación. Una vez se haya completado la descarga del archivo, hay que descomprimirlo haciendo doble clic en el icono de la aplicación e instalarlo. Se recomienda utilizar la primera opción para la instalación del software.



2.5.1.3. Instalando los drivers USB.

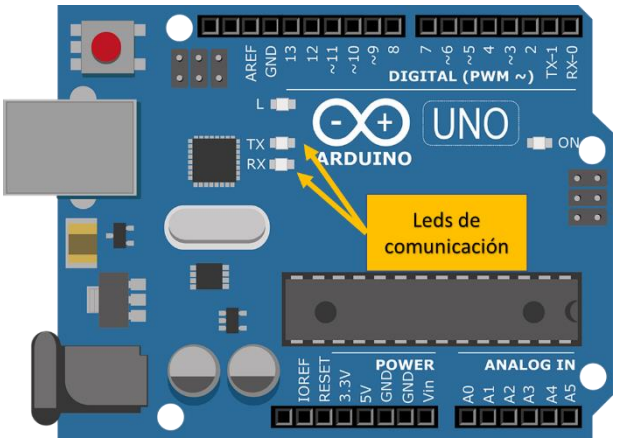


Cuando se ha descargado el Arduino IDE Installer se procede a la instalación del software, así como también de los componentes que queremos que se instalen. También podemos aceptar la instalación de los drivers como se muestra en la siguiente figura.

2.5.1.4. Instalando y conectando la placa Arduino.

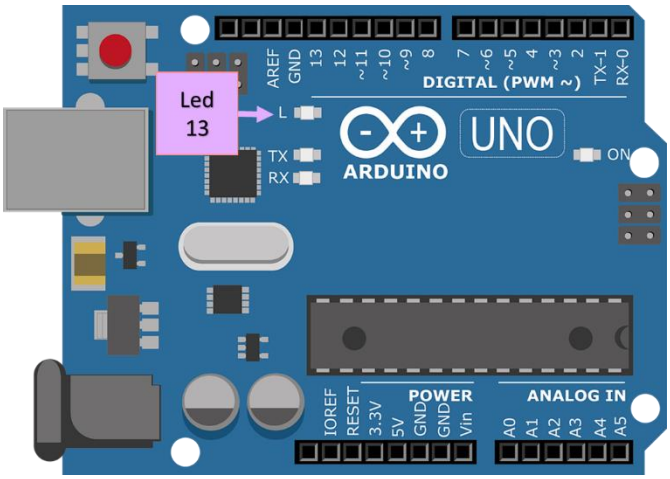
La conexión USB con la PC es necesaria para programar la placa. La placa obtiene energía automáticamente de la conexión USB a la computadora o de una fuente de alimentación externa.

Al conectar la placa Arduino a su computadora con el cable USB, se encienden:

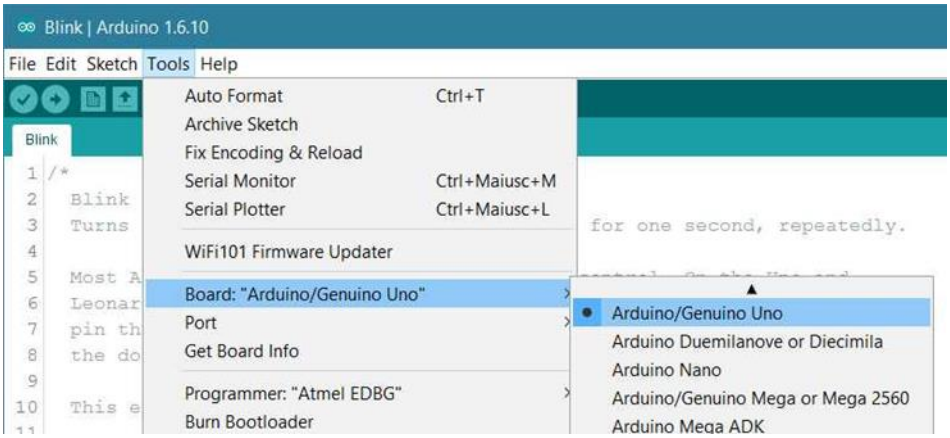


1. los leds TX Y RX indicando que se estableció la comunicación entre la placa y la computadora.

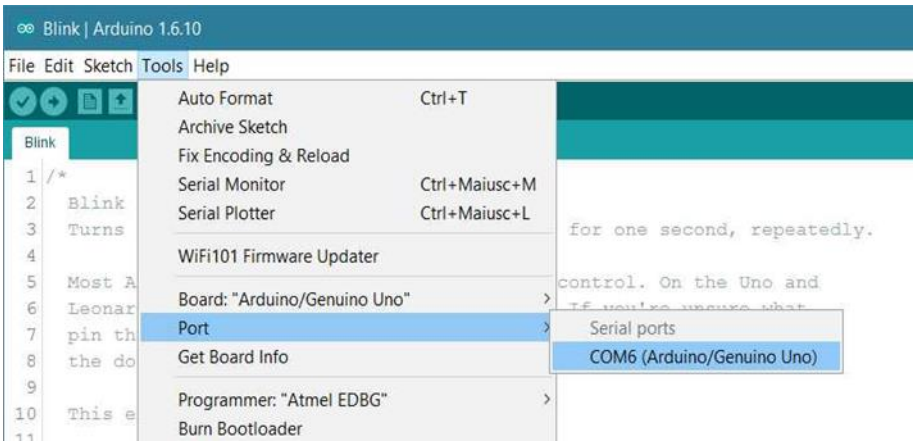
2. el led L que corresponde al pin digital 13 en la placa indicando que hay conexión con la placa, este continuará parpadeando mientras esté funcionando la placa.



Abrimos el IDE y seleccionamos en el menú la opción herramientas (tools) el tipo de placa y puerto.



Conecte la placa y seleccione el puerto. Es probable que sea COM3 o superior (COM1 y COM2 suelen estar reservados para puertos serie de hardware).



2.5..5. Ejecutando el entorno Arduino.

Para ejecutar el entorno Arduino debemos abrir el programa Arduino, se abrirá el IDE para que pueda escribir el programa que utilizará para indicarle a la placa lo que debe hacer. El lenguaje que usa Arduino es muy parecido a C ++ por lo que es muy fácil de aprender para ustedes.

Una vez escrito el programa, se compila. Si tiene errores en el código de su computadora, el compilador mostrará un mensaje de error en la parte inferior del IDE y resaltará la línea de código que parece ser el problema. Cuando el programa está libre de errores se sube a la placa.

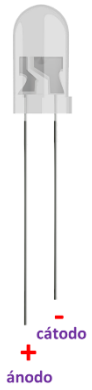
2.5..6. Configuración y descripción del entorno: Subiendo un programa.

Una vez que haya seleccionado el puerto serie y la placa correctos, presione el botón de carga en la barra de herramientas o seleccione el elemento de carga en el menú. En la mayoría de las placas, verá parpadear los LED RX y TX mientras se carga el boceto.

El software Arduino (IDE) mostrará un mensaje cuando se complete la carga o mostrará un error. Cuando carga un sketch, está utilizando el gestor de arranque Arduino, un pequeño programa que se ha cargado en el microcontrolador de su placa. Le permite cargar código sin utilizar ningún hardware adicional. El cargador de arranque está activo durante unos segundos cuando la placa se reinicia; luego comienza el sketch que se haya subido más recientemente al microcontrolador. El cargador de arranque hará parpadear el LED integrado (pin 13) cuando se inicie, es decir, cuando la placa se reinicie.

2.5..7. Buscando que el Led parpadee.

Uno de los primeros programas que se ejecuta en la placa de Arduino es el del led que parpadea. Este es un programa sencillo, cuyo objetivo es encender y apagar la luz del led. El parpadeo de un led, es el Hello World de los microcontroladores. La sigla LED significa Diodo Emisor de Luz (En inglés: Light Emitting Diode).



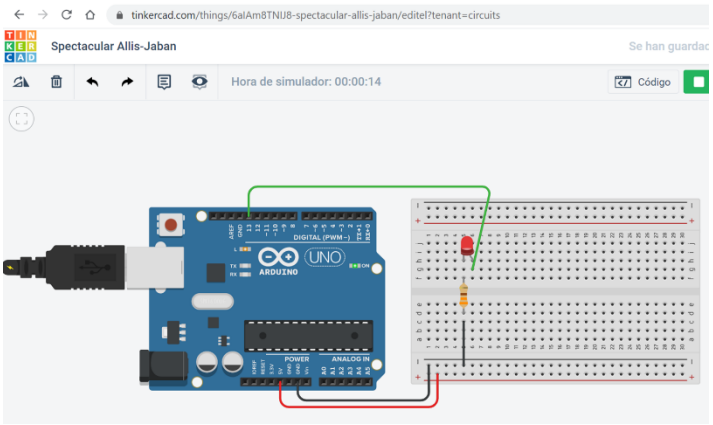
Los leds tienen dos polaridades:

- **positiva o ánodo:** corresponde a la pata o pin más largo.

- **negativa o cátodo:** corresponde a la pata o pin más corto. Si las patas o pines tienen el mismo largo, trata de visualizar el lado plano en la carcasa exterior del LED. El pin más cercano al lado plano va a ser el pin negativo, el cátodo.

De la forma más sencilla, se puede conectar el led directo a la placa de Arduino como se muestra a continuación.

La otra forma es utilizando el protoboard, con el cual podemos realizar conexiones de más componentes. Esta es la forma que se muestra a continuación utilizando los los siguientes componentes:

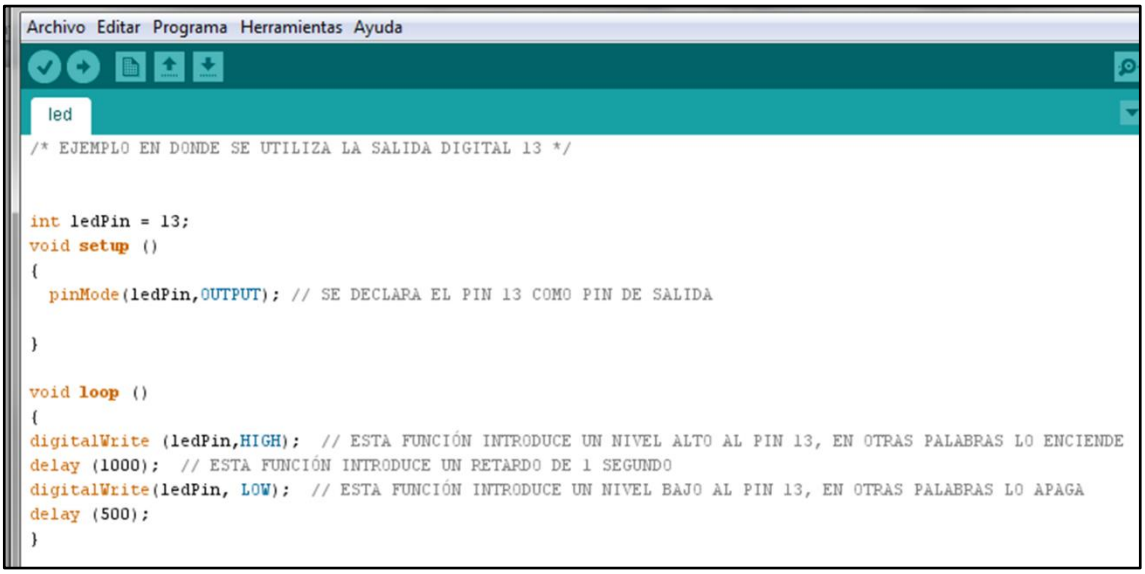


- 1 Arduino Uno
- 1 LED
- 1 Resistencia de 330Ω
- 1 protoboard
- Cables

Conectamos los componentes en la placa como se muestra en la imagen.

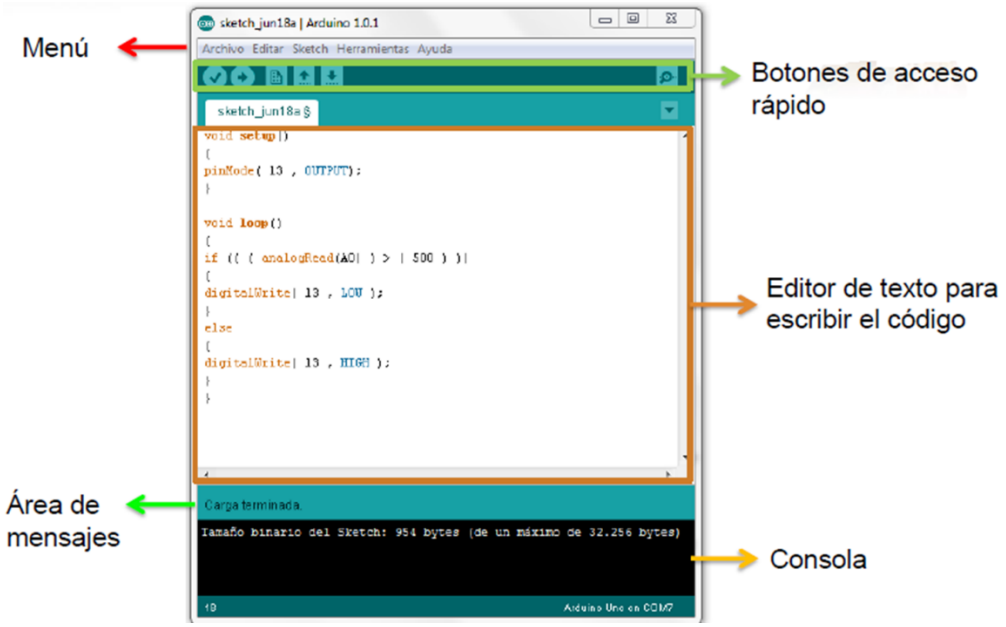
Sketch o programa

En el software Arduino IDE se debe escribir el siguiente programa que permite encender el led. En la imagen se muestra el programa documentado.



Las instrucciones del código serán explicadas con más detalle en la siguiente unidad, sin embargo, en el código del programa se encuentran debidamente documentado cada una de las instrucciones.

2.5.2. Preámbulo al Entorno de Trabajo del Software Arduino para Windows



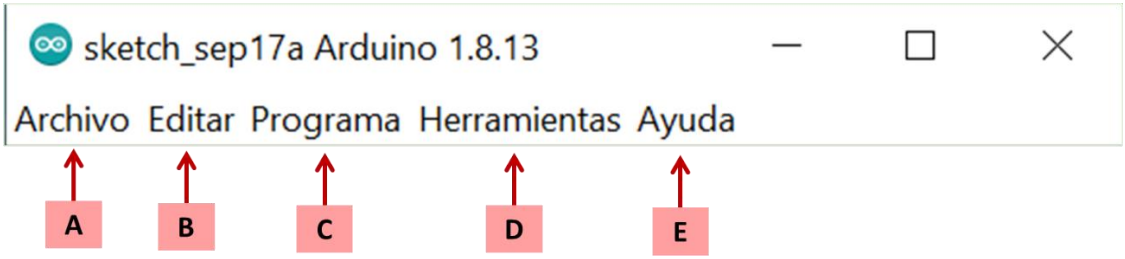
2.5.2.1. Barra de Herramientas del Entorno Arduino para Windows.

También conocida como botones de acceso rápido, nos permite ejecutar algunos de los comandos más básicos sin tener que entrar al menú. A continuación, se presenta la función de cada símbolo que aparece en la barra de herramientas Arduino IDE.

1. **Verificar:** se utiliza para comprobar si hay algún error de compilación.
2. **Subir:** se utiliza para cargar un programa en la placa Arduino.
3. **Nuevo:** utilizado para crear un nuevo sketch.
4. **Abrir:** se utiliza para abrir un nuevo sketch.
5. **Salvar:** se utiliza para guardar su sketch.
6. **Monitor Serie:** abre una consola con la que podemos comunicarnos con la placa Arduino.

2.5.2.2. Menús del Entorno Arduino para Windows.

Los comandos adicionales se encuentran dentro de los cinco menús: Archivo, Editar, Sketch, Herramientas, Ayuda. Algunas de las opciones contenidas dentro de estos menús se explican a continuación.



A. Archivo

- **Nuevo:** Crea una nueva instancia del editor, con la estructura mínima de un sketch ya en su lugar.
- **Abrir:** Permite cargar un archivo de programa o sketch navegando por las unidades y carpetas de la computadora.
- **Abrir reciente:** Proporciona una lista breve de los programas o sketches más recientes, listos para abrirse.
- **Ejemplos:** Cualquier ejemplo proporcionado por el software Arduino (IDE) o la biblioteca se muestra en este elemento de menú.
- **Preferencias:** Abre la ventana Preferencias donde se pueden personalizar algunas configuraciones del IDE.

B. Editar

- **Deshacer / Rehacer:** Retrocede uno o más pasos que realizó durante la edición.
- **Copiar al foro:** Copia el código de su sketch en el portapapeles en una forma adecuada para publicar en el foro, completa con colores de sintaxis.
- **Copiar como HTML:** Copia el código de su sketch al portapapeles como HTML, adecuado para incrustarlo en páginas web.
- **Comentar / Descomentar:** Pone o elimina el marcador de comentario // al principio de cada línea seleccionada.

- **Aumentar / Disminuir sangría:** Agrega o resta un espacio al principio de cada línea seleccionada, moviendo el texto un espacio a la derecha o eliminando un espacio al principio.
- **Buscar:** Abre la ventana Buscar y reemplazar, donde puede especificar el texto para buscar dentro del boceto actual de acuerdo con varias opciones.

C. Programa o sketch

- **Verificar / Compilar:** Verifica su sketch en busca de errores al compilarlo; informará el uso de la memoria para el código y las variables en el área de la consola.
- **Subir:** Compila y sube el archivo binario en la placa configurada a través del puerto configurado.
- **Exportar binario compilado:** Guarda un archivo .hex que puede guardarse como archivo o enviarse a la placa usando otras herramientas.
- **Incluir biblioteca:** Agrega una biblioteca a su boceto insertando declaraciones #include al comienzo de su código. Además, desde este elemento del menú puede acceder al Administrador de bibliotecas e importar nuevas bibliotecas desde archivos .zip.
- **Agregar archivo:** Agrega un archivo de origen al sketch (se copiará desde su ubicación actual). El nuevo archivo aparece en una nueva pestaña en la ventana de sketch.

D. Herramientas

- **Auto formato:** formatea su código, es decir, lo sangra para que las llaves de apertura y cierre se alineen, y que las declaraciones dentro de las llaves tengan más sangría.
- **Archivar sketch:** Archiva una copia del sketch actual en formato .zip. El archivo se coloca en el mismo directorio que el sketch.
- **Reparar codificación y recargar:** Corrige posibles discrepancias entre la codificación del mapa de caracteres del editor y los mapas de caracteres de otros sistemas operativos.

- **Monitor serie:** Abre la ventana del monitor serie e inicia el intercambio de datos con cualquier placa conectada en el puerto seleccionado actualmente.
- **Placa:** Seleccione la placa que está utilizando.
- **Puerto:** Este menú contiene todos los dispositivos seriales (reales o virtuales) de su máquina.

E. Ayuda

- Aquí encontrará fácil acceso a varios documentos que vienen con el software Arduino (IDE).

