

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE VIÑA DEL MAR – JOSÉ MIGUEL CARRERA

GUÍA TEMÁTICA DE ARDUINO UNO PARA AUTOAPRENDIZAJE

Trabajo de titulación para optar al título
Profesional de Ingeniero de Ejecución en
Control e Instrumentación Industrial.

Alumno:
Andrés Adolfo Fuentes Salazar.

Profesor guía:
Mag. Guelis Montenegro Zamora

RESUMEN

Keywords: ARDUINO UNO, MICROCONTROLADOR, GUÍA, AUTOAPRENDIZAJE.

Arduino UNO es una pequeña placa electrónica que integra todo lo necesario para el uso de un microcontrolador. Es importante recordar que un microcontrolador requiere de un cristal oscilador con sus capacitores, una fuente de alimentación continua y la placa o protoboard (en caso de un prototipo) para poder funcionar. Sin embargo, Arduino ahorra esos obstáculos junto con los problemas de contacto o de ruido, ofreciendo todo implementado y listo para comenzar a programar.

En este trabajo se presenta un curso compuesto por seis unidades. Las primeras cinco entregan conocimientos necesarios para la realización de la sexta unidad que consiste en 10 experiencias prácticas.

El curso está principalmente enfocado en enseñar conocimientos básicos de Arduino, pero también entrega nociones básicas para trabajar con instrumentos y componentes electrónicos, dado que tal información es vital para utilizar de forma correcta los dispositivos.

Además, de toda la información necesaria para la realización del curso, también se le entrega al alumno (en algunas guías específicas) información que le será de gran utilidad en caso de querer profundizar con proyectos electrónicos, o que desee aprender electrónica más avanzada. La idea principal es incentivar que el estudiante quiera continuar aprendiendo y de ningún modo limitar sus ganas de introducirse en el área electrónica y programación.

ÍNDICE

1.	PRESENTACIÓN DE LA GUÍA TEMÁTICA DE ARDUINO	3
1.1.	ORIGEN DE LA PROPUESTA PARA LA GUÍA TEMÁTICA	3
1.2.	OBJETIVOS DE LA GUÍA TEMÁTICA.....	4
1.3.	ELECCIÓN DEL KIT	4
1.4.	CONTENIDO DEL KIT	5
1.4.1.	Sobre la Placa Arduino UNO y su microcontrolador	6
1.5.	CONTENIDOS DE LA GUÍA TEMÁTICA	8
1.5.1.	Unidad 1: Teoremas y Conceptos Básicos Para el Análisis de Circuitos.....	8
1.5.2.	Unidad 2: Componentes Electrónicos y herramientas.....	9
1.5.3.	Unidad 3: Conceptos de microcontroladores.....	10
1.5.4.	Unidad 4: Apuntes para Programar	11
1.5.5.	Unidad 5: Apuntes y Conceptos útiles	12
1.5.6.	Unidad 6: Experiencias de Arduino.....	13
2.	EXPERIENCIA 8: CONTROL DE SERVOMOTOR.....	15
2.1	EXPERIENCIA 8.....	15
2.2	DESARROLLO DE EXPERIENCIA 8	21
2.2.1	Programa para resolver ejercicio de figura 2-4.....	21
2.2.2	Ejercicio opcional intermedio.....	22
2.2.3	Programa para resolver ejercicio de figura 2-5.....	23
3.	EXPERIENCIAS DE ARDUINO (UNIDAD 6).....	26
3.1	UNIDAD 6	26
3.1.1	Experiencia 1: Uso de multímetro y ley de Ohm	26
3.1.2	Experiencia 2: Secuencias con leds	27
3.1.3	Experiencia 3: Instrucción IF	28
3.1.4	Experiencia 4: Bucle FOR.....	29
3.1.5	Experiencia 5: Bucle WHILE.....	30
3.1.6	Experiencia 6: Entradas Analógicas	31
3.1.7	Experiencia 7: PWM	33
3.1.8	Experiencia 8: Control de servomotor	34
3.1.9	Experiencia 9: Comunicación Serial	34
3.1.10	Experiencia 10: Interrupciones	36
	CONCLUSIÓN.....	38
	BIBLIOGRAFÍA	39

ÍNDICE DE FIGURAS

Figura 1-1 Kit de Arduino UNO para estudiantes.....	4
Figura 1-2 Placa referencial de Arduino UNO	6
Figura 1-3 Microcontrolador ATmega328P incorporado en la placa de Arduino UNO	6
Figura 2-1 Imagen Referencial de un Servomotor Desarmado.....	15
Figura 2-2 Función de cada cable del servomotor	16
Figura 2-3 Señal 1ms para posicionar el servo en 0° y señal de 2ms para 180°	17
Figura 2-4 Circuito para controlar un servomotor con Arduino	19
Figura 2-5 Circuito para controlar servomotor con Arduino mediante un potenciómetro	20
Figura 3-1 Circuitos propuestos en la experiencia 1	26
Figura 3-2 Circuito propuesto en la experiencia 2	27
Figura 3-3 Circuito propuesto en la experiencia 3	28
Figura 3-4 Circuito propuesto en la experiencia 4	29
Figura 3-5 Proceso propuesto en la experiencia 5, control de nivel.....	30
Figura 3-6 Circuito 1 experiencia 6.....	31
Figura 3-7 Circuito 2 experiencia 6.....	32
Figura 3-8 Circuito 1 experiencia 7.....	33
Figura 3-9 Circuito 2 experiencia 7.....	33
Figura 3-10 Circuito 1 experiencia 9.....	34
Figura 3-11 Circuito 2 experiencia 9.....	35
Figura 3-12 Circuito 1 experiencia 10.....	36
Figura 3-13 Circuito 2 experiencia 10.....	37

SIGLAS Y SIMBOLOGÍA

A. SIGLAS

BUS	: Significado en ingles de Canal (en informática).
CPU	: Central Processing Unit (Unidad Central de Procesamiento).
DC	: Direct Current (Voltaje o Tensión Continua).
DT	: Duty Cycle (Ciclo de Trabajo)
EEPROM	: Electrically Erasable Programmable Read Only Memory (Memoria de solo lectura programable y borrable eléctricamente).
LCD	: Liquid Crystal Display (Pantalla de Cristal Líquido).
LED	: Light Emitting Diode (Diodo Emisor de Luz).
NTC	:Negative Temperature Coefficient (Coeficiente de Temperatura Negativo)
PC	: Personal Computer (Computador Personal)
PLC	: Programmable Logic Controller (Controlador Lógico Programable)
PTC	: Positive Temperature Coefficient (Coeficiente de Temperatura Positivo).
PYME	: Pequeña y Mediana Empresa
RAM	: Random Access Memory (Memoria de Acceso Aleatorio)
ROM	: Read Only Memory (Memoria de Solo Lectura)
SCADA	: Supervisory Control and Data Acquisition (supervisión, control y adquisición de datos)
SMD	: Surface Mount Technology (Tecnología de Montaje Superficial).
SRAM	: Static Random Access Memory (Memoria Estática de Acceso Aleatorio)
UART	: Universal Asynchronous Receiver-Transmitter (Trasmisor-Receptor Universal Asíncrono).
USM	: Universidad Santa María

B. SIMBOLOGÍA

A	: Amperes (Unidad de Corriente eléctrica).
B	: Byte (Unidad de Información).
°C	: Grados Celcius (Unidad de Temperatura).
F	: Faradios (Unidad de Capacidad Elécétrica).
G	: Giga (Prefijo del sistema internacional para anotar 10^9)
Hz	: Hertz (Unidad de Frecuencia).
K	: Kilo (Prefijo del sistema internacional para anotar 10^3)
M	: Mega (Prefijo del sistema internacional para anotar 10^6)
m	: mili (Prefijo del sistema internacional para anotar 10^{-3})
n	: nano (Prefijo del sistema internacional para anotar 10^{-9})
Ω	: Ohm (Unidad de Resistencia eléctrica).
u	: micro (Prefijo del sistema internacional para anotar 10^{-6})
V	: Volts (Unidad de voltaje o tensión).
W	: Watts (Unidad de Potencia eléctrica).
°	: Grados (Ángulos)

INTRODUCCION

Este trabajo consiste en la presentación de una guía temática de autoaprendizaje para Arduino UNO.

La guía compuesta por seis unidades busca entregar contenidos necesarios para que el alumno pueda aprender a trabajar con el dispositivo Arduino UNO sin conocimientos previos de electrónica, ni programación.

En el primer capítulo se realiza una presentación de la guía temática, el origen de la idea y como se llevó a cabo para hacer posible este trabajo. Se presenta el kit de Arduino con el que se trabajará y todas las unidades con sus objetivos y contenidos.

En el capítulo dos, para mostrar a modo de ejemplo el desarrollo de una guía, se presenta una de las experiencias prácticas de la unidad seis completa y luego se muestran las respuestas. Se explica en qué consiste la experiencia, el objetivo de la misma y la solución totalmente explicada punto por punto junto con la programación que debe ir en el software IDE.

En el capítulo tercero se presenta la última unidad completa que consiste en diez experiencias prácticas. Una por una, se explican los objetivos de cada guía y de cada programa que se pide redactar en el software IDE.

Finalmente se adjuntan los anexos en donde está la guía temática de Arduino UNO para autoaprendizaje, los anexos representan el núcleo de este trabajo constituyendo un total de 51 guías distribuidas en seis unidades.

CAPÍTULO 1: PRESENTACIÓN DE LA GUÍA TEMÁTICA DE ARDUINO UNO

1. PRESENTACIÓN DE LA GUÍA TEMÁTICA DE ARDUINO

Este capítulo está dedicado a presentar la guía temática de una forma general, se explica el origen de la propuesta para el presente trabajo de título y se presentan los materiales que serán utilizados para el desarrollo y aprendizaje. También se explica en qué consiste y los contenidos que serán entregados en cada unidad.

1.1.ORIGEN DE LA PROPUESTA PARA LA GUÍA TEMÁTICA

En conversaciones con la profesora Loreto Marín (jefa de la carrera de Ingeniería de Ejecución en Control e instrumentación Industrial), surgió la propuesta para realizar un trabajo de título enfocado en una guía temática de Arduino. El origen viene en base a la propuesta del profesor Rimsky Espíndola de la USM casa central.

Según los antecedentes del profesor Rimsky:

“Entre los años 1998 y 2000 en circunstancia de estar realizando estudios de postgrado en España, se logró conocer con bastante detalle varias instituciones educativas tanto universitarias como de formación técnica, así como empresas productivas PYMES en España y Francia.

Entre lo observado en esa época a nivel educativo ya destacaba una intensiva formación técnica aplicada, apoyada en paneles físicos y sobretodo en software de simulación.

En tanto en el ámbito industrial del ámbito PYME, era habitual encontrar un uso intensivo de soluciones basadas en automatismos con lógica cableada y un uso incipiente pero de sostenido crecimiento de soluciones basadas en lógica programada con PLC y PC, incluso integrados con soluciones de instrumentación virtual (LabView) y plataformas SCADA.

Han pasado más de 20 años y en nuestro país pareciera haber un estancamiento en el empleo sistemático de soluciones de automatización, en contraste con un mundo en el que la automatización ha pasado a constituirse en la herramienta fundamental para la competitividad y hasta supervivencia de las empresas.

En este escenario se hace necesario dar relevancia y presencia al tema de la automatización, estimulando su conocimiento básico temprano a nuevas generaciones y ayudando a posicionar su potencial en la comunidad tanto interna como externa.”

1.2.OBJETIVOS DE LA GUÍA TEMÁTICA

Promover la difusión de la automatización en la comunidad interna y externa de la USM mediante potenciamiento de los recursos de aprendizaje disponibles en Biblioteca USM.

1.3.ELECCIÓN DEL KIT

Luego de haber entablado conversaciones y reuniones con el profesor Rimsky y la profesora Loreto, se llegó al acuerdo que la guía temática se haría en base al kit de estudiantes de Arduino UNO mostrado en la figura 1-1.



Fuente: <https://mcielectronics.cl/shop/product/arduino-kit-de-estudiantes-arduino-28326/>

Figura 1-1 Kit de Arduino UNO para estudiantes

La elección de este kit se debe a que es el más básico para aprender conceptos de programación y electrónica, además no requiere experiencia previa y se puede enseñar a estudiantes inclusive desde los 11 años.

1.4.CONTENIDO DEL KIT

A continuación se tiene el listado de componentes, dispositivos y piezas que contiene el kit de estudiantes de Arduino UNO. Este listado viene directamente desde la página que comercializa este kit (<https://mcielectronics.cl/shop/product/arduino-kit-de-estudiantes-arduino-28326/>).

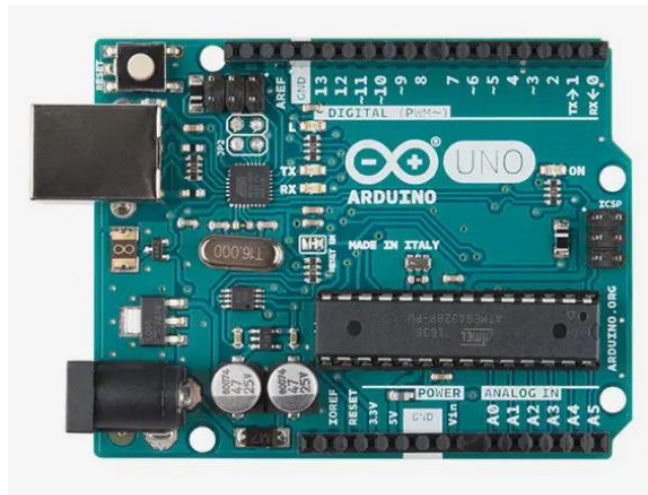
- 1x Placa Arduino UNO R3
- 1x Cable USB
- 1x Multímetro
- 1x Conector para batería 9v DC
- 1x Batería 9v DC
- 20x LED de 5mm (colores varios)
- 5x Resistencia 560Ω
- 5x Resistencia 220Ω
- 5x Resistencia 1KΩ
- 5x Resistencia 10KΩ
- 1x Protoboard de 400 puntos
- 1x Servo Motor
- 2x Potenciómetros 10KΩ
- 2x Potenciómetros Knob
- 2x Condensadores 100uF
- 5x Botones pulsadores
- 1x Fototransistor
- 2x Resistencias 4.7KΩ
- 1x Cable jumper (color negro)
- 1x Cable jumper (color rojo)
- 1x Cable jumper hembra(color negro)
- 1x Cable jumper hembra(color rojo)
- 3x Tornillos y tuercas

1.4.1. Sobre la Placa Arduino UNO y su microcontrolador

Para comenzar se debe explicar ¿qué es Arduino?, en palabras de la misma página de Arduino (<https://arduino.cl/que-es-arduino/>) se ha encontrado la siguiente definición:

“Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables dupont)”

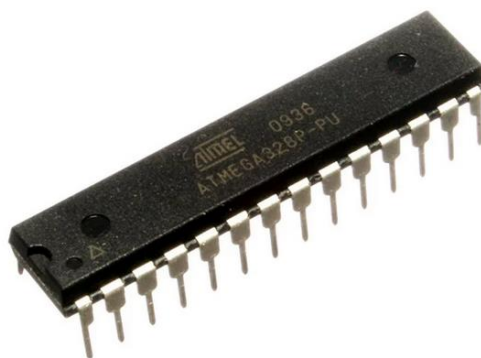
Sabiendo que en la plataforma existen muchas placas de desarrollo y kits, se da la elección del kit con la placa Arduino UNO de la figura 1-2.



Fuente: <https://arduino.cl/producto/arduino-uno/>

Figura 1-2 Placa referencial de Arduino UNO

El kit Arduino UNO consta de una placa de desarrollo que incorpora el microcontrolador “ATmega328P” de la figura 1-3.



Fuente: <https://afel.cl/producto/microcontrolador-atmega328p-pu-28-pines-20-mhz-32kb/>

Figura 1-3 Microcontrolador ATmega328P incorporado en la placa de Arduino UNO

1.4.1.1. Sobre el microcontrolador ATmega328P

El microcontrolador ATmega328P posee las siguientes características:

- Fabricante: Atmel
- Núcleo: AVR (tipo de arquitectura)
- Ancho de bus de datos: 8 bits
- Frecuencia de reloj máxima: 20 MHz
- Tamaño de memoria del programa: 32 KB
- Tamaño de RAM de datos: 1 KB
- Paquete / Cubierta: PDIP-28
- Conversión analógico-digital en chip: Si
- Tamaño de ROM de datos: 1KB
- Tipo de interfaz: 2-Wire, SPI, UART
- Temperatura de trabajo máxima: 85 C
- Temperatura de trabajo mínima: – 40 C
- Número de entradas / salidas programables: 23
- Número de temporizadores: 3
- Voltaje de alimentación Máx: 5.5v DC
- Voltaje de alimentación Mín: 1.8v DC

1.5.CONTENIDOS DE LA GUÍA TEMÁTICA

La guía temática se divide en seis unidades, cinco de ellas son teóricas y una práctica. Cada una con distintos objetivos, las cinco primeras buscan entregar conocimientos y herramientas para el desarrollo de las experiencias en la unidad seis. También se entregan conocimientos para que el alumno pueda realizar más experiencias fuera de las propuestas o proyectos más avanzados si así lo desea.

1.5.1. Unidad 1: Teoremas y Conceptos Básicos Para el Análisis de Circuitos

Objetivos:

- Enseñar teoremas y conceptos básicos para el análisis de circuitos de modo que el estudiante poco a poco se familiarice con la simbología y la matemática requerida para trabajar con Arduino UNO u otros dispositivos y componentes electrónicos.

Para comenzar el curso, se comienza a enseñar contenidos asumiendo que el estudiante no tiene ningún conocimiento previo, por lo tanto, el punto de inicio es desde el átomo y los electrones, para luego avanzar hacia la corriente y el voltaje.

Además de todos los teoremas que se enseñan, también se presentan en algunas guías, ejercicios con circuitos enteramente resistivos para que el alumno practique las ecuaciones de ley de Ohm, ley de Watt, LKV y LKI. Esta unidad busca entregar de forma muy básica los conocimientos para trabajar con circuitos de corriente continua.

1.5.1.1. Listado de Guías Unidad 1:

- El voltaje y la corriente.
- Ley de Ohm y código de colores
- Ley de Watt.
- Esquemáticos y simbología
- Circuitos serie, paralelo y mixto.
- Ley de Voltaje y corrientes de Kirchhoff.
- Ecuación para el cálculo de la resistencia de un LED

1.5.2. Unidad 2: Componentes Electrónicos y herramientas

Objetivos:

- Enseñar el modo de empleo del protoboard y el multitéster digital para que el alumno aprenda a medir corriente, voltaje, resistencia y demás funciones incluidas.
- Que el alumno aprenda el funcionamiento básico de los componentes electrónicos más utilizados (resistencias, diodo rectificador, transistor, bobinas, condensadores, circuitos integrados, reguladores de voltaje y compuertas lógicas).
- Entregar nociones para que el alumno sepa identificar componentes electrónicos en distintas presentaciones.

La unidad 2 está enfocada en entregar conocimientos de electrónica básica, para finalmente comenzar a enseñar electrónica digital (compuertas lógicas), de este modo poco a poco se empieza a llevar al estudiante hacia el área de Arduino y microcontroladores.

Lo más destacable de la unidad se encuentra en la guía 8 en donde se enseña que un microcontrolador permite convertir un circuito de control alambrado en un circuito de control programado.

1.5.2.1. Listado de Guías Unidad 2

- El multitéster digital.
- El protoboard.
- La resistencia y sus variantes
- El diodo rectificador
- El LED
- El transistor
- La bobina y el condensador
- El circuito integrado
- Reguladores de voltaje
- Compuertas lógicas

1.5.3. Unidad 3: Conceptos de microcontroladores

Objetivos:

- Entregar los contenidos necesarios de electrónica digital y microcontroladores para comprender el funcionamiento de ARDUINO UNO y su microcontrolador ATmega328P.
- Enseñar que a lo largo de los años se han desarrollado diferentes lenguajes de programación y junto con ello, los fabricantes han personalizado los adelantos en las arquitecturas de microcontroladores.

Como visión general de la unidad, se explica que los microcontroladores son dispositivos electrónicos digitales y sólo trabajan con 0 y 1 (binario).

Con un enfoque en entregar el conocimiento básico y también a la historia de los microcontroladores, la unidad 3 se concentra en explicar cómo estos dispositivos revolucionaron el área de la automatización.

Se explica que a pesar de que los microcontroladores solo entienden código máquina, con el tiempo se han desarrollado lenguajes que han hecho más amigable la programación.

En caso de que el alumno quiera trabajar con un microcontrolador diferente al que viene en Arduino UNO, se entrega toda la información acerca de los componentes necesarios para hacerlo funcionar fuera de una placa de desarrollo.

Finalmente se enseña que para programar ARDUINO UNO se utilizará lenguaje C++.

1.5.3.1. Listado de Guías Unidad 3

- Sistemas de numeración.
- Lógicas TTL y CMOS.
- Electrónica Análoga vs electrónica digital.
- El microcontrolador por dentro.
- Componentes necesarios externos para el microcontrolador.
- Lenguajes de programación.

1.5.4. Unidad 4: Apuntes para Programar

Objetivos:

- Enseñar el entorno de programación en el software IDE.
- Explicar las reglas de sintaxis del lenguaje C++.
- Profundizar en los conocimientos del funcionamiento interno de un microcontrolador explicando cómo operan las diferentes memorias internas.
- Enseñar que existen diferentes arquitecturas de microcontroladores.

El enfoque de la unidad 4 es enseñar lo más básico de la programación en C++.

Las primeras cuatro guías presentan un enfoque explicativo del lenguaje C++, enseñando la instalación del software IDE, sintaxis y orden de la programación en la guía cuatro (estructura de un programa). Se incluye la información acerca de las memorias del microcontrolador y se explica el rol que cumple cada una en la guía tres (tipos de memoria).

Las seis guías restantes entregan contenido mucho más específico acerca de cada una de las “secciones” de un programa, relacionando el contenido de programación con el hardware de Arduino. Como ejemplo de esto en la guía ocho se presentan los contenidos para trabajar con entradas analógicas y salidas PWM en el software IDE, junto con ello se indica cómo identificar los pines que corresponden a las entradas y salidas mencionadas.

1.5.4.1. Listado de Guías unidad 4

- Instalación del software IDE
- Sintaxis
- Tipos de memoria
- Estructura de un programa
- Configuración de pines (Entradas y salidas)
- Funciones
- Entradas y salidas analógicas (PWM)
- Comunicación serial
- Uso de librerías

1.5.5. Unidad 5: Apuntes y Conceptos útiles

Objetivos:

- Enseñar contenidos basados en experiencias laborales y universitarias.
- Entregar contenidos extras que le den un valor agregado al curso.
- Enseñar conocimientos para que el estudiante pueda realizar proyectos más avanzados fuera de los enseñados en este curso.

Como una forma de entregar conocimientos extras, la unidad 5 está basada en experiencias personales, de este modo se busca enseñar pequeños detalles muy importantes que podrían hacer la diferencia en alguna experiencia futura.

La unidad tiene un enfoque general, es decir, entrega algunos conocimientos básicos y también otros avanzados, pero no hace distinción en uno u otro.

Como contenidos más destacables de la unidad 5 se tiene los siguientes:

Guía 1: Mucho se habla durante la etapa de estudio y también en el ambiente laboral acerca del cortocircuito, es por eso que se dedica esta guía a enseñar específicamente el por qué ocurre y por qué se produce esto.

Guía 3: Durante las experiencias laborales, se logró observar que uno de los problemas más comunes que aparecen en todo tipo de proyectos es el ruido, es por ello que se ha dedicado esta guía para enseñar específicamente ese tipo de inconvenientes.

Guía 7: A pesar de que este curso está enfocado en Arduino UNO, también se enseña el modo de empleo para cualquier microcontrolador, esta guía enseña algunas formas de aislar las salidas y se basa en experiencias universitarias en la que fueron quemados microcontroladores del tipo PIC16F84A.

1.5.5.1. Listado de Guías unidad 5

- Que es un cortocircuito
- Efecto de carga
- Ruido
- Voltaje Average
- Diagrama en bloques
- Diagrama de flujo
- Aislar salidas

- Teorema de Kenelly
- Teorema de Thevenin

1.5.6. Unidad 6: Experiencias de Arduino

Objetivos:

- Aplicar conocimientos de las unidades anteriores de forma práctica.
- Familiarizarse con instrumentos y componentes electrónicos.
- Programar en lenguaje C++.
- Aprender mediante el lenguaje C++ que existen diferentes soluciones para un mismo problema.
- Aprender a leer datos con entradas digitales y analógicas.
- Utilizar salidas digitales y del tipo PWM.
- Aprender acerca del funcionamiento del servomotor y como controlarlo.
- Aprender qué son las interrupciones en los microcontroladores, cómo activarlas y programarlas en Arduino UNO.

La última unidad de este curso se enfoca totalmente en experiencias prácticas, aunque también se entregan conocimientos teóricos en algunas guías, dado que para desarrollarlas se requiere una introducción previa y saber algunos comandos específicos.

1.5.6.1. Listado de Experiencias Unidad 6

- Uso de multímetro y ley de Ohm
- Experiencia con LEDs.
- Instrucción IF
- Bucle FOR
- Bucle WHILE
- Entradas Analógicas
- PWM
- Experiencia con Servomotor
- Comunicación Serial
- Interrupciones

CAPÍTULO 2: EXPERIENCIA 8: CONTROL DE SERVOMOTOR

2. EXPERIENCIA 8: CONTROL DE SERVOMOTOR

Para explicar a modo de ejemplo cómo pueden ser desarrolladas las experiencias prácticas de la guía, se presenta la experiencia práctica número 8 del capítulo 6.

Esta elección se da debido a que se ha considerado que la dificultad es mayor a las otras.

2.1 EXPERIENCIA 8

Objetivos:

- Aprender a controlar un servomotor.
- Introducción sobre el servomotor

El servomotor es un motor que permite tener un movimiento muy preciso y con un gran torque. Internamente está compuesto por un motor de corriente continua, un circuito de control que entrega 3 cables y una caja de engranajes que reducirá la velocidad y aumentará el torque. A través de los cables se puede conectar el terminal positivo, tierra y la señal de información para el movimiento del motor. Todo este sistema (Figura 2-1) estará ensamblado dentro de una caja plástica, lo que es visible desde el exterior son el eje del motor y los 3 cables para alimentación y control.