2.5.2.3. Preferencias del Entorno Arduino para Windows.

Algunas preferencias se pueden controlar desde el cuadro de diálogo Preferencias dentro del entorno Arduino. Acceda a él desde el menú Archivo en Windows o Linux, o desde el menú Arduino en Mac.

Otras preferencias deben cambiarse en el archivoferences.txt. La ubicación de este archivo se muestra en el cuadro de diálogo de preferencias.

Solo edite el archivo cuando Arduino no se esté ejecutando; de lo contrario, sus cambios se sobrescribirán cuando Arduino salga.

Módulo III: ENTORNO DE PROGRAMACIÓN EN EL SISTEMA DE ROBÓTICA ARDUINO

3. CAPÍTULO 3. LENGUAJE DE PROGRAMACIÓN PARA ARDUINO.

3.1. Estructura básica del lenguaje de programación Arduino.

La estructura básica del lenguaje de programación de Arduino se compone principalmente de dos funciones necesarias para que el programa trabaje, la función setup() y la función loop(), las cuales encierran bloques que contienen declaraciones e instrucciones. La función setup() es la parte encargada de recoger la configuración y la función loop() contiene el programa que se ejecutará cíclicamente

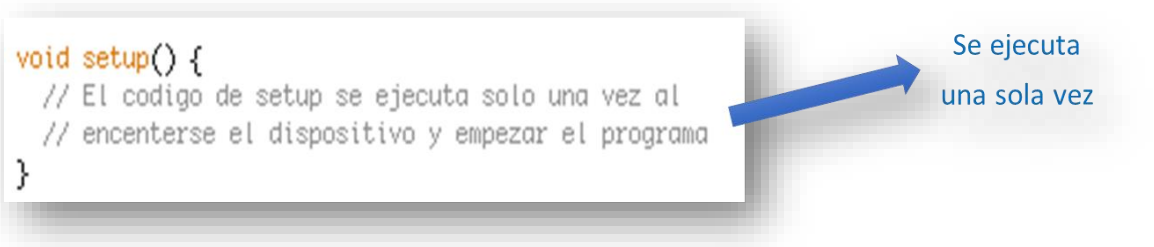
3.1.1. Función setup ()

Es la primera función a ejecutar en el programa. Se utiliza para inicializar las variables, configurar los modos de trabajo de los pines, configuración de la comunicación en serie, comenzar a usar bibliotecas, entre otros. Además, se usa para configurar la placa Arduino. La función de configuración solo se ejecutará una vez, después de cada encendido o reinicio de la placa Arduino. Adicionalmente, como la función es de tipo void, la misma, no devuelve información.

La estructura básica de esta función es:

```
void setup ( )
{
 // el código entre las llaves solo se ejecuta vez
}
```

Esta función debe ser incluida en el programa, aunque no haya declaración que ejecutar.



3.1.2. Función loop ().

La función loop () es donde residirá el cuerpo del programa. En esta función todo el código que se encuentre entre las llaves se repite una y otra vez, en un ciclo. La función loop () no devuelve ningún valor.

La función loop() contiene el código (lectura de entradas, activación de salidas, etc) que se ejecutará continuamente en el programa. Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

La estructura básica de esta función es:

void loop( )

{

//el Código que se coloque aquí se ejecutará una y otra vez

}

Esta función permite que el programa responda continuamente ante los eventos que se producen en la tarjeta.

3.1.3. Funciones del lenguaje de programación Arduino.

Una función es un bloque de código que tiene un nombre y un conjunto de instrucciones que son ejecutados cuando se llama a la función. Pueden ser escritas para realizar tareas repetitivas y para reducir el tamaño de un programa.

Algunas ventajas de utilizar funciones son:

- Reduce la posibilidad de errores en la modificación, si es necesario cambiar el código.
- Hacen que sea más fácil reutilizar el código en otros programas haciéndolo más modular.
- Hace que el código sea más legible.
- Hacen que todo el sketch sea más pequeño y compacto porque las secciones de código se reutilizan muchas veces.

3.1.3.1. Declaración del tipo de la función

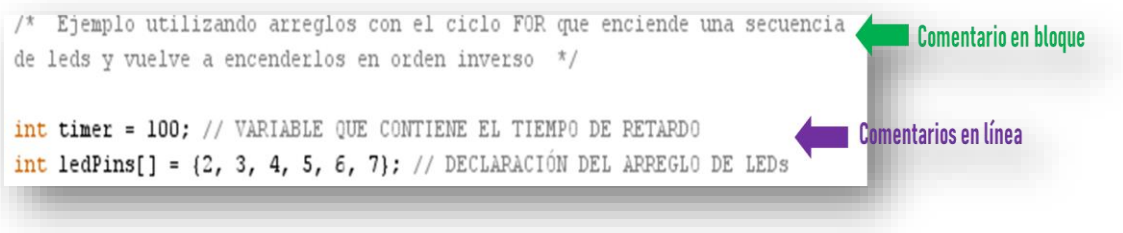
Declarar el tipo de la función, está asociado al valor retornado por la función. Este valor puede ser de tipo int, byte, long, float, void.

3.1.3.2. Declaración del nombre de la función

Después de declarar el tipo de dato que devuelve la función, se declara el nombre de la función y entre paréntesis se escribirán, si es necesario, los parámetros que se deben pasar a la función para que se ejecute. Los parámetros pasados a la función, deben ser definidos también con sus tipos y nombres.

3.1.3.3. Bloques de comentarios /*...*/

Los comentarios nos permiten documentar el programa que estamos realizando. Los bloques de comentarios se usan cuando hay mucho que explicar sobre un código y para esto es necesario utilizar varias líneas.



3.1.3.4. Comentarios de línea //

Se utiliza para aquellos que son comentarios cortos, generalmente pueden colocarse en una línea. Para este tipo de comentarios se utiliza la doble barra invertida.

3.1.3.5. Llaves { }

Las llaves se utilizan en distintas codificaciones, para encerrar las instrucciones realizadas por una función, en las estructuras condicionales o en las de repetición, etc. Siempre se utilizan en par, es decir, una llave de apertura siempre debe tener una llave de cierre. Si olvida cerrar una llave, el compilador enviará un mensaje de error.

3.1.3.6. Punto y coma (;)

Se usa para finalizar una declaración. Por ejemplo:

int LEDpin = 9;

int var = 1;

Olvidar terminar una línea con un punto y coma resultará en un error del compilador. El texto de error puede ser obvio y hacer referencia a un punto y coma que falta, o puede que no. Si le aparece un error de compilación aparentemente ilógico, una de las primeras cosas que hay que comprobar es que falta un punto y coma, en las inmediaciones, antes de la línea en la que se produjo el error.

3.1.4. Variables del lenguaje de programación Arduino

Una variable es una forma de nombrar y almacenar un valor para su uso posterior por parte del programa, como los datos de un sensor o un valor intermedio utilizado en un cálculo.

3.1.4.1. Declaración de variable

Antes de utilizar una variable, debe ser declarada y se le puede asignar un valor. En la declaración de la variable se indica el tipo de datos que almacenará (int, float, long, etc).

Sintaxis para declarar una variable

type var = val;

Parámetros utilizados en la declaración de una variable:

- type: tipo de variable
- var: nombre de la variable.
- val: el valor a asignar a esa variable.

3.1.4.2. Ámbito de la variable.

El sitio en el que la variable es declarada determina el ámbito de la misma. El ámbito determina en qué partes del programa puede ser usada la variable. Esto nos permite volver a definir una variable con un mismo nombre en diferentes partes del programa sin que haya conflictos entre ellos.

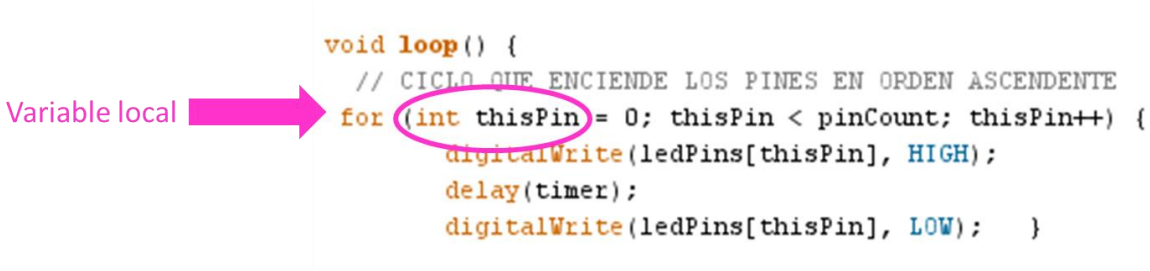
Las variables se pueden declarar:

- Dentro de una función o un bloque, que se llama variables locales.

- En la definición de parámetros de función, que se denomina parámetros formales.
- Fuera de todas las funciones, que se llama variables globales.

Variables locales

Son variables que se declaran dentro de una función o bloque. Estas serán válidas solo dentro de esa función o bloque y se destruirá al terminar la función o bloque. A continuación, se muestra un ejemplo que utiliza variables locales:



Variables globales

Las variables globales se definen fuera de todas las funciones, generalmente en la parte superior del programa. Cualquier función o bloque puede acceder a ella y modificarla. Las variables globales mantendrán su valor durante toda la vida útil de su programa.

3.1.5. Constantes del lenguaje de programación Arduino

El lenguaje de programación de Arduino tiene unos valores predeterminados, que son llamados constantes. Las constantes se clasifican en grupos. A continuación, se explican las constantes más utilizadas.

3.1.5.1. True / False

Son las dos constantes booleanas predefinidas que se utilizan para representar que un valor es verdadero o falso.

False: se define como 0 (cero).

True: se define como 1.

3.1.5.2. High / Low

Estas constantes definen los niveles de los pines como HIGH o LOW y son empleados cuando se leen o escriben en las entradas o salidas digitales. HIGH se define como el nivel lógico 1 (ON) o 5 V y funciona de la siguiente forma:

- Si el pin está configurado como INPUT, dará HIGH si hay un voltaje de:
  - 3 voltios en el pin para tarjetas de 5V
  - 2 voltios en el pin para tarjetas de tarjetas de 3.3V
- si el pin está configurado como OUTPUT, dará HIGH si hay un voltaje de:
  - 5 voltios para tarjetas de 5V
  - 3.3 voltios para tarjetas de 3.3V

LOW es el nivel lógico 0, OFF, o 0 V. Cuando un pin se configura en Output (salida) con pinMode (), y se establece en LOW con digitalWrite (), el pin está a 0 voltios en placas de 5V y 3.3V.

3.1.5.3. Input / Output

Son las constantes empleadas con la función pinMode() para definir el tipo de un pin digital usado como entrada INPUT o salida OUTPUT. Por ejemplo:

pinMode(13, OUTPUT);

3.1.6. Tipos de datos del lenguaje de programación Arduino.

Arduino permite manejar los siguientes tipos de datos:

3.1.6.1. Byte

Almacena un valor numérico de 8 bits. Tienen un rango de 0-255.

3.1.6.2. Int.

Almacena un valor entero de 16 bits con un rango de 32,767 a -32,768.

3.1.6.3. Long

Valor entero almacenado en 32 bits con un rango de 2,147,483,647 a -2,147,483,648.

3.1.6.4. Float

Tipo coma flotante almacenado en 32 bits con un rango de 3.4028235E+38 a -3.4028235E+38.

3.1.6.5. Arrays

Se trata de una colección de valores que pueden ser accedidos con un número de índice, en donde el primer valor del índice es 0. Ejemplos de utilización:

Definición y asignación:

```
int myArray[] = {2, 6, 4}; // declara  e inicializa un arreglo de 3 enteros
```

Definición:

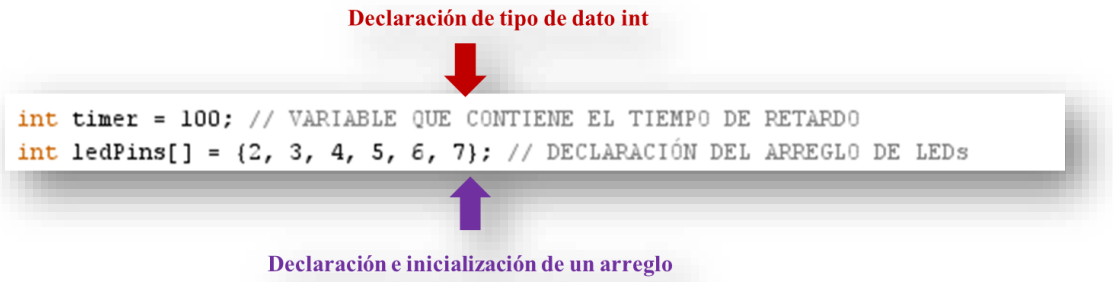
```
int myArray[5];  // declara un array de 6 enteros
```

Asignación de un valor a una posición específica:

```
myArray[3] = 10; // asigna el valor de 10 a la cuarta posición del arreglo
```

Recuperación de la información de una posición en el arreglo:

```
x = myArray[3]; // recupera el valor de la cuarta posición del arreglo y lo asigna a x
```



3.1.7. Aritmética: Operadores aritméticos del lenguaje de programación Arduino.

3.1.7.1. Operadores Aritméticos.

Son operadores que devuelven el resultado de operaciones tales como la suma, la diferencia, el producto o el cociente (respectivamente) de los dos operandos. La operación se realiza utilizando el tipo de datos de los operandos, por ejemplo, para la siguiente operación 9/4 en donde 9 y 4 son enteros (tipo int) da como resultado 2 manteniendo el tipo de datos entero (int).

Esto también significa que la operación puede desbordarse si el resultado es mayor que el que se puede almacenar en el tipo de datos. Si los operandos son de diferentes tipos, se utiliza el tipo "más grande" para el cálculo.

3.1.7.2. Asignaciones Compuestas.

Son operaciones que combinan una operación aritmética con una variable asignada. Este tipo de operaciones son usualmente utilizadas en los ciclos, en especial en el ciclo for. A continuación, se listan las operaciones de asignación compuesta.

- **Operador de incremento:** aumenta el valor entero en uno.

Ejemplo:

x ++ es equivalente a la operación x = x + 1

- **Operador de decremento:** decrementa el valor entero en uno.

  Ejemplo:

  x -- es equivalente a la operación x = x - 1

- **Adición compuesta:** agrega el operando derecho al operando izquierdo y asigna el resultado al operando izquierdo

  Ejemplo:

  x += y es equivalente a la operación x = x + y

- **Sustracción compuesta:** resta el operando derecho del operando izquierdo y asigna el resultado al operando izquierdo

  Ejemplo:

  x -= y es equivalente a la operación x = x – y

- **Multiplicación compuesta:** multiplica el operando derecho con el operando izquierdo y asigna el resultado al operando izquierdo

  Ejemplo:

  x *= y es equivalente a la operación x = x * y

- **División compuesta:** divide el operando izquierdo entre el operando derecho y asigna el resultado al operando izquierdo

  Ejemplo:

  x /= y es equivalente a la operación x = x / y

- **Modulo compuesto:** toma módulo usando dos operandos y asigna el resultado al operando izquierdo

Ejemplo:

x %= y es equivalente a la operación x = x % y

3.1.7.3. Operadores de Comparación.

Se utilizan para realizar operaciones comparativas entre variables o contantes entre sí. Son muy utilizados en estructuras de control condicionales, tales como, if, while, etc. Los operadores de comparación disponibles en Arduino son las siguientes:

- **== (igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando los dos operandos son iguales.

Ejemplo:

x == y; // es verdadero si x es igual a y, es falso si x no es igual a y

- **!= (no igual que):** compara la variable de la izquierda con el valor o la variable de la derecha del operador. Devuelve verdadero cuando los dos operandos no son iguales.

Ejemplo:

x != y; // es falso si x es igual a y, es verdadero si x no es igual a y

- **< (menor que):** la comparación es verdadera si x es igual a, y es falso si x no es igual a y

Ejemplo:

x < y; // es verdadero si x es menor que y, es falso si x es igual o mayor que y

- **> (mayor que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es mayor (más grande) que el operando de la derecha.

  Ejemplo:

  x > y; // es verdadero si x es mayor que y, es falso si x es igual o menor que y

- **<= (menor o igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es menor o igual que el operando de la derecha.

  Ejemplo:

  x <= y; // es verdadero si x es menor o igual que y, es falso si x es mayor que y

- **>= (mayor o igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es mayor o igual que el operando de la derecha.

  Ejemplo:

  x >= y; // es verdadero si x es mayor o igual que y, es falso si x es menor que y

3.1.7.4. Operadores Lógicos.

Los operadores lógicos o booleanos permiten comparar dos variables o constantes entre sí. Estos devuelven un VERDADERO o FALSO dependiendo del operador. Existen tres operadores booleanos:

- **&& (y):** la operación y (and) lógico da como resultado verdadero solo si ambos operandos son verdaderos. Este operador se puede utilizar dentro de la condición de una sentencia if, while, etc.

  Ejemplo:

```
if (digitalRead(2) == HIGH  && digitalRead(3) == HIGH)
{    // Si AMBOS los interruptores leen HIGH
   // instrucciones
}
```

- **|| (o):** la operación o (or) lógico da como resultado verdadero si cualquiera de los dos operandos es verdadero. Este operador se puede utilizar dentro de la condición de una sentencia if, while, etc.

  Ejemplo:

```
if (x > 0 || y > 0)
{   // si x o y es mayor que cero
  // instrucciones
}
```

- **! (no):** El operador no (not) lógico da como resultado verdadero si el operando es falso y viceversa. Se puede utilizar para invertir el valor booleano como se muestra en la primera instrucción de ejemplo o dentro de la condición de una sentencia if, while, etc.

  Ejemplo 1:

```
x = !y;   // el valor invertido de y se almacena en x
```

  Ejemplo 2:

```
if (!x)
{
```

```
    // si x no es cierto
    // instrucciones
}
```

3.1.8. Control de flujo: Sentencias condicionales del lenguaje Arduino.

Son estructuras que nos permiten tomar decisiones durante la ejecución del programa. Estas requieren que se especifique una o más condiciones para ser evaluadas o probadas por el programa. Debe ir junto con una declaración o declaraciones que se ejecutarán si se determina que la condición es verdadera y, opcionalmente, otras declaraciones que se ejecutarán si se determina que la condición es falsa. Entre estas estructuras se explicarán la if y if … else. Además, se presentan las estructuras repetitivas for, while y do … while.

3.1.8.1. If: toma una expresión entre paréntesis y una declaración o bloque de declaraciones. Si la expresión es verdadera, la declaración o el bloque de declaraciones se ejecuta; de lo contrario, estas declaraciones se omiten.

Ejemplo:

```
if (A > B)
    A++;
```

3.1.8.2. If. . . else: una instrucción if puede ir seguida de una instrucción else opcional, que se ejecuta cuando la expresión es falsa.

Ejemplo:

```
if (A > B)
{
    A++;
  }
```

```
else
{
    B -= A;
  }
```

3.1.8.3. **for:** se usa para repetir un bloque de declaraciones entre llaves. Por lo general, se usa un contador de incrementos para incrementar y terminar el ciclo. La instrucción for es útil para cualquier operación repetitiva y, a menudo, se usa en combinación con matrices para operar en colecciones de datos / pines.

Ejemplo:

```
void loop()
{
  for (int i=0; i <= 255; i++){
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

3.1.8.4. **while:** este ciclo se repetirá de forma continua e infinita, hasta que la expresión dentro del paréntesis, () se vuelva falsa. Algo debe cambiar la variable probada, o el ciclo while nunca saldrá. Esto podría estar en su código, como una variable incrementada, o una condición externa, como probar un sensor.

Ejemplo:

```
var = 0;
```

```
while(var < 200)
{
  // hacer algo repetitivo 200 veces
  var++;
}
```

3.1.8.5. Do. . . while: este ciclo funciona de la misma manera que el bucle while, con la excepción de que la condición se prueba al final del bucle, por lo que el ciclo do siempre se ejecutará al menos una vez.

Ejemplo:

```
do
{
  delay(50);          // esperar a que los sensores se estabilicen
  x = readSensors();  // comprobar los sensores
} while (x < 100);
```

3.1.9. Entradas y salidas digitales del lenguaje de programación Arduino.

Los pines digitales se pueden configurar como entradas o salidas que reciben niveles altos (5V) o bajos (0V) de tensión y que son interpretados como un 1 o un 0 respectivamente.

Para controlar estas salidas /entradas, se utilizan las siguientes funciones:

3.1.9.1. pinMode(pin, mode): configura el pin como entrada o salida, en donde, pin corresponde al número del pin y mode puede ser INPUT o OUTPUT.

3.1.9.2. digitalRead(pin): Lee el valor de un pin digital.

3.1.9.3. digitalWrite(pin, value): Escribe un 0 o un 1 (0 o 5V) en el pin especificado

Ejemplo:

```
int ledPin = 13; // LED conectado al pin 13
int inPin = 7;   // pulsador conectado al pin 7
int val = 0;     // Variable para almacenar el valor leído

void setup()
{
 pinMode(ledPin, OUTPUT);    // configura el pin 13 como salida
 pinMode(inPin, INPUT);      // configure el pin 7 como entrada
}

void loop()
{
 val = digitalRead(inPin);   // Lee el valor del pin 7
 digitalWrite(ledPin,val);   // Enciende el LED si el pulsador está presionado
}
```

3.1.10. Entradas y salidas analógicas del lenguaje de programación Arduino.

Los pines análogos son entradas análogas que reciben tensiones entre 5V y 0V (v –voltios). Los pines análogos, al contrario de los digitales, no necesitan ser declarados como modo INPUT o OUTPUT.

Dos funciones muy utilizadas en las entradas y salidas analógicas son:

3.1.10.1. analogRead(pin): lee o captura el valor de entrada del pin análogo especificado. La placa Arduino realiza una conversión

análoga a digital de 10 bits. Esto quiere decir que mapeará los valores de voltaje de entrada, entre 0 y 5 voltios, a valores enteros comprendidos entre 0 y 1023.

3.1.10.2. analogWrite(pin, value): escribe un valor analógico (onda PWM) en un pin. Se puede utilizar para encender un LED con distintos brillos o impulsar un motor a distintas velocidades. Después de una llamada a analogWrite (), el pin generará una onda cuadrada constante del ciclo de trabajo especificado hasta la próxima llamada a analogWrite () (o una llamada a digitalRead () o digitalWrite ()) en el mismo pin. La frecuencia de la señal PWM en la mayoría de los pines es de aproximadamente 490 Hz. En el Uno y tableros similares, los pines 5 y 6 tienen una frecuencia de aproximadamente 980 Hz.

Esta función tiene la siguiente sintaxis:
analogWrite(pin, value)
en donde los parámetros indican
pin: el pin al que escribir. Tipos de datos permitidos: int.
value: el ciclo de trabajo: entre 0 (siempre apagado) y 255 (siempre encendido). Tipos de datos permitidos: int.

Es importante recordar que las salidas PWM generadas en los pines 5 y 6 tendrán ciclos de trabajo más altos de lo esperado. Esto se debe a las interacciones con las funciones millis () y delay (), que comparten el mismo temporizador interno utilizado para generar esas salidas PWM. Esto se notará principalmente en configuraciones de ciclo de trabajo bajo (por ejemplo, 0-10) y puede resultar en un valor de 0 que no apague completamente la salida en los pines 5 y 6.

Ejemplo:

```
int ledPin = 9;   // LED conectado al pin digital  9
int analogPin = 3;   // potenciómetro conectado al pin análogo 3
int val = 0;         // variable para almacenar el valor que se leerá

void setup()
{
  pinMode(ledPin, OUTPUT);   // declara el pin como salida
}

void loop()
{
  val = analogRead(analogPin);   /* lee el valor del pin de entrada y se lo asigna
                                    a la variable val */
  analogWrite(ledPin, val / 4);  /* Los valores de analogRead van de 0 a 1023,
                                    los valores de analogWrite de 0 a 255 */
}
```

3.1.11. Funciones de tiempo del lenguaje de programación Arduino.

Son funciones muy utilizadas en distintas aplicaciones en las cuales es necesario trabajar con tiempo. Las más utilizadas son:

3.1.11.1. delay(ms): esta función pausa el programa durante el tiempo (en milisegundos) especificado como parámetro. (Hay 1000 milisegundos en un segundo).

Esta función tiene la siguiente sintaxis:

delay(ms)

en donde el parámetro indica

ms: la cantidad de milisegundos para pausar.

3.1.11.2. millis(): devuelve el número de milisegundos desde que la placa Arduino comenzó a ejecutar el programa actual. Este número se desbordará (volverá a cero) después de aproximadamente 50 días.

Tiene la siguiente sintaxis
time = millis()

Es importante mencionar que el dato que regresa la función de Arduino millis, requiere de una variable del tipo unsigned long. Si se usará otra más pequeña pueden ocurrir errores de lógica.

Ejemplo:

```
unsigned long time;

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

void loop()
{
  Serial.print("Time: ");
  time = millis();
  Serial.println(time);   // imprime el tiempo desde que se inició el programa
  delay(1000);  /* espera un segundo para no enviar cantidades masivas de
                   datos */
}
```

3.1.12. Funciones matemáticas del lenguaje de programación Arduino.

El lenguaje Arduino posee varias funciones matemáticas que nos ayudan al momento de programar la placa. En esta sección se presentan dos de las funciones más utilizadas.

3.1.12.1. Min(x,y)

Es una función matemática que calcula el valor menor o mínimo de dos números.

Sintaxis

min (x, y)

Parámetros

x: el primer número, cualquier tipo de datos

y: el segundo número, cualquier tipo de datos

Ejemplo:

```
minval = min(x, 20);
// asigna a la variable minval el valor menor entre lo que contiene la variable x o 20
```

3.1.12.2. Max(x, y)

Es una función matemática que calcula el valor mayor o máximo de dos números.

Sintaxis

max (x, y)

Parámetros

x: el primer número, cualquier tipo de datos

y: el segundo número, cualquier tipo de datos

Ejemplo:

```
maxval = max(x, 20);
```

// asigna a la variable maxval el valor mayor entre lo que contiene la variable x o 20

3.1.13. Funciones de generación aleatoria del lenguaje de programación Arduino.

Para generar números aleatorios, puede utilizar las funciones de números aleatorios de Arduino. Tenemos dos funciones que se explican a continuación.

3.1.13.1. Randomseed(seed)

Esta función inicializa el generador de números pseudoaleatorios. La secuencia es siempre la misma, aunque es muy larga y aleatoria. La función hace que se inicie en un punto arbitrario de la misma.

Si es necesario que la secuencia de valores generada por random () difiera, se puede usar la función randomSeed () para inicializar el generador de números aleatorios con una entrada aleatoria como analogRead () en un pin no conectado.

Ejemplo:

```
long randNumber;

void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop()
{
  randNumber = random(300);
  Serial.println(randNumber);
  delay(50);
}
```

3.1.13.2. Random(max) y Random(min, max)

Al igual que la anterior la función genera números pseudoaleatorios.

Sintaxis

random(max)

random(min, max)

Parámetros

min: límite inferior del valor aleatorio, es opcional

max: límite superior del valor aleatorio

Ejemplo:

```
long randNumber;
void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop()
{
  // imprime un número aleatorio del 0 al 299
  randNumber = random(300);
  Serial.println(randNumber);

  // imprime un número aleatorio del 10 al 19
  randNumber = random(10, 20);
  Serial.println(randNumber);
  delay(50);
}
```

3.1.14. Puerto serie del lenguaje de programación Arduino.

Se utiliza para la comunicación entre la placa Arduino y una computadora u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie, se comunica en los pines digitales 0 (RX) y 1 (TX) así como con la computadora a través de USB. También se puede usar el monitor serial integrado del entorno Arduino para comunicarse con una placa Arduino.

3.1.14.1. Serial.begin(rate)

Abre un puerto serie y establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. La velocidad típica de comunicación es de 9600 pero también puede utilizar otras velocidades.

Sintaxis

Serial.begin(speed)

Serial.begin(speed, config)

Parámetros

speed: en bits por segundo (baudios)

config: establece datos, paridad y bits de parada.

Ejemplo:

```
// Programa Hola Mundo

/* Los datos son enviados desde la placa de Arduino hacia el computador a través del puerto USB */

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

void loop()
```

```
{
  Serial.println("Hola Mundo");
}
```

3.1.14.2. Serial.println(data)

Imprime datos en el puerto serie como texto ASCII legible por las personas. Esta instrucción está seguida de un carácter de retorno de carro y un carácter de nueva línea.

Sintaxis

Serial.println (val)

Serial.println (val, formato)

Parámetros

val: el valor para imprimir, es cualquier tipo de datos

formato: especifica la base numérica o el número de lugares decimales (para tipos de coma flotante)

Ejemplo:

```
int analogValue = 0;    // variable que mantiene el valor analógico
void setup()
{
      //abre el puerto serial y establece la velocidad de datos a 9600 bps
      Serial.begin(9600);
}

void loop()
{
  // lee la entrada analógica en el pin 0:
```

```
  analogValue = analogRead(0);

  // imprime la saida en formato decimal codificado en ASCII, decimal y binario
 Serial.println(analogValue);
 Serial.println(analogValue, DEC);
  Serial.println(analogValue, BIN);

   // retrasar 10 milisegundos antes de la siguiente lectura:
   delay(10);
}
```

3.1.14.3. Serial.read().

Lee datos seriales entrantes. La instrucción read () hereda de la clase de utilidad Stream. La función retorna el primer byte de datos seriales entrantes disponible o -1 si no hay datos disponibles.

Sintaxis

```
Serial.read()
```

Ejemplo:

```
int incomingByte = 0;   // para datos en serie entrantes
void setup()
{
      //abre el puerto serial y establece la velocidad de datos a 9600 bps
      Serial.begin(9600);
}

void loop()
{
      // envíe datos solo cuando reciba datos:
```

```
      if (Serial.available() > 0)
    {
            // lee el byte entrante
            incomingByte = Serial.read();

            // imprime lo que recibió
            Serial.print("He recibido: ");
            Serial.println(incomingByte, DEC);
     }
}
```

3.1.14.4. Serial.available().

Devuelve el número de caracteres disponibles para leer desde el puerto serie. La cantidad de datos se obtiene de los datos que ya llegaron y se almacenaron en el búfer de recepción en serie. available () hereda de la clase de utilidad Stream.

Sintaxis

Serial.available()

Ejemplo:

```
int incomingByte = 0;
void setup()
{
        Serial.begin(9600);
}

void loop()
{
        // envía los datos cuando los recibe
        if (Serial.available() > 0)
        {
```

```
            incomingByte = Serial.read();
            Serial.print("I received: ");
            Serial.println(incomingByte, DEC);
        }
}
```

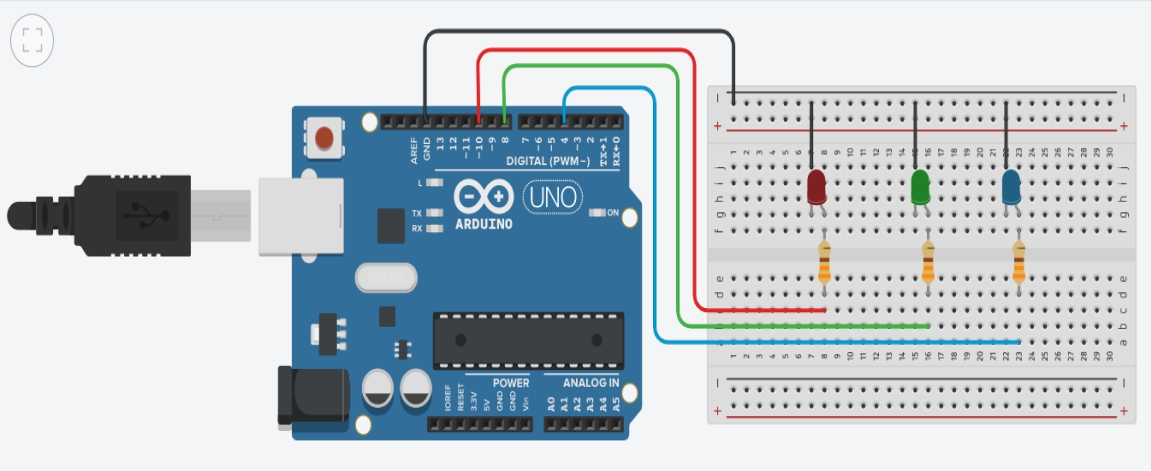
3.1.15. Ejemplos de código del lenguaje de programación Arduino

3.1.15.1. Ejemplo 1 -- Salida digital

Construya el esquema del circuito y escriba el código que le permite encender tres leds a la vez y que luego los apague. Los pines de los leds son: 4, 8 y 10.

Solución

Esquema del circuito



Código del programa en Arduino

```
/* EJEMPLO EN DONDE SE ENCIENDEN Y APAGAN JUNTOS TRES LEDS*/
```

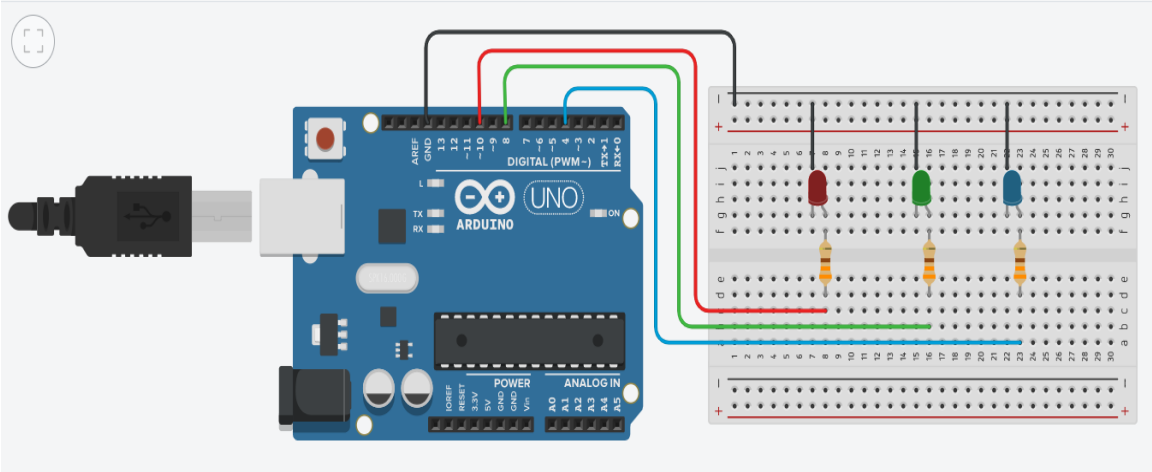
```
int led1 = 10;
int led2 = 8;
int led3 = 4;
void setup ()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}

void loop ()
{
    digitalWrite (led1, HIGH);
    digitalWrite (led2, HIGH);
    digitalWrite (led3, HIGH);
    delay (1000);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite (led3, LOW);
    delay (500);
}
```

3.1.15.2. Ejemplo 2 -- Salida digital II

Construya el esquema del circuito y escriba el código que le permite encender tres leds en serie, es decir, primero enciende el led 4, luego el 8 y finalmente el 10. Los pines de los leds son: 4, 8 y 10.

Solución

Esquema del circuito



Código del programa en Arduino

```
int led1 = 10;
int led2 = 8;
int led3 = 4;
void setup ()
{
   pinMode(led1, OUTPUT);
   pinMode(led2, OUTPUT);
   pinMode(led3, OUTPUT);
}

void loop ()
{
     // SE APAGAN TODOS LOS LEDS
     digitalWrite (led1, LOW);
     delay (1000);
     digitalWrite(led2, LOW);
     delay (1000);
     digitalWrite (led3, LOW);
```

```
    delay (1000);

    // COMIENZA LA SERIE
    digitalWrite (led1, HIGH);
    delay (1000);
    digitalWrite(led2, LOW);
    delay (1000);
    digitalWrite (led3, LOW);
    delay (1000);
    digitalWrite (led1, LOW);
    delay (1000);
    digitalWrite(led2, HIGH);
    delay (1000);
    digitalWrite (led3, LOW);
    delay (1000);
    digitalWrite (led1, LOW);
    delay (1000);
    digitalWrite(led2, LOW);
    delay (1000);
    digitalWrite (led3, HIGH);
    delay (1000);
}
```

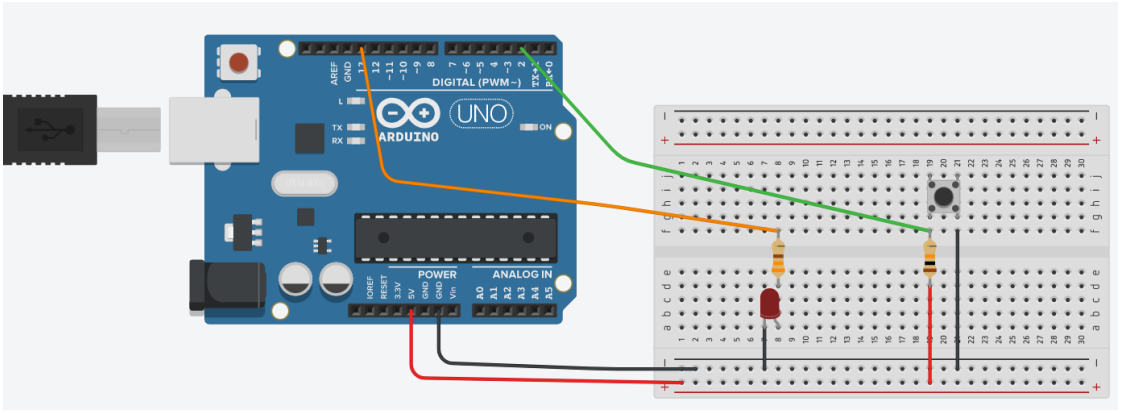
3.1.15.3. Ejemplo 3 -- Entrada digital

Una de las formas más sencillas de entrada es utilizando un simple switch o pulsador que nos permite sólo dos posibles estados: encendido o apagado. Al pulsar el botón se apaga el led.

Construya el esquema del circuito y escriba el código que le permite encender un led utilizando un pulsador o botón.

Solución

Esquema del circuito



Código del programa en Arduino