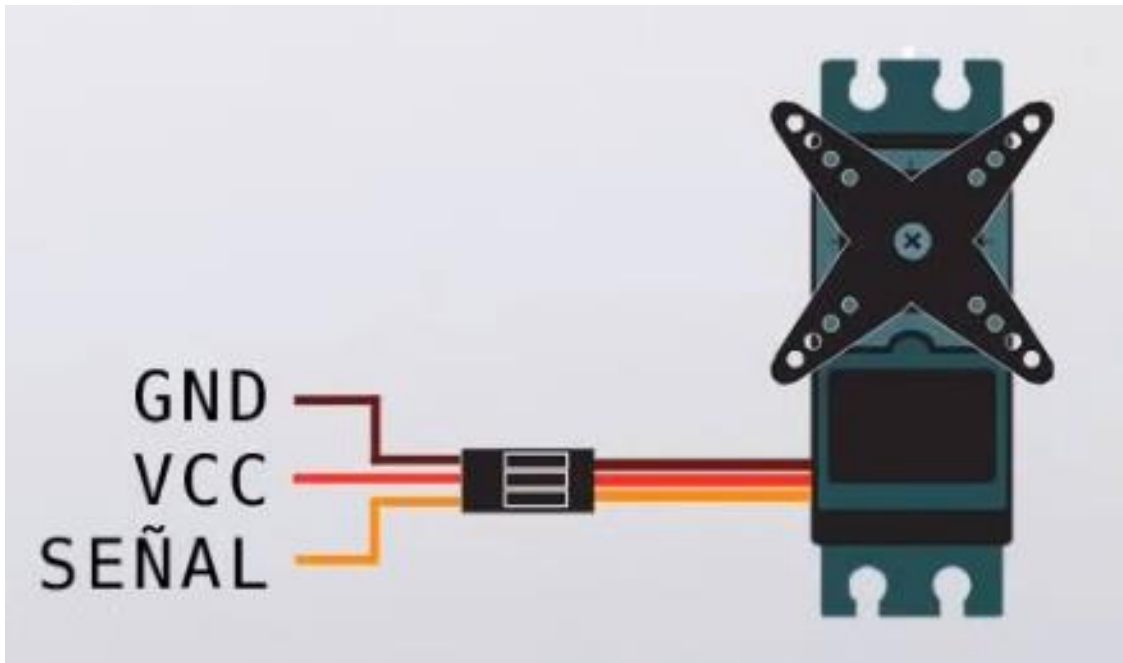


Fuente: <https://fidiasrodriguez.com/que-es-un-servomotor/>

Figura 2-1 Imagen Referencial de un Servomotor Desarmado

- Conexión del servomotor

Como se ha mencionado, para conectarlo se tienen 3 cables (Figura 2-2). Generalmente el cable negro es para conectar la tierra y el cable rojo para la alimentación.



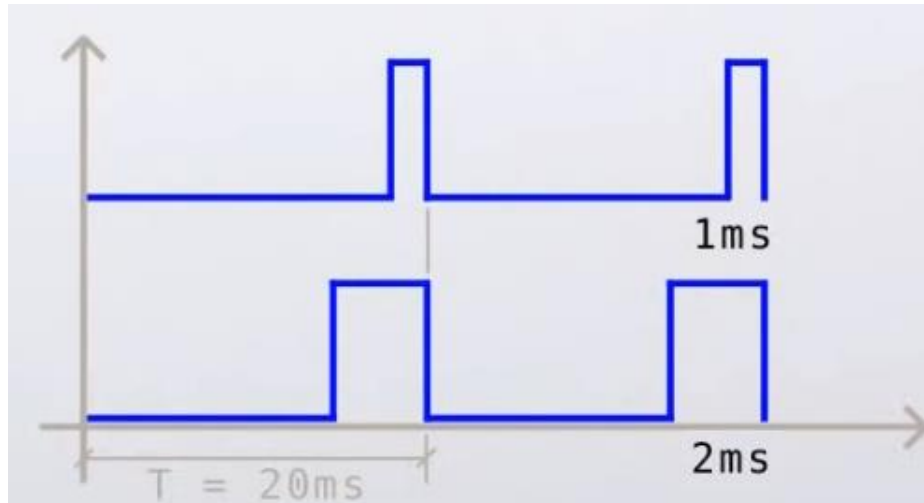
Fuente: <https://descubrearduino.com/servomotores/>

Figura 2-2 Función de cada cable del servomotor

Existen más combinaciones de colores para los cables del servomotor, en caso de que existan dudas sobre cómo conectarlo, se deberá consultar en la página del fabricante con el modelo del servo.

Se debe tener en consideración que las imágenes de la figura 2-1 y la figura 2-2 son solo referenciales, de modo que el servo podría tener otra presentación o tener otro brazo actuador distinto conectado a su eje.

- Señal de control para el servomotor



Fuente: <https://saraih1008.wixsite.com/roboticaeducativa/copia-de-4-2-led-rgb>

Figura 2-3 Señal 1ms para posicionar el servo en 0° y señal de 2ms para 180°

Las señales de la figura 2-3 son para que el servo se mueva a sus límites mínimos y máximos, para ajustarlo en alguna posición intermedia habrá que variar el ancho del pulso entre 1 y 2 mili segundos. Estas señales son del tipo PWM.

- Pasos necesarios en software IDE para operar el servomotor

Paso 1: Importar la librería dedicada a servomotores, para ello se debe escribir:

```
#include <Servo.h>
```

Paso 2: Dar nombre al motor servo para poder referenciarlo más adelante en el programa.

```
Servo nombre_del_servo;
```

Este paso crea un objeto del tipo servo y lo definirá con el nombre que se escriba en el parámetro "nombre_del_servo".

Paso 3: Dentro de la función `void setup()` se utilizará la siguiente función:

```
nombre_del_servo.attach(pin_servo, mínimo, máximo);
```

El nombre del servo será el mismo que haya sido asignado en el paso 2.

El parámetro `pin_servo` se refiere al pin del arduino en que está conectada la señal de control del servo.

Los parámetros mínimo y máximo se refieren a la duración de los pulsos de la señal de control para posicionar el servo en 0° y 180°, respectivamente. En estos parámetros se debe escribir 1000 para el mínimo y 2000 para el máximo ya que esta función trabaja con microsegundos.

Paso 4: Dentro de la función `void loop()` se utilizará la siguiente instrucción para mover el servo a la posición deseada.

`Nombre_del_servo.write(posición);`

En el parámetro posición se debe escribir el ángulo al que se desea mover el servo, debe ser un valor entre 0 y 180.

Paso 5 (adicional): Función necesaria para convertir valores

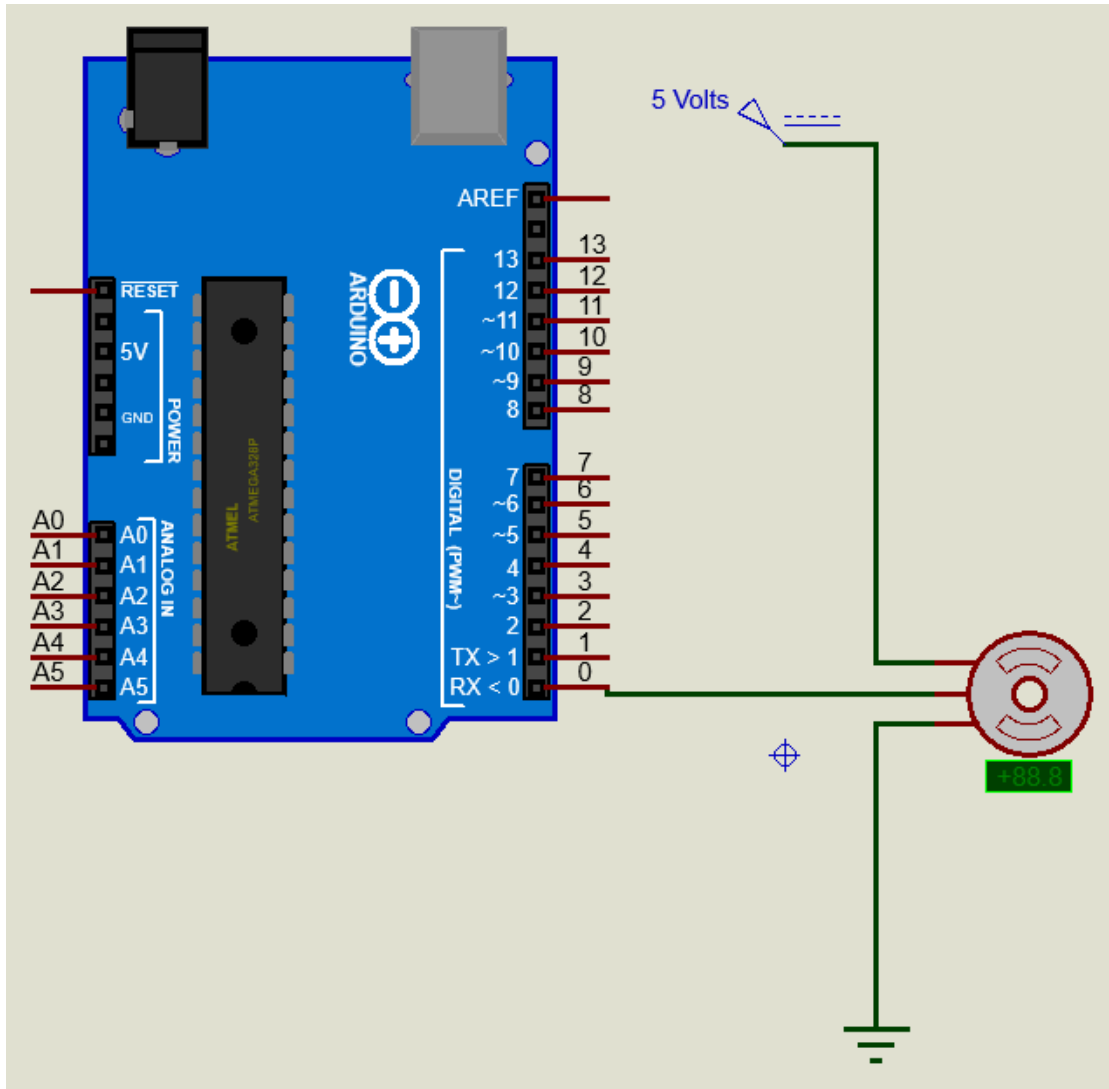
`map(variable, valor_mínimo1, valor_máximo1, valor_mínimo2, valor_máximo2);`

Esta función sirve para escalar valores de una variable que tengan mínimos y máximos a otra nueva escala. Es muy utilizada cuando se trabaja con valores analógicos y convertirlos.

Los valores mínimo 1 y máximo 1 corresponden a la escala en que trabaja la variable.

Los valores mínimo 2 y máximo 2 corresponden a la nueva escala a la que será convertido el valor de la variable.

- Teniendo el circuito de la figura 2-4



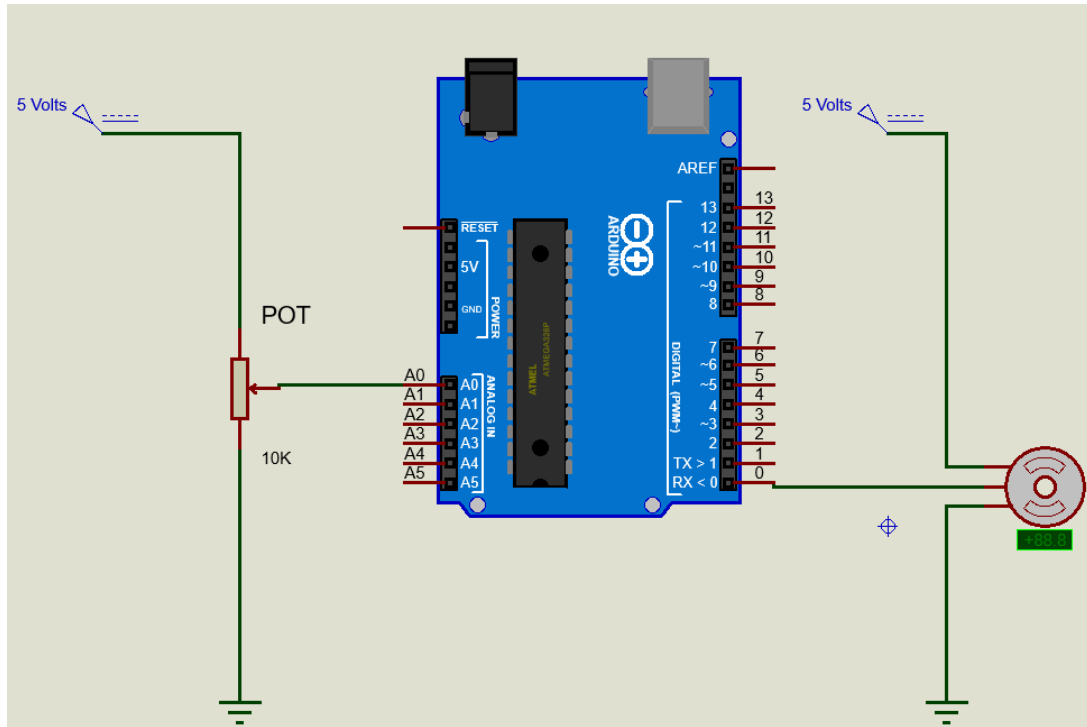
Fuente: Elaboración propia

Figura 2-4 Circuito para controlar un servomotor con Arduino

1.- Implementar el circuito y conectar el Arduino a tu PC.

2.- Diseñar un programa que mueva el servo a 0°, 90° y 180°. El servo deberá cambiar de posición cada 3 segundos, la secuencia será: 0° - 90° - 180° - 90°. El Arduino deberá repetir la secuencia periódicamente.

- Teniendo el circuito de la figura 2-5



Fuente: Elaboración propia

Figura 2-5 Circuito para controlar servomotor con Arduino mediante un potenciómetro

- 1.- Implementar el circuito y conectar el Arduino a tu PC.
- 2.- Diseñar un programa en que el servomotor se mueva según la variación de voltaje generada por el potenciómetro. Los límites mínimos y máximos serán $0v \leftrightarrow 0^\circ$ y $5v \leftrightarrow 180^\circ$ respectivamente.

2.2 DESARROLLO DE EXPERIENCIA 8

En el primer circuito de la figura 2-4 se pide implementar el circuito y luego escribir el programa para controlar el servomotor, a continuación se presenta el programa que resuelve el ejercicio propuesto.

2.2.1 Programa para resolver ejercicio de figura 2-4

`include <Servo.h>` ;incluimos librería especial para servomotores

`Servo servomotor;` /*se crea un objeto del tipo servo y se le nombra servomotor */

`int pinservomotor = 0;` /* se le asigna este número en donde se conecta el cable de señal del servomotor */

`int pmin = 1000;` /* se asigna 1000 para que el pulso mínimo sea de 1000 microsegundos equivalente a 0 grados */

`int pmax = 2000;` /* se asigna 2000 para que el pulso máximo sea de 2000 microsegundos equivalente a 180 grados */

`void setup ()`{

`servomotor.attach(pinservomotor,pmin,pmax);`

/* Esta función sirve para inicializar el servo, el primer parámetro es el pin en donde se encuentra conectado el servomotor, luego se ponen los valores de pulso mínimo y pulso máximo, tales valores requieren estar indicados en microsegundos, por este motivo es que hemos se han configurado las variables pmin y pmax con 1000 y 2000 respectivamente */

}

`void loop()`{

`servomotor.write(0);`

/*la función write incluida en la librería servo, envía el ángulo al cual se debe posicionar el servomotor, y se encarga de generar la señal de pulso con la duración y frecuencia adecuadas, por lo tanto posiciona el servomotor en 0 grados */

`delay(3000);` se esperan los 3 segundos indicados en la guía

`servomotor.write(90);` se posiciona el servomotor en 90 grados

`delay(3000);`

`servomotor.write(180);` se posiciona el servomotor en 180 grados

`delay(3000);`

`servomotor.write(90);` se posiciona el servomotor en 90 grados

`delay(3000);`

}

Una vez conectado el circuito y cargado el programa, se debe poder visualizar el movimiento del servo a las posiciones de 0, 90, 180 y 90 grados de forma periódica.

2.2.2 Ejercicio opcional intermedio

Antes de comenzar a redactar el segundo programa, se debe tener claro que el ejercicio de la figura 2-4 tiene dos objetivos, el primero es verificar el correcto funcionamiento del servomotor y el segundo es para realizar un ajuste fino para el siguiente programa.

Como objetivo opcional para el siguiente problema se puede buscar la totalidad del rango de movimiento en que opera el servomotor.

Los pulsos mínimo y máximo de 1000 y 2000 microsegundos respectivamente son solamente valores aproximados, por lo que con estos valores, no es posible obtener el máximo rango de movimiento para el servomotor, es por eso que se tendrá que hacer un proceso previo de prueba y error, es decir, se debe ir modificando los valores poco a poco para obtener los microsegundos en que se logre utilizar la totalidad del rango del servomotor.

- Ejemplo:

Como ya se tiene el programa escrito, en la declaración de variables se modificará poco a poco los valores y cargando el programa al Arduino, es decir, se reducirá el valor de pulso mínimo (pmin) e irá aumentando el valor del pulso máximo (pmax) de modo que se logre encontrar los valores en que el servo llegue a los límites mínimo y máximo respectivamente.

```
int pmin = 900;
```

```
int pmax = 2100;
```

2.2.3 Programa para resolver ejercicio de figura 2-5

Habiendo realizado el ejercicio de la figura 2-4, ahora se debe modificar el circuito para agregar el potenciómetro, de modo que mediante él se pueda controlar el servomotor.

Teniendo totalmente conectado el circuito de la figura 2-5 se presenta la solución al ejercicio a continuación.

- Programa:

```
include <Servo.h> ;se incluye la librería especial para servomotores
```

```
Servo servomotor; /* se crea un objeto del tipo servo y se le nombra servomotor */
```

```
int pinservomotor = 2; /* se le asigna este número en donde se conecta el cable de señal del servomotor */
```

```
int pmin = 1000; /* se asigna 1000 para que el pulso mínimo sea de 1000 microsegundos equivalente a 0 grados */
```

```
int pmax = 2000; /* se asigna 2000 para que el pulso máximo sea de 2000 microsegundos equivalente a 180 grados */
```

```
int valorpot; /* Esta variable será utilizada para guardar el voltaje del cursor del potenciómetro */
```

```
int angulo; /* Aquí se guardará el valor del voltaje del potenciómetro una vez que se haya
convertido desde una escala entre 0 y 1023 a otra de 0 y 180 grados */
```

```
int pinpote = 0; /* se le asigna este número en donde estará conectado el cursor del
potenciómetro */
```

```
void setup (){
    servomotor.attach(pinservomotor,pmin,pmax); /* se Inicializa el servomotor igual que en el
primer ejercicio */
}
```

```
void loop(){
    valorpot=analogread(pinpote); /* se lee el valor del voltaje en el cursor del potenciómetro */

    angulo=map(valorpot,0,1023,0,180); /* se convierte la lectura analógica que se acaba de
realizar a un valor entre 0 y 180 grados y luego se guarda en la variable ángulo */

    servomotor.write(angulo); /* se envía el ángulo al que se desea mover el servomotor */

    delay(30); /* se le damos un retardo que puede ser entre 20 y 50 milisegundos dependiendo
de la carga que tenga el servomotor para que alcance a llegar al ángulo */
}
```

Con el circuito totalmente conectado y el programa cargado en el Arduino, al mover el cursor del potenciómetro también se moverá el servomotor replicando el movimiento.

CAPÍTULO 3: EXPERIENCIAS DE ARDUINO (UNIDAD 6)

3. EXPERIENCIAS DE ARDUINO (UNIDAD 6)

En este capítulo se presentan las experiencias prácticas que componen la unidad 6 de este curso, cada una tiene diferentes objetivos de modo que sean aplicados la mayor cantidad de contenidos posibles.

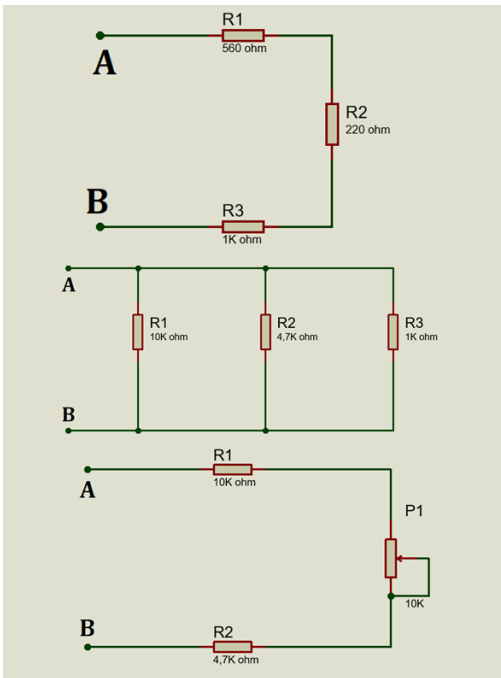
3.1 UNIDAD 6

Esta unidad contiene diez experiencias prácticas enfocadas a la programación de lenguaje C++ mediante el software IDE de Arduino.

Como en todo curso, se comienza con experiencias básicas para que luego el estudiante se enfrente a problemas cada vez más complejos.

3.1.1 Experiencia 1: Uso de multímetro y ley de Ohm

La experiencia presenta tres circuitos mostrados en la figura 3-1, a través de ellos se busca que el estudiante se familiarice con los instrumentos y componentes electrónicos. En el primer circuito de la guía se pide que el estudiante mida circuitos resistivos con el multímetro en la función óhmetro. Posteriormente se presenta un circuito paralelo con resistencias y un circuito serie también compuesto por resistencias pero con un potenciómetro.



Fuente: Elaboración propia

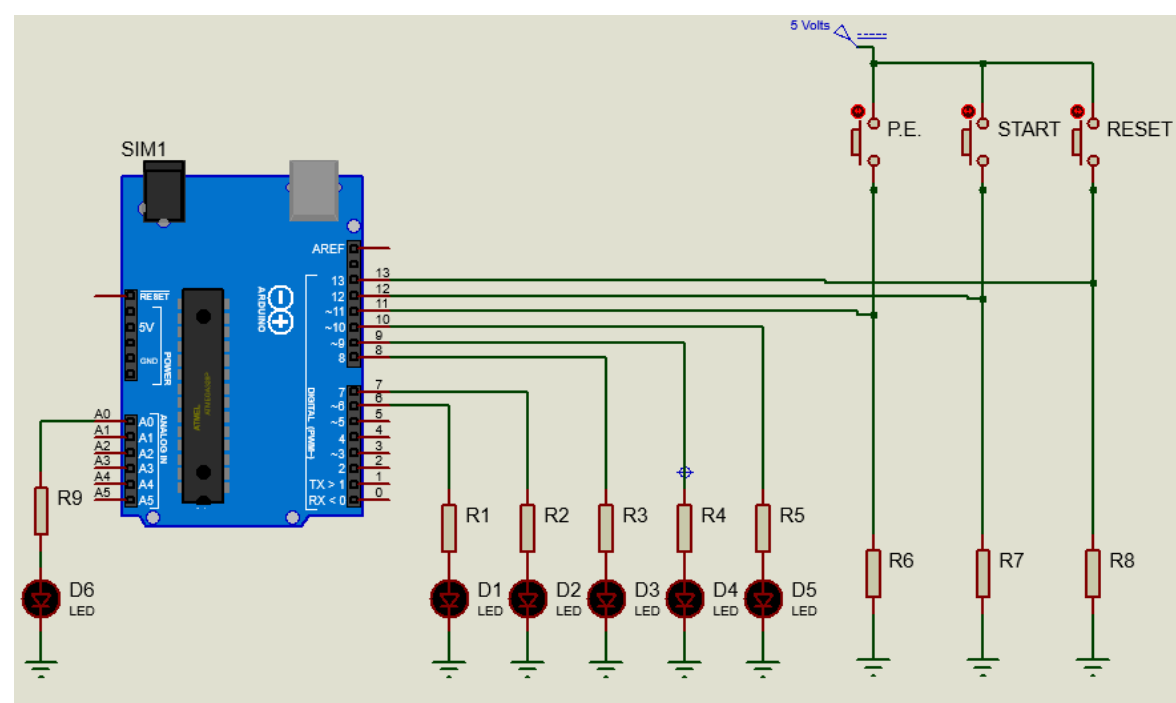
Figura 3-1 Circuitos propuestos en la experiencia 1

3.1.3 Experiencia 3: Instrucción IF

La tercera experiencia tiene como objetivo que el estudiante aprenda a utilizar las instrucciones if, else if y else, para ello se presenta el circuito de la figura 3-3.

La guía cuenta con una introducción en donde se enseña a utilizar las instrucciones señaladas y para que se logre comprender mejor se presentan ejemplos de cada caso.

Esta experiencia busca que el alumno logre hacer que el Arduino tome decisiones.



Fuente: Elaboración propia

Figura 3-3 Circuito propuesto en la experiencia 3

El circuito de la figura 3-3 representa un sistema que controla un proceso. Se explica la equivalencia de cada uno de los componentes conectados al Arduino.

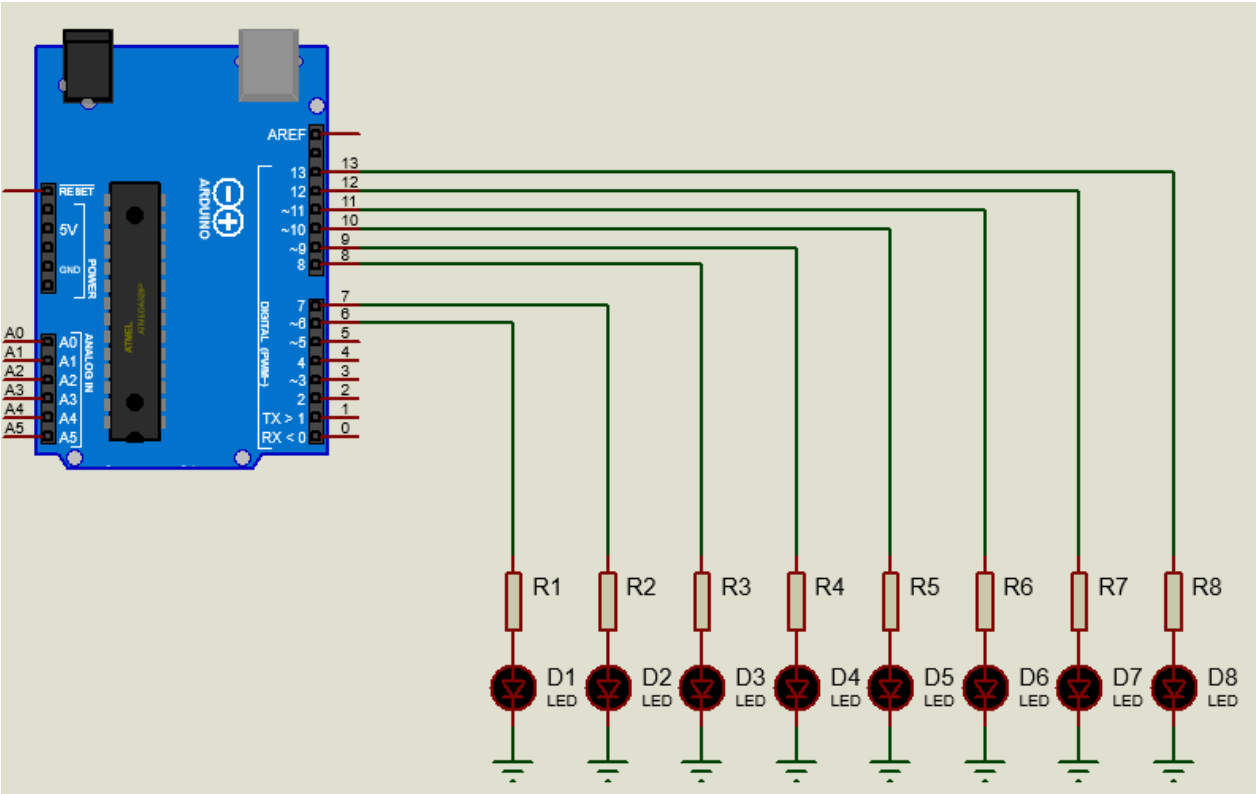
La experiencia contiene instrucciones y reglas de funcionamiento para el circuito, de modo que el estudiante deba simular e imaginar que está trabajando en un entorno industrial.

Antes de pedir la escritura del programa que solucione el problema, se pide diseñar un diagrama de flujo (tema enseñado en unidad 5), esto es para que resulte más sencilla y ordenada la estructura del código.

3.1.4 Experiencia 4: Bucle FOR

El objetivo de la experiencia cuatro es que el alumno aprenda y domine el uso de la instrucción for, se tomó la decisión de dedicar una guía exclusivamente a este bucle debido a que en base a experiencias personales, muchos estudiantes no sabían cómo utilizarla.

El circuito de la figura 3-4 es el propuesto para esta experiencia.



Fuente: Elaboración propia

Figura 3-4 Circuito propuesto en la experiencia 4

La guía consta de una introducción en donde se explican los tres parámetros para el funcionamiento de la instrucción for, también se incluyen las reglas de sintaxis y ejemplos de programas.

Como todas las guías, primero se pide calcular las resistencias conectadas a cada led, luego se pide configurar los pines utilizando la función for, esto para que el alumno aprenda una forma de ahorrar líneas de código. En el ejemplo presentado se enseña a configurar los pines.

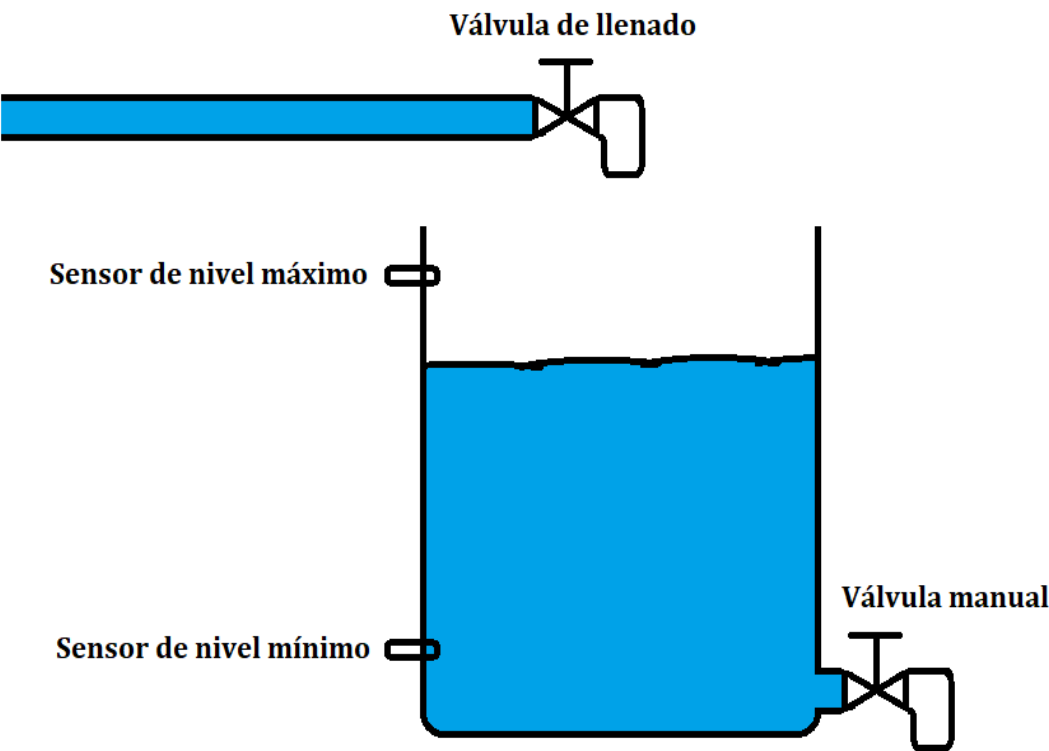
En el punto tres se pide que el estudiante escriba un programa que cree un efecto visual de desplazamiento de los leds hacia la derecha para utilizar la función for con un incremento y luego en el punto cuatro se pide lo contrario, es decir, que se vea un efecto visual de desplazamiento de los leds hacia la izquierda para utilizar decremento.

Finalmente en el punto cinco se piden unir los programas de los puntos tres y cuatro para crear un efecto de metrónomo.

3.1.5 Experiencia 5: Bucle WHILE

La instrucción while es una de las más utilizadas cuando se programa Arduino, sobre todo cuando se aplica con algún tipo de sensor. El objetivo de esta experiencia es enseñar a utilizar esta instrucción.

Esta experiencia es diferente de las demás, ya que no propone ningún circuito, por el contrario, se presenta el proceso de la figura 3-5 y se pide al estudiante que diseñe un circuito para que posteriormente escriba un programa simulando el control del nivel de agua en el estanque.



Fuente: Elaboración propia

Figura 3-5 Proceso propuesto en la experiencia 5, control de nivel

La guía comienza con una introducción en donde se explica el funcionamiento de la instrucción while y que (a diferencia del caso anterior con la instrucción for) solamente requiere el parámetro de la condición para funcionar.

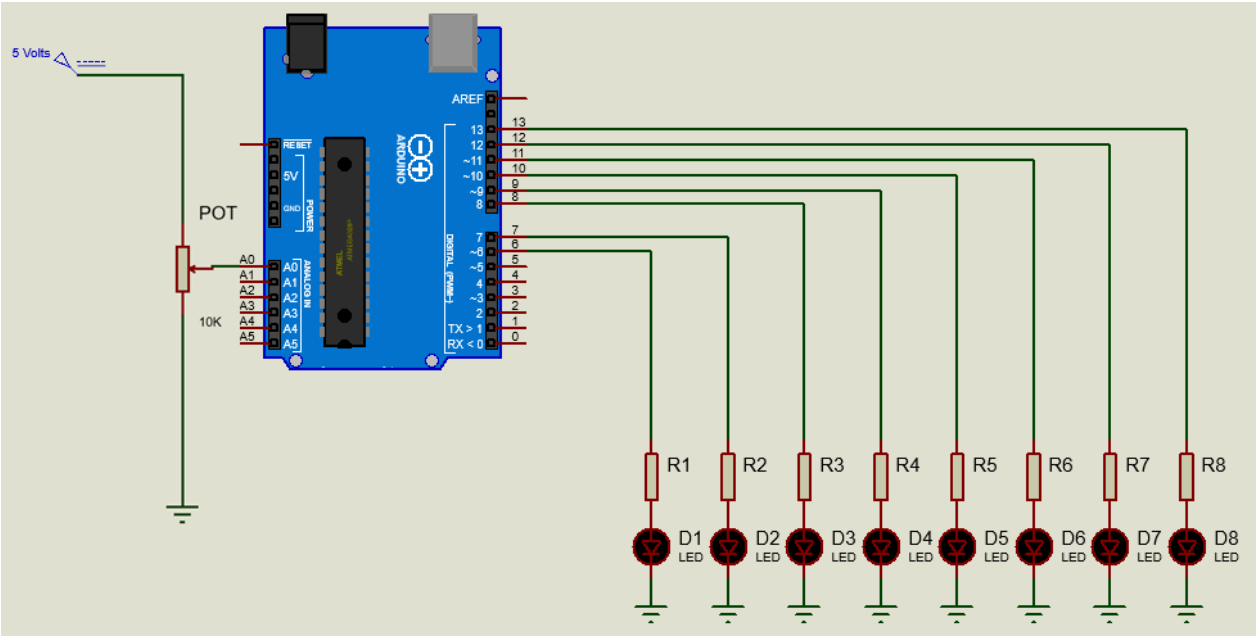
Después de la introducción se presenta un ejemplo con instrucciones digitalWrite() y digitalWrite().

Finalmente se presenta el esquema de la figura 3-5 con sus reglas de funcionamiento para luego pedir un programa que sea capaz de controlar el proceso utilizando instrucción while.

3.1.6 Experiencia 6: Entradas Analógicas

El objetivo de esta experiencia es utilizar las entradas analógicas e interpretar la información.

Como se ha enseñado a trabajar con entradas analógicas en la unidad cuatro, la guía seis de esta unidad no tiene introducción, por ello se presentan inmediatamente las dos experiencias, la primera propone el circuito de la figura 3-6.



Fuente: Elaboración propia

Figura 3-6 Circuito 1 experiencia 6

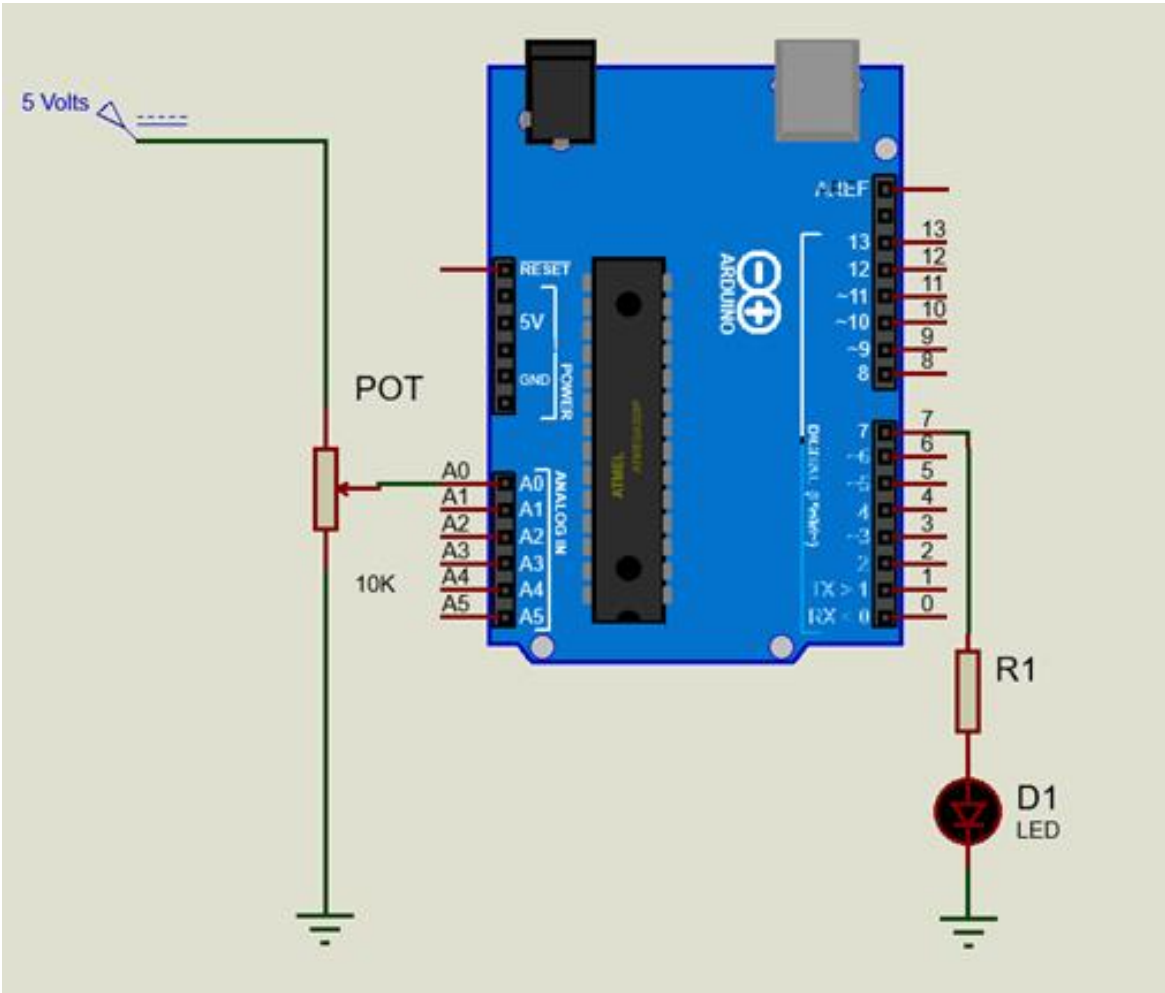
Antes de comenzar la escritura del programa el alumno debe calcular las resistencias para los leds, luego implementar el circuito y conectar el Arduino a su pc.

Para la experiencia del circuito en la figura 3-6 se debe escribir un programa que opere detectando el voltaje del potenciómetro en la entrada analógica y vaya encendiendo un led por cada 625mV a medida que el voltaje aumenta. El programa también debe apagar los leds a medida que el voltaje disminuye.

El circuito está diseñado para que cada led se encienda exactamente con la misma variación de voltaje, de este modo se distribuyen los ocho leds en los 5 volts, ese es el motivo por el que se pide que los leds se vayan encendiendo por cada 625mV.

Para la segunda experiencia se propone el circuito de la figura 3-7.

La segunda parte de la guía propone escribir un programa para que el Arduino genere una señal cuadrada en que su frecuencia varíe en función del voltaje en el potenciómetro, es decir, mientras menor sea el voltaje, el led parpadeará más lentamente y mientras el voltaje aumenta, el led parpadeará más rápido.



Fuente: Elaboración propia

Figura 3-7 Circuito 2 experiencia 6

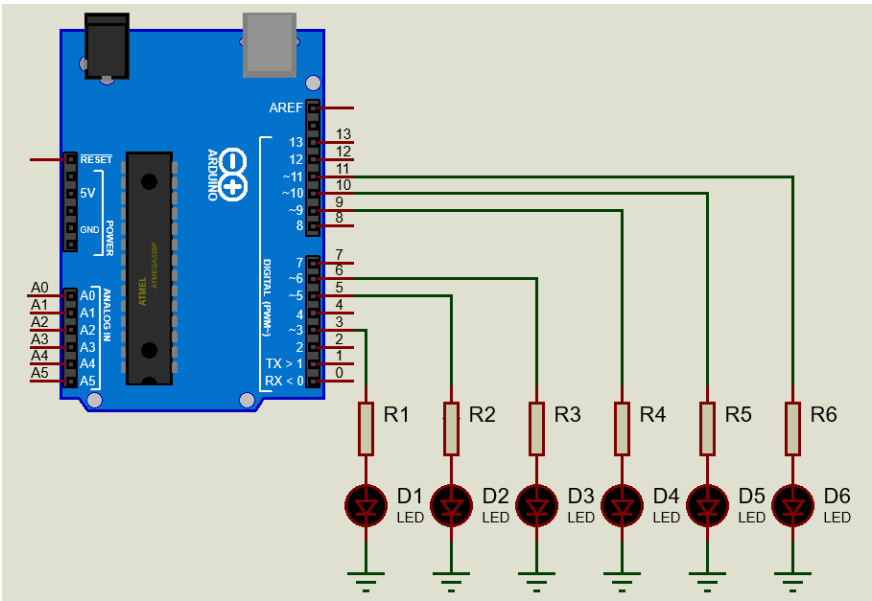
Como instrucción final y para añadirle más dificultad, se pide considerar que si en la entrada analógica hay 0v el led debe estar apagado y si se detectan 5v el led debe estar encendido.

3.1.7 Experiencia 7: PWM

Los objetivos principales de esta experiencia son utilizar las salidas PWM y aplicar conocimientos de la guía anterior.

Al igual que la guía anterior, ya se ha enseñado a trabajar con salidas del tipo PWM en la unidad cuatro, por lo tanto esta guía tampoco tiene introducción.

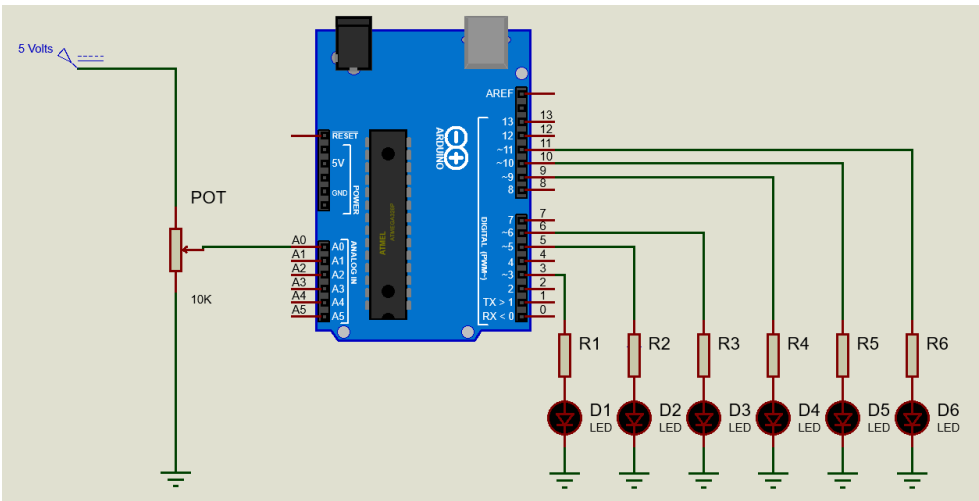
En el primer circuito de la figura 3-8 se propone crear un programa que varíe la luminosidad de los leds en forma gradual con un periodo de 4 segundos, 2 segundos para el encendido y 2 para el apagado.



Fuente: Elaboración propia

Figura 3-8 Circuito 1 experiencia 7

Para la segunda parte de la experiencia, se propone el circuito de la figura 3-9



Fuente: Elaboración propia

Figura 3-9 Circuito 2 experiencia 7

Se debe escribir un programa que varíe la luminosidad de los leds en función del voltaje detectado en el potenciómetro, de este modo se pide aplicar los contenidos de salidas PWM y de entradas analógicas.

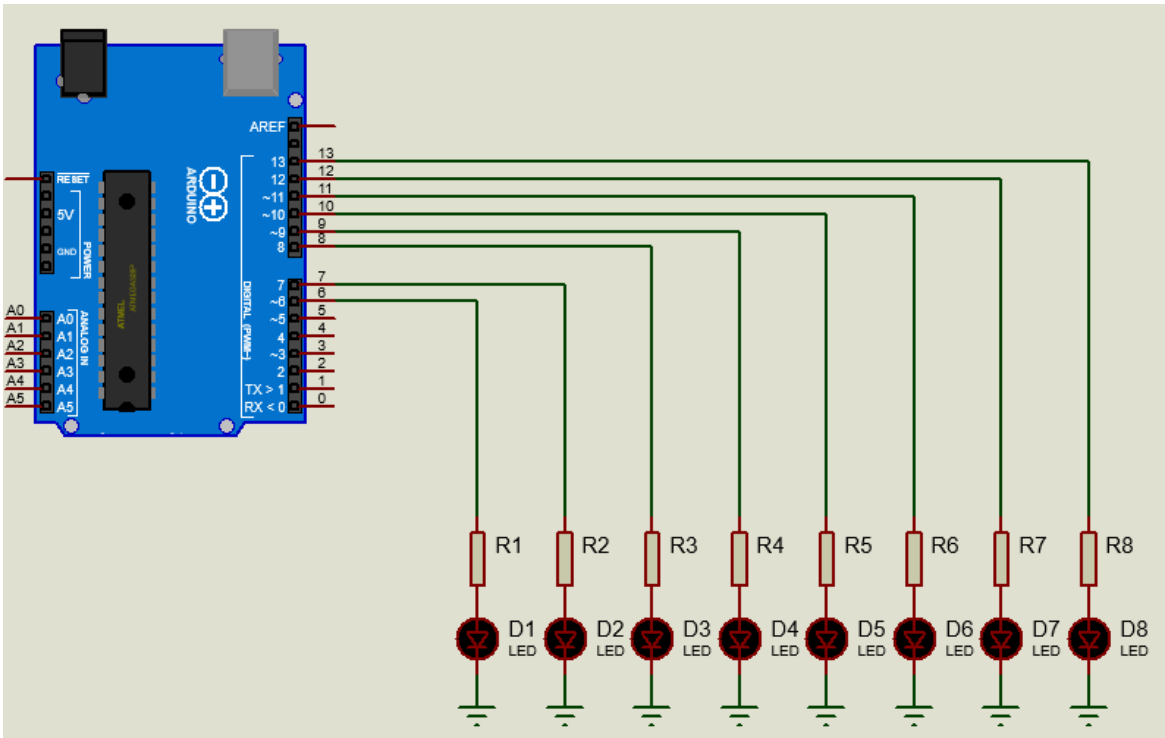
3.1.8 Experiencia 8: Control de servomotor

Esta experiencia está totalmente presentada y desarrollada en el capítulo dos del presente trabajo de título.

3.1.9 Experiencia 9: Comunicación Serial

La experiencia nueve tiene como objetivo que el alumno aplique los contenidos de comunicación serial incluidos en la unidad cuatro.

En el primer circuito de la figura 3-10 se pide escribir un programa en que el Arduino reciba datos desde el PC y dependiendo de lo que reciba se enciendan diferentes secuencias de leds.

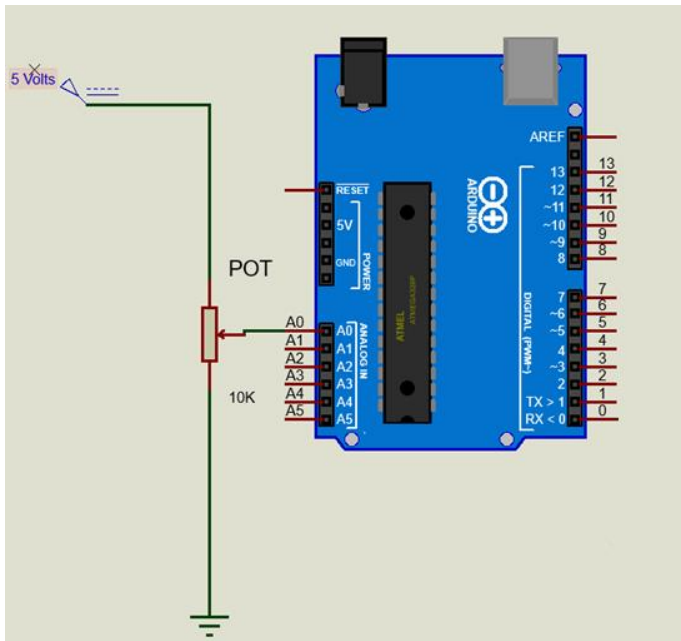


Fuente: Elaboración propia

Figura 3-10 Circuito 1 experiencia 9

El objetivo de esta experiencia es que el Arduino reciba datos en el puerto serial provenientes del teclado del PC para posteriormente interpretar esos datos y tomar una decisión.

En la segunda parte de la experiencia 9 se propone el circuito de la figura 3-11



Fuente: Elaboración propia

Figura 3-11 Circuito 2 experiencia 9

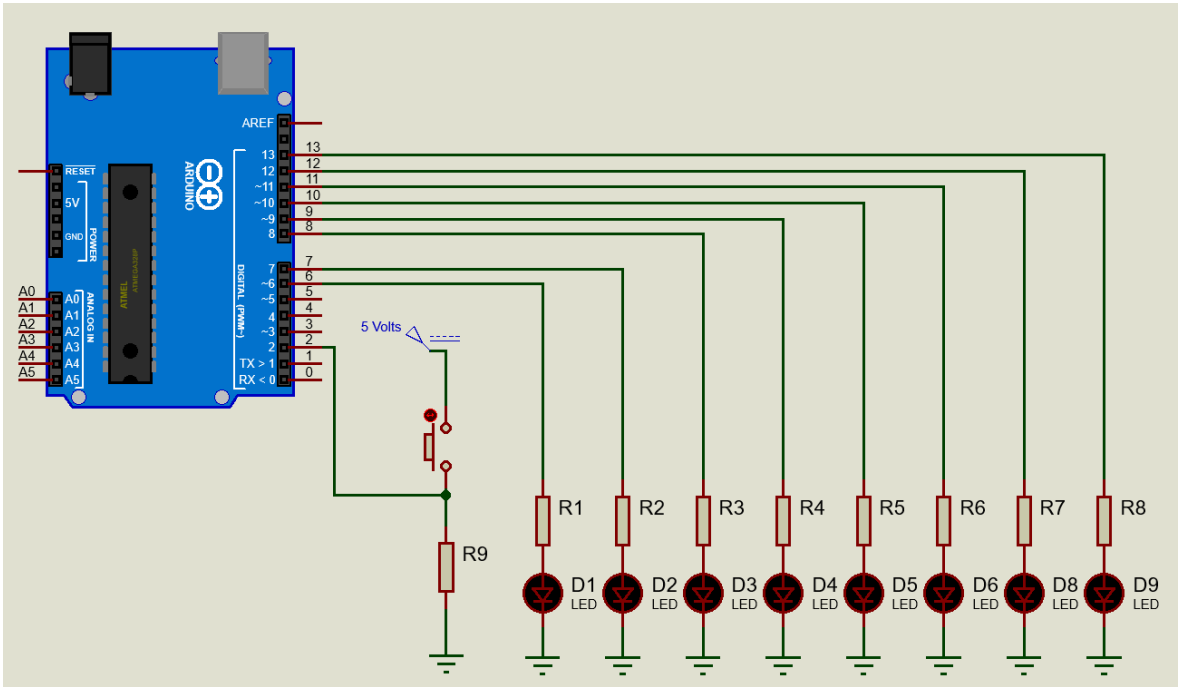
Se pide escribir un programa para que el Arduino envíe el valor de voltaje del potenciómetro en tiempo real a través del puerto serial al PC, de este modo, esta segunda parte consiste en poner en práctica el envío de datos y la lectura de ellos mediante el monitor serial en el software IDE.

3.1.10 Experiencia 10: Interrupciones

La última experiencia aborda el tema de las interrupciones y su objetivo es aprender a programarlas, esto para que el estudiante aprenda a tomar decisiones con el Arduino en caso de que ocurra algún evento externo.

Como no se ha entregado los contenidos de interrupciones en las unidades anteriores, la guía inicia con una introducción en donde se explica qué es una interrupción, qué pines se pueden habilitar para detectar una interrupción, la instrucción para configurarla y los parámetros necesarios para su funcionamiento.