```
// programa que enciende un led utilizando un pulsador
const int buttonPin = 2;     // el número del pin del boton
const int ledPin =  13;      // el número del pin del LED
int buttonState = 0;         // variable para leer el estado del botón
void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop()
{
   buttonState = digitalRead(buttonPin); /* lee el valor del estado del botón */

   if (buttonState == HIGH)
        digitalWrite(ledPin, HIGH);  /* enciende el led */
   else
       digitalWrite(ledPin, LOW); /* apaga el led */
```

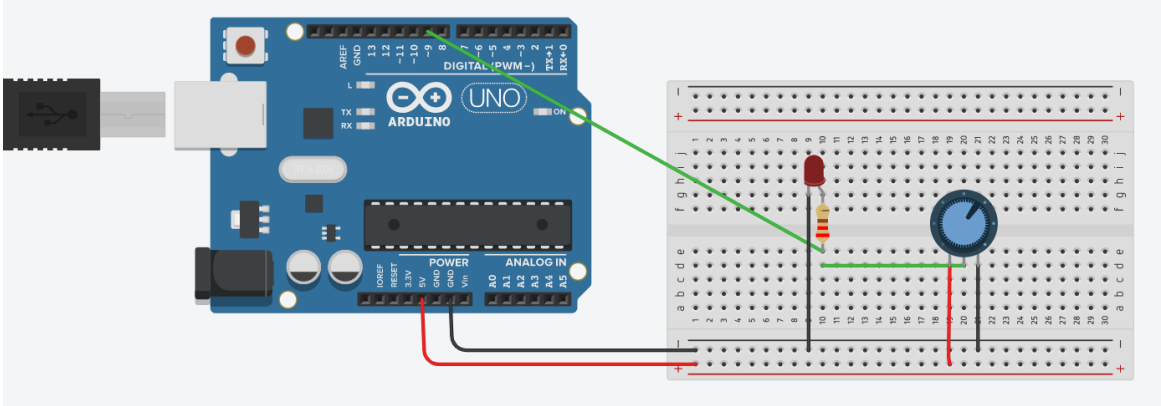
}

3.1.15.4. Ejemplo 4 -- Salida PWM

Construya el esquema del circuito y escriba el código para modificar el brillo de un LED utilizando la Modulación de Impulsos en Frecuencia (PWM).

Solución

Esquema del circuito



Código del programa en Arduino

```
int ledPin = 9; // pin PWM para el LED
void setup()  {} // no es necesario configurar nada


void loop()
{
    for (int i=0; i<=255; i++)
    {
```

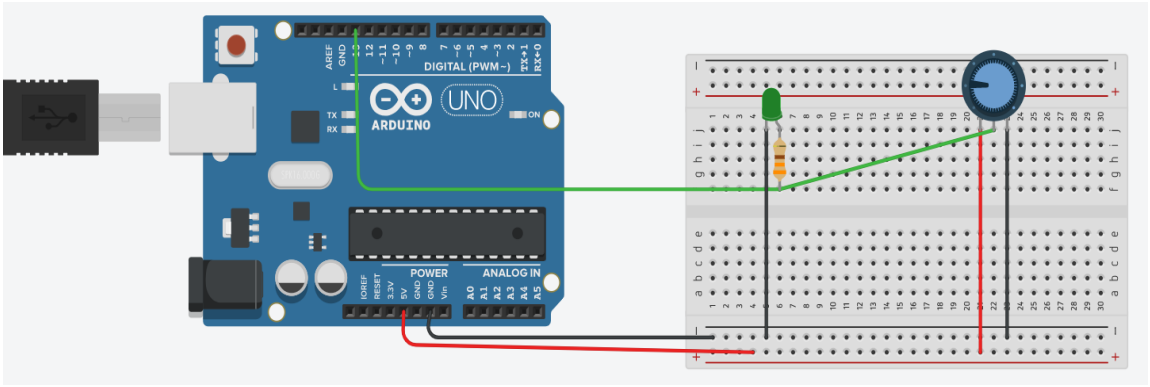
```
        analogWrite(ledPin, i);
        delay(100);
    }
    for (int i=255; i>=0; i--)
    {
        analogWrite(ledPin, i);
        delay(100);
    }
}
```

3.1.15.5. Ejemplo 5 -- Entrada a partir de un potenciómetro

Construya el esquema del circuito y escriba el código para utilizar un potenciómetro para controlar un el tiempo de parpadeo de un LED.

Solución

Esquema del circuito



Código del programa en Arduino

```
int potPin = 0; // pin entrada para potenciómetro
int ledPin = 13; // pin de salida para el LED
void setup()
{
    pinMode(ledPin, OUTPUT); // declara ledPin como SALIDA
}

void loop()
{
   digitalWrite(ledPin, HIGH); // pone ledPin en on
   delay(analogRead(potPin)); // detiene la ejecución un tiempo “potPin”
   digitalWrite(ledPin, LOW); // pone ledPin en off
   delay(analogRead(potPin)); // detiene la ejecución un tiempo “potPin”
}
```

Módulo IV: CONSTRUCCIÓN EN EL SISTEMA DE ROBÓTICA ARDUINO

4. CAPÍTULO 4. APLICACIONES SOBRE ARDUINO.

4.1. Codificación mínima de la programación con arduino.

Cuando realizamos un proyecto en Arduino existen algunos aspectos que son indispensables explicar para poder realizar de manera apropiada el mismo. En esta sección se detallan cada uno de los requisitos que se deben explicar cuando se desarrolla un proyecto con Arduino.

4.1.1. Objetivo.

Cada proyecto que se desarrolle debe poseer, por lo menos, un objetivo que constituye la pieza clave del diseño. El mismo, expresan la meta que se desea alcanzar.

Los objetivos se redactan comenzando por un verbo en infinitivo y deben ser evaluables permitiendo comprobar si se alcanza el resultado esperado. Estos deben definirse forma clara, que puedan ser medibles, esto es, que tenga un resultado alcanzable y reales.

4.1.2. Requisitos de Hardware.

Una parte muy importante al momento de realizar un proyecto es enumerar los componentes de hardware que se necesitaran para la elaboración del mismo. Es importante elaborar un listado de todos los componentes, incluyendo el tipo de placa que se utilizará.

4.1.3. Construcción del Circuito.

Constituye una representación de la conexión de los distintos componentes pertenecientes al proyecto. Es de gran ayuda para aquellos que desean implementar el proyecto a futuro.

4.1.4. Codificación de la programación con Arduino.

Es el código del programa, el cual debe estar documentado para una mejor comprensión del desarrollo del proyecto.

4.2. Ejemplos de programas en Arduino

A continuación, se presentan diferentes ejemplos utilizando distintos conceptos explicados en los módulos anteriores.

4.2.1. Ejemplo 1-- Encendido de leds utilizando estructura condicional

Escriba un programa utilizando la estructura condicional y el residuo (mòdulo) que le permita encender un led ubicado en el pin número dos cuando el residuo sea par o que encienda un led ubicado en el pin número tres cuando el residuo sea impar.

Solución

Hardware requerido

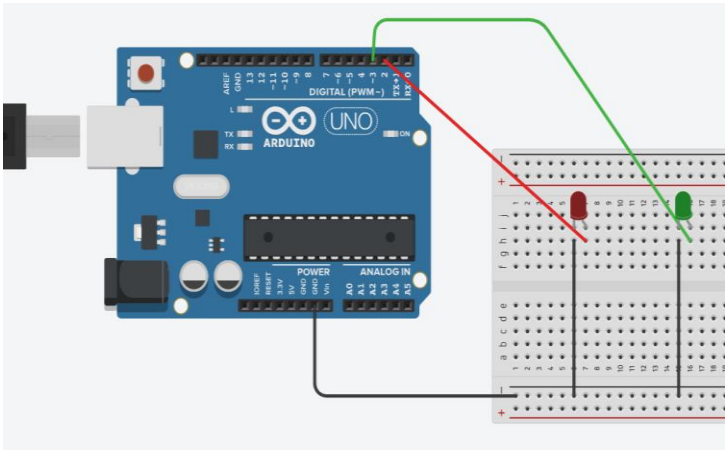
tarjeta de arduino

2 leds

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
// Ejemplo del uso de la estructura condicional compuesta y el módulo o residuo con leds

int pin2 = 2;
int pin3 = 3;
int val = 2;
int res = 0;

void setup() {
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);  }

void loop() {
  res = val%2;
  if (res == 0)
   { digitalWrite(pin2, HIGH);
    digitalWrite(pin3, LOW);
    delay(1000); }
  else
    { digitalWrite(pin3, HIGH);
     digitalWrite(pin2, LOW);
     delay(1000); }

val = val + 1;  }
```

4.2.2. Ejemplo 2 -- Encendido de cuatro leds utilizando estructura condicional

Escriba un programa utilizando la estructura condicional y el residuo (mòdulo) que le permita encender dos leds ubicados en los pines números 2 y 4 cuando el

residuo sea par o que encienda dos leds ubicados en los pines números 3 y 5 cuando el residuo sea impar.

Solución

Hardware requerido

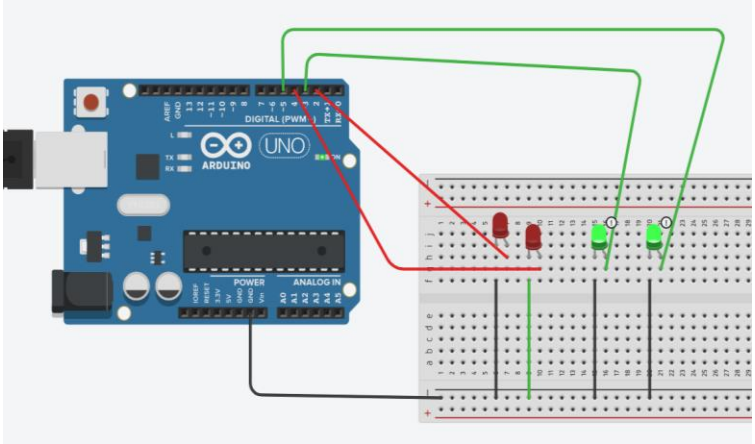
tarjeta de arduino

4 leds

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
// Ejemplo del uso de la estructura condicional compuesta y el módulo con 4 leds

int pin2 = 2;
int pin3 = 3;
int pin4 = 4;
int pin5 = 5;
int val = 2;
int res = 0;
```

```
void setup() {
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);
  pinMode(pin5, OUTPUT); }

void loop() {
  res = val%2;
  if (res == 0)
   { digitalWrite(pin2, HIGH);
     digitalWrite(pin4, HIGH);
    digitalWrite(pin3, LOW);
    digitalWrite(pin5, LOW);
    delay(1000); }
  else
    { digitalWrite(pin3, HIGH);
      digitalWrite(pin5, HIGH);
     digitalWrite(pin2, LOW);
     digitalWrite(pin4, LOW);
     delay(1000); }

val = val + 1;  }
```

4.2.3. Ejemplo 3 -- Encendido de cuatro leds utilizando switch case

Escriba un programa utilizando switch case y el residuo (mòdulo) que le permita encender dos leds ubicados en los pines números 2 y 4 cuando el residuo sea par o que encienda dos leds ubicados en los pines números 3 y 5 cuando el residuo sea impar.

Solución

Hardware requerido

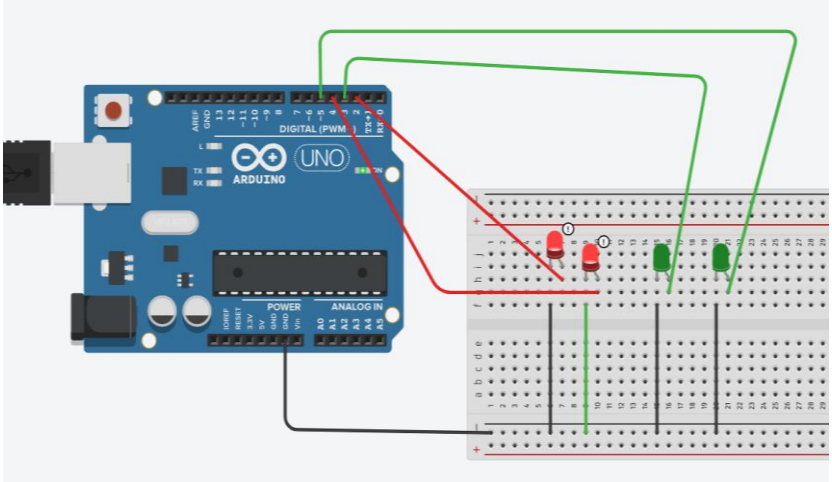
tarjeta de arduino

4 leds

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
//  Ejemplo utilizando SWITCH CASE
int val = 2;    int res = 0;
int pin2 = 2;   int pin3 = 3;
int pin4 = 4;   int pin5 = 5;
int pin6 = 6;   int pin7 = 7;

void setup()
 {
```

```
  pinMode(pin2, OUTPUT);  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);  pinMode(pin5, OUTPUT);
  pinMode(pin6, OUTPUT);  pinMode(pin7, OUTPUT);
  }

void loop() { res = val%2;
  switch (res) {
    case 0:
      digitalWrite(pin2, HIGH); digitalWrite(pin4, HIGH);
      digitalWrite(pin6, HIGH); digitalWrite(pin3, LOW);
      digitalWrite(pin5, LOW);  digitalWrite(pin7, LOW);
      delay(1000);  break;
   case 1:
      digitalWrite(pin2, LOW); digitalWrite(pin4, LOW);
      digitalWrite(pin6, LOW); digitalWrite(pin3, HIGH);
      digitalWrite(pin5, HIGH); digitalWrite(pin7, HIGH);
      delay(1000);  break;
 }
val++;
}
```

4.2.4. Ejemplo 4 -- Encendido de leds utilizando arreglos y while

Escriba un programa utilizando arreglos y el ciclo while que encienda una secuencia de leds, los apague y luego los vuelva a encender en orden inverso

Solución

Hardware requerido

tarjeta de arduino

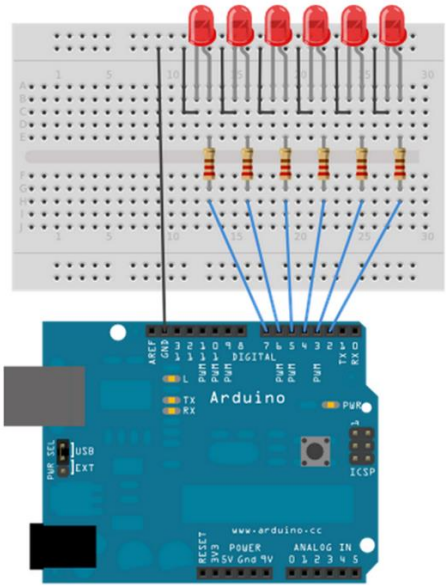
6 resistencias de 220 ohm

6 leds

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
/*  Ejemplo utilizando arreglos con While que enciende una secuencia de leds
 y vuelve a encenderlos en orden inverso  */

int timer = 100;           // VARIABLE QUE CONTIENE EL TIEMPO DE RETARDO
int ledPins[] = {2, 3, 4, 5, 6, 7};       // DECLARACIÓN DEL ARREGLO DE LEDs
int pinCount = 6;   // VARIABLE QUE CONTIENE LA CANTIDAD  DE PINES (EQUIVALENTE AL TAMAÑO DEL ARREGLO)
int thisPin = 0;

void setup() {
```

```
  // LOS ELEMENTOS DEL ARREGLO SON NUMERADOS DESDE 0 A  (pinCount
- 1).
  // USO DEL FOR PARA INICIALIZAR CADA PIN COMO SALIDA
    while (thisPin < pinCount)  {
    pinMode(ledPins[thisPin], OUTPUT);
     thisPin++;  }   }

void loop() {
  thisPin = 0;
  while (thisPin < pinCount)  {
    digitalWrite(ledPins[thisPin], LOW);
    delay(timer);
    thisPin++;  }

  thisPin = 0;
  while (thisPin < pinCount)  {
    digitalWrite(ledPins[thisPin], HIGH);
    delay(timer);
    thisPin++;  }
 }
```

4.2.5. Ejemplo 5 -- Encendido de leds utilizando arreglos y for

Escriba un programa utilizando arreglos y el ciclo while que encienda una secuencia de leds, los apague y luego los vuelva a encender en orden inverso

Solución

Hardware requerido

tarjeta de arduino

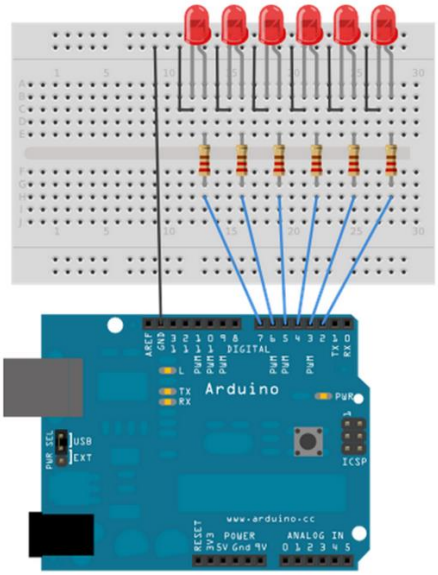
6 resistencias de 220 ohm

6 leds

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
/*  Ejemplo utilizando arreglos con el ciclo FOR que enciende una secuencia
de leds y vuelve a encenderlos en orden inverso  */

int timer = 100; // VARIABLE QUE CONTIENE EL TIEMPO DE RETARDO
int ledPins[] = {2, 3, 4, 5, 6, 7}; // DECLARACIÓN DEL ARREGLO DE LEDs
int pinCount = 6; /* VARIABLE QUE CONTIENE LA CANTIDAD DE PINES
                  (EQUIVALENTE AL TAMAÑO DEL ARREGLO) */
void setup() {
  /* LOS ELEMENTOS DEL ARREGLO SON NUMERADOS DESDE 0 A  (pinCount
- 1) */
  // USO DEL FOR PARA INICIALIZAR CADA PIN COMO SALIDA
  for (int thisPin = 0; thisPin < pinCount; thisPin++)  {
    pinMode(ledPins[thisPin], OUTPUT);    }   }
```

```
void loop() {
  // CICLO QUE ENCIENDE LOS PINES EN ORDEN ASCENDENTE
 for (int thisPin = 0; thisPin < pinCount; thisPin++) {
        digitalWrite(ledPins[thisPin], HIGH);
        delay(timer);
        digitalWrite(ledPins[thisPin], LOW);   }

  // CICLO QUE ENCIENDE LOS PINES EN ORDEN DESCENDENTE
  for (int thisPin = pinCount - 1; thisPin >= 0; thisPin--) {
        digitalWrite(ledPins[thisPin], HIGH);
        delay(timer);
        digitalWrite(ledPins[thisPin], LOW);  }   }
```

4.2.6. Ejemplo 6 – Utilización de la bocina

Escriba un programa que toca una pequeña melodía utilizando una bocina o piezo.