La primera parte propone el circuito de la figura 3-12

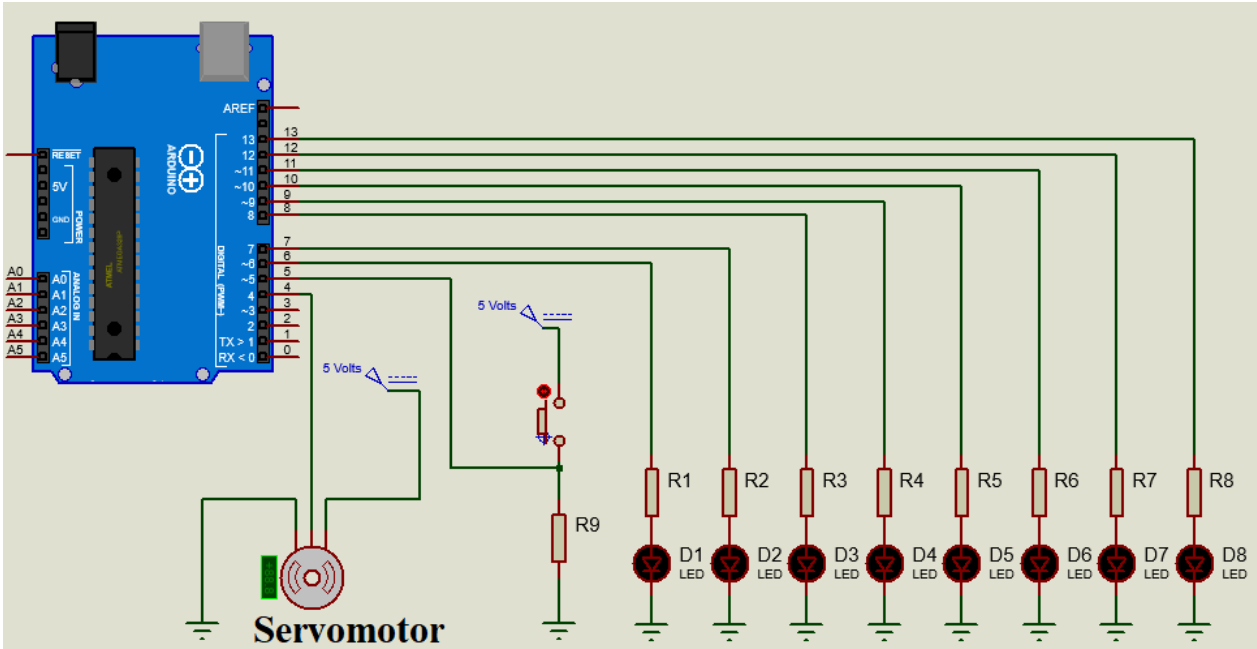


Fuente: Elaboración propia

Figura 3-12 Circuito 1 experiencia 10

El circuito de la figura 3-12 propone escribir un programa en que el Arduino haga una secuencia de leds, en el momento en que se presione el pulsador deberán quedar los leds encendidos durante 10 segundos a través de una interrupción.

La segunda parte de la experiencia propone el circuito de la figura 3-13.



Fuente: Elaboración propia

Figura 3-13 Circuito 2 experiencia 10

En este caso se pide diseñar en un programa en que el Arduino realice una secuencia de leds. En el momento en que se presione el pulsador, mediante una interrupción los leds deberán apagarse y el servomotor deberá moverse dos veces de entre 0° y 180°.

El objetivo de esta segunda parte es aplicar los contenidos de interrupciones y también volver a programar el control de un servomotor.

CONCLUSIÓN

Arduino es una buena herramienta para comenzar a aprender programación en C++, sus ventajas de ofrecer un precio de costo bajo, ser una plataforma abierta de desarrollo y de entregar todo armado y listo para comenzar, lo hacen muy accesible y amigable para cualquier persona.

Es importante tener conocimientos previos de electrónica básica antes de empezar a trabajar con el hardware de Arduino, a pesar de que toda su plataforma de desarrollo es muy sencilla para el aprendizaje.

Aprender electrónica y programación es relativamente sencillo, lo realmente difícil es entregar el conocimiento, ya que una vez que uno como profesional ha aprendido la temática lo ve muy sencillo, sin embargo para un alumno puede resultar más complicado.

Desde el punto de vista de los ramos administrativos de la carrera, este trabajo ha servido como una forma de volver al inicio, algo muy importante que realizan las grandes empresas y que ha sido de gran utilidad.

La docencia es un trabajo muy complicado, demandante y exigente, en definitiva, requiere mucha vocación, entrega y esfuerzo.

Es posible en un futuro aplicar mejoras o extender este curso con contenidos más avanzados e incorporar el uso de kits que puedan ser controlados con Arduino, algunos ejemplos pueden ser los siguientes dispositivos:

- Módulo de comunicación MODBUS
- Comunicación con otros Arduinos
- Dispositivos Xbee o Zigbee
- Pantallas LCD

BIBLIOGRAFÍA

Links visitados año 2022:

<https://arduino.cl/>

<https://programarfacil.com/categoria/blog/arduino-blog/>

<https://www.youtube.com/playlist?list=PLZHVfZzF2DYJeLXXxz6YtpBj4u7FoGPWN>

https://www.youtube.com/playlist?list=PLaOwYHXnu18u0p8888L_DiQNCE5Mcpp92

<https://www.youtube.com/playlist?list=PLyLh25DppBle40j3VBAsInVfs4Pz-B3ZB>

<https://www.youtube.com/playlist?list=PLkjnQ3NFTPnY1eNyLDGi547gkVui1vyn2>

Software Proteus 8.12 SP0 para Windows 10 utilizado en dibujo de circuitos y componentes electrónicos

Software Arduino IDE 2.0.2 para escribir los programas ejemplo

ANEXO: GUÍA TEMÁTICA DE ARDUINO PARA AUTOAPRENDIZAJE

ANEXO A:

UNIDAD 1: TEOREMAS Y CONCEPTOS BÁSICOS PARA EL ANÁLISIS DE CIRCUITOS

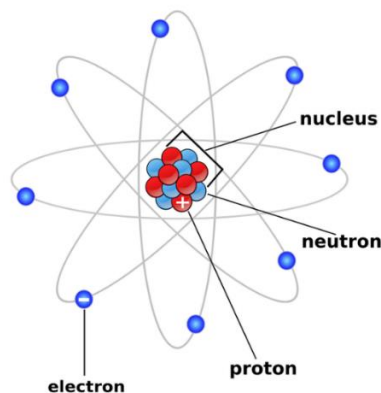
Guía 1: El Voltaje y la Corriente

- El átomo

El átomo está compuesto por neutrones, protones y electrones. En el núcleo se puede encontrar los neutrones y protones, mientras que girando alrededor del núcleo están los electrones. Todo está compuesto por átomos de distintos elementos, sin embargo, no todos los átomos reúnen las condiciones necesarias para ser utilizados con corriente y/o voltaje.

El protón tiene carga positiva, el electrón tiene carga negativa y el neutrón tiene carga neutra.

En la figura 4-1 se tiene un ejemplo de un átomo.



Fuente: <https://cienciadesofa.com/2015/05/es-sistema-solar-atomo-gigante.html>

Figura 4-1 Imagen referencial de un átomo

- El Voltaje

Existen muchas definiciones, pero para este curso, se definirá el voltaje como la fuerza que provoca el movimiento de electrones (flujo) de un punto a otro. Su unidad de medida es en Volts o voltios y su símbolo es la letra (V).

El voltaje también se define como diferencia de potencial y tensión eléctrica o simplemente “Tensión”.

Para medir voltaje se utiliza un voltímetro y la medida debe ser tomada entre dos puntos distintos.

Como el voltaje se aplica entre dos puntos distintos, se generará una polaridad, un punto será el polo positivo y el otro punto será el polo negativo, para medir voltaje se conectará el instrumento de medición en paralelo.

- La Corriente

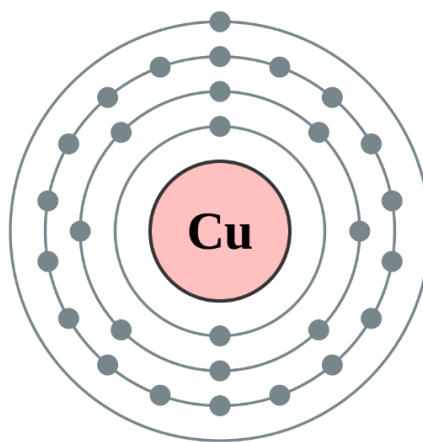
Es el flujo de cargas positivas, su unidad de medida es en Amperes y su símbolo es la letra (A). Circula en dirección contraria al flujo de electrones. El tamaño del flujo de cargas positivas se define como intensidad de corriente eléctrica o simplemente “Intensidad” y su símbolo es la letra (I).

Para medir corriente se utiliza un amperímetro y la medida la se debe tomar en un solo punto, es decir, conectar el instrumento de medición en serie.

Anteriormente se mencionó que no todos los átomos reúnen las condiciones necesarias para ser utilizados con corriente y/o voltaje. Los elementos compuestos por átomos que reúnan esas condiciones se definen como conductores, mientras que en el caso contrario se definen como aislantes.

¿Cómo saber cuándo un elemento es conductor o aislante?

A través de la capa de valencia de sus átomos. Los átomos tienen sus electrones girando a distintas distancias de su núcleo, los que se encuentran más lejanos son electrones de valencia (que se encuentran en la capa del mismo nombre). Los elementos compuestos por átomos como los de la figura 4-2 (átomo de cobre) son conductores.



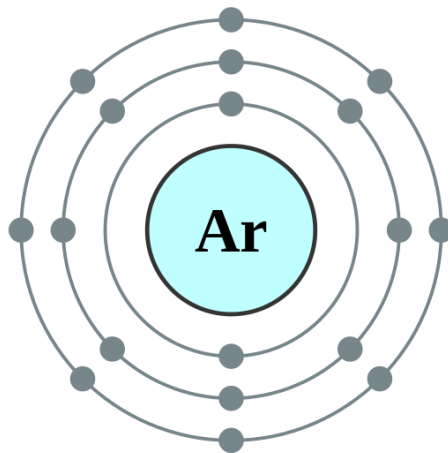
Fuente: <https://www.freepng.es/png-t99avg/>

Figura 4-2 Átomo de Cobre (Conductor)

El átomo de cobre reúne las condiciones ideales para conducir electricidad con solamente un electrón en su capa de valencia.

Un material es un conductor eléctrico cuando sus átomos tienen pocos electrones en la capa de valencia, por lo tanto, no es necesario aplicar mucho voltaje para que sus electrones de valencia salten de un átomo a otro y se genere corriente.

Un elemento aislante, al contrario de los conductores, tiene mayor cantidad de electrones en la capa de valencia. En la figura 4-3 se presenta un ejemplo con un átomo de Argón.



Fuente: <https://www.rolscience.net/2020/06/datos-y-curiosidades-sobre-el-argon.html>

Figura 4-3 Átomo de Argón (Aislante)

El átomo de Argón tiene 8 electrones en su capa de valencia, además de ser un gas noble, presenta las condiciones ideales de un elemento aislante.

Cuando un material es aislante, al aplicar voltaje, este se distribuye de modo uniforme en sus electrones de valencia, por lo tanto, para que los electrones puedan saltar y generar corriente, se requiere una gran cantidad de voltaje.

- Datos curiosos

Una pregunta frecuente es ¿Cuál es el mejor conductor?

La Plata es el mejor conductor, pero debido al costo económico se utiliza el cobre que tiene propiedades muy similares en lo que respecta a la conductividad.

¿Y cuáles son los mejores aislantes?

Los más utilizados son el plástico, el vidrio, la cerámica, la madera, el papel y el teflón.

Guía 2: La ley de Ohm y el código de colores

Una vez aprendido lo que es corriente y voltaje, solo falta un parámetro para completar la ley de ohm: la resistencia eléctrica.

- La Resistencia Eléctrica

Es la capacidad que tiene un material para oponerse al flujo de corriente. Su unidad de medida es en Ohm y su símbolo es la letra griega (Ω).

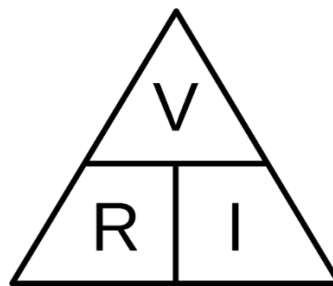
- Ley de Ohm

La ley de ohm establece que el voltaje aplicado en un conductor es directamente proporcional a la corriente que circula a través de él, mientras que la resistencia es inversamente proporcional a la corriente. Los parámetros son Voltaje (V), Corriente (I) y Resistencia (R).

- Ecuación

$$V = I \cdot R$$

En la figura 4-4 se tiene la ecuación de la ley de ohm representada en un triángulo.



Fuente: Elaboración propia

Figura 4-4 Triángulo de la Ley de Ohm

La ley de ohm es la ecuación más utilizada en las áreas de electricidad y electrónica, por lo que es fundamental dominarla. La figura 4-4 será de gran utilidad durante todo el curso.

- La resistencia como componente electrónico

El resistor (llamado simplemente “resistencia” en jerga electrónica) es un componente electrónico utilizado para limitar o reducir la corriente entre dos puntos. Está construido principalmente con carbón y tiene un valor en [ohm] dentro de la tolerancia de fabricación, de modo que se pueda calcular cuánto se desea reducir la corriente.



Fuente: <http://corrienteycircuitoselectricos2015.blogspot.com/2015/04/resistencia-electrica-r.html>

Figura 4-5 Imagen referencial de una Resistencia

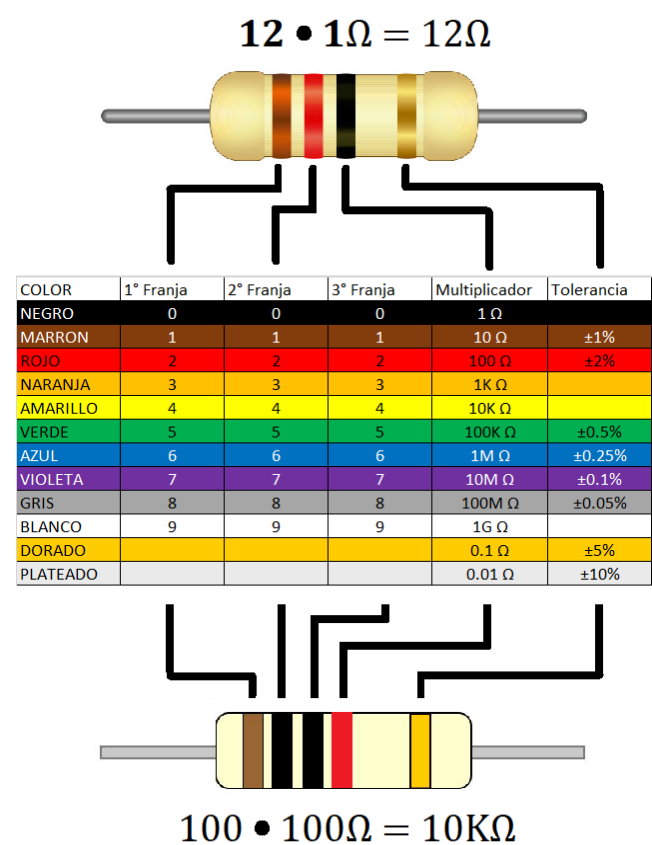
La Figura 4-5 muestra el tipo de resistencia más común que se utilizará en el curso. Las franjas de colores determinan el valor en ohm (Ω) de la resistencia y los valores de cada color están determinados en el código de colores.

La resistencia no tiene polaridad, por lo que se puede conectar en cualquier sentido al armar los circuitos.

- Código de colores

Las resistencias pueden tener 4, 5 y hasta 6 franjas de colores. Las primeras franjas (incluido el multiplicador) indican el valor, pero siempre las dos últimas serán el multiplicador y la tolerancia, respectivamente.

La tolerancia indica el porcentaje que puede variar el valor de la resistencia respecto del valor nominal, es decir, al tener una resistencia de 12 ohm (como la del ejemplo en la figura 3) y su última franja es dorada, quiere decir que su valor real puede estar entre un 5% más (12.6 Ω) y un 5% menos (11.4 Ω). La franja de la tolerancia siempre está un poco más separada de las otras franjas.



Fuente: Elaboración propia

Figura 4-6 Código de colores con 2 ejemplos.

- Datos Curiosos

No existe el conductor perfecto, es decir, que tenga resistencia 0 ohm, todos los materiales (incluidos los conductores) tienen resistencia, pero al diseñar circuitos, no se consideran esos valores porque son despreciables.

Quando se habla de resistencia de materiales, se usa el término resistividad, también se mide en ohm (Ω) y su símbolo es la letra griega rho (ρ).

Guía 3: Ley de Watt

- Potencia Eléctrica

La potencia eléctrica es la cantidad de energía que consume o disipa un componente o circuito durante un rango de tiempo determinado, es decir, corresponde al gasto de energía durante un intervalo de tiempo. En el caso de dispositivos que entregan corriente y/o voltaje se referirá a la potencia que son capaces de entregar.

Existen distintas unidades para medir la potencia eléctrica, pero en este curso sólo se verán circuitos de voltaje y corriente continua, por lo tanto, su unidad de medida son los Watts y su símbolo es la letra (W).

- Ley de Watt

La ley de Watt establece que la potencia eléctrica consumida, disipada o entregada por un componente o circuito es directamente proporcional al voltaje de alimentación y a la corriente que fluye a través de él.

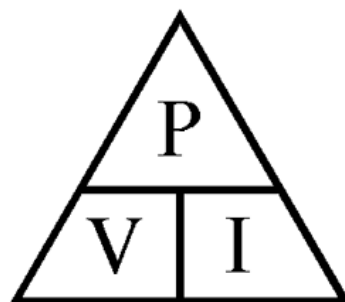
Para efectos de cálculos y ecuaciones, se utilizará la letra P como expresión algebraica de la potencia.

Se puede obtener la potencia calculando el producto del voltaje y la corriente.

- Ecuación

$$P = V \cdot I$$

O también se puede utilizar el triángulo de la ley de Watt mostrado en la figura 4-7 que (al igual que la ley de ohm) será de gran utilidad.



Fuente: Elaboración propia

Figura 4-7 Triángulo de la ley de Watt

Otras ecuaciones para el cálculo de la potencia son las siguientes:

Si se conoce el voltaje y la resistencia.

$$P = \frac{V^2}{R}$$

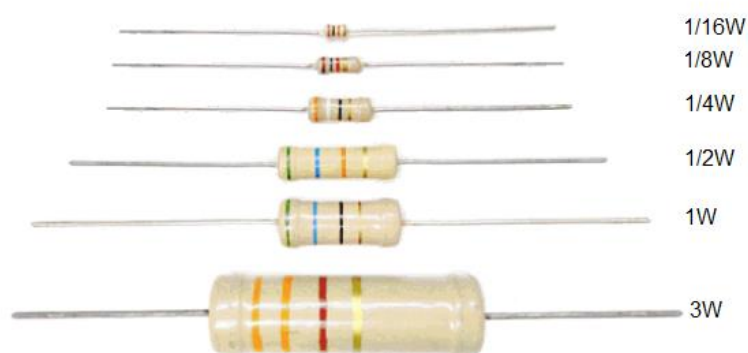
Si se conoce la corriente y la resistencia.

$$P = I^2 \cdot R$$

Las ecuaciones presentadas vienen desde la ley de ohm.

- Dato Importante

Al comenzar las experiencias prácticas se podrá notar que existen resistencias con distintos tamaños, esto es porque las más pequeñas son de más baja potencia (hasta 1/8 de watt) mientras que las más grandes aguantan una potencia mayor. En la figura 4-8 se presenta una imagen referencial de algunas resistencias presentes en el kit de Arduino UNO.



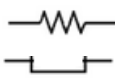
Fuente: <https://geeelectronica.com/resistencias-fijas-o-resistores-electricos/>

Figura 4-8 Imagen referencial con resistencias de distintas potencias.

Guía 4: Simbología Electrónica

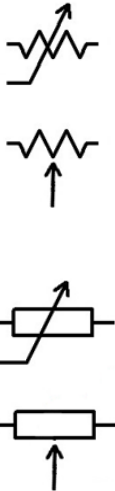
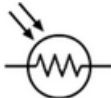
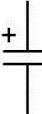
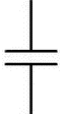
Ahora que ya son conocidas las leyes de Ohm y de Watt, el siguiente paso es aprender cómo aplicarlas. Para ello es necesario aprender la simbología y los esquemáticos. Lógicamente no se presentan todos los símbolos, solo los que serán usados durante este curso y también algunos más que probablemente se puedan utilizar para aprender de forma autodidacta. Se pueden ver desde la tabla 1-1 a la tabla 1-3.

Tabla 1-1. Simbología necesaria para la Unidad 1

	Polaridad Positiva		Polaridad Negativa
	Corriente Continua (CC) Direct Current (DC en Inglés)		Fuente de Voltaje Continuo
	Pila		Batería
	Resistencia Eléctrica (ambos símbolos son válidos)		LED (Diodo Emisor de luz)
	Cable o puente		Interruptor Abierto
	Cruce de cables sin conexión		Interruptor Cerrado

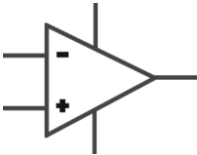
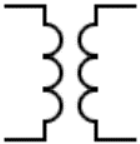
Fuente: Elaboración propia

Tabla 1-2 Simbología Necesaria para la Unidad 2

	Amperímetro		Voltímetro
	Ohmetro		Fusible
	Potenciómetro (Resistencia Variable)		LDR (Resistencia Dependiente de la luz) Sensor de Luz
			Diodo Rectificador
			Transistor NPN
			Transistor PNP
	Bobina		Condensador electrolítico
	Condensador		Compuerta AND
	Compuerta OR		Compuerta NOT
	Compuerta NAND		Compuerta NOR
	Compuerta OR Exclusiva (tambien llamada XOR)		Compuerta NOR Exclusiva (también llamada XNOR)

Fuente: Elaboración propia

Tabla 1-3 Otros Símbolos

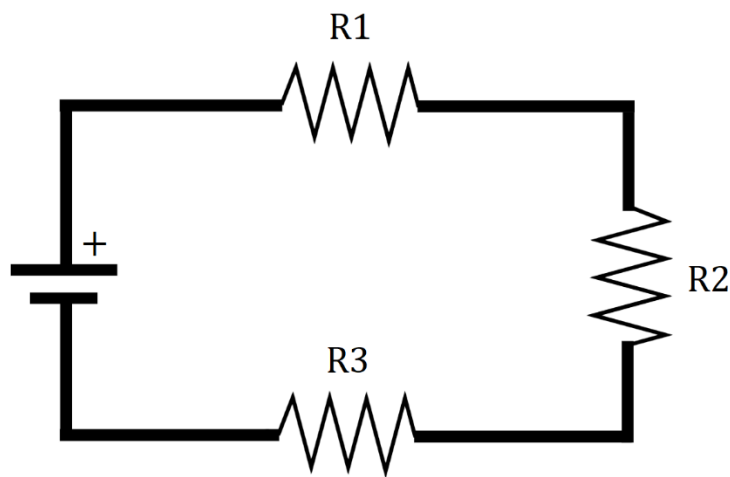
	Diodo Zener		Amplificador Operacional
	Oscilador de Cristal		Transformador (Símbolo Genérico)
	Ampolleta		Motor Eléctrico (Símbolo Genérico)
	Pulsador N.A. (Normalmente Abierto)		Altavoz o Parlante
	Pulsador N.C. (Normalmente Cerrado)		Zumbador

Fuente: Elaboración Propia

Guía 5: Circuitos Serie, Paralelo y Mixto

- Circuito Serie.

El circuito serie es un tipo de circuito en el que la corriente sigue una misma trayectoria durante todo su recorrido. Las resistencias están conectadas una después de la otra como en la figura 4-9.



Fuente: Elaboración propia

Figura 4-9 Ejemplo de circuito serie

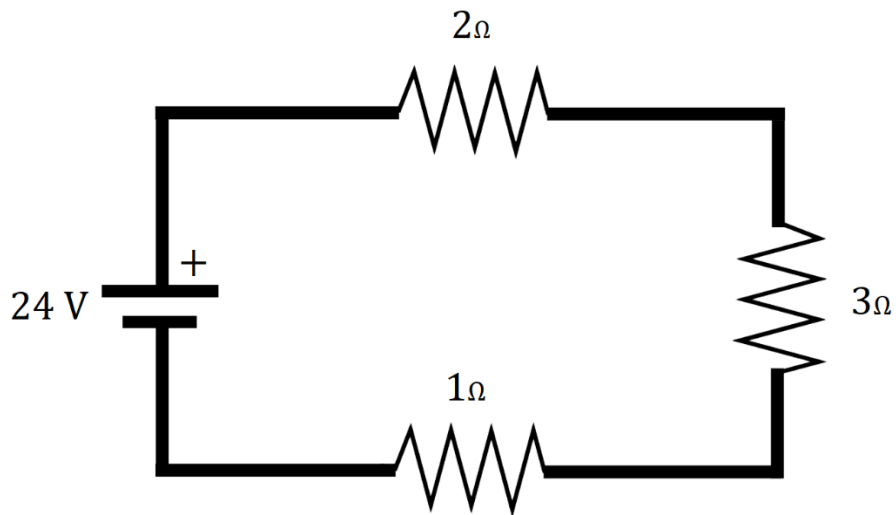
La cantidad de resistencias del circuito dependerá del diseño (uno define la cantidad de resistencias).

La resistencia total del circuito se obtiene sumando todas las resistencias.

$$R_{\text{Total}} = R_1 + R_2 + R_3$$

Si se sabe que la cantidad de resistencias es definida por uno mismo al igual que los valores en ohm, se debe tener cuidado en no elegir valores de resistencias lo suficientemente bajos de modo que no circule una corriente demasiado elevada en el circuito y que pudiera dañar las resistencias o la fuente.

En la figura 4-10 se presenta un ejemplo de un circuito en el que podría producirse una corriente muy alta.



Fuente: Elaboración propia

Figura 4-10 Circuito serie en el que podría producirse sobrecarga por exceso de corriente

Se calcula la resistencia total del circuito de la siguiente forma:

$$R_{\text{Total}} = 2\Omega + 3\Omega + 1\Omega$$

$$R_{\text{Total}} = 6\Omega$$

Teniendo la resistencia total, se aplica ley de ohm para obtener la corriente:

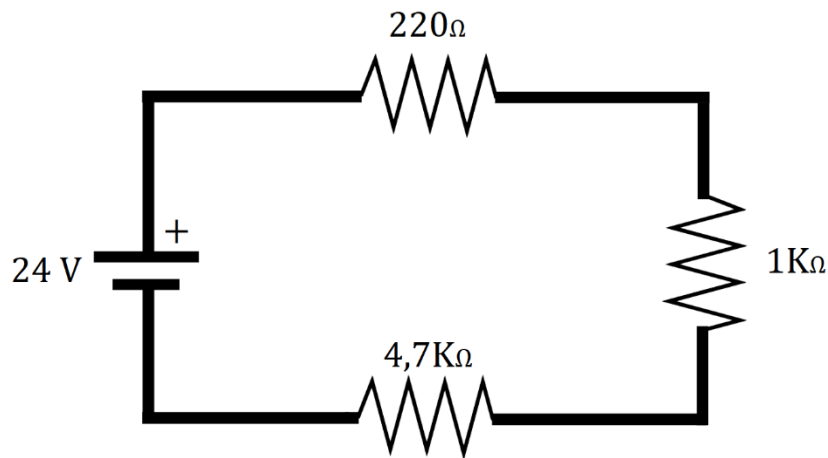
$$I = \frac{V}{R_{\text{Total}}}$$

$$I = \frac{24V}{6\Omega}$$

$$I = 4A$$

En un circuito electrónico, una corriente de 4A se puede considerar una corriente muy elevada, ya que generalmente en electrónica se trabaja con corrientes mucho más pequeñas.

En la figura 4-11 se presenta un diseño correcto de un circuito serie en el que la corriente se mantiene en el orden de los miliamperes (valores suficientemente pequeños), por lo tanto el circuito es estable.