Solución

Hardware requerido

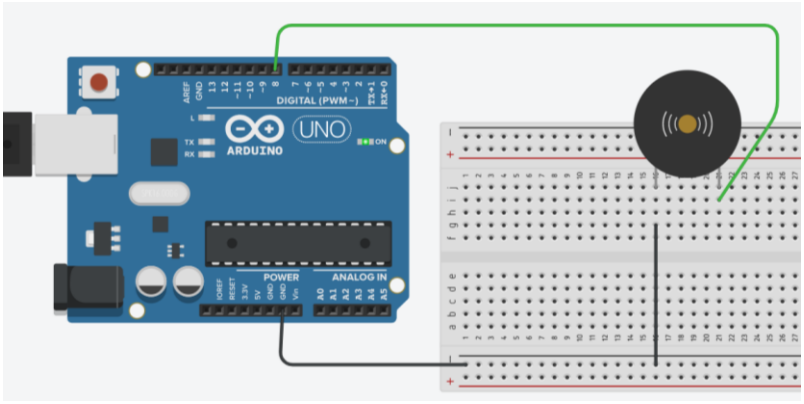
tarjeta de arduino

bocina o piezo

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
// notas de la melodìa:
int melody[] = {262, 196, 196, 220, 196, 0, 247, 262};
// duraiòn de las notas: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {4, 8, 8, 4,4,4,4,4 };

void setup() { // iterar sobre las notas de la melodía:

  for (int thisNote = 0; thisNote < 8; thisNote++) {
    // para calcular la duración de la nota, toma un segundo
    // dividido por el tipo de nota.
    // Por ejemplo cuarto de nota = 1000 / 4, octavo de nota = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);

    // para distinguir las notas se establece un tiempo mínimo entre ellas.
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // detenie la reproducción del tono:
    noTone(8);      }  }

void loop() { /* no es necesario repetir la melodía.*/ }
```

4.2.7. Ejemplo 7 – Utilización de fotocelda y la bocina

Escriba un programa que activa la bocina o piezo cuando una fotocelda LDR como sensor de luz.

Solución

Hardware requerido

tarjeta de arduino

bocina o piezo

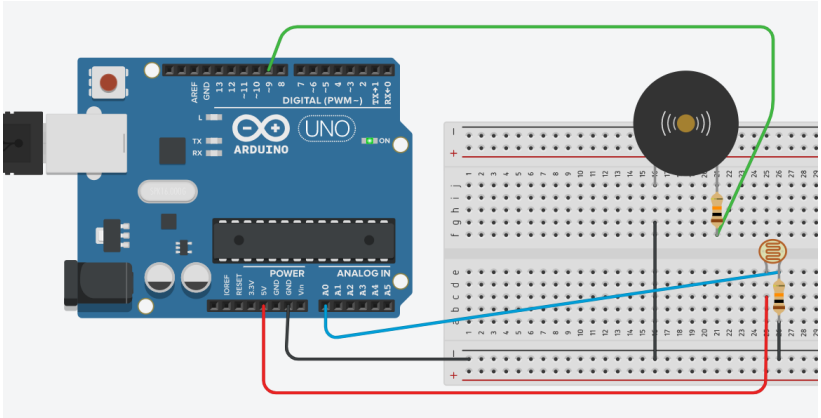
fotocelda LDR

2 resistencias de 10 KΩ

protoboard

cables

Esquema del circuito



Código del programa en Arduino

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

void loop() {
  //lectura de la fotoresistencia
  int sensorReading = analogRead(A0);
    Serial.println(sensorReading);
```

```
// mapea el rango de la entrada analoga (400 - 1000 de la fotoresistencia)
// a la salida del rango del tono (120 - 1500Hz)
  int thisPitch = map(sensorReading, 400, 1000, 120, 1500);

// ejecuta el tono:
tone(9, thisPitch, 10);
delay(1);    }
```

4.2.8. Ejemplo 8 – Utilización de fotocelda y led

Escriba un programa que encienda un led cuando una fotocelda LDR reciba luz.

Solución

Hardware requerido

tarjeta de arduino

led

fotocelda LDR

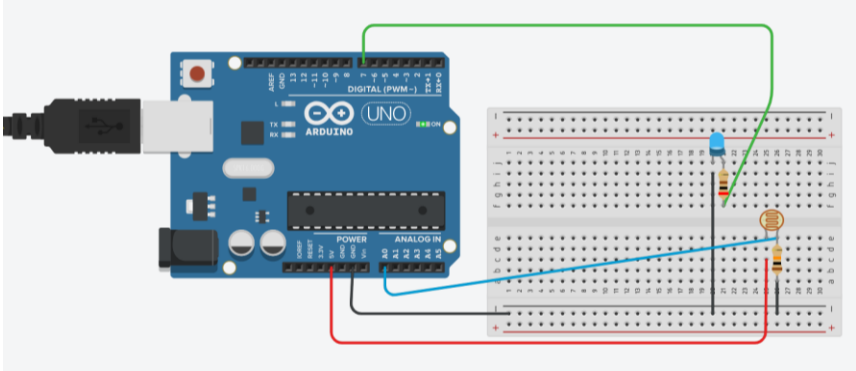
1 resistencias de 200 KΩ

1 resistencias de 10 KΩ

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
int sensor_luz = A0;  //Pin análogo en donde va conectada la fotocelda
int led_rojo = 7;
int valor_luz;
int brillo_led;

void setup() {
  pinMode(sensor_luz,INPUT);
  pinMode(led_rojo,OUTPUT);  }

void loop() {

  valor_luz = 1023 - analogRead(A0); //Mapeo inverso de los valores que toma la fotocelda
  brillo_led = map(valor_luz, 0, 1023, 0, 255);   //Almacena los valores en una variable
  analogWrite(led_rojo, brillo_led);   //Escribe los valores en forma de PWM al LED
}
```

4.2.9. Ejemplo 9 – Utilización de un servomotor

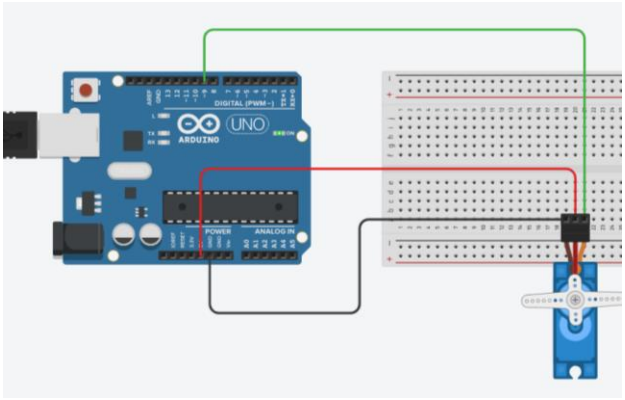
Escriba un programa que haga girar un servo motor.

Solución

Hardware requerido

tarjeta de arduino
servomotor
protoboard
cables

Esquema del circuito



Código del programa en Arduino

```
#include <Servo.h>
Servo myservo;  // crear objeto servo para controlar un servo
int pos = 0;    // variable para almacenar la posición del servo

void setup() {
  myservo.attach(9);  // conecta el servo en el pin 9 al objeto servo
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // va de 0 grados a 180 grados
    // in steps of 1 degree
    myservo.write(pos);              // le indica al servo que vaya a la posición en la variable 'pos'
    delay(15);                       }

for (pos = 180; pos >= 0; pos -= 1) { // va de 180 grados a 0 grados
    myservo.write(pos);              // le indica al servo que vaya a la posición en la variable 'pos'
    delay(15);                 }  }
```

4.2.10. Ejemplo 10 – Utilización del sensor ultrasónico

Escriba un programa que utilice un sensor ultrasónico de tal forma que cuando tenga un objeto a una distancia menor o igual a 15 cm se encienda el led.

Solución

Hardware requerido

tarjeta de arduino

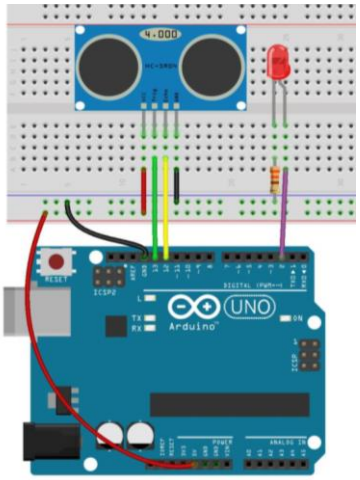
sensor ultrasónico

led

protoboard

cables

Esquema del circuito



Código del programa en Arduino

```
const int pinecho = 12;     const int pintrigger = 13;
const int pinled = 2;     unsigned int tiempo, distancia;

void setup() {
  Serial.begin(9600);              pinMode(pinecho, INPUT);
  pinMode(pintrigger, OUTPUT);     pinMode(2, OUTPUT);  }
```

```
void loop() {  // enviar pulso de disparo en el pin "trigger"
  digitalWrite(pintrigger, LOW);    delayMicroseconds(2);
  digitalWrite(pintrigger, HIGH);   delayMicroseconds(10);
  digitalWrite(pintrigger, LOW);

  tiempo = pulseIn(pinecho, HIGH);  // mide el tiempo en estado alto del pin "echo"

  /* la velocidad del sonido es de 340 m/s o 29 microsegundos por centímetro dividimos el tiempo del pulso entre 58, tiempo que tarda en recorrer ida y vuelta un centímetro la onda sonora */
  distancia = tiempo / 58;

  // enviar el resultado al monitor serial
  Serial.print(distancia);  Serial.println(" cm"); delay(200);

  // encender el led cuando se cumpla con cierta distancia
  if (distancia <= 15)
      { digitalWrite(13, HIGH);     delay(500);}
  else { digitalWrite(13, LOW);    }   }
```

ANEXO 1: PRUEBAS RAPIDAS

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
ASIGNACIÓN #1

Nombre de la Asignatura: INGENIERÍA DE SISTEMAS ROBÓTICOS
Módulo 1: Fundamentos de Robótica
Docente Responsable: Carlos Rovetto
Objetivo: Conocer los antecedentes históricos de la robótica
Recursos: lápiz, papel, bolígrafo, internet, computadora.
Instrucciones
✓ El trabajo debe ser entregado antes de terminar la clase.
Grupo: _____ Fecha: _____ **Puntos obtenidos: $\frac{\quad}{100}$**
Nombre: _____ #_____ Cédula: _____

INDICACIONES:
En la clase magistral previa se presentaron distintos tipos de robots y sus características principales. Utilizando los nombres ubicados en el cuadro, investigue y ubique en los espacios en blancos de la tabla el año y el nombre del robot correspondiente a la descripción del robot presentada.

PUMA	Electro	Brazos Boston y Stanford	AIBO	E1
Unimate	Irb6	Genghis	ASIMO	P2

AÑO	NOMBRE DEL ROBOT	DESCRIPCIÓN DEL ROBOT
		Podía caminar por comando de voz, sumar, volar globos y mover la cabeza y los brazos.
		Se instaló en la Ford Motors Company para atender una máquina de fundición de troquel.
		Visión Artificial para vehículos auto guiados. Stanford dotado de una cámara y controlado por computadora.
		Accionamiento completamente eléctrico.
		Brazo manipulador capaz de mover objetos. Es la base de la mayoría de los robots actuales.
		Prototipo que andaba con movimientos de las dos piernas.
		Primer robot andante.
		Primer robot humanoide bípedo auto regulable.
		Perro mascota robótico.
		Robot Humanoide.

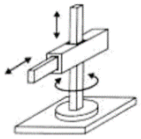
UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBOTICOS
ASIGNACIÓN #2

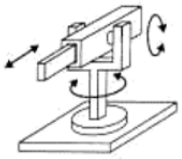
Nombre de la Asignatura: INGENIERÍA DE SISTEMAS ROBÓTICOS
Módulo 1: Introducción a la Robótica
Docente Responsable: Carlos Rovetto
Objetivo: Conocer la clasificación de los robots
Instrucciones

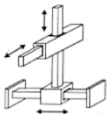
- ✓ Investigar sobre las diferentes configuraciones de robots según las combinaciones de las articulaciones y los tipos de articulaciones
- ✓ El trabajo debe ser entregado antes de la terminación de la clase.

Nombre: _____ Cédula: _____
Grupo: _____ Fecha: _____ **Puntos obtenidos: ___/100**

1. **Coloque el nombre correspondiente a cada robots según las combinaciones de las articulaciones (40 PUNTOS)**

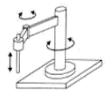

_____


_____

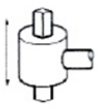

_____


_____

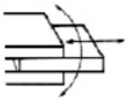
_____


_____

1. **Coloque el nombre y los grados de libertad correspondiente a cada articulación (60 PUNTOS)**


_____

_____


_____

_____


_____

_____


_____

_____


_____


_____

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
ASIGNACIÓN #3

Nombre de la Asignatura: INGENIERÍA DE SISTEMAS ROBÓTICOS
Módulo 1: Fundamentos de Robótica
Docente Responsable: Carlos Rovetto
Objetivo: Conocer las aplicaciones de la robótica
Recursos: lápiz, papel, bolígrafo, internet, computadora.
Instrucciones
✓ El trabajo debe ser entregado antes de terminar la clase.
Grupo: _____ Fecha: _____ **Puntos obtenidos: ___/100**
Nombre: _____ Cédula: _____

INDICACIONES:
Utilizando las áreas de aplicación ubicadas en el cuadro, investigue y ubique en los espacios en blancos la letra correspondiente a la categoría de la descripción dada.

I – Aplicaciones industriales de los robots
S – Aplicaciones de los robots de servicio
E – Plataformas de Robótica Educativa

_____ Fundición por inyección
_____ Medicina
_____ FisherTecnich
_____ Bioloid
_____ Transferencia de material
_____ Agricultura
_____ MakeBlock
_____ Control de Calidad
_____ Vehículos Submarinos
_____ Paletización
_____ Neulog Sense
_____ Montaje
_____ Espacio
_____ Construcción
_____ LEGO WeDo
_____ Laboratorios
_____ littleBits
_____ Operaciones de Procesamiento
_____ Makeblock
_____ LEGO Mindstorms EV3

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
ASIGNACIÓN #4

Nombre de la Asignatura: INGENIERÍA DE SISTEMAS ROBÓTICOS
Docente Responsable: Carlos Rovetto
Objetivo: Identificar un problema que pueda ser solucionado con Arduino
Recursos: papel, bolígrafo, internet, computadora.
Instrucciones
✓ El trabajo debe ser entregado antes de terminar la clase.

Grupo: _____ Fecha: _____ **Puntos obtenidos: ___/100**

Nombre: _____ Cédula: _____

Instrucciones
Investigar sobre un problema en el que usted pueda aportar la solución utilizando arduino. Se debe entregar la siguiente información al docente la cual constituye la propuesta del proyecto final que presentará el grupo proponente en la feria de la Facultad.

Criterios de Evaluación

	Puntos por Criterio	**Puntos Obtenidos**
Identificación del problema	30 puntos	_____
Solución propuesta	30 puntos	_____
Listado de materiales a utilizar	20 puntos	_____
Nombre del Proyecto	20 puntos	_____
Total		**___/100**

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
LABORATORIO #1

Nombre de la Asignatura: INGENIERÍA DE SISTEMAS ROBÓTICOS
Módulo 4: Construcción en el Sistema de Robótica Arduino
Docente Responsable: Carlos Rovetto
Objetivo: Aplicar los conceptos básicos para encender y apagar Led
Recursos: computadora, arduino, leds y otros componentes
Instrucciones
✓ Trabajar de forma grupal.
✓ El trabajo debe ser entregado antes de terminar la clase.

Nombre: _____ Cédula: _____

Grupo: _____ Fecha: _____ **Puntos obtenidos: ___/100**

ENUNCIADO

- Programar el encendido de tres leds (led4, led8 y led10) ubicados en las salidas digitales 4, 8 y 10 en serie de tal forma que cuando uno se encuentre encendido los otros dos restantes estén apagados. Compile el programa. **(15 PUNTOS)**
- Construya el circuito. **(10 PUNTOS)**
- Cargue el programa en la tarjeta de Arduino. Ejecute el programa **(30 PUNTOS)**
- Una vez comprobado su programa responda las siguientes preguntas y entregue a su profesor para su evaluación.

1. Para la construcción del circuito, ¿dónde se debe conectar el led4? **(5 PUNTOS)**

_____

2. Escriba la declaración de las variables que utilizó para: **(10 PUNTOS)**

Led8: _____

Led10: _____

3. Escriba el procedimiento ubicado en el loop que le permite ejecutar de forma correcta el programa. **(30 PUNTOS)**

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
LABORATORIO #2

Docente Responsable: Carlos Rovetto
Objetivo: Aplicar los conceptos de la estructura condicional y switch-case
Recursos: computadora, arduino, leds y otros componentes
Instrucciones
✓ El trabajo debe ser entregado antes de terminar la clase.

Nombre: _____ Cédula: _____

Grupo: _____ Fecha: _____ **Puntos obtenidos: ___/100**

ENUNCIADO
Programar el encendido de cuatro leds (led2, led3, led4 y led5) ubicados en las salidas digitales 2, 3, 4, y 5 de tal forma que se enciendan los pares y los impares se apagen. Luego de 1 segundo se enciendan los impares y se apaguen los pares. Esto se debe hacer utillizando:

a. Construya el circuito. **(10 PUNTOS)**
b. IF-THEN-ELSE (funcionamiento) **(20 PUNTOS)**

c. SWITCH-CASE (funcionamiento) **(20 PUNTOS)**

4. Escriba el procedimiento ubicado en el loop que le permite ejecutar de forma correcta el programa con la estructura condicional. **(25 PUNTOS)**

5. Escriba el procedimiento ubicado en el loop que le permite ejecutar de forma correcta el programa con SWITCH-CASE. **(25 PUNTOS)**

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS
INGENIERÍA DE SISTEMAS ROBÓTICOS
PARCIAL #1

Nombre: _____ Cédula: _____

Grupo: _____ Fecha: _____ **Puntos obtenidos:** $\frac{\quad}{100}$

1. Programar el potenciómetro de tal forma que cuando tenga el valor de 0 se encienda un led de un color y cuando tenga el valor de 1023 se encienda otro led de distinto color. Los valores del potenciómetro se deben mostrar en el monitor serial.
   - **d.** Construya el circuito. **(15 PUNTOS)**
   - **e.** Funcionamiento correcto **(15 PUNTOS)**
   - **f.** Escriba el código que le permite ejecutar de forma correcta el programa. **(30 PUNTOS)**

2. Programar un servomotor de tal forma que cuando gire en una dirección se encienda un led de un color y cuando el giro sea contrario se encienda otro led de distinto color.
   - **g.** Construya el circuito. **(10 PUNTOS)**
   - **h.** Funcionamiento correcto **(10 PUNTOS)**
   - **i.** Escriba el código que le permite ejecutar de forma correcta el programa. **(20 PUNTOS)**

BUENA SUERTE

UNIVERSIDAD TECNOLÓGICA DE PANAMÁ FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS INGENIERÍA DE SISTEMAS ROBÓTICOS
PARCIAL #2

Nombre: _____Cédula: _____

Grupo:_____ Fecha: _____ Puntos obtenidos: ____ 100

NO SE PUEDEN PRESTAR LAS PLACAS SIN LA AUTORIZACIÓN DEL DOCENTE Y LA PREVIA VERIFICACIÓN POR EL DOCENTE DEL PROGRAMA GRABADO EN LA MISMA.

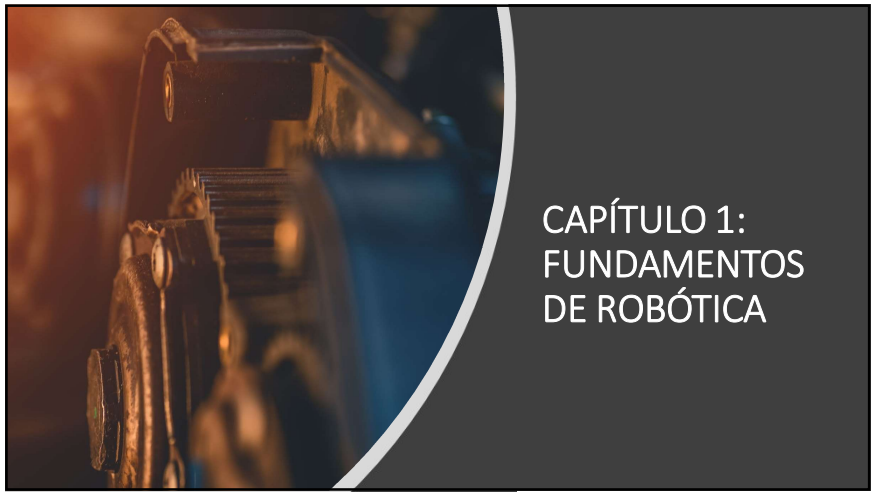
1. Programar el sensor ultrasónico de tal forma que cuando tenga un objeto a una distancia menor o igual a 10 cm se active una señal sonora con la bocina y cuando sea mayor se active otra señal sonora con la bocina. La distancia del objeto se debe mostrar en el monitor serial.
   - **j.** Construya el circuito. **(20 PUNTOS)**
   - **k.** Funcionamiento correcto. **(20 PUNTOS)**
   - **l.** Escriba el código que le permite ejecutar de forma correcta el programa. **(10 PUNTOS)**

2. Programar un fotoresistor (LDR) de tal forma que al disminuir la luz en un valor menor o igual a 700 se encienda un led de un color y se active un sonido con la bocina. Cuando el valor sea superior a 700 se encienda otro led de distinto color y se active otro sonido distinto con la bocina. También se debe mostrar en el monitor serial el valor del fotoresistor.
   - **a.** Construya el circuito. **(20 PUNTOS)**
   - **b.** Funcionamiento correcto. **(20 PUNTOS)**
   - **c.** Escriba el código que le permite ejecutar de forma correcta el programa. **(10 PUNTOS)**

BUENA SUERTE

ANEXO 2: PRESENTACIONES

1



2

CAPÍTULO 1: FUNDAMENTOS DE ROBÓTICA

Antecedentes históricos

Aunque la ciencia de la robótica solo surgió en el siglo XX, la historia de los robots y la automatización inventada por humanos tiene un pasado mucho más largo.

Los griegos construyeron estatuas que operaban con sistemas hidráulicos, los cuales se utilizaban para fascinar a los adoradores de los templos.

Muchas fuentes dan fe de la popularidad de los autómatas en la antigüedad y la Edad Media.

En la Edad Media, tanto en Europa como en Oriente Medio, los autómatas eran populares como parte de los relojes y el culto religioso.

En 1805 Henry Maillardet construyó una muñeca mecánica capaz de hacer dibujos.

Esculturas animadas egipcias (2000 a.C)

3

Estas son las cuatro leyes de la robótica establecidas por Asimov:

Ley Cero: Un robot no puede dañar a la humanidad, o a través de su inacción, permitir que se dañe a la humanidad.

Primera Ley: Un robot no puede dañar a un ser humano, o a través de su inacción, permitir que se dañe a un ser humano.

Segunda Ley: Un robot debe obedecer las órdenes dadas por los seres humanos, excepto cuando tales órdenes estén en contra de la Primera Ley.

Tercera Ley: Un robot debe proteger su propia existencia, siempre y cuando esta protección no entre en conflicto con la Primera y la Segunda Ley.

4

Origen y desarrollo de la robótica

Luego de los primeros autómatas los progenitores más directos de los robots fueron los manipuladores teleoperados. Con el objetivo de manejar elementos radioactivos sin riesgo para el operador se desarrolló el primer sistema de telemanipulación en 1948 por R. C. Goertz del Argonne National Laboratory.

En 1973, ASEA construyó el IRb6, el primer robot con accionamiento totalmente eléctrico.

En 1975 el ingeniero mecánico Victor Scheinman desarrolló un manipulador polivalente realmente flexible conocido como Brazo Manipulador Universal Programable (PUMA, siglas en inglés).

En 1987 se crea el E1 que es un prototipo que andaba en un paso estático a 0,25 Km/h con una cierta distinción entre el movimiento de las dos piernas.

En 1989 se crea el Genghis, el primer robot andante, es desarrollado en el Massachusetts Institute of Technology.

5

Definición del Robot

- Definición general de robot: un robot es una máquina programable que es capaz de moverse en la realización de una tarea.
- Definición de Robot Industrial Manipulador (ISO 8373): manipulador de 3 o más ejes, con control automático, reprogramable, multiaplicación, móvil o no, destinado a ser utilizado en aplicaciones de automatización industrial. Incluye al manipulador (sistema mecánico y accionadores) y al sistema de control (software y hardware de control y potencia).
- Robots de servicio (IFR): un robot que opera de manera semi o totalmente autónoma para realizar servicios útiles a los humanos y equipos, excluidas las operaciones de manu- factura.
- Robot doméstico: aquel robot destinado a ser usado por humanos sin formación técnica específica, al objeto de servirle como ayudante o colaborador en sus quehaceres o actividades diaria.

Otras definiciones muy importantes de robots son:

- Robot móvil (ISO 8373): robot que contiene todo lo necesario para su pilotaje y movimiento (potencia, control y sistema de navegación).
- Robot Teleoperado (ISO 8373): un robot que puede ser controlado remotamente por un operador humano, extendiendo las capacidades sensoriales y motoras de éste a localizaciones remotas.

6

Clasificación de los Robots

Un robot puede ser clasificado atendiendo a diferentes criterios o características o a la aplicación o tarea a que se destinan. A continuación, se presentan algunas de las clasificaciones más utilizadas.

Clasificación de los Robots según su generación o cronología

- **1ª Generación -- Manipuladores.**

Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

- **2ª Generación -- Robots de aprendizaje.**

Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano

- **3ª Generación -- Robots con control sensorizado.**

El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

- **4ª Generación -- Robots inteligentes.**

Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso

7

Clasificación de los Robots según su arquitectura

- **Poliarticulados**

Son robots cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo.

- Móviles

Estos robots tienen una gran capacidad de desplazamiento

- **Androides**

Estos robots intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano

- **Zoomórficos**

Constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos

- **Híbridos**

Estos robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas.

8

Morfología del robot

Un robot está formado por los siguientes elementos: estructura mecánica, transmisiones, sistema de accionamiento, sistema sensorial, sistema de potencia y control, y elementos terminales. En las siguientes secciones se explicarán cada uno de estos elementos.

9

Estructura Mecánica de un Robot

Un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos.

Tipos de Articulación:

- **Articulación de rotación (1GDL):** suministra un grado de libertad consistente en una rotación alrededor del eje de la articulación. Está articulación es una de la más empleada.
- **Articulación prismática(1GDL):** el grado de libertad consiste en una traslación a lo largo del eje de la articulación.
- **Articulación cilíndrica (2 GDL):** existen dos grados de libertad: una rotación y una traslación.
- **Articulación planar (2 GDL):** está caracterizada por el movimiento de desplazamiento en un plano existiendo, por lo tanto, dos grados de libertad.
- **Articulación esférica o rótula (3 GDL):** combina tres giros en tres direcciones perpendiculares en el espacio.
- **Articulación de tornillo (1 GDL):** se define por consistir en un movimiento rotacional que traslada verticalmente, similar a la articulación cilíndrica.

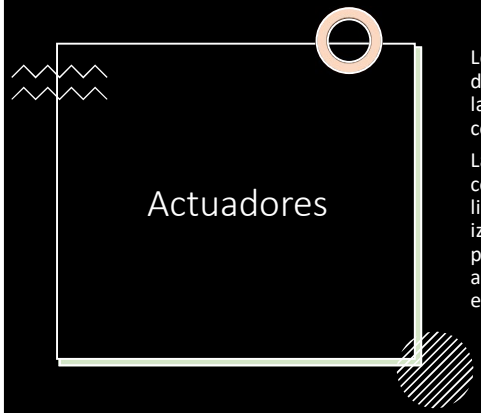
10

Transmisiones y reductores

Las transmisiones son elementos encargados de transmitir el movimiento desde los actuadores hasta las articulaciones, especialmente aquellas situadas en el extremo del robot, para reducir al máximo el momento de inercia del robot debido a que el mismo mueve sus extremos con aceleraciones elevadas.

Los reductores o engranajes son elementos encargados de adaptar el par y la velocidad de la salida del actuador a los valores adecuados para el movimiento de los elementos del robot. Generalmente se reduce la velocidad del actuador (de ahí el nombre).

11



Los actuadores generan el movimiento de los elementos del robot siguiendo las órdenes recibidas de la unidad de control.

La mayoría de los actuadores simples controlan únicamente 1 grado de libertad (GDL), el cual puede ser: izquierda-derecha o arriba-abajo. Los primeros robots industriales utilizaban actuadores hidráulicos para mover sus elementos.

12

Clasificación de los actuadores dependiendo del tipo de energía utilizada por los ejes principales del robot

- **Actuadores Neumáticos**

La fuente de energía es el aire a presión entre 5 y 10 bar. Los actuadores neumáticos se pueden utilizar para producir tanto movimiento de rotación como movimiento lineal.

- **Actuadores Hidráulicos**

Utilizan aceites minerales a una presión comprendida normalmente entre los 50 y 10 bar, en ocasiones supera los 300 bar.

- **Actuadores Eléctricos**

Son los más adecuados para robots que no requieren gran velocidad o potencia, pero que sí requieren precisión y repetitividad, como es el caso de la robótica industrial.

13

Sensores Internos

Los sensores son los dispositivos que permiten a un robot percibir su entorno. Estos permiten a los robots comprender y medir las propiedades geométricas y físicas de los objetos en su entorno circundante, como posición, orientación, velocidad, aceleración, distancia, tamaño, fuerza, momento, temperatura, luminancia, peso, etc.

14

Elementos Terminales

También llamados efectores finales, es el dispositivo o herramienta al final del brazo robótico. Tiene como objetivo interactuar con el entorno laboral. Los efectores finales varían y dependen de la aplicación del robot. Los accesorios utilizados pueden ser para agarrar, levantar, soldar, pintar y muchos más.

Esto significa que el robot estándar puede llevar a cabo una amplia gama de aplicaciones diferentes dependiendo del efector final que se le instale. Los pies de un robot humanoide o las ruedas de un robot móvil no se consideran efectores finales; se consideran parte de la movilidad del robot.

- **Tipos de efectores finales**

Herramientas: se utilizan en operaciones como la soldadura, pintar o perforar. Sus formas y los tipos son numerosos y variados.

Pinzas o dedos mecánicos: son el mecanismo de agarre de los robots. La pinza puede ser de dos dedos, tres dedos o incluso cinco dedos.

15

Categorías básicas de pinzas

Pinza mecánica: se utiliza como efector final en un robot para agarrar los objetos con sus dedos operados mecánicamente.

Pinza neumática: es un tipo específico de actuador neumático que normalmente involucra movimiento paralelo o angular de superficies.

Pinzas magnéticas: se utilizan con mayor frecuencia en un robot como efector final para agarrar materiales ferrosos.

16

Programación de Robot

A través de la programación le indicamos al robot la secuencia de acciones que debe llevar a cabo durante la realización de su tarea. Estas instrucciones contenidas en los algoritmos se traducirán después en un lenguaje de programación inteligible por el sistema de control del robot.

17

Métodos de Programación de Robots

Existen muchos criterios para clasificar los métodos de programación de robots. Esto se debe a la falta de estandarización de los métodos de programación para robots ya que la mayoría de los fabricantes de hardware de robot también proporcionan su propio software.

- Programación guiada o directa

En este tipo de programación el operario interviene guiando manualmente el brazo del robot, se describen los movimientos y se trazan las trayectorias necesarias para cumplir la función del robot.

- Programación textual o indirecta

Las instrucciones se encuentran escritas en un lenguaje de programación específico, luego este programa se graba en la memoria del robot para que el mismo realice las acciones indicadas en el mismo.

18

Niveles de la programación textual

Nivel Robot: se deben especificar cada uno de los movimientos que realizará el robot, así como también otros valores como lo son: la velocidad, direcciones de aproximación, salida y apertura y cierre de la pinza.

Nivel Objeto: en este tipo de programación las instrucciones se dan en función de los objetos a manejar.

Nivel tarea: en este tipo de programación se especifica qué es lo que debe hacer el robot en lugar de como hacerlo, esto reduce el programa a una única sentencia.

19

Requerimientos de un sistema de programación de robots

Actualmente, existen una serie de necesidades comunes en los métodos de programación de los robots que nos permite establecer una serie de características generales que se manifiestan en los elementos de programación que éstos contienen. Estos requerimientos generales para un sistema de programación de robots son los siguientes:

- **Entorno de programación:** está relacionado con la programación de las acciones que debe realizar el robot.
- **Modelado del entorno:** es la representación que tiene el robot de los objetos con los que interacciona.
- **Tipo de datos:** además de los tipos de datos usados en los lenguajes de programación, enteros, reales, booleanos, etc
- **Manejo de entradas/salidas:** estás le permiten al robot comunicarse con otras máquinas o procesos que cooperan con él.
- **Comunicaciones:** necesarias para que el robot pueda comunicarse con los otros robots o máquinas que estén participando de la producción.