Fuente: Elaboración propia

Figura 4-11 Ejemplo correcto de un circuito serie

Se realizan los mismos cálculos que hechos en el circuito anterior para obtener la resistencia total:

$$R_{\text{Total}} = 220\Omega + 1K\Omega + 4,7K\Omega$$

$$R_{\text{Total}} = 5,92K\Omega$$

Teniendo la resistencia total, se aplica ley de ohm para calcular la corriente:

$$I = \frac{V}{R_{\text{Total}}}$$

$$I = \frac{24V}{5,92K\Omega}$$

$$I = 0,00405A = 4,05mA$$

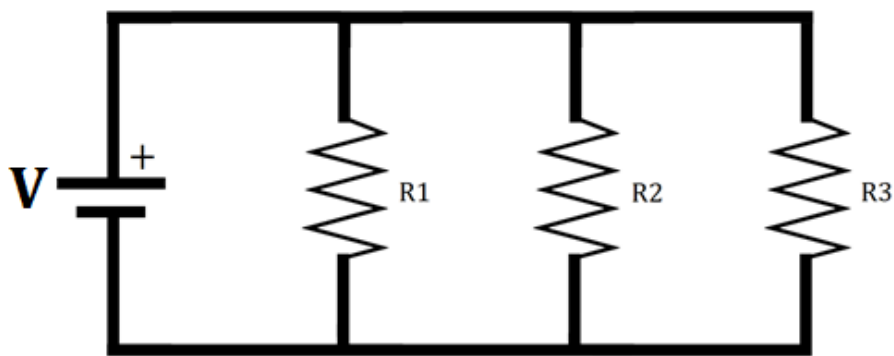
En un circuito electrónico, una corriente de aproximadamente 4 miliamperes es una corriente pequeña y que no representa un riesgo de sobrecarga, de este modo el circuito se mantiene estable.

- Consideraciones importantes del circuito serie

1. En un circuito serie los voltajes se dividen entre las cargas (resistencias).
2. En un circuito serie la corriente es la misma en todo el circuito.
3. En caso de falla de alguna de las cargas o en caso de alguna desconexión, el circuito completo dejará de funcionar.

- Circuito Paralelo.

El circuito paralelo es un circuito en el que las resistencias se conectan de un modo en el que sus terminales de entrada están conectados todos juntos, al igual que los terminales de salida como en el ejemplo de la figura 4-12.



Fuente: Elaboración propia

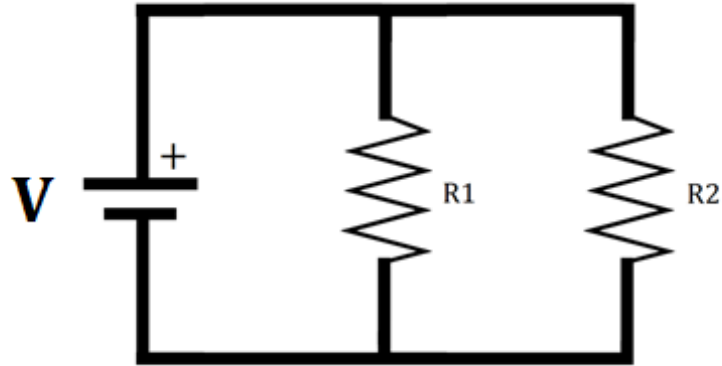
Figura 4-12 Ejemplo de circuito paralelo con más de dos resistencias

Al igual que en el circuito serie, la cantidad de resistencias (cargas) las define uno mismo.

Para obtener la resistencia total de un circuito en paralelo con más de dos resistencias, se debe sumar el inverso multiplicativo de cada resistencia de la siguiente forma, en el caso que hayan 3 resistencias en paralelo:

$$\frac{1}{R_{total}} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3}$$

En el caso de un circuito en paralelo con dos resistencias como el de la figura 4-13, se puede calcular el producto de ambas resistencias y dividirlo por la suma.



Fuente: Elaboración propia

Figura 4-13 Ejemplo de circuito paralelo con dos resistencias

Ecuación para el cálculo de la resistencia total en un circuito paralelo con dos resistencias:

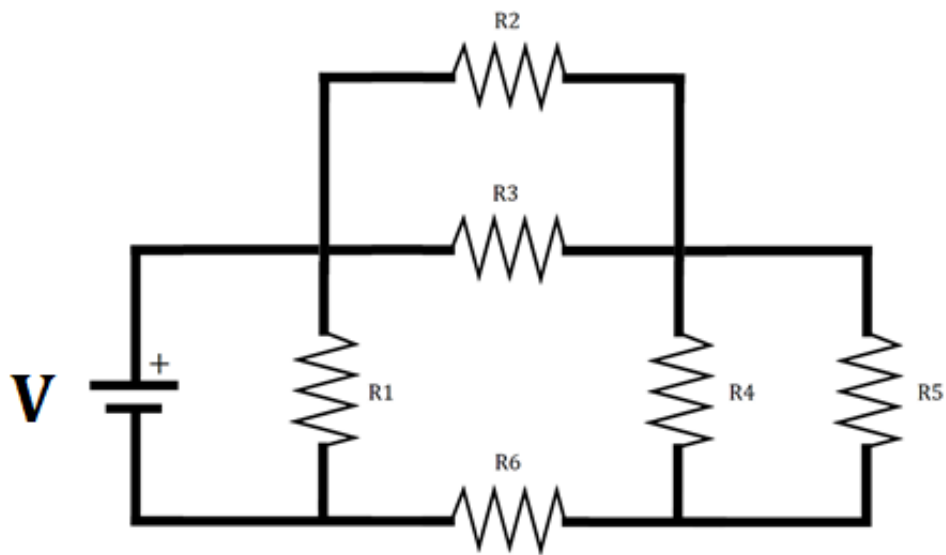
$$R_{Total} = \frac{R1 \cdot R2}{R1 + R2}$$

- Consideraciones importantes del circuito paralelo
1. En un circuito paralelo, el voltaje en cada resistencia es el mismo.
 2. En un circuito paralelo las corrientes se dividen.
 3. A diferencia del circuito serie, en el circuito paralelo si se desconecta alguna resistencia (carga) el circuito continuará funcionando.
 4. A medida que vayamos agregando más resistencias al circuito paralelo, la resistencia total disminuirá y por lo tanto la corriente aumentará.

- Circuito Mixto

El circuito mixto es un tipo de circuito compuesto por resistencias conectadas en serie y en paralelo como el que se presenta en la figura 4-14.

Para poder calcular la resistencia total, se debe aplicar las ecuaciones mencionadas en los circuitos serie y paralelo cuando corresponda.



Fuente: Elaboración propia

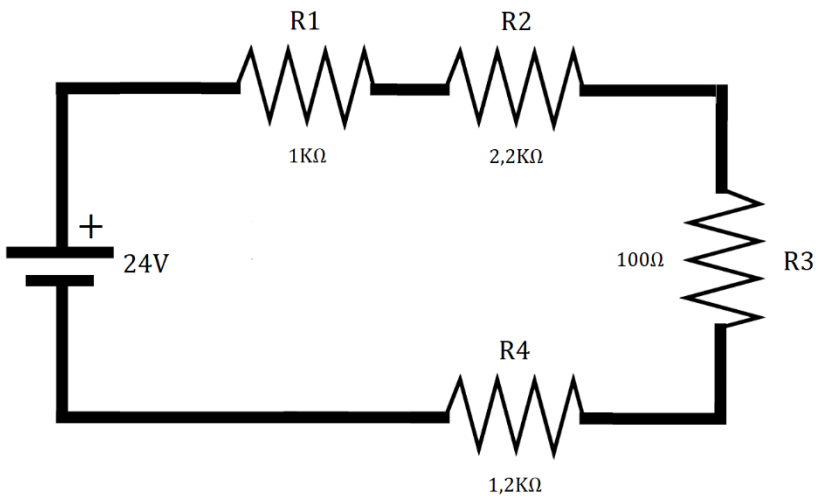
Figura 4-14 Ejemplo de circuito mixto

En el circuito de la figura 4-14 se puede comenzar reduciendo las resistencias R_2 y R_3 están conectadas en paralelo, al igual que las resistencias R_4 y R_5 . Luego se puede redibujar el circuito con las nuevas resistencias para poder reducirlo al máximo y obtener la resistencia total.

Ejercicios

Aplicando Ley de Ohm, Ley de Watt y ecuaciones de circuitos serie y paralelo, encuentre la corriente total, potencia total y resistencia total en los siguientes circuitos

1.



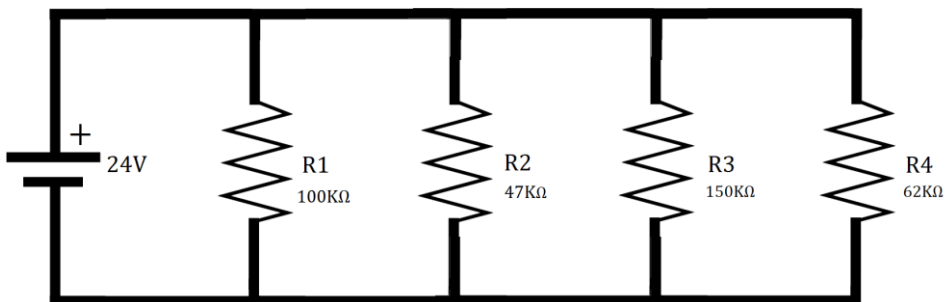
Respuestas:

$$R_{\text{total}} = 4,5\text{K}\Omega$$

$$I_{\text{total}} = 5,333\text{ mA}$$

$$P_{\text{total}} = 128\text{ mW}$$

2.



Respuestas:

$$R_{\text{total}} = 18,49\text{K}\Omega$$

$$I_{\text{total}} = 1,29\text{ mA}$$

$$P_{\text{total}} = 31\text{ mW}$$

Guía 6: Leyes de Voltajes y de Corrientes de Kirchhoff

- Malla eléctrica

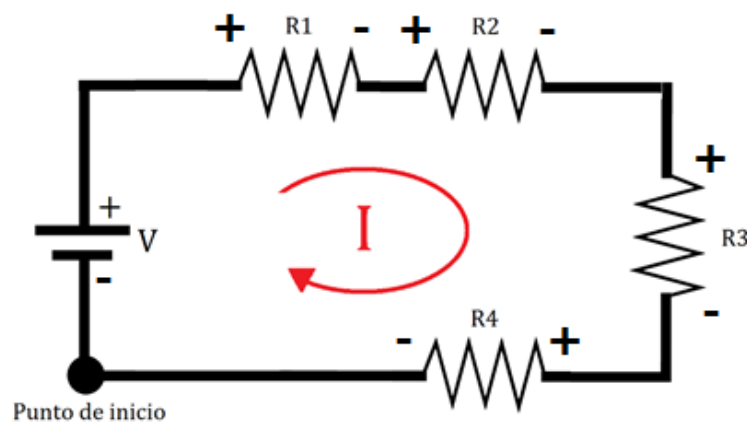
Es un camino por donde circula la corriente, a través de él puede haber fuentes de voltaje, resistencias o cualquier otro componente. En este caso se hará un análisis con resistencias. Para considerar un camino como una malla, se empieza desde un punto y se realiza el recorrido hasta volver al punto de inicio. El camino de la malla debe ser un circuito cerrado para que exista circulación de corriente.

- Ley de Voltajes de Kirchhoff o Teorema de Mallas

La ley de Voltajes de Kirchhoff o también conocida como teorema de mallas establece que al realizar el recorrido a través de una malla eléctrica (desde el punto de inicio hasta llegar al mismo punto), la sumatoria de todos sus voltajes es igual a cero.

El objetivo es obtener la corriente que circula a través de una malla mediante sistemas de ecuaciones, para ello se utiliza la ley de Ohm principalmente.

Este teorema generalmente se utiliza en circuitos que tengan más de una malla, pero se hará el análisis en el circuito de la figura 4-15 con una sola malla a modo de ejemplo.



Fuente: Elaboración propia

Figura 4-15 Ejemplo de circuito con una malla

Teniendo el circuito de la Figura 4-15, se realiza el recorrido en sentido horario y se anota cada voltaje con su respectiva polaridad (que serán los signos en la ecuación), de este modo se obtiene lo siguiente

- - Voltaje de la fuente
- +Voltaje en la resistencia 1
- +Voltaje en la resistencia 2
- +Voltaje en la resistencia 3
- +Voltaje en la resistencia 4

Luego se ponen los términos en la ecuación:

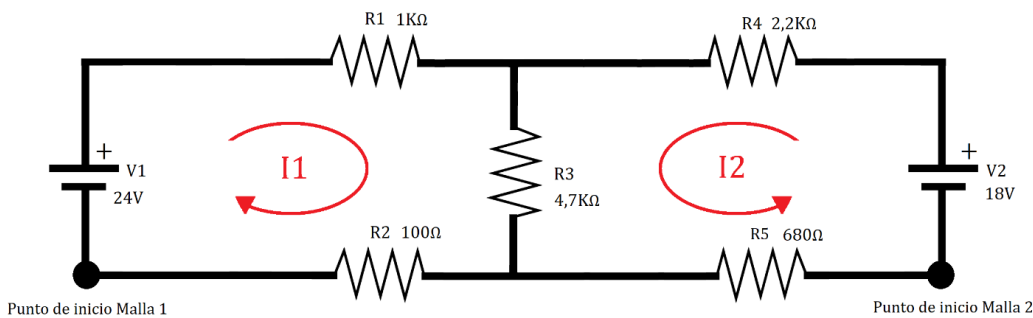
$$-V + V_{R1} + V_{R2} + V_{R3} + V_{R4} = 0$$

Y finalmente se expande la ecuación utilizando ley de Ohm, debe quedar así al terminar el planteo de la ecuación; en casa de ser conveniente, se puede factorizar la corriente (I) ya que en este caso es la misma en todas las resistencias.

$$-V + I \cdot R_1 + I \cdot R_2 + I \cdot R_3 + I \cdot R_4 = 0$$

- Ejemplo de cómo utilizar el teorema de mallas:

En la Figura 4-16 se presenta un circuito con dos mallas. Se encontrarán los valores de las corrientes que circulan en cada una.



Fuente: Elaboración propia

Figura 4-16 Circuito con dos mallas

Es importante tener en cuenta que a través de la R3 están circulando las dos corrientes, por ello se debe considerar este detalle al plantear las ecuaciones.

A continuación se plantean las ecuaciones de cada malla, para la malla 1 se realiza el recorrido en sentido horario y en la malla 2 en sentido anti horario.

Ecuación Malla 1:

$$-V_1 + V_{R1} + V_{R3} + V_{R2} = 0$$

Se expande la ecuación de la Malla 1:

$$-24V + I_1 \cdot R_1 + I_1 \cdot R_3 + I_2 \cdot R_3 + I_1 \cdot R_2 = 0$$

En el caso del voltaje en R3, se considera que a través de ella están circulando las corrientes I1 y I2, es por eso que en la ecuación se tiene $I_1 \cdot R_3 + I_2 \cdot R_3$.

Se factoriza, se reemplazan por los valores reales y se deja la ecuación lista para utilizar en un sistema de ecuaciones:

$$I_1(1K_{\Omega} + 4,7K_{\Omega} + 100_{\Omega}) + I_2 \cdot 4,7K_{\Omega} = 24V$$

Ecuación Malla 2:

$$-V_2 + V_{R4} + V_{R3} + V_{R5} = 0$$

Se expande la ecuación de la Malla 2 y se reemplazan los valores por los reales:

$$-18V + I_2 \cdot R_4 + I_2 \cdot R_3 + I_1 \cdot R_3 + I_2 \cdot R_5 = 0$$

Para el caso de R3 se tiene la misma consideración que en el caso de la malla 1.

Al factorizar se reemplazan por los valores reales y se deja la ecuación lista para utilizar en un sistema de ecuaciones:

$$I_1 \cdot 4,7K_{\Omega} + I_2(2,2K_{\Omega} + 4,7K_{\Omega} + 680_{\Omega}) = 18V$$

Finalmente se tiene el sistemas de ecuaciones con dos incógnitas: I_1 y I_2 .

$$I_1(1K_{\Omega} + 4,7K_{\Omega} + 100_{\Omega}) + I_2 \bullet 4,7K_{\Omega} = 24V$$

$$I_1 \bullet 4,7K_{\Omega} + I_2(2,2K_{\Omega} + 4,7K_{\Omega} + 680_{\Omega}) + = 18V$$

Para resolverlo se puede utilizar una calculadora científica o el método que resulte más conveniente, a modo de recordatorio se mencionan los métodos para resolver un sistema de ecuaciones con dos incógnitas, ellos son los métodos de igualación, sustitución, reducción y determinantes (cramer).

Si se ha resuelto de forma correcta el sistema de ecuaciones, los valores de corrientes encontrados deben ser los siguientes:

$$I_1 = 4,45mA$$

$$I_2 = -384\mu A$$

La corriente I_2 dio un resultado negativo, esto significa que está circulando en dirección inversa a la que se ha definido en la figura 4-16.

- Nodo eléctrico.

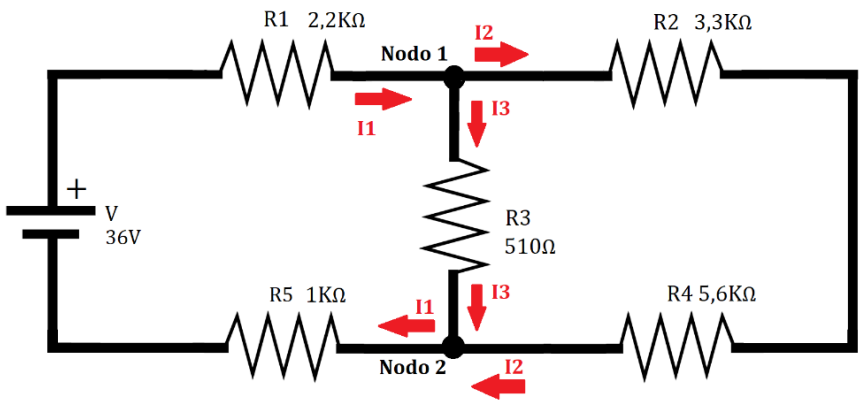
Es un punto común de un circuito en el que dos o más componentes (o fuentes de voltaje) están conectados. En el circuito se simboliza simplemente con un punto en donde se unen las conexiones.

- Ley de Corrientes de Kirchhoff o Teorema de Nodos.

Por definición, esta ley establece que “Las corrientes de entrada de un nodo son iguales a la suma de las corrientes de salida”. La ley sirve para obtener el voltaje en cada nodo; una vez obtenidos los voltajes se puede obtener las corrientes de malla.

En el momento en que se haga el análisis se debe escribir una ecuación por cada nodo en el circuito. Por lo tanto, para poder aplicar este teorema, es necesario que el circuito tenga como mínimo dos mallas.

Tal como en el teorema de mallas se asignan etiquetas a cada malla o a las corrientes (Corriente 1, Corriente 2, Malla 1 o Malla 2), aquí se ponen etiquetas a los nodos o también a los voltajes en los nodos como en circuito de la figura 4-17.



Fuente: Elaboración propia

Figura 4-17 Circuito con dos nodos

Para realizar el análisis, se utilizan las siguientes etiquetas en los nodos.

V_{N1} = Voltaje en el Nodo 1

V_{N2} = Voltaje en el Nodo 2

- Ecuaciones de Nodos:

Nodo 1:

$$I_1 = I_2 + I_3$$

Teniendo esta ecuación, se aplica ley de ohm para expandirla:

$$\frac{36V - V_{N1}}{2,2K\Omega} = \frac{V_{N1} - V_{N2}}{8,9K\Omega} + \frac{V_{N1} - V_{N2}}{510\Omega}$$

Ahora se despejan las variables y se simplifica la ecuación para insertarla en el futuro sistema de ecuaciones. La primera ecuación debe quedar así:

$$5,561V_{N1} - 4,561V_{N2} = 36V$$

Nodo 2

$$I_2 + I_3 = I_1$$

Se expande la ecuación con ley de ohm

$$\frac{V_{N1} - V_{N2}}{8,9K\Omega} + \frac{V_{N1} - V_{N2}}{510\Omega} = \frac{V_{N2}}{1K\Omega}$$

Se despejan las variables y se simplifica la ecuación

$$2,072V_{N1} - 3,072V_{N2} = 0V$$

Se insertan las dos ecuaciones y se plantea el sistema dos incógnitas: V_{N1} y V_{N2} .

$$5,561V_{N1} - 4,561V_{N2} = 36V$$

$$2,072V_{N1} - 3,072V_{N2} = 0V$$

Al igual que en el caso de mallas, para resolverlo se puede recurrir al método que sea más cómodo. Si el sistema es resuelto correctamente, se debe llegar a los siguientes valores.

$$V_{N1} = 14,49V$$

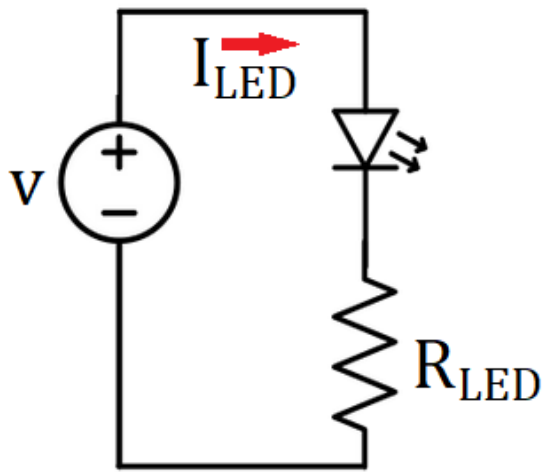
$$V_{N2} = 9,77V$$

Teniendo los voltajes en cada nodo, se pueden calcular los voltajes en las resistencias y todas las corrientes.

Guía 7: Ecuación para el cálculo de la resistencia de un led

Los LED no son simples luces, son componentes electrónicos llamados diodos, pero con la diferencia que emiten luz cuando están polarizados. La palabra LED son las iniciales del nombre del componente en inglés (Light Emitting Diode).

Durante la unidad 2 se enseñará más acerca del LED, por ahora solo se aprenderá a calcular la resistencia que necesita para funcionar.



Fuente: Elaboración propia

Figura 4-18 Circuito de polarización de un led

- Ecuación para calcular la resistencia de un LED:

$$R_{LED} = \frac{V - V_{LED}}{I_{LED}}$$

Donde:

- R_{LED} : Resistencia del LED
- V : Voltaje de polarizacion
- V_{LED} : Voltaje nominal del LED
- I_{LED} : Corriente nominal del LED

El voltaje de polarización corresponde al voltaje de alimentación del circuito.

El voltaje nominal del LED corresponde a la caída de tensión en el LED cuando está encendido, el valor se encuentra dado de fábrica.

La corriente nominal del LED corresponde a la corriente necesaria que debe circular a través del LED para que se encienda, este valor también viene dado de fábrica.

- Datos importantes

Siempre se debe utilizar una resistencia para polarizar un LED, de lo contrario se quemará.

El valor de I_{LED} y V_{LED} varía según el fabricante, el color y el tipo de led.

UNIDAD 2: COMPONENTES ELECTRÓNICOS Y HERRAMIENTAS

Guía 1: El Multitéster Digital

Será la herramienta principal (Figura 4-19) durante gran parte del curso, en jerga se le llama simplemente “Téster”. Dependiendo del modelo (y en muchos casos del valor), dependerá la cantidad de unidades de medida en que se podrá medir. Consta de una pantalla con un cristal LCD que en algunos casos tiene luz, un selector en el centro con el que se puede seleccionar lo que se desea medir y en la parte más baja los terminales en donde se conectan las puntas de prueba. En algunos téster se debe seleccionar el rango de medida de lo que se va a medir, es decir, que en el selector aparecerá el máximo medible, mientras que en otros no es necesario seleccionarlo ya que seleccionan el rango de forma automática.



Fuente: Elaboración propia

Figura 4-19 Multitester digital con selector manual (izquierda) y multitester digital con selector automático (derecha)

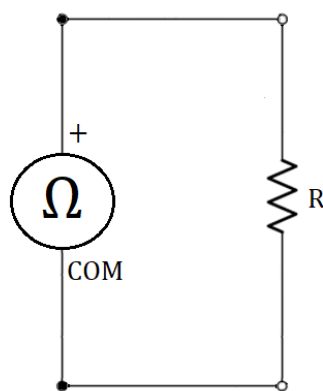
Las funciones incluidas en un multitéster son los siguientes:

- Ohmetro (Para medir resistencia)
- Voltímetro (para medir voltaje)
- Amperímetro (para medir corriente)
- Medidor de continuidad (para medir conductividad de conectores)
- Capacímetro (para medir capacitancia de condensadores)
- Termómetro (solo en algunos casos en los que vengan con termocupla, para medir temperatura)
- Frecuencímetro (para medir frecuencia)
- Medidor de ganancia Hfe (medidor de ganancia de transistores)
- Prueba de diodos (para probar el estado de diodos)

Con él se puede medir todo lo descrito en el curso 1 (excepto Potencia). Pero cada función del téster opera de forma diferente, es por eso que es necesario tener cuidado en cómo conectarlo al realizar mediciones.

Los instrumentos más importantes del téster son:

- Ohmetro:



Fuente: Elaboración propia

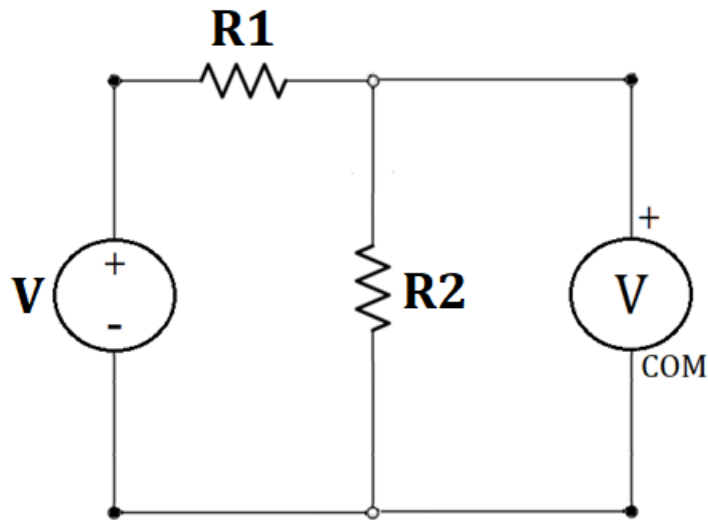
Figura 4-20 Circuito esquemático de cómo conectar un óhmetro

El óhmetro se conecta en paralelo con la resistencia que se desea medir (Figura 4-20), no se debe estar tocando los terminales de la resistencia porque se generará una lectura errónea. Para poder realizar la medición de forma más sencilla es mejor tener conectada la resistencia en un protoboard para que quede fija.

Lo más importante es que nunca se debe medir con el óhmetro teniendo el circuito energizado porque el instrumento resultará dañado.

De este mismo modo se puede conectar los cables al téster al hacer pruebas de continuidad, o también un capacímetro (al medir condensadores).

- Voltímetro:



Fuente: Elaboración propia

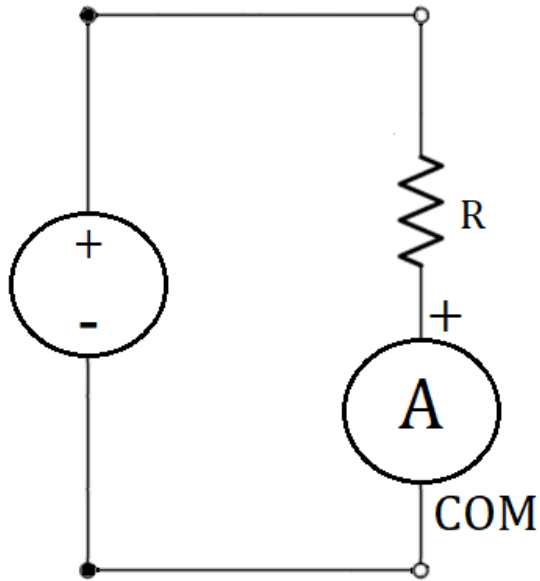
Figura 4-21 Voltímetro conectado en paralelo a la carga

El voltímetro es un instrumento que tiene una resistencia interna muy alta y para realizar la medición se debe conectar en paralelo a la resistencia.

Muy importante es que siempre que se mida voltaje y se desconozca la tensión en la carga medida, en caso de que el téster no posea selector de rango automático, siempre se debe seleccionar el rango más alto, de lo contrario si la tensión es muy elevada es probable que el téster sufra daños o inclusive podría explotar.

De este mismo modo se puede conectar un frecuencímetro en caso de realizar mediciones de frecuencia.

- Amperímetro:



Fuente: Elaboración propia

Figura 4-22 Amperímetro conectado en serie con la carga

El amperímetro tiene una resistencia interna muy baja (tiende a cero), para poder medir se debe conectar en serie con la carga (Figura 4-22), de este modo la corriente ingresa al instrumento a través del terminal positivo.

Nunca se debe medir corriente en paralelo, si se hace esto se generará un cortocircuito que quemará los fusibles del téster dejándolo inoperativo.

Si se desea medir corrientes sobre 200 mA, será necesario conectar el terminal positivo en el conector del téster que dice “10A” o “10A max” (ver figura 4-19).

- Otros instrumentos del téster

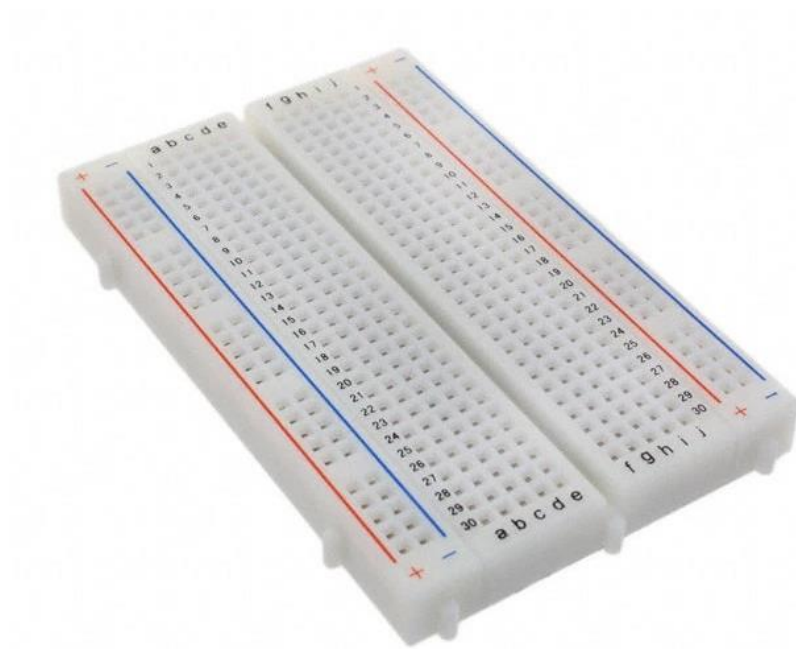
En caso de medir temperatura, hay algunos téster que vienen con termómetro, solo cuando venga con una termocupla y tenga bornes para conectarla.

Para medir ganancia, también el téster tiene bornes especiales en donde se podrá introducir los terminales del transistor.

Para utilizar la prueba de diodos, se debe conectar el positivo del téster con el positivo del diodo y lo mismo con los terminales negativos.

Guía 2: El Protoboard

Es una especie de tablero eléctrico en miniatura (Figura 4-23), en él se pueden armar circuitos en forma práctica. Tiene orificios en los que se puede insertar los terminales de los componentes y alambres de conexión. Los orificios del protoboard están internamente conectados. En casi todos, las conexiones internas son casi siempre las mismas, siguiendo una estructura determinada.



Fuente: Elaboración propia

Figura 4-23 Imagen referencial de un protoboard

Internamente tiene “bandas” metálicas (Figura 4-24) que conectan los puntos de conexión (orificios) entre ellos, de modo que se pueda establecer conexión entre los nodos del circuito armado. Las bandas están fijadas a una carcasa plástica blanca que actúa como aislante para que no exista contacto entre ellas.



Fuente: Elaboración propia

Figura 4-24 Imagen referencial de una banda metálica

Están diseñadas de ese modo para que al insertar los terminales de los componentes o alambres de conexión, estos sean sujetados por las bandas. En la figura 4-25 se muestra una vista en corte de un protoboard y de cómo están organizadas las bandas.



Fuente: Elaboración propia

Figura 4-25 Vista en corte de un protoboard

Las bandas laterales indicadas en color rojo y azul son continuas desde principio a fin, mientras que en el centro están formadas por líneas paralelas, cada una con 5 puntos. La división del centro es para poner los circuitos integrados.

Guía 3: La Resistencia y sus variantes

En la unidad anterior se ha hablado acerca de la resistencia y el código de colores. Existen más tipos de resistencias pero en diferentes presentaciones. En este curso solo se utilizará la resistencia y el potenciómetro (Figura 4-26), pero también se enseñará otro tipo de resistencias que se utilizan en circuitos de instrumentación.

- El potenciómetro

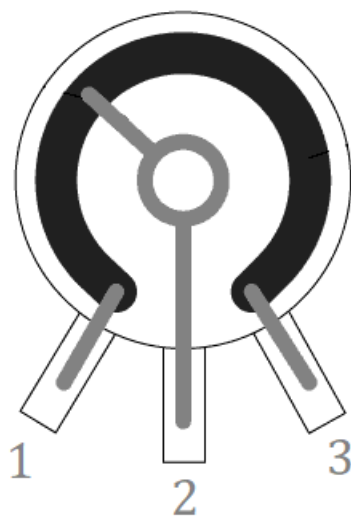
Es un tipo de resistencia que se puede hacer variar su valor varíe.



Fuente: Elaboración propia

Figura 4-26 Imagen referencial de un potenciómetro

Tiene un valor predeterminado de fábrica, entre los terminales de los extremos está el valor máximo de resistencia, pero al medir con un óhmetro entre el terminal del centro y cualquier otro de los dos, se puede notar que la resistencia varía al mover el cursor.



Fuente: Elaboración propia

Figura 4-27 Esquema de un potenciómetro por dentro

En la figura 4-27, el material resistivo o la resistencia total del potenciómetro aparece en color negro, los números 1, 2 y 3 indican los terminales.

Existen casos en los que es necesario realizar la variación de resistencia de un modo más preciso, es por ello que existen los potenciómetros multivoltas (Figura 4-28), que ofrecen una variación de resistencia mucho más lenta que los potenciómetros convencionales.



Fuente: Elaboración propia

Figura 4-28 Potenciómetro multivoltas

Su principio de funcionamiento es el mismo que el caso anterior, pero para hacer variar la resistencia tendremos que girar el tornillo dorado que tiene en su parte superior.

- LDR

El LDR (Light-Dependent Resistor) de la figura 4-29 es tipo de resistencia que varía su valor dependiendo de la intensidad de luz que reciba. Comúnmente es utilizado como sensor de luz.



Fuente: Elaboración propia

Figura 4-29 Imagen referencial de un LDR

- Sensor NTC o PTC

Es un tipo de resistencia que varía su valor según la temperatura, el sensor puede ser tipo NTC (Variación Negativa) o tipo PTC (Variación Positiva) y su presentación es como se muestra en la figura 4-30.



Fuente: Elaboración propia

Figura 4-30 Imagen referencial de un sensor NTC

- Dato importante

Al utilizar alguno de los sensores mencionados, será necesario diseñar un circuito para acondicionar la señal.

Guía 4: El diodo rectificador

En la primera unidad se ha enseñado que hay materiales conductores y aislantes, pero existe un tercer tipo de material: los semiconductores. Los elementos más utilizados son el Silicio y el Germanio.

El diodo rectificador (Figura 4-31) es un componente electrónico construido con Silicio (también existen de Germanio pero en el curso no será visto) y está diseñado para permitir el paso de la corriente en un solo sentido, es una especie de válvula electrónica anti retorno (aunque también tiene otros usos específicos).

Físicamente es de color negro con letras plateadas, la línea plateada en uno de sus extremos indica que ese es el terminal negativo (Cátodo) mientras que el otro es terminal positivo (Ánodo).



Fuente: Elaboración propia

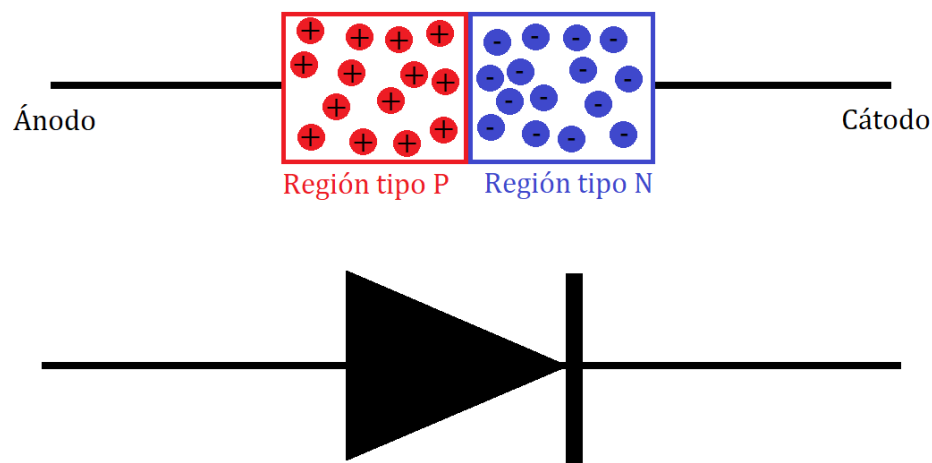
Figura 4-31 Diodo rectificador

- El diodo por dentro

Internamente está compuesto por dos regiones, una conectada con cada terminal y ambas compuestas de silicio pero con una gran diferencia:

1. La primera región tipo P tiene átomos modificados de modo que contengan más protones que electrones, por lo tanto son átomos con carga positiva.
2. La segunda región tipo N tiene átomos modificados, con más electrones que protones, por lo tanto son átomos con carga negativa.

Las dos regiones están unidas en el centro a través de una unión PN.

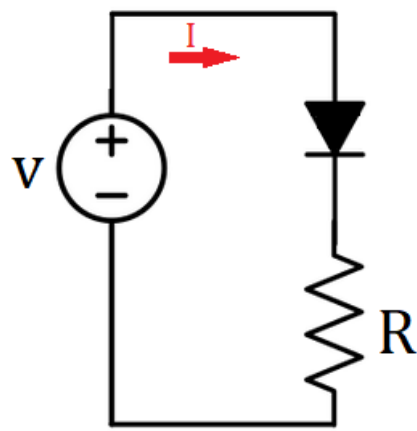


Fuente: Elaboración propia

Figura 4-32 Diodo por dentro y su símbolo

Se ha mencionado que el diodo es un dispositivo que permite la circulación de corriente en un solo, sentido y que al polarizarlo en sentido contrario se bloqueará, pero a continuación se explica cómo funciona.

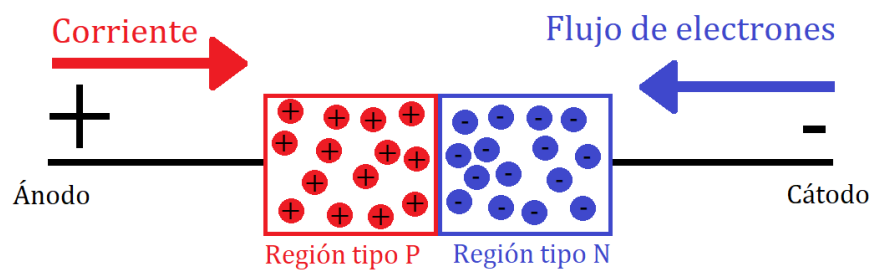
- Funcionamiento en polarización directa



Fuente: Elaboración propia

Figura 4-33 Ejemplo de circuito con un diodo en polarización directa

Análisis de lo que ocurre en el diodo cuando está polarizado de forma directa.



Fuente: Elaboración propia

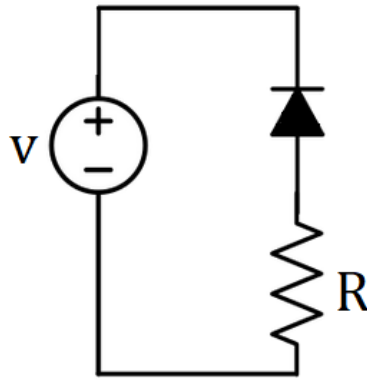
Figura 4-34 Funcionamiento de un diodo en polarización directa

Teniendo el diodo energizado se genera una polaridad positiva en su ánodo y una negativa en su cátodo, sabiendo que ambos terminales están conectados a las regiones internas del diodo y que esas regiones tienen material con su polaridad modificada ocurre lo que se muestra en la figura 4-34.

Como los polos opuestos se atraen y hay una mayor cantidad de electrones en la región tipo N, estos empiezan a moverse a la región tipo P produciéndose flujo de electrones desde la región tipo N hacia la región tipo P por lo tanto se produce un flujo de cargas positivas (corriente) hacia la región tipo N.

Para que pueda fluir corriente a través del diodo se requiere una tensión mínima de 0,7v (en caso de diodos de silicio), de lo contrario no conducirá.

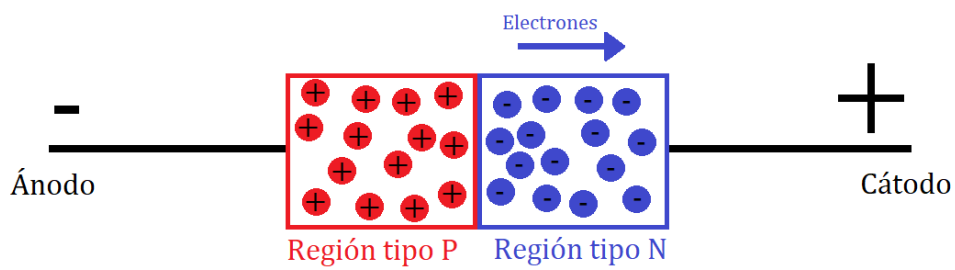
- Funcionamiento en polarización inversa



Fuente: Elaboración propia

Figura 4-35 Circuito con un diodo en polarización inversa

Al realizar el mismo análisis que el caso anterior, pero con la diferencia que no se produce circulación de corriente, por eso es que no aparece dibujada en la figura 4-36



Fuente: Elaboración propia

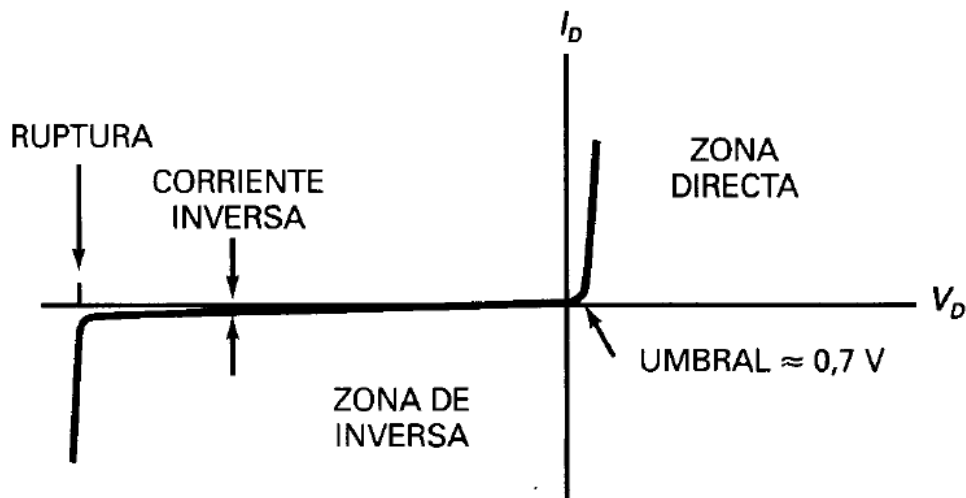
Figura 4-36 Funcionamiento de un diodo en polarización inversa

Al polarizar en forma inversa, se produce una polaridad positiva en el Cátodo y los electrones intentarán ir hacia allá, pero en este caso los electrones no tienen espacio para saltar, porque el material con espacio disponible se encuentra en la región tipo P, por lo tanto el diodo se bloquea y no existe corriente ni flujo de electrones.

- Datos importantes

El diodo comienza a conducir a 0,7v, después de eso la corriente se dispara, porque lo que es necesario limitar la corriente que circula a través de él.

En polarización inversa, el diodo soporta hasta una tensión determinada como “tensión de ruptura”, al exceder ese voltaje inverso, el diodo conducirá en forma inversa, pero se destruirá.



Fuente: Elaboración propia

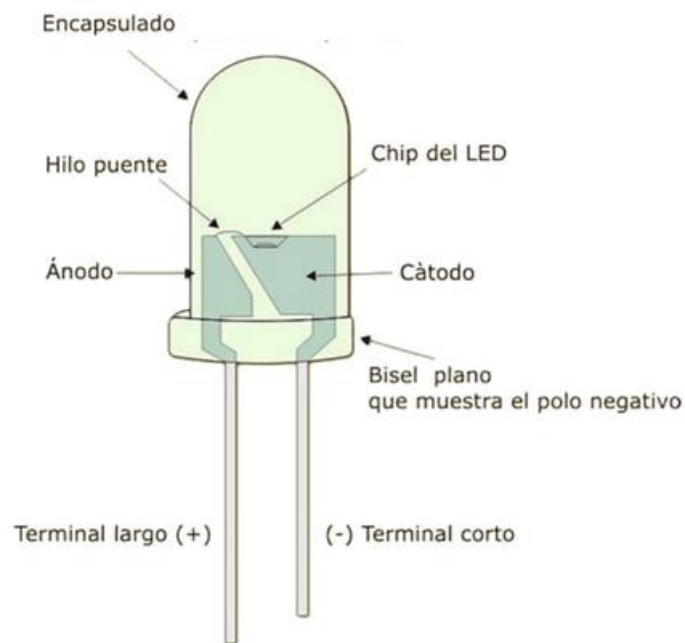
Figura 4-37 Curva de voltaje vs corriente de un diodo

Guía 5: El LED

El LED tiene el mismo principio de funcionamiento que el diodo rectificador, conduce la corriente en un solo sentido, pero la diferencia es que emite luz, y su voltaje necesario para activarlo es distinto, también soporta menos corriente.

En la primera unidad se ha enseñado que existe una ecuación para polarizar un LED, dado que al exceder la corriente necesaria, el LED se quemará. El voltaje del led (V_{LED}) es fijo, pero dependiendo de cuál sea su color será distinto ese voltaje.

En la actualidad existen muchos tipos de leds, en este curso solo se verán leds difusos y de alta luminosidad.



Fuente: Elaboración propia

Figura 4-38 Imagen referencial de un LED e indicaciones de sus terminales.

Los leds difusos son muy utilizados, generalmente en cursos de aprendizaje, pero más comúnmente en cualquier dispositivo, su función es ser indicador de que el dispositivo se encuentra energizado u operativo.

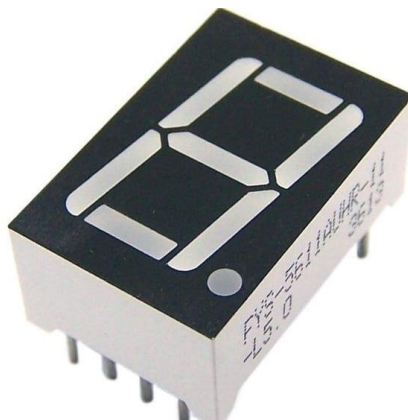


Fuente: Elaboración propia

Figura 4-39 Imagen referencial de un led de alta luminosidad

Este tipo de LED no es tan antiguo como el difuso, se utiliza bastante debido a su eficiencia energética, las aplicaciones más comunes actualmente es en semáforos o también en juegos de luces, en algunos dispositivos han reemplazado a los leds difusos.

- Otras presentaciones de leds



Fuente: Elaboración propia

Figura 4-40 Display de 7 segmentos

Los displays de 7 segmentos internamente tienen leds, cada segmento del número es un led. Para poder utilizarlo se debe usar un juego de resistencias y algunos circuitos integrados.



Fuente: Elaboración propia

Figura 4-41 Led hexagonal multichip

En los últimos años se inventaron los leds multichips, consisten en que hay varios leds dentro de un mismo led, al utilizarlos se debe acompañar (además de resistencias) de disipador de temperatura, porque se calientan cuando están en funcionamiento.



Fuente: Elaboración propia

Figura 4-42 Led SMD

Los componentes SMD van directamente soldados en las placas electrónicas a ras de piso, estos leds son los que están en nuestra plaquita de Arduino UNO.

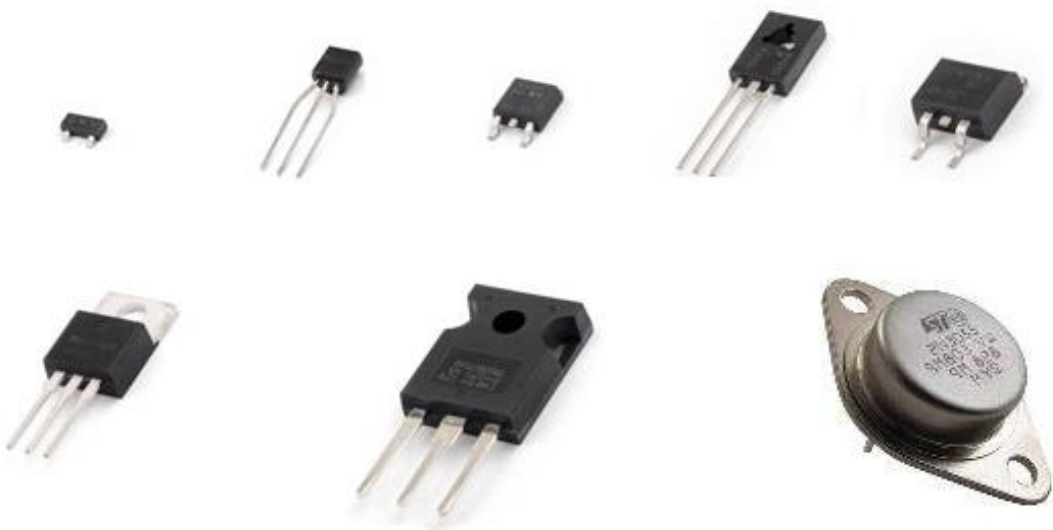
Guía 6: El Transistor

Es un componente electrónico que se inventó en el año 1947 y supuso un gran avance en la electrónica. En la actualidad se encuentran presente en todo dispositivo electrónico.

La palabra transistor viene del inglés, de la contracción de las palabras transfer resistor (transferencia de resistencia), significa que es un dispositivo que controla la resistencia eléctrica.

Más allá del significado de la palabra, el transistor tiene dos usos: Amplificador de señales o interruptor.

Tiene 3 terminales: Base, Colector y Emisor. Físicamente tiene muchas presentaciones, pero su tamaño está directamente relacionado con la potencia, mientras más pequeño, manejará menos potencia y lo mismo en el caso contrario.



Fuente: Elaboración propia

Figura 4-43 Distintos encapsulados de transistores.

Los transistores más pequeños (que manejan potencias bajas) se les llaman “transistores de pequeña señal”, mientras que a los más grandes se les llama “transistores de potencia”.

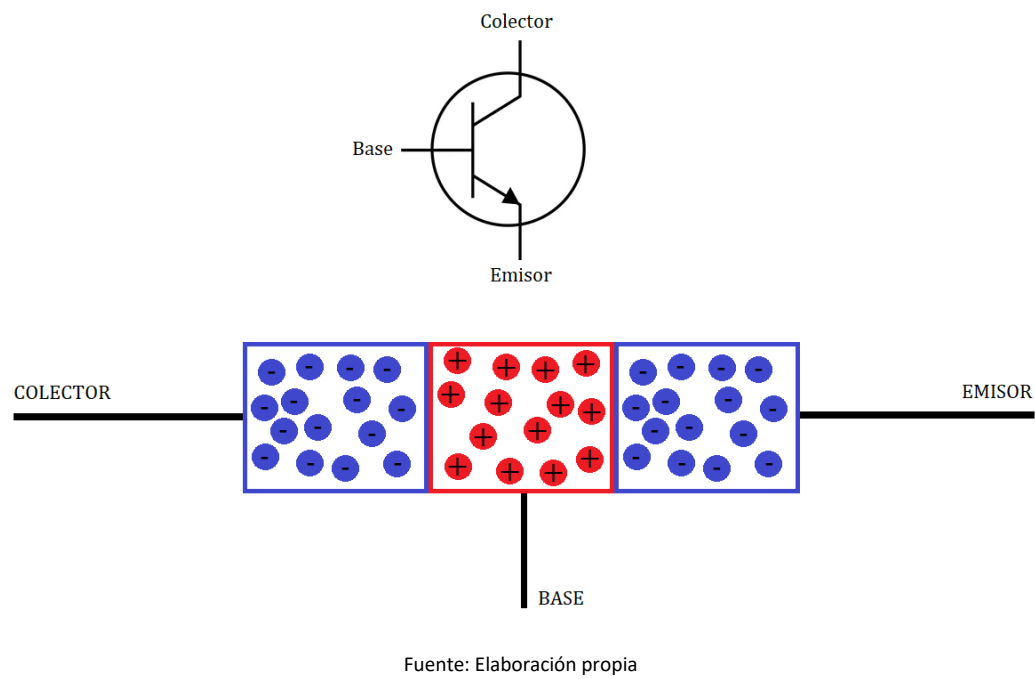


Figura 4-44 Dibujo de un transistor NPN por dentro junto con su símbolo

El transistor NPN es el más utilizado y más común. Sus regiones son Negativo-Positivo-Negativo.

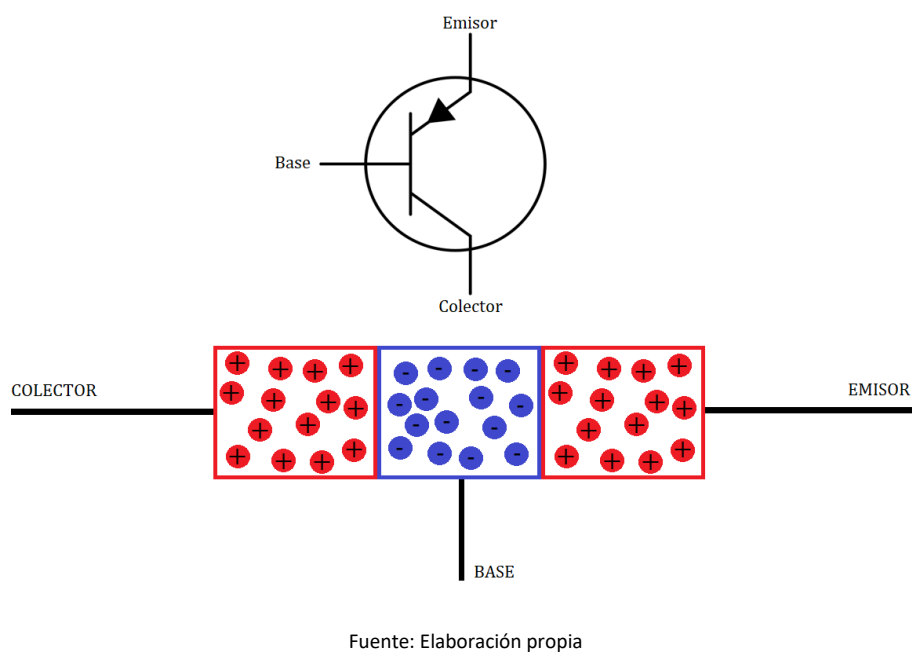


Figura 4-45 Dibujo de un transistor PNP por dentro junto con su símbolo

El transistor PNP es menos común y menos utilizado, aunque no por eso es menos importante. Sus regiones son Positivo-Negativo-Positivo.