- **Control de movimiento:** especifica el movimiento del robot a su destino incluyendo: la trayectoria que puede ser punto a punto, coordinadas o trayectoria continua (línea recta, interpolación circular, etc.),
- **Control del flujo de ejecución del programa:** en donde se utilizan las clásicas estructuras de control utilizadas en los lenguajes de programación para especificar el flujo de ejecución de las operaciones que debe realizar el robot.

20

Estandarización

Tener una estandarización en los métodos de programación de los robots no ha sido una tarea fácil. A través de los años han existido diferentes intentos para establecer una estandarización que sea aceptada a nivel mundial.

Una de las propuestas que ha ganado cierta aceptación, siendo norma DIN desde el año 1996 [DIN-96] y que fue sido incluido como modo de programación en algunos simuladores como COSIMIR de FESTO es la presentada por el Industrial Robot Language (IRL).

21

Diseño de control de una célula de trabajo

El correcto diseño de una célula de trabajo debe definir los distintos elementos con los que contará la célula de trabajo, tales como:

- elementos periféricos pasivos: mesas, alimentadores, utillajes, etc.
- elementos periféricos activos: manipuladores secuenciales, máquinas de Control Numérico, etc.
- seleccionar el robot más adecuada para la aplicación que se va a desarrollar
- definir y seleccionar la arquitectura de control, hardware y software, que todo sistema flexible de fabricación debe incluir

22

Características a considerar en la selección de un robot

Al momento de seleccionar un robot existen muchas características que se deben considerar, una de la más importante es evaluar para qué tipo de aplicaciones y de procesos va a realizar el robot. En función de esta primera evaluación, se debe considerar:

- la carga máxima que el robot puede soportar en su trabajo teniendo en cuenta si el robot cargará la pieza de trabajo objetivo de una posición a otra.
- el número de ejes que serán configurados para el robot.
- las características del controlador del robot, entre ellas, el tamaño compacto y peso ligero
- debe tener un software de programación fuera de línea asequible
- debe tener un bajo consumo energético.
- que el robot cumpla todos los códigos de seguridad actuales para proteger a los empleados y limitar la responsabilidad de la empresa
- que el robot tenga un momento de inercia máximo admisible elevado.

23

Seguridad en instalaciones robotizadas

La seguridad es un aspecto crítico durante el desarrollo y explotación de una célula robotizada. Esto se debe a que el robot posee un mayor índice de riesgo a un accidente que otra máquina de características similares.

Es por ello, que en los sistemas robotizados podemos dividir la seguridad en dos grandes áreas: una de ellas está directamente relacionada con el robot y que involucra la supervisión del sistema de control, velocidad máxima limitada, frenos mecánicos adicionales, comprobación de señales de autodiagnóstico, paradas de emergencia, entre otros, cuya responsabilidad recae en el fabricante.

24

Justificación económica

Se deben considerar distintos factores antes de realizar un proyecto de instalación de una célula robotizada. Para realizar el análisis económico es necesario conocer, además de los beneficios resultantes del proyecto relacionados con el aumento de la productividad debido a la utilización de un sistema robotizado y que es consecuencia directa de la realización del trabajo de forma más rápida y con menores interrupciones, lo siguiente:

- el tipo de instalación que se va a desarrollar
- el coste del robot
- coste de inversión que incluye el coste de herramientas y equipos especiales
- costes de instalación

25

Mercado de Robots

En 2018 los cinco mercados principales representaron el 74% de los robots industriales instalados a nivel global: China, Japón, República de Corea, Estados Unidos y Alemania.

Asia sigue siendo el mercado de robots industriales que lidera el ranking a nivel mundial. En 2018, las unidades instaladas en China y República de Corea disminuyeron, mientras que en Japón aumentaron considerablemente. En total, Asia creció un 1%.

En Europa, el segundo mercado más grande del mundo, las instalaciones de robots aumentaron un 14% y alcanzaron un nuevo pico por sexto año consecutivo.

En las Américas, la tasa de crecimiento subió un 20%, lo que también constituye un nuevo récord por sexto año consecutivo.

Para 2022 se espera que el segmento de robots industriales abarque un 72% del mercado y siga predominando, según los resultados publicados por la IDC

26

Aplicación de Robots

En la actualidad podemos ver una gran variedad de robots utilizados para diferentes aplicaciones en el mundo y, a medida que avanza la tecnología, es probable que los robots desempeñen un papel cada vez más importante en nuestras vidas.

Los diferentes tipos de robots y sus aplicaciones varían mucho, desde robots industriales hasta robots de conserjería en edificios comerciales. A continuación, se presentan las distintas aplicaciones para los robots industriales y los de servicio.

27

Aplicaciones industriales de los robots

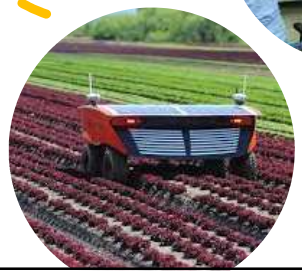
- **Soldadura por arco:** la soldadura por arco, o soldadura por robot, se convirtió en algo común en la década de 1980.
- **Soldadura por puntos:** la soldadura por puntos une dos superficies metálicas en contacto dirigiendo una gran corriente a través del punto, que derrite el metal y forma la soldadura entregada al punto en muy poco tiempo
- **Manipulación de materiales:** los robots de manipulación de materiales se utilizan para mover, embalar y seleccionar productos.
- **Cuidado de máquinas:** la automatización robótica para el cuidado de máquinas es el proceso de carga y descarga de materias primas en la maquinaria para procesar y supervisar la máquina mientras realiza un trabajo.
- **Pintura:** la pintura robótica se utiliza en la producción de automóviles y en muchas otras industrias, ya que aumenta la calidad y consistencia del producto.
- **Embalaje y Paletizado:** la mayoría de los productos se manipulan varias veces antes del envío final.
- **Montaje:** los robots ensamblan productos de manera rutinaria, eliminando tareas tediosas y tediosas.
- **Corte, rectificado, desbarbado y pulido mecánicos:** incorporar destreza a los robots proporciona una opción de fabricación que, de otro modo, sería muy difícil de automatizar.
- **Materiales de pegado, sellado adhesivo y pulverización:** los robots selladores están construidos con numerosas configuraciones de brazos robóticos que permiten al robot aplicar adhesivos a cualquier tipo de producto.

28

Aplicaciones de los robots de servicio

- **Robots de servicio minorista:** pueden descubrir cualquier error en las etiquetas de los estantes y evitar mostrar un precio diferente al que se muestra en la etiqueta del estante.
- **Robots de servicio doméstico:** se encargan de las tareas habituales que los humanos realizarían normalmente fuera del entorno industrial.
- **Robots de servicio científico:** llevan a cabo tareas recurrentes comúnmente involucradas en la investigación, como la recopilación de datos, el análisis y la formación de hipótesis.
- **Robots de servicio de eventos:** se utilizan para atender a clientes comerciales y visitantes en eventos.
- **Robots de servicio agrícola:** se están incorporando cada día más a todas las facetas de la actividad agrícola.
- **Robots médicos:** son robots de servicio profesional que se utilizan dentro y fuera de los entornos hospitalarios para mejorar el nivel de atención al paciente.
- **Robots de servicios médicos:** incluyen robots de telepresencia para cuidados remotos, robots desinfectantes para reducir las infecciones adquiridas en el hospital, robots que pueden extraer sangre de manera precisa y eficiente y exoesqueletos robóticos que brindan apoyo externo y entrenamiento muscular para la rehabilitación.
- **Robots de servicios logísticos:** son vehículos móviles guiados automatizados en almacenes e instalaciones de almacenamiento para transportar mercancías.





29

CAPÍTULO 2. INTRODUCCIÓN A LA ROBÓTICA CON ARDUINO.

30

CAPÍTULO 2. INTRODUCCIÓN A LA ROBÓTICA CON ARDUINO.

- **Arduino como Plataforma Electrónica de Código Abierto**

Arduino nació en el Ivrea Interaction Design Institute como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación.

Tan pronto se popularizó, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta desde placas simples de 8 bits hasta productos para aplicaciones de IoT, impresión 3D y entornos integrados.

31

Medidas de Seguridad con la Plataforma Electrónica Arduino

Se debe tener en cuenta que la plataforma es electrónica y aunque sea de baja potencia, se puede dañar tanto la placa como el ordenador donde se tiene conectada.

Algunas medidas que se deben tener presentes al trabajar con esta plataforma para garantizar la seguridad son:

- No trabajar sobre una mesa metálica o cualquier otro material conductor de electricidad.
- Se debe desconectar la placa cada vez que se deba cambiar o mover los cables, quitar o poner un led, etc.
- No conectar dispositivos que requieran más de 40mA de manera directa.
- Utilizar fuentes de alimentación estables y seguras.
- Tener un área de trabajo limpia y evitar tener vasos o bebidas cerca que se puedan derramar.
- Se puede medir con un multímetro y múltiples conexiones, la conductividad entre tierra y los pines de potencia para asegurar que no haya ningún corto.

32

Uso Arduino en la Robótica

Esta plataforma de código abierto ha sido de gran ayuda tanto para principiantes como para expertos. Miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos se han realizado con Arduino.

Las contribuciones de la comunidad que la utiliza se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda para los que trabajan con ella.

33

Hardware Arduino

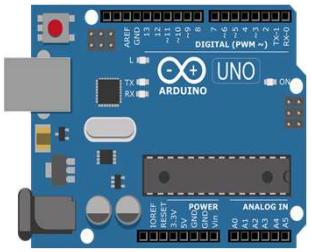
El hardware de Arduino está formado por distintas partes como son entradas, salidas, alimentación, comunicación y shields. Cada una de ellas se explicarán a continuación.

- **Placas Arduino de Entrada / Salida**

Hay varios tipos de placas Arduino disponibles en función de los diferentes microcontroladores utilizados. Todas las placas Arduino están programadas a través del IDE de Arduino.

- **Entradas/Salidas y esquema de pines de la placa Arduino**

En las siguientes dos subsecciones de explican los tipos de entradas y salidas de la placa Arduino y su ubicación en la misma.



34

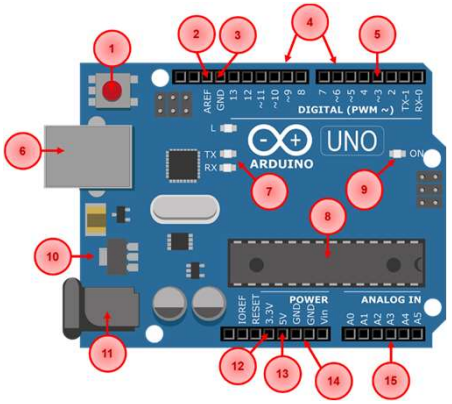
Entradas/Salidas y esquema de pines de la placa Arduino

En las siguientes dos subsecciones de explican los tipos de entradas y salidas de la placa Arduino y su ubicación en la misma.

- **Botón de reinicio:** esto reiniciará cualquier código que se cargue en la placa Arduino.
- **AREF:** Significa "Referencia analógica" y se utiliza para establecer una referencia de voltaje externo.
- **Pin de tierra:** hay algunos pines de tierra en el Arduino y todos funcionan igual.
- **Entrada / salida digital:** los pines 0-13 se pueden usar para entrada o salida digital.
- **PWM:** The pins marked with the (~) symbol can simulate analog output.
- **Conexión USB:** se utiliza para encender su Arduino y cargar proyectos.
- **TX / RX:** LED de indicación de transmisión y recepción de datos.
- **Microcontrolador ATmega:** este es el cerebro y es donde se almacenan los programas.
- **Indicador LED de alimentación:** este LED se enciende cada vez que la placa está conectada a una fuente de alimentación.
- **Regulador de voltaje:** controla la cantidad de voltaje que ingresa a la placa Arduino.
- **Conector de barril de alimentación de CC:** se utiliza para alimentar su Arduino con una fuente de alimentación.
- **Pin de 3,3 V:** este pin proporciona 3,3 voltios de potencia a sus proyectos.
- **Pin de 5V:** este pin suministra 5 voltios de potencia a sus proyectos.
- **Pines de tierra:** hay algunos pines de tierra en el Arduino y todos funcionan igual.
- **Pines analógicos:** estos pines pueden leer la señal de un sensor analógico y convertirla en digital.

35

Esquema y pines de la placa Arduino



36

Placa Arduino

Dependiendo de nuestras necesidades Arduino dispone de una amplia variedad de placas y shields para usar. Un shield es una placa compatible que se puede colocar en la parte superior de los arduinos y permite extender las capacidades del arduino.

A continuación, se explica con mayor detalle las características que poseen las placas de Arduino.

37

Características técnicas y físicas de la placa Arduino

- Las placas Arduino pueden leer señales de entrada analógicas o digitales de diferentes sensores y convertirlas en una salida, como activar un motor, encender / apagar LED, conectarse a la nube y muchas otras acciones.
- Puede controlar las funciones de su placa enviando un conjunto de instrucciones al microcontrolador en la placa a través de Arduino IDE (conocido como software de carga).
- A diferencia de la mayoría de las placas de circuitos programables anteriores, Arduino no necesita una pieza adicional de hardware (llamada programador) para cargar un nuevo código en la placa. Simplemente puede usar un cable USB.
- El IDE de Arduino utiliza una versión simplificada de C ++, lo que facilita el aprendizaje de la programación.
- Arduino proporciona un factor de forma estándar que divide las funciones del microcontrolador en un paquete más accesible.
- Para trabajar con Arduino es importante una placa de pruebas sin soldadura.
- Se utilizan cables para formar los circuitos conectando resistencias, interruptores y otros componentes juntos.

38

Entrada y Salida de la placa Arduino

Existen dos tipos de entradas y salidas de la placa Arduino las analógicas y las digitales.

Entradas/salidas analógicas

Las entradas analógicas van marcadas con una A delante. La cantidad de entradas que tenga cada placa depende del modelo, por ejemplo, el Arduino UNO posee 6 entradas analógicas (A0-A5), mientras que otros modelos como el NANO posee 8 entradas y el MEGA, 16 entradas

Entradas/salidas digitales

Son pines que pueden ser activado, o sea poner tensión, o desactivados, es decir quitarle la tensión. Este proceso es similar a escribir 0 que equivale a 0 V y 1 para 5V.

39

Forma de alimentación de la placa Arduino

Debido a que Arduino puede suministrar solo alrededor de 40ma de corriente a través de cualquiera de sus pines de salida, es severamente limitado a lo que puede alimentar por sí mismo.

Un LED rojo típico de 5 mm requiere aproximadamente 30ma de corriente, por lo que el Arduino no tiene problemas para encenderlo hasta el 100%, pero cualquier otro dispositivo que requiera más corriente tendrá problemas para activarlo.

Usar el Arduino para controlar un dispositivo de alta potencia requiere el uso de un "amplificador". También llamado "búfer de señal", un amplificador simplemente reproduce una señal de entrada de baja potencia, con una potencia de salida mucho mayor para conducir una carga.

40

Forma de comunicación de la placa Arduino

Se han definido muchos protocolos de comunicación para lograr el intercambio de datos. Cada protocolo se puede clasificar en una de las dos categorías: paralelo o en serie.

- **Comunicación paralela**

La conexión en paralelo entre el Arduino y los periféricos a través de puertos de entrada / salida es la solución ideal para distancias más cortas de hasta varios metros.

- **Módulos de comunicación en serie**

Hoy en día, la mayoría de las placas Arduino están construidas con varios sistemas diferentes para la comunicación en serie como equipo estándar.

La decisión de cuál sistema utilizar depende de los siguientes factores:

- ¿Con cuántos dispositivos tiene que intercambiar datos el microcontrolador?
- ¿Qué tan rápido debe ser el intercambio de datos?
- ¿Cuál es la distancia entre estos dispositivos?
- ¿Es necesario enviar y recibir datos simultáneamente?

41

Forma de programación de la placa Arduino

- Una vez que se haya creado el circuito en la placa de pruebas, deberá cargar el programa, conocido como sketch (boceto), en Arduino.

Sintaxis

Un punto y coma debe seguir cada declaración escrita en el lenguaje de programación Arduino.

Se utiliza doble barra inclinada (//) para agregar comentarios al código.

Las llaves se usan para encerrar instrucciones adicionales realizadas por una función. Siempre hay una llave de apertura y una llave de cierre. Si olvida cerrar una llave, el compilador enviará un código de error.

42

Tipos de Memoria

Memoria flash (espacio del programa), es donde se almacena el boceto de Arduino.

SRAM (memoria estática de acceso aleatorio) es donde el boceto crea y manipula variables cuando se ejecuta.

EEPROM es un espacio de memoria que los programadores

43

Protección de sobrecarga del USB de la placa Arduino

Aunque la mayoría de las computadoras ofrecen protección interna, la cual incorpora un fusible rearmable de intensidad máxima 500mA con la intención de proteger tanto la placa Arduino como el bus USB de sobrecargas y cortocircuitos.

Si circula una intensidad mayor a 500mA por el bus USB (Intensidad máxima de funcionamiento), el fusible salta rompiendo la conexión de la alimentación.

44

Reseteo automático de la placa Arduino

Se puede restablecer la placa Arduino de dos formas.

Primero, usando el botón de reinicio que se encuentra en la placa.

En segundo lugar, puede conectar un botón de reinicio externo al pin Arduino etiquetado RESET.

45

Software Arduino

- El software Arduino o entorno de desarrollo integrado de Arduino (IDE), es donde se escriben los programas que luego se cargarán al hardware Arduino y que permite establecer la comunicación entre ellos.
- El entorno de desarrollo integrado de Arduino (IDE) contiene un editor de texto para escribir código, un área de mensajes, una consola de texto, una barra de herramientas con botones para funciones comunes y una serie de menús.

46

Instalando el Software Arduino para Windows.

- **Obteniendo una placa Arduino y un cable USB**

Actualmente, existen muchas variaciones de las placas Arduino que son las oficiales, mientras que, hay cientos más de competidores que ofrecen clones. A la hora de elegir una placa se debe considerar el tipo de proyecto que se va a realizar.

Las placas oficiales son aquellas placas con el nombre Arduino. Dentro de la familia de placas Arduino, la placa Arduino Uno es la más utilizada y documentada del mercado debido a sus excelentes funciones y facilidad de uso.

47

Descargando e instalando el entorno Arduino para Windows.

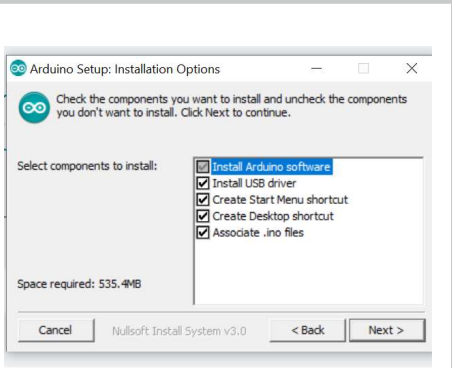
El IDE de Arduino se puede descargar desde la página en el sitio web oficial de Arduino https://www.arduino.cc/en/Main/Software. Debe seleccionar el software, que sea compatible con su sistema operativo (Windows, IOS o Linux).

48

Instalando los Drivers USB

Cuando se ha descargado el Arduino IDE Installer se procede a la instalación del software, así como también de los componentes que queremos que se instalen.

También podemos aceptar la instalación de los drivers como se muestra en la siguiente figura.



49

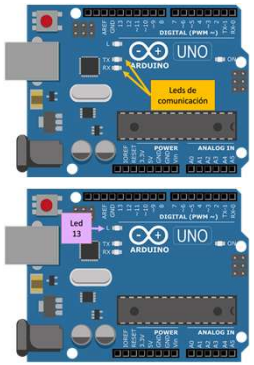
Instalando y conectando la placa Arduino.

La conexión USB con la PC es necesaria para programar la placa. La placa obtiene energía automáticamente de la conexión USB a la computadora o de una fuente de alimentación externa.

Al conectar la placa Arduino a su computadora con el cable USB, se encienden:

los leds TX Y RX indicando que se estableció la comunicación entre la placa y la computadora.

el led L que corresponde al pin digital 13 en la placa indicando que hay conexión con la placa, este continuará parpadeando mientras esté funcionando la placa.



50

Ejecutando el entorno Arduino.

Para ejecutar el entorno Arduino debemos abrir el programa Arduino, se abrirá el IDE para que pueda escribir el programa que utilizará para indicarle a la placa lo que debe hacer.

El lenguaje que usa Arduino es muy parecido a C ++ por lo que es muy fácil de aprender para ustedes.

51

Configuración y descripción del entorno: Subiendo un programa.

Una vez que haya seleccionado el puerto serie y la placa correctos, presione el botón de carga en la barra de herramientas o seleccione el elemento de carga en el menú. En la mayoría de las placas, verá parpadear los LED RX y TX mientras se carga el boceto.

52

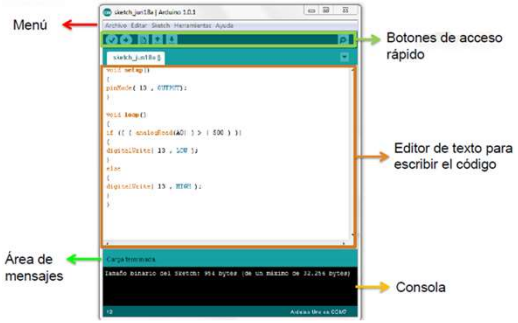
Buscando que el Led parpadee.

Uno de los primeros programas que se ejecuta en la placa de Arduino es el del led que parpadea. Este es un programa sencillo, cuyo objetivo es encender y apagar la luz del led. El parpadeo de un led, es el Hello World de los microcontroladores. La sigla LED significa Diodo Emisor de Luz (En inglés: Light Emitting Diode)

Sketch o Programa: El software Arduino IDE se debe escribir el siguiente programa que permite encender el led. En la imagen se muestra el programa documentado.

53

Preámbulo al Entorno de Trabajo del Software Arduino para Windows



54

Barra de Herramientas del Entorno Arduino para Windows.

También conocida como botones de acceso rápido, nos permite ejecutar algunos de los comandos más básicos sin tener que entrar al menú. A continuación, se presenta la función de cada símbolo que aparece en la barra de herramientas Arduino IDE.

1. **Verificar:** se utiliza para comprobar si hay algún error de compilación.
2. **Subir:** se utiliza para cargar un programa en la placa Arduino.
3. **Nuevo:** utilizado para crear un nuevo sketch.
4. **Abrir:** se utiliza para abrir un nuevo sketch.
5. **Salvar:** se utiliza para guardar su sketch.
6. **Monitor Serie:** abre una consola con la que podemos comunicarnos con la placa Arduino.