Teniendo los esquemas, surge la pregunta sobre cuál es la diferencia y cómo saber cuál utilizar, la diferencia entre uno y otro es el sentido en el que circulan las corrientes, mientras que para activar el transistor NPN requiere aplicar una corriente positiva en la base, en el caso del PNP es necesaria una corriente negativa.

- Zonas de operación de un transistor

Al utilizar el transistor como amplificador de señal se encontrará operando en zona activa.

Al utilizar el transistor como interruptor, lo estará operando en zona de saturación (interruptor cerrado) o en zona de corte (interruptor abierto).

Zona Activa: el transistor tiene voltaje y corriente equilibrados y su respuesta es lineal. En esta zona de trabajo existe un parámetro llamado H_{fe} (en jerga electrónica se le llama Beta β) que corresponde a la ganancia de corriente. Este parámetro relaciona la corriente de la base (necesaria para activar el transistor) versus la corriente del colector (corriente de salida).

Zona de Saturación: el transistor tiene muy poco voltaje, pero una gran corriente. En este caso el transistor está actuando como un interruptor cerrado, permitiendo el paso de la corriente.

Zona de Corte: el transistor tiene casi todo el voltaje de polarización (V_{cc}) y la corriente que circula a través de él es cero, en esta zona el transistor actúa como un interruptor abierto, bloqueando el paso de la corriente.

Al utilizar el transistor en zonas de corte y saturación, su respuesta será no lineal.

- El transistor como amplificador

Generalmente en este tipo de aplicaciones, se utilizan transistores de pequeña señal debido a su rápida respuesta en frecuencia, aunque también se pueden utilizar transistores de potencia para ciertas aplicaciones, un ejemplo son los circuitos de audio.

Generalmente el proceso de amplificar señales se realiza por etapas, primero una etapa de pequeña señal en el que la señal de entrada es muy pequeña (muchas veces pueden ser incluso microvolts) para luego introducirla a una segunda etapa de potencia.

- El transistor como interruptor

En este caso es muy común el uso de transistores de potencia, porque generalmente se utilizan para controlar cargas que consumen corrientes grandes, algunos ejemplos son en puente H (para controlar motores de corriente continua), para circuitos de audio (controlar parlantes) o para activar relés que controlan otros dispositivos.

- Datos importantes

En los transistores con encapsulado metálico, el colector está conectado directamente a la carcasa.

Para determinar gráficamente en qué zona está trabajando el transistor se utiliza un término llamado “Punto Q”, dependiendo de su ubicación en un gráfico de corriente vs voltaje es en donde estará trabajando el transistor.

Existe una cuarta zona llamada “zona prohibida” y consiste en que el transistor tiene mucho voltaje y una gran cantidad de corriente circulando a través de él, por lo tanto está disipando una gran potencia, esta zona es peligrosa ya que si el transistor opera aquí por un periodo de tiempo importante se quemará (por eso el nombre de zona prohibida).

Guía 7: La bobina y el condensador

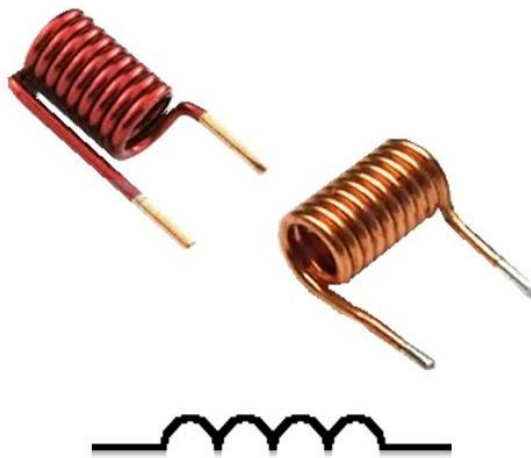
La bobina y el condensador son componentes electrónicos capaces de almacenar energía. La bobina almacena energía en forma de campo magnético mientras que el condensador lo hace en forma de campo eléctrico.

Para entender cómo funcionan se analizará en detalle cada componente.

- La bobina:

La bobina o inductor, físicamente está construida por un conductor (puede ser un cable o alambre) enrollado de distintas formas a un núcleo, la forma de enrollado al igual que el tipo de núcleo da lugar a que existan distintos tipos de bobinas.

Así como el valor de las resistencias se miden en ohm, el valor de inductancia de las bobinas se miden en Henry (en jerga electrónica se le dice Henrios). Pero es importante destacar que las bobinas no se comportan igual que las resistencias, son componentes totalmente diferentes.



Fuente: Elaboración propia

Figura 4-46 Bobina con núcleo de aire y su símbolo



Fuente: Elaboración propia

Figura 4-47 Bobina con núcleo de hierro y su símbolo



Fuente: Elaboración propia

Figura 4-48 Bobina con núcleo de ferrita y su símbolo

Las bobinas tienen un comportamiento diferente cuando se aplica tensión continua que cuando se aplica tensión alterna. Para este curso solo se verá el comportamiento en corriente continua.

- Comportamiento de la bobina en corriente continua

Cuando circula una corriente continua a través de ella, se comporta como un cortocircuito, al no haber variación de corriente, tampoco varía el campo magnético interno, por lo tanto no se genera ningún voltaje inverso inducido.

- El condensador

Al igual que la bobina, el condensador también se comporta de forma diferente al aplicar corriente continua que al aplicar corriente alterna. Físicamente existen muchos tipos de condensadores, entre los tipos que existen podemos nombrar los condensadores de aire, de papel, de cerámica, de polyester o electrolíticos.

Su unidad de medida es el Faradio, a pesar de ello, los condensadores se encuentran siempre en unidades de micro o nano Faradios.

El único condensador que tiene polaridad es el electrolítico, siempre debe ser polarizado de forma correcta (voltaje positivo en terminal positivo y voltaje negativo o tierra en terminal negativo) al utilizarlo en aplicaciones con corriente continua, en caso de polarizarlo en forma inversa se destruirá (explotará).



Fuente: Elaboración propia

Figura 4-49 Imagen referencial de un condensador de 4700 μ F

El condensador electrolítico tiene una banda gris que indica el terminal negativo, de este modo se puede saber cómo conectarlo correctamente.



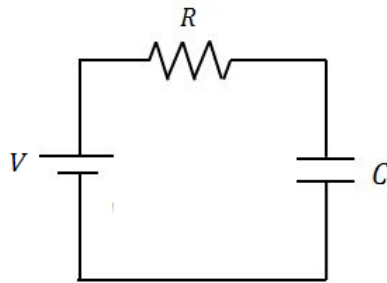
Fuente: Elaboración propia

Figura 4-50 Imagen referencial de un condensador cerámico

Este tipo de condensadores de la figura 4-50 no tiene polaridad, puede ser conectado sin una posición determinada.

- Comportamiento del condensador en corriente continua

Al aplicar corriente continua en un condensador, en un principio y si se encuentra totalmente descargado, se comportará como un cortocircuito y comenzará a cargarse, pero una vez que esté totalmente cargado se comportará como un circuito abierto impidiendo el paso de la corriente. Para explicar este funcionamiento existe una ecuación llamada “ecuación de carga y descarga”. Esta ecuación es solo aplicable en circuito del tipo RC (Resistencia en serie con un condensador).



Fuente: Elaboración propia

Figura 4-51 Circuito RC

Ecuación de carga

$$V_C = V_F (1 - e^{-\frac{t}{\tau}})$$

Ecuación de descarga

$$V_C = V_i \cdot e^{-\frac{t}{\tau}}$$

Donde

V_C = Voltaje del condensador

V_F = Voltaje de la fuente

V_i = Voltaje inicial del condensador ($t=0$)

t = tiempo en segundos

τ (Tau) = $R \cdot C$ (Valor de la multiplicación de la resistencia en ohm y del condensador en Faradios)

Estas ecuaciones se aplican cuando el voltaje es continuo.

- Información muy importante

Existen muchos más análisis del comportamiento de estos componentes, pero requieren un análisis con matemáticas más avanzadas en el que entran en juego las integrales, derivadas y también el plano de Laplace.

Todo lo mencionado en esta mini guía es para aprender nociones básicas acerca de cómo funcionan estos componentes.

Guía 8: El circuito integrado

El circuito integrado (conocido comúnmente como chip), internamente tiene estructuras de circuitos electrónicos muy pequeños, están hechos generalmente de silicio (material semiconductor) por ello es que la circuitería interna tiene muchos transistores.

Externamente tienen pines (muchas veces se les llama patitas) para conectarlos y el encapsulado que protege al circuito interno es de color negro.

Se inventaron a finales de los años 1940 siendo un gran avance, ya que supuso la reducción del tamaño de los circuitos y una rebaja considerable en los costos.

Una de las desventajas es que se pueden medir con el multímetro para saber su estado, la única forma es probarlo en un circuito.



Fuente: Elaboración propia

Figura 4-52 Imagen referencial de un NE555

Uno de los circuitos integrados más usados es el NE555 (figura 4-52), también conocido como LM555 o simplemente 555. Dependiendo de la aplicación puede ser usado como monoestable o astable.

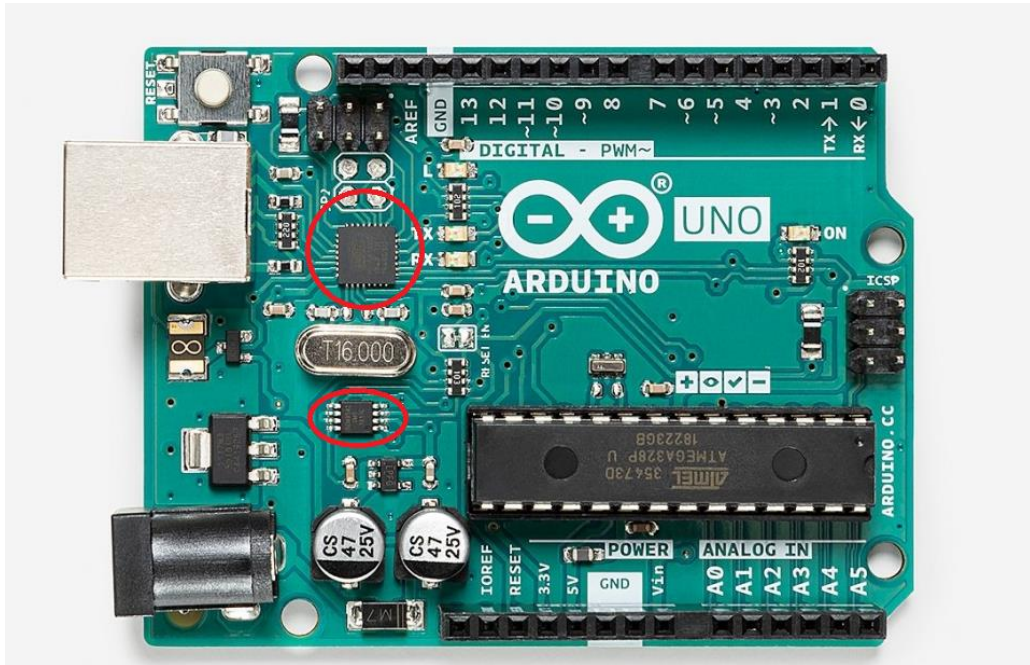
La cantidad de pines es variable dependiendo del circuito integrado.

Al igual que el diodo y el transistor, también es posible encontrar circuitos integrados del tipo SMD, muy comunes en cualquier dispositivo electrónico en la actualidad.

Para saber cómo funciona o para saber la función de cada uno de sus pines se debe recurrir al datasheet.

En el datasheet se encuentra la información del pin positivo y del pin negativo o tierra. Generalmente el pin positivo está indicado como Vcc y el pin negativo como GND.

Se debe tener especial cuidado de conectar la alimentación de forma correcta ya que si se polariza en forma inversa se quemarán.



Fuente: Elaboración propia

Figura 4-53 Placa referencial de Arduino UNO

En la figura 4-53 se encuentran encerrados en rojo circuitos integrados tipo SMD presentes en la placa de Arduino UNO.

Otra característica muy importante es que existen circuitos integrados llamados “microcontroladores”, y funcionan de un modo completamente distinto al resto.

Los microcontroladores son chips que pueden ser programados, de este modo se puede evitar realizar grandes cálculos o grandes circuitos para lograr la aplicación que se desea desarrollar.

El microcontrolador permite convertir un circuito de control alambrado a uno de control programado.

Arduino UNO es una placa de pruebas que utiliza el microcontrolador “ATmega328P” por lo que en el curso se aprenderá a programar el chip mencionado en lenguaje C++.

Un detalle muy importante a considerar es que los microcontroladores no están diseñados para manejar grandes corrientes, su aplicación es solo para circuitos de control (corrientes muy pequeñas). Si se desea controlar cargas que consuman corrientes elevadas, es necesario utilizar un circuito de adaptación o un driver de corriente.

Guía 9: Reguladores de Voltaje

Los reguladores de voltaje son componentes diseñados para estabilizar voltaje. Existen para regular tensión positiva y también para tensión negativa.

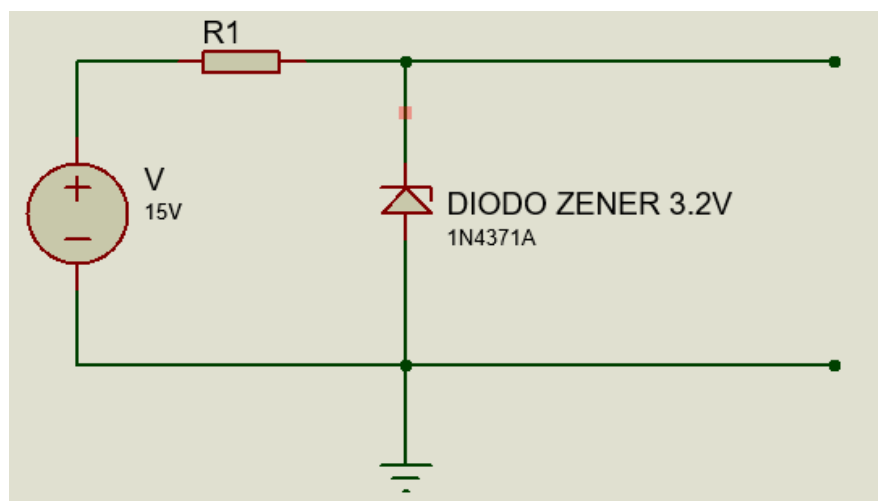
El objetivo del regulador es tomar una tensión de entrada inestable y entregar en la salida un voltaje sin variaciones (constante). También se pueden usar como limitadores de voltaje, de modo que al tener una tensión de entrada más elevada de lo que necesario, con un regulador se puede adaptar la tensión.

El uso más común de los reguladores de voltaje es en fuentes de voltaje reguladas.

- Tipos de reguladores de voltaje

1.- Diodo Zener

El diodo Zener es un tipo de diodo que funciona como regulador de voltaje al polarizarlo de forma inversa, por el contrario, si se polariza de forma directa funciona como diodo rectificador.



Fuente: Elaboración propia

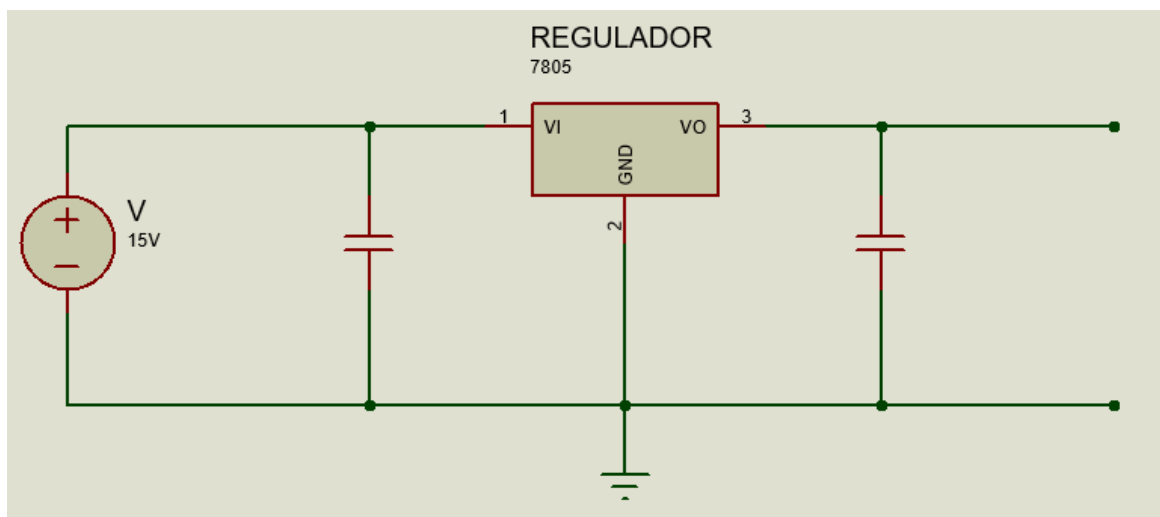
Figura 4-54 Circuito regulador de voltaje con diodo Zener

En la figura 4-54 se presenta un circuito con un diodo Zener que regula la tensión. Entre los terminales del lado derecho habrá 3,2 volts, el resto del voltaje caerá en la resistencia R1. Siempre que se utilice este circuito se debe poner una resistencia como la R1 en serie con el diodo Zener para limitar la corriente, de lo contrario el diodo Zener podría resultar dañado o quemado.

2.- Circuitos integrados 78xx

Los circuitos integrados tipo 78XX constan de 3 pines (Entrada, Salida y Tierra) y al igual que el diodo Zener, son capaces de tomar una tensión de entrada inestable o con valores más altos de lo necesitado y regularla.

Las letras XX corresponden a la tensión de salida que entrega el regulador, de este modo están los circuitos integrados 7803 (3v), 7805 (5v), 7806 (6v), 7808 (8v), 7809 (9v), 7810 (10v), 7812 (12v), 7815 (15v), 7818 (18v), 7824 (24v) y 7833 (33v).



Fuente: Elaboración propia

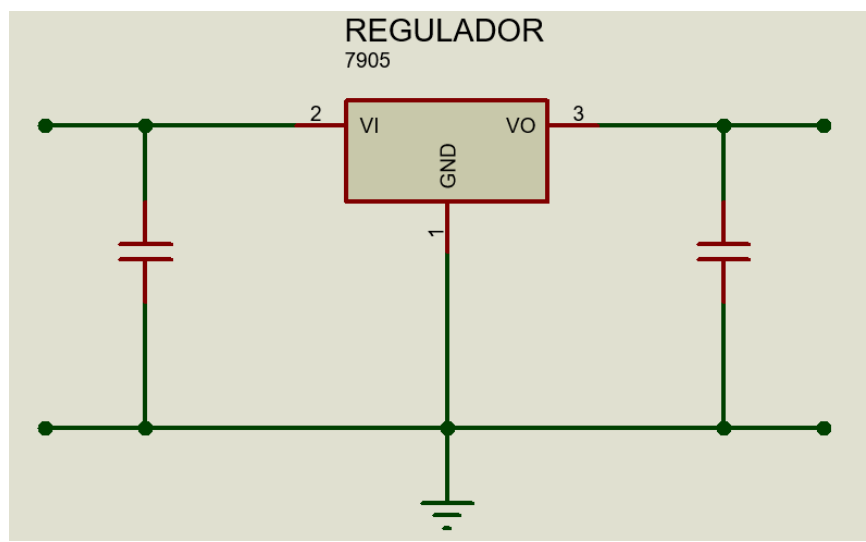
Figura 4-55 Circuito con regulador de voltaje 7805

En la figura 4-55 se presenta un circuito con un regulador 7805 que regula la tensión a 5 volts en la salida, los pines pueden ser distintos dependiendo del fabricante, por ello siempre se debe observar el datasheet para saber la función de cada pin.

Se utilizan condensadores en la entrada y la salida para eliminar alguna transiente o ruido que podría dañar el regulador.

3.- Circuitos integrados 79xx

Los reguladores tipo 79XX tienen el mismo principio de funcionamiento que los 78XX, la única diferencia es que regulan tensión negativa, por lo tanto el voltaje de entrada debe ser negativo.



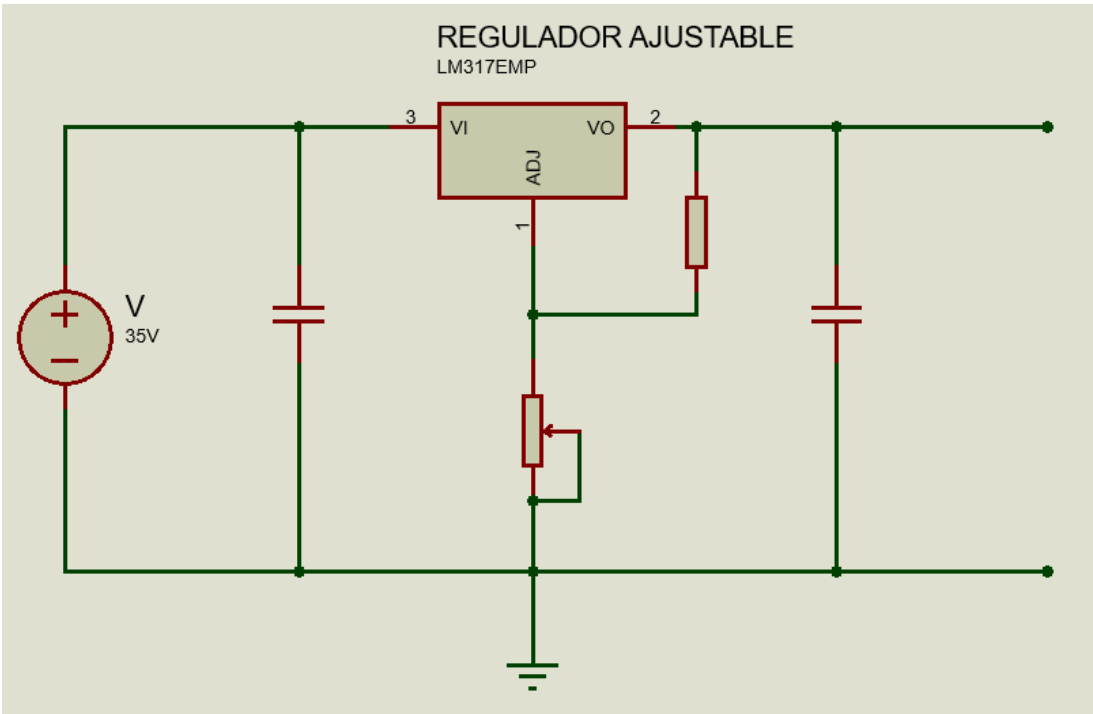
Fuente: Elaboración propia

Figura 4-56 Circuito con regulador de voltaje 79XX

En la figura 4-56 se tiene un circuito con regulador 7905, en los extremos del lado derecho habrá -5 volts.

4.- Regulador LM317

Este regulador funciona de forma similar a los 78XX y 79XX, la gran diferencia es que es un regulador ajustable, es decir, se puede variar el voltaje de salida en función de la aplicación.



Fuente: Elaboración propia

Figura 4-57 Circuito con regulador LM317

En la figura 4-57 se muestra el regulador LM317 en un pequeño circuito, en los terminales del lado derecho está el voltaje de salida regulado, para variar el voltaje solo se debe ajustar el potenciómetro conectado en el pin de ajuste. Los valores de voltaje que puede entregar dependerán del fabricante, pero generalmente van desde 1,2 a 37 volts.

- Información muy importante

Es necesario tener especial cuidado de conectar la alimentación de forma correcta ya que al polarizar en forma inversa se quemarán.

Guía 10: Compuertas Lógicas

Las compuertas lógicas son circuitos hechos con transistores, están diseñados para realizar operaciones matemáticas básicas (suma, multiplicación y negación). Solo operan con señales booleanas (estados lógicos 0 y 1).

- Estados Lógicos

Estado lógico 0: representa un 0 lógico, es decir, 0 volts o ausencia de tensión.

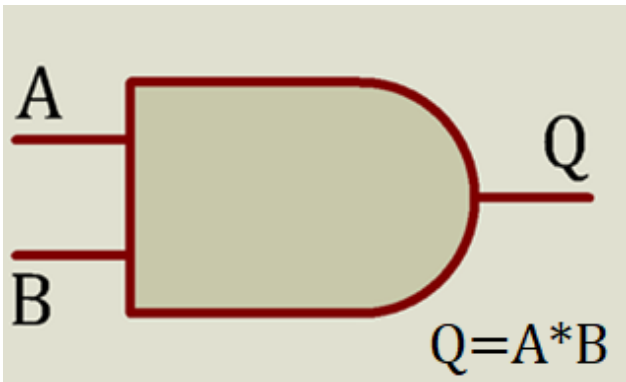
Estado lógico 1: representa un 1 lógico (presencia de tensión). Generalmente corresponde a 5 volts, pero algunos circuitos tipo CMOS pueden operar con voltajes entre 4 y hasta 18 volts.

Para saber cómo funciona cada compuerta lógica se puede ver su tabla de verdad, que contiene todas las posibles combinaciones de valores en su entrada y todos los resultados de salida.

- Compuertas lógicas

1.- Compuerta AND (Y)

Realiza la multiplicación de los valores de entrada (A y B) y el resultado lo entrega en la salida Q



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

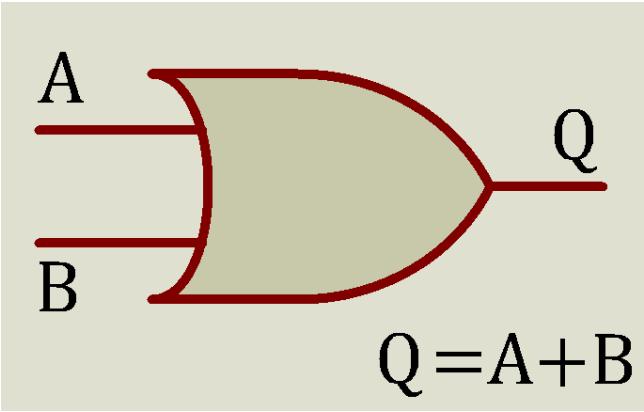
Fuente: Elaboración propia

Figura 4-58 Compuerta AND con su tabla de verdad

Para que en la salida haya un 1, ambas entradas deberán tener un 1, de lo contrario la salida será 0.

2.- Compuerta OR (O)

Realiza la suma de los valores de entrada (A y B) y el resultado lo entrega en la salida Q.