55

Menús del Entorno Arduino para Windows.

Editar, Sketch, Herramientas, Ayuda. Algunas de las opciones contenidas dentro de estos menús se explican a continuación.

Archivo

- **Nuevo:** Crea una nueva instancia del editor, con la estructura mínima de un sketch ya en su lugar.
- **Abrir:** Permite cargar un archivo de programa o sketch navegando por las unidades y carpetas de la computadora.
- **Abrir reciente:** Proporciona una lista breve de los programas o sketches más recientes, listos para abrirse.
- **Ejemplos:** Cualquier ejemplo proporcionado por el software Arduino (IDE) o la biblioteca se muestra en este elemento de menú.
- **Preferencias:** Abre la ventana Preferencias donde se pueden personalizar algunas configuraciones del IDE.

56

Editar

- **Deshacer / Rehacer:** Retrocede uno o más pasos que realizó durante la edición.
- **Copiar al foro:** Copia el código de su sketch en el portapapeles en una forma adecuada para publicar en el foro, completa con colores de sintaxis.
- **Copiar como HTML:** Copia el código de su sketch al portapapeles como HTML, adecuado para incrustarlo en páginas web.
- **Comentar / Descomentar:** Pone o elimina el marcador de comentario // al principio de cada línea seleccionada.
- **Aumentar / Disminuir sangría:** Agrega o resta un espacio al principio de cada línea seleccionada, moviendo el texto un espacio a la derecha o eliminando un espacio al principio.
- **Buscar:** Abre la ventana Buscar y reemplazar, donde puede especificar el texto para buscar dentro del boceto actual de acuerdo con varias opciones.

Programa o sketch

- **Verificar / Compilar:** Verifica su sketch en busca de errores al compilarlo; informará el uso de la memoria para el código y las variables en el área de la consola.
- **Subir:** Compila y sube el archivo binario en la placa configurada a través del puerto configurado.
- **Exportar binario compilado:** Guarda un archivo .hex que puede guardarse como archivo o enviarse a la placa usando otras herramientas.
- **Incluir biblioteca:** Agrega una biblioteca a su boceto insertando declaraciones #include al comienzo de su código. Además, desde este elemento del menú puede acceder al Administrador de bibliotecas e importar nuevas bibliotecas desde archivos .zip.
- **Agregar archivo:** Agrega un archivo de origen al sketch (se copiará desde su ubicación actual). El nuevo archivo aparece en una nueva pestaña en la ventana de sketch.

57

Herramientas

- **Auto formato:** formatea su código, es decir, lo sangra para que las llaves de apertura y cierre se alineen, y que las declaraciones dentro de las llaves tengan más sangría.
- **Archivar sketch:** Archiva una copia del sketch actual en formato .zip. El archivo se coloca en el mismo directorio que el sketch.
- **Reparar codificación y recargar:** Corrige posibles discrepancias entre la codificación del mapa de caracteres del editor y los mapas de caracteres de otros sistemas operativos.
- **Monitor serie:** Abre la ventana del monitor serie e inicia el intercambio de datos con cualquier placa conectada en el puerto seleccionado actualmente.
- **Placa:** Seleccione la placa que está utilizando.
- **Puerto:** Este menú contiene todos los dispositivos seriales (reales o virtuales) de su máquina.

Ayuda

- Aquí encontrará fácil acceso a varios documentos que vienen con el software Arduino (IDE).

58

Preferencias del Entorno Arduino para Windows.

Algunas preferencias se pueden controlar desde el cuadro de diálogo Preferencias dentro del entorno Arduino. Acceda a él desde el menú Archivo en Windows o Linux, o desde el menú Arduino en Mac.

Otras preferencias deben cambiarse en el archivoferences.txt. La ubicación de este archivo se muestra en el cuadro de diálogo de preferencias.

Solo edite el archivo cuando Arduino no se esté ejecutando; de lo contrario, sus cambios se sobrescribirán cuando Arduino salga.

59

CAPÍTULO 3. LENGUAJE DE PROGRAMACIÓN PARA ARDUINO.

60

CAPÍTULO 3.LENGUAJE DE PROGRAMACIÓN PARA ARDUINO.

- **Estructura básica del lenguaje de programación Arduino.**

La estructura básica del lenguaje de programación de Arduino se compone principalmente de dos funciones necesarias para que el programa trabaje, la función setup() y la función loop(), las cuales encierran bloques que contienen declaraciones e instrucciones. La función setup() es la parte encargada de recoger la configuración y la función loop() contiene el programa que se ejecutará cíclicamente

- **Función setup ()**

Es la primera función a ejecutar en el programa. Se utiliza para inicializar las variables, configurar los modos de trabajo de los pines, configuración de la comunicación en serie, comenzar a usar bibliotecas, entre otros.

- **Función loop ().**

La función loop () es donde residirá el cuerpo del programa. En esta función todo el código que se encuentre entre las llaves se repite una y otra vez, en un ciclo. La función loop () no devuelve ningún valor.

61

Funciones del lenguaje de programación Arduino.

Una función es un bloque de código que tiene un nombre y un conjunto de instrucciones que son ejecutados cuando se llama a la función. Pueden ser escritas para realizar tareas repetitivas y para reducir el tamaño de un programa.

- Algunas ventajas de utilizar funciones son:
- Reduce la posibilidad de errores en la modificación, si es necesario cambiar el código.
- Hacen que sea más fácil reutilizar el código en otros programas haciéndolo más modular.
- Hace que el código sea más legible.
- Hacen que todo el sketch sea más pequeño y compacto porque las secciones de código se reutilizan muchas veces.

Declaración del tipo de la función Declarar el tipo de la función, está asociado al valor retornado por la función. Este valor puede ser de tipo int, byte, long, float, void.

Declaración del nombre de la función Después de declarar el tipo de dato que devuelve la función, se declara el nombre de la función y entre paréntesis se escribirán, si es necesario, los parámetros que se deben pasar a la función para que se ejecutar.

Comentarios de línea // Se utiliza para aquellos que son comentarios cortos, generalmente pueden colocarse en una línea. Para este tipo de comentarios se utiliza la doble barra invertida.

Llaves { } Las llaves se utilizan en distintas codificaciones, para encerrar las instrucciones realizadas por una función, en las estructuras condicionales o en las de repetición, etc

Punto y coma (;) Se usa para finalizar una declaración

62

Variables del lenguaje de programación Arduino

Una variable es una forma de nombrar y almacenar un valor para su uso posterior por parte del programa, como los datos de un sensor o un valor intermedio utilizado en un cálculo.

Declaración de variable

Antes de utilizar una variable, debe ser declarada y se le puede asignar un valor. En la declaración de la variable se indica el tipo de datos que almacenará (int, float, long, etc).

Sintaxis para declarar una variable

type var = val;

- Parámetros utilizados en la declaración de una variable:

type: tipo de variable

var: nombre de la variable.

val: el valor a asignar a esa variable.

63

Ámbito de la variable.

El sitio en el que la variable es declarada determina el ámbito de la misma. El ámbito determina en qué partes del programa puede ser usada la variable.

Variables locales

Son variables que se declaran dentro de una función o bloque. Estas serán válidas solo dentro de esa función o bloque y se destruirá al terminar la función o bloque

Variables globales

Las variables globales se definen fuera de todas las funciones, generalmente en la parte superior del programa.

64

Constantes del lenguaje de programación Arduino

El lenguaje de programación de Arduino tiene unos valores predeterminados, que son llamados constantes. Las constantes se clasifican en grupos. A continuación, se explican las constantes más utilizadas.

True / False

- Son las dos constantes booleanas predefinidas que se utilizan para representar que un valor es verdadero o falso.

False: se define como 0 (cero).

True: se define como 1.

High / Low

- Estas constantes definen los niveles de los pines como HIGH o LOW y son empleados cuando se leen o escriben en las entradas o salidas digitales.

Input / Output

- Son las constantes empleadas con la función pinMode() para definir el tipo de un pin digital usado como entrada INPUT o salida OUTPUT.

65

Tipos de datos del lenguaje de programación Arduino.

- **Byte:** Almacena un valor numérico de 8 bits. Tienen un rango de 0-255.
- **Int. :** Almacena un valor entero de 16 bits con un rango de 32,767 a -32,768.
- **Long:** Valor entero almacenado en 32 bits con un rango de 2,147,483,647 a -2,147,483,648.
- **Float:** Tipo coma flotante almacenado en 32 bits con un rango de 3.4028235E+38 a -3.4028235E+38.
- **Arrays:** Se trata de una colección de valores que pueden ser accedidos con un número de índice, en donde el primer valor del índice es 0.

66

Aritmética: Operadores aritméticos del lenguaje de programación Arduino.

Operadores Aritméticos.

- Son operadores que devuelven el resultado de operaciones tales como la suma, la diferencia, el producto o el cociente (respectivamente) de los dos operandos. La operación se realiza utilizando el tipo de datos de los operandos, por ejemplo, para la siguiente operación 9/4 en donde 9 y 4 son enteros (tipo int) da como resultado 2 manteniendo el tipo de datos entero (int).
- Esto también significa que la operación puede desbordarse si el resultado es mayor que el que se puede almacenar en el tipo de datos. Si los operandos son de diferentes tipos, se utiliza el tipo "más grande" para el cálculo.

67

Asignaciones Compuestas.

Son operaciones que combinan una operación aritmética con una variable asignada. Este tipo de operaciones son usualmente utilizadas en los ciclos, en especial en el ciclo for. A continuación, se listan las operaciones de asignación compuesta.

- **Operador de incremento:** aumenta el valor entero en uno.
- **Operador de decremento:** decrementa el valor entero en uno.
- **Adición compuesta:** agrega el operando derecho al operando izquierdo y asigna el resultado al operando izquierdo
- **Sustracción compuesta:** resta el operando derecho del operando izquierdo y asigna el resultado al operando izquierdo
- **Multiplicación compuesta:** multiplica el operando derecho con el operando izquierdo y asigna el resultado al operando izquierdo
- **División compuesta:** divide el operando izquierdo entre el operando derecho y asigna el resultado al operando izquierdo
- **Modulo compuesto:** toma módulo usando dos operandos y asigna el resultado al operando izquierdo.

68

Operadores de Comparación.

Se utilizan para realizar operaciones comparativas entre variables o contantes entre sí. Son muy utilizados en estructuras de control condicionales, tales como, if, while, etc. Los operadores de comparación disponibles en Arduino son las siguientes:

- **== (igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando los dos operandos son iguales.
- **!= (no igual que):** compara la variable de la izquierda con el valor o la variable de la derecha del operador. Devuelve verdadero cuando los dos operandos no son iguales.
- **< (menor que):** la comparación es verdadera si x es igual a, y es falso si x no es igual a y
- **> (mayor que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es mayor (más grande) que el operando de la derecha.
- **<= (menor o igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es menor o igual que el operando de la derecha.
- **>= (mayor o igual que):** compara la variable de la izquierda con el valor de la variable de la derecha del operador. Devuelve verdadero cuando el operando de la izquierda es mayor o igual que el operando de la derecha.

69

Operadores Lógicos.

Los operadores lógicos o booleanos permiten comparar dos variables o constantes entre sí. Estos devuelven un VERDADERO o FALSO dependiendo del operador. Existen tres operadores booleanos:

&& (y): la operación y (and) lógico da como resultado verdadero solo si ambos operandos son verdaderos. Este operador se puede utilizar dentro de la condición de una sentencia if, while, etc.

|| (o): la operación o (or) lógico da como resultado verdadero si cualquiera de los dos operandos es verdadero. Este operador se puede utilizar dentro de la condición de una sentencia if, while, etc.

! (no): El operador no (not) lógico da como resultado verdadero si el operando es falso y viceversa. Se puede utilizar para invertir el valor booleano como se muestra en la primera instrucción de ejemplo o dentro de la condición de una sentencia if, while, etc.

70

Control de flujo: Sentencias condicionales del lenguaje Arduino.

Son estructuras que nos permiten tomar decisiones durante la ejecución del programa. Estas requieren que se especifique una o más condiciones para ser evaluadas o probadas por el programa.

- **If:** toma una expresión entre paréntesis y una declaración o bloque de declaraciones.
- **If. . . else:** una instrucción if puede ir seguida de una instrucción else opcional, que se ejecuta cuando la expresión es falsa.
- **for:** se usa para repetir un bloque de declaraciones entre llaves. Por lo general, se usa un contador de incrementos para incrementar y terminar el ciclo.
- **while:** este ciclo se repetirá de forma continua e infinita, hasta que la expresión dentro del paréntesis, () se vuelva falsa.
- **Do. . . while:** este ciclo funciona de la misma manera que el bucle while, con la excepción de que la condición se prueba al final del bucle

71

Entradas y salidas digitales del lenguaje de programación Arduino.

Los pines digitales se pueden configurar como entradas o salidas que reciben niveles altos (5V) o bajos (0V) de tensión y que son interpretados como un 1 o un 0 respectivamente.

Para controlar estas salidas /entradas, se utilizan las siguientes funciones:

- ✓**pinMode(pin, mode):** configura el pin como entrada o salida, en donde, pin corresponde al número del pin y mode puede ser INPUT o OUTPUT.
- ✓**digitalRead(pin):** Lee el valor de un pin digital.
- ✓**digitalWrite(pin, value):** Escribe un 0 o un 1 (0 o 5V) en el pin especificado

72

Entradas y salidas analógicas del lenguaje de programación Arduino.

Los pines análogos son entradas análogas que reciben tensiones entre 5V y 0V (v – voltios). Los pines análogos, al contrario de los digitales, no necesitan ser declarados como modo INPUT o OUTPUT.

Dos funciones muy utilizadas en las entradas y salidas analógicas son:

- **analogRead(pin):** lee o captura el valor de entrada del pin análogo especificado
- **analogWrite(pin, value):** escribe un valor analógico (onda PWM) en un pin. Se puede utilizar para encender un LED con distintos brillos o impulsar un motor a distintas velocidades.

73

Funciones de tiempo del lenguaje de programación Arduino.

Son funciones muy utilizadas en distintas aplicaciones en las cuales es necesario trabajar con tiempo. Las más utilizadas son:

- **delay(ms):** esta función pausa el programa durante el tiempo (en milisegundos) especificado como parámetro. (Hay 1000 milisegundos en un segundo).
- **millis():** devuelve el número de milisegundos desde que la placa Arduino comenzó a ejecutar el programa actual.

74

Funciones matemáticas del lenguaje de programación Arduino.

El lenguaje Arduino posee varias funciones matemáticas que nos ayudan al momento de programar la placa. En esta sección se presentan dos de las funciones más utilizadas.

Min(x,y)

- Es una función matemática que calcula el valor menor o mínimo de dos números.

Max(x, y)

- Es una función matemática que calcula el valor mayor o máximo de dos números.

75

Funciones de generación aleatoria del lenguaje de programación Arduino.

Para generar números aleatorios, puede utilizar las funciones de números aleatorios de Arduino. Tenemos dos funciones que se explican a continuación.

- **Randomseed(seed)**

Esta función inicializa el generador de números pseudoaleatorios. La secuencia es siempre la misma, aunque es muy larga y aleatoria

- **Random(max) y Random(min, max)**

Al igual que la anterior la función genera números pseudoaleatorios.

76

Puerto serie del lenguaje de programación Arduino.

Se utiliza para la comunicación entre la placa Arduino y una computadora u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie, se comunica en los pines digitales 0 (RX) y 1 (TX) así como con la computadora a través de USB.

Serial.begin(rate) Abre un puerto serie y establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. La velocidad típica de comunicación es de 9600 pero también puede utilizar otras velocidades.

Serial.println(data) Imprime datos en el puerto serie como texto ASCII legible por las personas. Esta instrucción está seguida de un carácter de retorno de carro y un carácter de nueva línea.

Serial.read(). Lee datos seriales entrantes. La instrucción read () hereda de la clase de utilidad Stream. La función retorna el primer byte de datos seriales entrantes disponible o -1 si no hay datos disponibles.

Serial.available(). Devuelve el número de caracteres disponibles para leer desde el puerto serie. La cantidad de datos se obtiene de los datos que ya llegaron y se almacenaron en el búfer de recepción en serie. available () hereda de la clase de utilidad Stream.

77

Ejemplos de código del lenguaje de programación Arduino

- **Ejemplo 1 -- Salida digital**

Construya el esquema del circuito y escriba el código que le permite encender tres leds a la vez y que luego los apague. Los pines de los leds son: 4, 8 y 10.

- **Ejemplo 2 -- Salida digital II**

Construya el esquema del circuito y escriba el código que le permite encender tres leds en serie, es decir, primero enciende el led 4, luego el 8 y finalmente el 10. Los pines de los leds son: 4, 8 y 10.

- **Ejemplo 3 -- Entrada digital**

Una de las formas más sencillas de entrada es utilizando un simple switch o pulsador que nos permite sólo dos posibles estados: encendido o apagado. Al pulsar el botón se apaga el led.

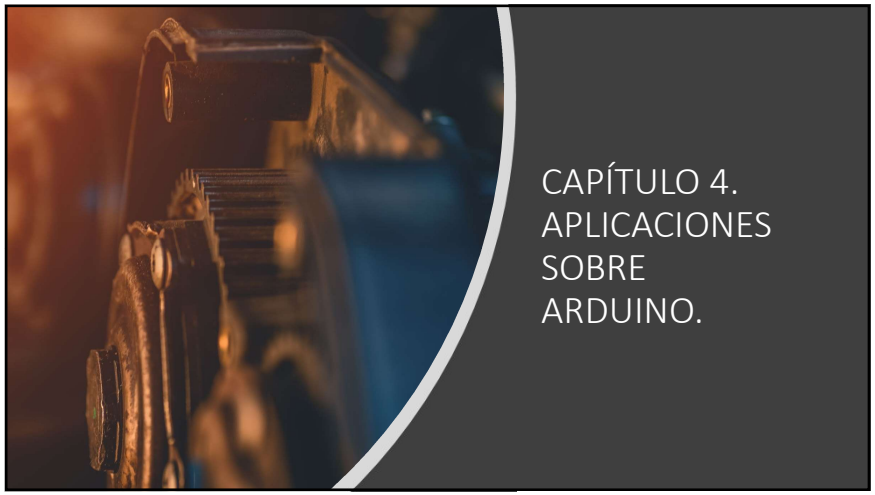
- **Ejemplo 4 -- Salida PWM**

Construya el esquema del circuito y escriba el código para modificar el brillo de un LED utilizando la Modulación de Impulsos en Frecuencia (PWM).

- **Ejemplo 5 -- Entrada a partir de un potenciómetro**

Construya el esquema del circuito y escriba el código para utilizar un potenciómetro para controlar un el tiempo de parpadeo de un LED.

78



CAPÍTULO 4. APLICACIONES SOBRE ARDUINO.

79

CAPÍTULO 4. APLICACIONES SOBRE ARDUINO.

Codificación mínima de la programación con arduino.

Cuando realizamos un proyecto en Arduino existen algunos aspectos que son indispensables explicar para poder realizar de manera apropiada el mismo. En esta sección se detallan cada uno de los requisitos que se deben explicar cuando se desarrolla un proyecto con Arduino.

- **Objetivo:** Cada proyecto que se desarrolle debe poseer, por lo menos, un objetivo que constituye la pieza clave del diseño. El mismo, expresan la meta que se desea alcanzar.
- **Requisitos de Hardware:** Una parte muy importante al momento de realizar un proyecto es enumerar los componentes de hardware que se necesitaran para la elaboración del mismo.
- **Construcción del Circuito:** Constituye una representación de la conexión de los distintos componentes pertenecientes al proyecto.
- **Codificación de la programación con Arduino:** Es el código del programa, el cual debe estar documentado para una mejor comprensión del desarrollo del proyecto.

80

Ejemplos de programas en Arduino

Ejemplo 1-- Encendido de leds utilizando estructura condicional

Escriba un programa utilizando la estructura condicional y el residuo (mòdulo) que le permita encender un led ubicado en el pin número dos cuando el residuo sea par o que encienda un led ubicado en el pin número tres cuando el residuo sea impar.

Ejemplo 2 -- Encendido de cuatro leds utilizando estructura condicional

Escriba un programa utilizando la estructura condicional y el residuo (mòdulo) que le permita encender dos leds ubicados en los pines números 2 y 4 cuando el residuo sea par o que encienda dos leds ubicados en los pines números 3 y 5 cuando el residuo sea impar.

81

Ejemplos de programas en Arduino

Ejemplo 3 -- Encendido de cuatro leds utilizando switch case

Escriba un programa utilizando switch case y el residuo (mòdulo) que le permita encender dos leds ubicados en los pines números 2 y 4 cuando el residuo sea par o que encienda dos leds ubicados en los pines números 3 y 5 cuando el residuo sea impar.

Ejemplo 4 -- Encendido de leds utilizando arreglos y while

Escriba un programa utilizando arreglos y el ciclo while que encienda una secuencia de leds, los apague y luego los vuelva a encender en orden inverso

82

Ejemplos de programas en Arduino

Ejemplo 5 -- Encendido de leds utilizando arreglos y for

Escriba un programa utilizando arreglos y el ciclo while que encienda una secuencia de leds, los apague y luego los vuelva a encender en orden inverso

Ejemplo 6 – Utilización de la bocina

Escriba un programa que toca una pequeña melodía utilizando una bocina o piezo.

Ejemplo 7 – Utilización de fotocelda y la bocina

Escriba un programa que activa la bocina o piezo cuando una fotocelda LDR como sensor de luz.

83

Ejemplos de programas en Arduino

Ejemplo 8 – Utilización de fotocelda y led

Escriba un programa que encienda un led cuando una fotocelda LDR reciba luz.

Ejemplo 9 – Utilización de un servomotor

Escriba un programa que haga girar un servo motor.

Ejemplo 10 – Utilización del sensor ultrasónico

Escriba un programa que utilice un sensor ultrasónico de tal forma que cuando tenga un objeto a una distancia menor o igual a 15 cm se encienda el led.

84