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

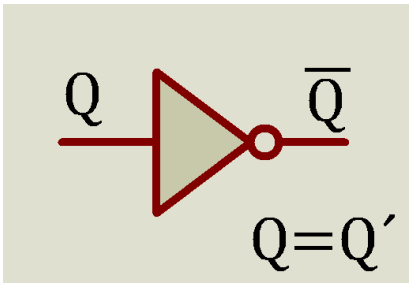
Fuente: Elaboración propia

Figura 4-59 Compuerta OR con su tabla de verdad

Para que en la salida haya un 1 solo es necesario activar cualquiera de las entradas o ambas.

3.- Compuerta NOT

Realiza la negación del valor que haya en la entrada entregándolo en la salida.



Q	Q'
0	1
1	0

Fuente: Elaboración propia

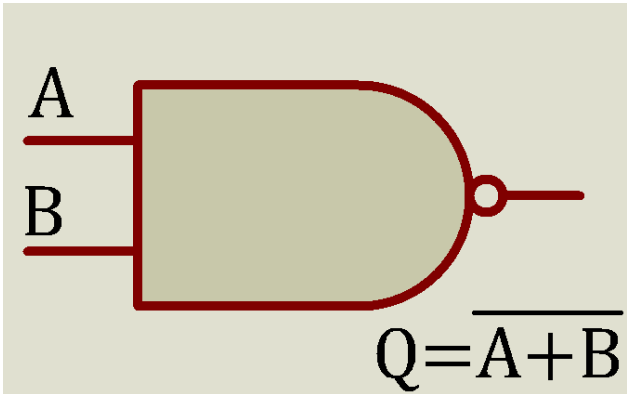
Figura 4-60 Compuerta NOT con su tabla de verdad

Si el valor de entrada es Q en la salida el valor será Q negado.

Cuando un valor se encuentre negado se anotará con una línea sobre él o con un apóstrofo (comilla).

4.- Compuerta NAND

Realiza la multiplicación de los valores de entrada (A y B), luego niega el resultado y lo entrega en la salida.



A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

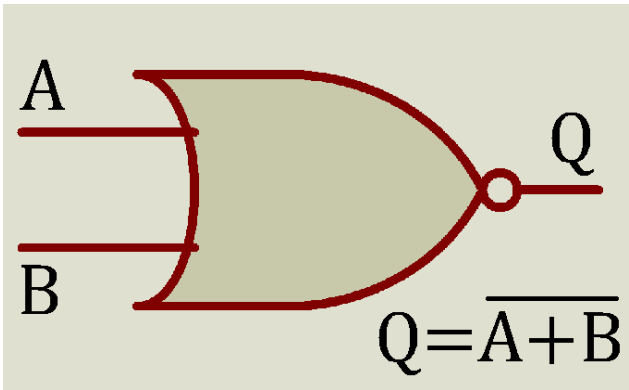
Fuente: Elaboración propia

Figura 4-61 Compuerta NAND con su tabla de verdad

Esta compuerta equivale a tener una compuerta AND y luego una compuerta NOT.

5.- Compuerta NOR

Realiza la suma de las entradas (A y B), luego niega el resultado y lo entrega en la salida



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

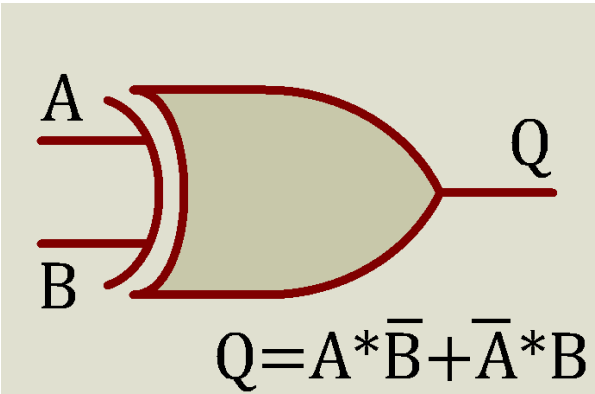
Fuente: Elaboración propia

Figura 4-62 Compuerta NOR con su tabla de verdad

Esta compuerta equivale a tener una compuerta OR y luego una compuerta NOT.

6.- Compuerta XOR

La compuerta XOR o también llamada OR exclusiva, funciona entregando un 0 cuando sus valores de entrada tienen el mismo valor y entregando un 1 cuando sus entradas son distintas.



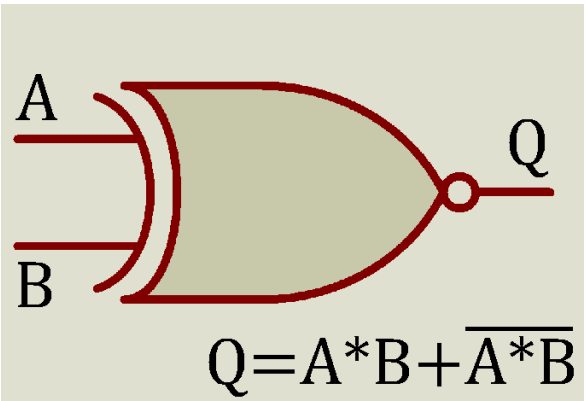
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Fuente: Elaboración propia

Figura 4-63 Compuerta XOR con su tabla de verdad

7.- Compuerta XNOR

Esta compuerta se comporta en forma inversa a la XOR, cuando sus entradas son iguales entrega un 1 y cuando sus salidas son distintas entrega un 0.



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

Fuente: Elaboración propia

Figura 4-64 Compuerta XNOR y su tabla de verdad

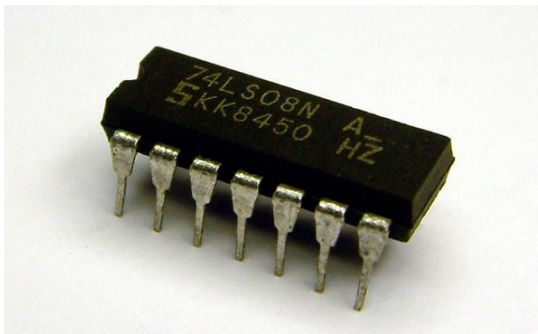
Equivale a tener una compuerta XOR y luego una compuerta NOT.

- Encapsulado físico

La presentación de las compuertas lógicas es en circuitos integrados, la cantidad de pines puede variar según la cantidad de compuertas que tenga. No se debe pasar por alto que además de los pines de entrada y salida, también están los pines de alimentación.

Para saber qué pines son entradas, salidas y de alimentación, se debe revisar el datasheet.

Se debe tener especial cuidado de conectar la alimentación de forma correcta ya que al polarizar en forma inversa se quemarán.



Fuente: Elaboración propia

Figura 4-65 Circuito integrado 7408 de compuertas lógicas tipo AND.

UNIDAD 3: CONCEPTOS DE MICROCONTROLADORES

Guía 1: Sistemas de Numeración

Al hablar de un sistema de numeración, se hace referencia a un reglamento en el que se define la cantidad de números válidos y sus símbolos. Un ejemplo es el sistema decimal (el usado siempre), que contiene 10 números válidos (desde 0 a 9) y sus símbolos son los mismos números.

Un ejemplo de un sistema distinto al decimal es el que se ha visto en las compuertas lógicas, que utilizan un sistema de numeración binario en el que sólo existen 2 números válidos (0 y 1).

Otro sistema de numeración que existe pero ya prácticamente no se encuentra en uso es el octal, consiste en 8 números válidos que van desde el 0 hasta el 7.

Un sistema bastante utilizado es el hexadecimal, consiste en 16 números válidos, pero como son más de 10 números, los otros 6 restantes se representan con letras (A, B, C, D, E y F). Este sistema es probable que sea utilizado al programar el Arduino.

Decimal	Binario	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Fuente: Elaboración propia

Figura 4-66 Tabla con sistema decimal y sus equivalentes en otros sistemas.

- Información importante

Existen algunas calculadoras científicas en las que se puede hacer conversión de números de un sistema a otro, el modo de conversión suele llamarse "Base N"

También se puede hacer la conversión de un número de un sistema a otro en forma manual, para ello se debe investigar el teorema fundamental de numeración.

Guía 2: Lógicas TTL y CMOS

Desde que existe la electrónica digital, se han desarrollado distintas tecnologías para la creación de circuitos integrados, a pesar de que existen muchas más. En esta guía solo serán vistas las tecnologías TTL y CMOS que son las más importantes.

- Lógica TTL

Las siglas TTL significan Transistor-Transistor Logic, o en español: “Lógica Transistor a Transistor”.

La tecnología TTL significa que los circuitos integrados digitales están contruidos en base a transistores tipo NPN o PNP.

Características principales de la tecnología TTL:

- Su tensión de alimentación es de 5V, aunque también puede aceptar variación de aproximadamente un 5%, al exceder los 5V el circuito integrado se quemará.
- El nivel de estado lógico 1 son 5V (igual a la tensión de alimentación), para el estado lógico 0 puede ser desde 0V hasta 1V.
- Una de las desventajas de esta tecnología es su consumo, ya que al estar hecha de transistores, necesita circulación de corriente para funcionar.
- Existen variaciones de la tecnología TTL en la que además de transistores emplean otros componentes (diodos) para mejorar la velocidad o intentar disminuir el consumo.

Es importante recordar que a pesar de que los circuitos de lógica TTL están hechos con transistores, no están diseñados para manejar grandes corrientes; si se busca controlar algún dispositivo que tenga un alto consumo (por ejemplo: ampolletas, motores o parlantes) y se desea controlarlo con compuertas lógicas o algún otro circuito combinacional, habrá que usar un driver de potencia o adaptador de impedancias.

- Lógica CMOS

Las siglas CMOS significan Complementary Metal-Oxide-Semiconductor que en español significa “Semiconductor Complementario de Óxido Metálico”.

Es una tecnología que (a diferencia de la tecnología TTL) utiliza transistores de efecto de campo (FET) tanto de canal tipo N como de canal tipo P para la construcción de circuitos integrados.

Al ser una tecnología en que sus circuitos integrados están contruidos en base a FETs, su consumo es muy bajo y cuando el circuito se encuentra en estado de reposo prácticamente no existe circulación de corriente. Por el contrario, sus únicas pérdidas son fugas que pudieren tener los circuitos o también conocidas como corrientes parásitas.

Otra característica es que su voltaje de polarización puede variar, generalmente es entre 4 y 15 volts, pero existen algunos circuitos que soportan hasta 18 volts. Para averiguar estos parámetros de cada circuito integrado se debe revisar el datasheet.

A medida que el circuito funcione a una mayor frecuencia, su consumo será más elevado.

-Ventajas

- 1.- Bajo consumo de potencia debido a la alta impedancia de entrada que poseen los FETs.
- 2.- Los circuitos CMOS pueden restaurar las señales digitales, es decir, al tener en la entrada un nivel lógico de 4,7 volts y su alimentación es de 5 volts, la salida debe dar un resultado lógico 1, por lo tanto habrá 5 volts en la salida (y no la señal dañada inicial de 4,7 volts).

- Diferencias entre TTL y CMOS

- 1.- Mientras TTL presenta un consumo elevado de potencia, CMOS es más eficiente.
- 2.- Los circuitos TTL pueden manejar corrientes más elevadas que los CMOS.
- 3.- Los circuitos TTL tienen mejor respuesta en frecuencia y más rango de operación que los CMOS.
- 4.- A pesar de que el ruido es un problema general de todos los circuitos, CMOS tiene mejor inmunidad al ruido que los circuitos TTL.

Guía 3: Electrónica Analógica vs Electrónica Digital

Existen muchas formas de separar ramas de la electrónica, algunos ejemplos pueden ser por la cantidad de potencia que manejan (electrónica de potencia), si su comportamiento es sinusoidal (electrónica lineal) o por el contrario si sufre cambios bruscos (electrónica no lineal). En esta guía se verá la diferencia entre electrónica analógica y electrónica digital y qué implica cada rama.

- Electrónica analógica

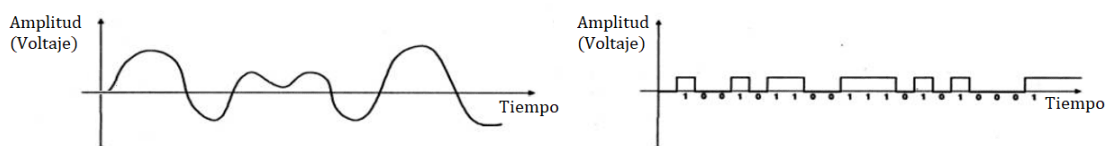
La electrónica analógica abarca todos los circuitos en que el comportamiento de sus voltajes pueda tomar cualquier valor, incluso valores infinitos (solamente en forma teórica).

- Electrónica digital

En esta rama de la electrónica los valores de sus voltajes solo pueden adoptar valores lógicos 0 y 1, es decir pueden ser 0 y 5v (para lógicas TTL). Para tecnología CMOS sus voltajes pueden ser 0v para un 0 lógico y desde 4v hasta 18v para un 1 lógico.

Arduino UNO es un tipo de circuito digital y sus niveles lógicos pueden ser 0v y 5v.

- Ejemplo



Fuente: Elaboración propia

Figura 4-67 Ejemplo de señal analógica vs señal digital

En la figura 4-67 se tiene una comparación entre una señal analógica vs una digital. En las señales analógicas existe continuidad del voltaje, a diferencia de la señal digital que trabaja solamente con 0 y 1.

A pesar de que Arduino UNO es un dispositivo digital, se encuentra equipado internamente con un módulo conversor análogo-digital capaz de recibir señales analógicas en sus entradas.

Para emular señales analógicas en sus salidas cuenta con módulos del tipo modulación por ancho de pulso (PWM).

Al trabajar convirtiendo señales analógicas en digitales o a la inversa, aparece un parámetro llamado resolución que indica la calidad del muestreo. La resolución se mide en bits, mientras mayor sea la cantidad de bits, mejor será la calidad de muestreo.

En la figura 4-68 se presentan dos ejemplos de una señal analógica convertida en una señal digital con diferente resolución.



Fuente: Elaboración propia

Figura 4-68 Comparación de señal analógica vs señales digitales con diferente resolución.

Evidentemente que si se realiza una comparación, se notará que en una señal digital se pierde información. Mientras mayor sea la resolución, mejor será la calidad de la señal digital.

Guía 4: El Microcontrolador por dentro

Antes de poder comenzar con la programación de Arduino, se debe tener claro que la placa contiene un microcontrolador, por lo tanto, se hace necesario saber cómo funciona por dentro y cuáles son las ventajas y desventajas de su uso.

- Partes internas de un microcontrolador

Todo microcontrolador funciona como si fuera un computador en miniatura y de este modo puede ser separado en bloques de operaciones, cada uno con tareas específicas.

1.- Unidad de control o microprocesador

Corresponde a la Unidad Central de Proceso (CPU) o simplemente el procesador, se puede decir que es el “cerebro del computador”, encargado de ejecutar todas las tareas y realizar las operaciones lógicas. El procesador también tiene bloques internos:

-ALU (Aritmetic Logic Unit): La unidad aritmética lógica es la unidad encargada de realizar las operaciones lógicas básicas como sumas y multiplicación. Se dedica un bloque especial dentro del procesador ya que su importancia es vital en un computador o microcontrolador.

-Unidad de Control: Como su nombre lo indica, es la unidad encargada de controlar la ALU, los registros, los buses y de realizar todos los procesos lógicos necesarios para controlar las unidades mencionadas.

-Registros: Corresponden a una pequeña memoria que tiene el procesador necesaria para almacenar los datos de las operaciones que realizará, generalmente se hace mención a ellos en los procesadores de computadores como memoria caché, la idea principal de esta memoria es almacenar datos de instrucciones que ejecutará el procesador, de este modo es posible obtener un mayor rendimiento.

2.- Memoria

En un microcontrolador al igual que en un computador, existen dos tipos de memoria, la primera es la memoria no volátil que equivale al “disco duro”, en ella quedará almacenado el programa. La segunda memoria es la RAM (Random Access Memory) y corresponde a una memoria temporal que utilizará el procesador para realizar alguna operación, es una

memoria volátil, ello implica que al quitar la alimentación al microcontrolador, la información guardada ahí se perderá.

3.- Periféricos

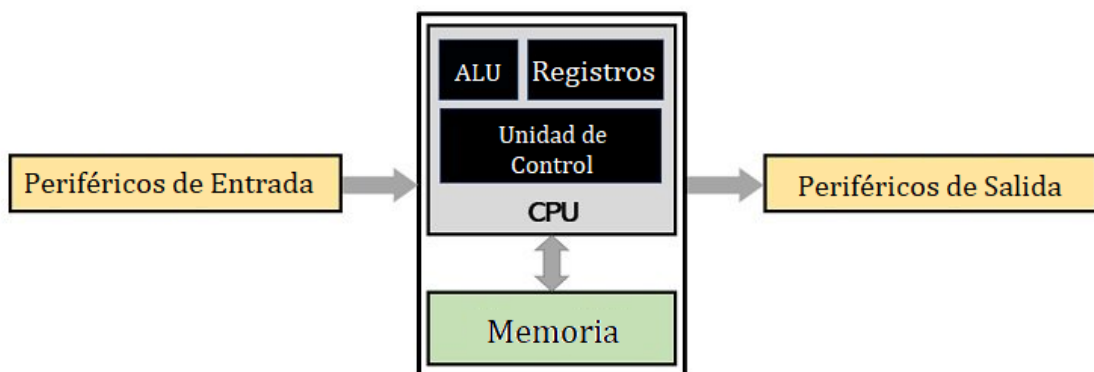
Corresponden a los puertos del microcontrolador. En ellos están las entradas y salidas digitales, entradas y salidas análogas, PWM y pines de transmisión y recepción.

- Arquitecturas de microcontroladores

Ahora que se sabe cuáles son los bloques internos de un microcontrolador, es posible continuar para ver cómo se encuentran organizados, esto se traduce en dos tipos de arquitecturas.

1.- Arquitectura de Von Neumann

Es la primera arquitectura de microcontroladores y debe su nombre en honor al físico y matemático John von Neumann.



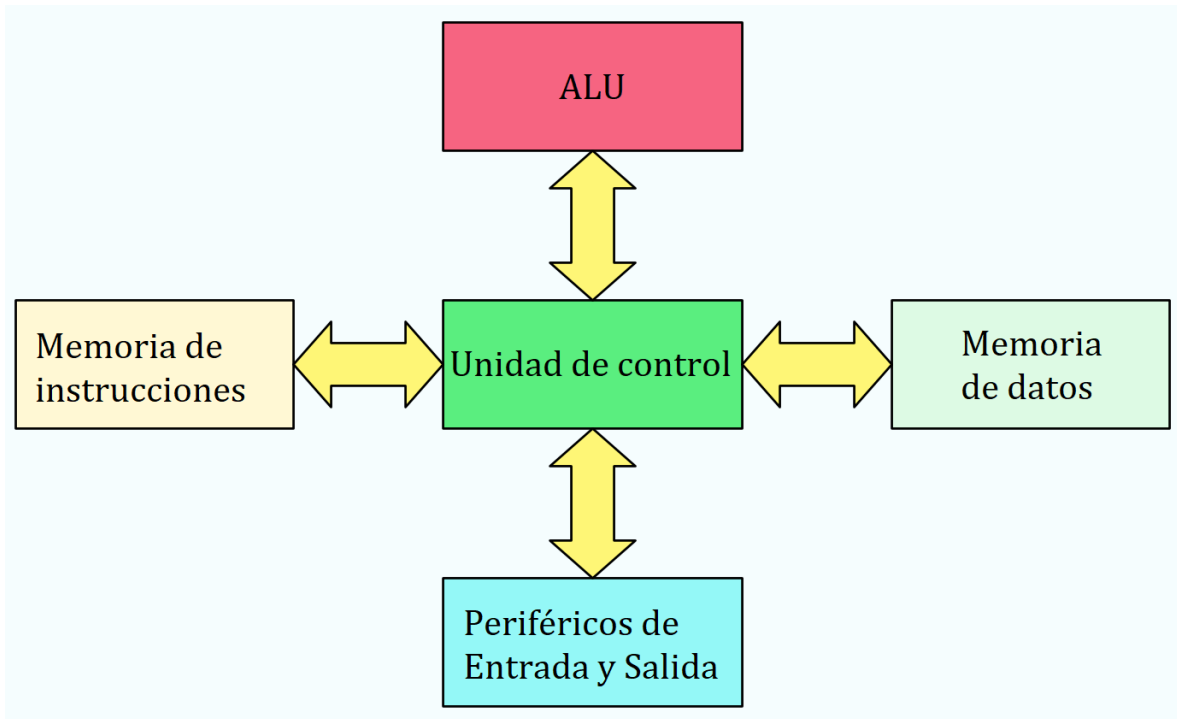
Fuente: Elaboración propia

Figura 4-69 Esquema de arquitectura Von Neumann

Esta arquitectura actualmente está obsoleta debido a que presenta problemas en el bus de datos de la memoria y la CPU, pero de todas formas es la base para la creación de una arquitectura más avanzada.

2.- Arquitectura Harvard

Es la arquitectura utilizada hoy en día en computadores y microcontroladores, en este último caso se aprovecha el diseño teniendo dos memorias separadas, de este modo se puede definir cierta cantidad de bits del bus para cada memoria (X bits para la memoria de instrucciones y Z bits para la transferencia de datos).



Fuente: Elaboración propia

Figura 4-70 Esquema de arquitectura Harvard

A pesar de la existencia del esquema, cada fabricante le realiza sus propias modificaciones y mejoras a sus dispositivos.

Arduino UNO está diseñado en base a arquitectura Harvard pero con las modificaciones del fabricante, de este modo el nombre de la arquitectura es RISC 8-bit.

Guía 5: Componentes externos necesarios para el microcontrolador

Para que el microcontrolador pueda funcionar, requiere una serie de elementos externos.

1.- Alimentación

Para cualquier dispositivo electrónico es necesaria una alimentación, en el caso de los microcontroladores, debe ser una fuente de voltaje continuo.

Es importante recordar que si el microcontrolador está construido en base lógica TTL es necesario una fuente de 5v, y si es en base a lógica CMOS se debe consultar el datasheet. De todos modos su alimentación puede estar entre 4v y 18v.

2.- Cristal de cuarzo

El cristal de cuarzo o también llamado oscilador de cristal es una pequeña cápsula que contiene un circuito piezoeléctrico. El circuito está diseñado para que el cristal genere una señal eléctrica de precisión. Por ejemplo: si el cristal dice que oscila a 4MHz (4.000.000 de Hz), oscilará exactamente a esa frecuencia. La señal generada proporciona al Arduino UNO una señal de reloj estable.



Cristal de Cuarzo

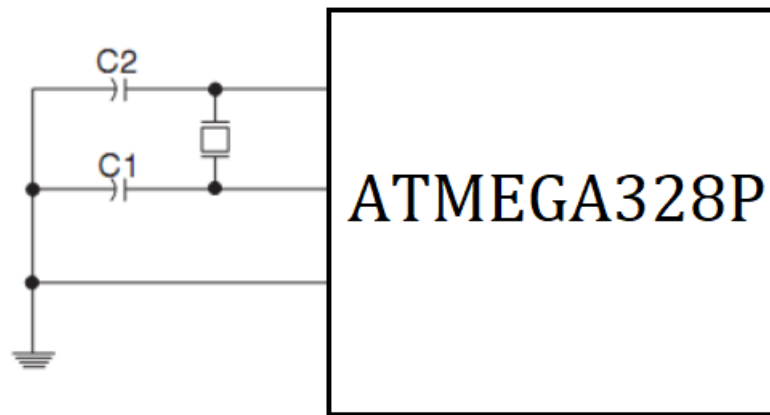
Fuente: Elaboración propia

Figura 4-71 Cristal de Cuarzo en placa de Arduino UNO

El microcontrolador ATMEGA328P presente en el Arduino UNO utiliza un cristal de cuarzo de 16MHz.

3.- Condensadores

Siempre que se utilice un microcontrolador, junto con el cristal de cuarzo, habrá que conectar dos condensadores, uno en cada extremo del cristal y los otros terminales a tierra, el valor viene definido por el fabricante, se emplean para que la frecuencia a la que oscila el cristal sea exacta. De este modo los condensadores eliminan capacitancias parásitas del circuito y de la placa.



Fuente: Elaboración propia

Figura 4-72 Circuito esquemático de conexión con cristal de cuarzo y condensadores

Guía 6: Lenguajes de Programación

Para que el microcontrolador ejecute o controle lo deseado, primero debe saber lo que se hará, esto lo es posible conseguirlo cargando un programa en su memoria de datos.

Un programa es una serie de instrucciones o comandos ordenados de forma coherente. El programa trabajará sobre los datos que el usuario defina para lograr el propósito.

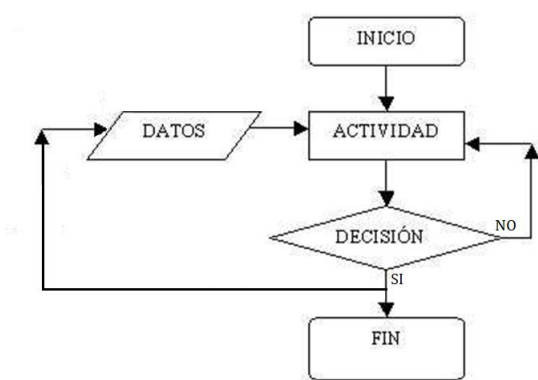
Es necesario recordar que el procesador solo entiende ceros y unos, por lo tanto se debe hacer que la información llegue al procesador de este modo. Existe un lenguaje que entiende el procesador llamado código máquina, pero el programa no se escribe en ese lenguaje porque solo lo entiende el procesador.

Para llegar al código máquina existen otros lenguajes en los que sí se se puede escribir el programa y luego convertirlo.

- Lenguaje Assembler

El lenguaje Assembler o Ensamblador (en español) es un lenguaje de bajo nivel en el que se utilizan palabras que el microcontrolador (o el dispositivo que se esté programando) interpreta como instrucciones, es el lenguaje más parecido al código máquina. Dado lo complejo de utilizar este lenguaje es que prácticamente no se utiliza.

A pesar de que está muy desplazado por el lenguaje C y C++, existen aplicaciones que hacen más simple la programación en assembler y consisten en diagramas de flujo (Figura 4-73).



Fuente: Elaboración propia

Figura 4-73 Ejemplo genérico de un diagrama de flujo

Las aplicaciones que utilizan diagramas de flujo, consisten en dibujar el diagrama insertando los datos con los que se va a trabajar y automáticamente la aplicación creará el programa.

Cabe destacar que los diagramas de flujo son muy útiles en todos los lenguajes de programación, más allá de que si se utiliza o no alguna aplicación para ellos, servirán para poder dibujar la estructura del programa, las decisiones que tendrá que tomar el microcontrolador y los datos con los que se trabajará.

- Lenguaje C++

Es el lenguaje en que se escribirá el programa de Arduino, es un lenguaje de alto nivel. Su modo de programar consiste en la misma sintaxis que el lenguaje C pero con algunas modificaciones para permitir controlar cosas. A diferencia de assembler, C++ no trabaja directamente en la memoria y hace más sencillo el control del microcontrolador, además es mucho más amigable porque permite agrupar funciones (una función dentro de otra).

Como es el lenguaje con el que se programará el Arduino, se profundizará más en las siguientes unidades.

UNIDAD 4: APUNTES PARA PROGRAMAR

Guía 1: Instalación del Software IDE

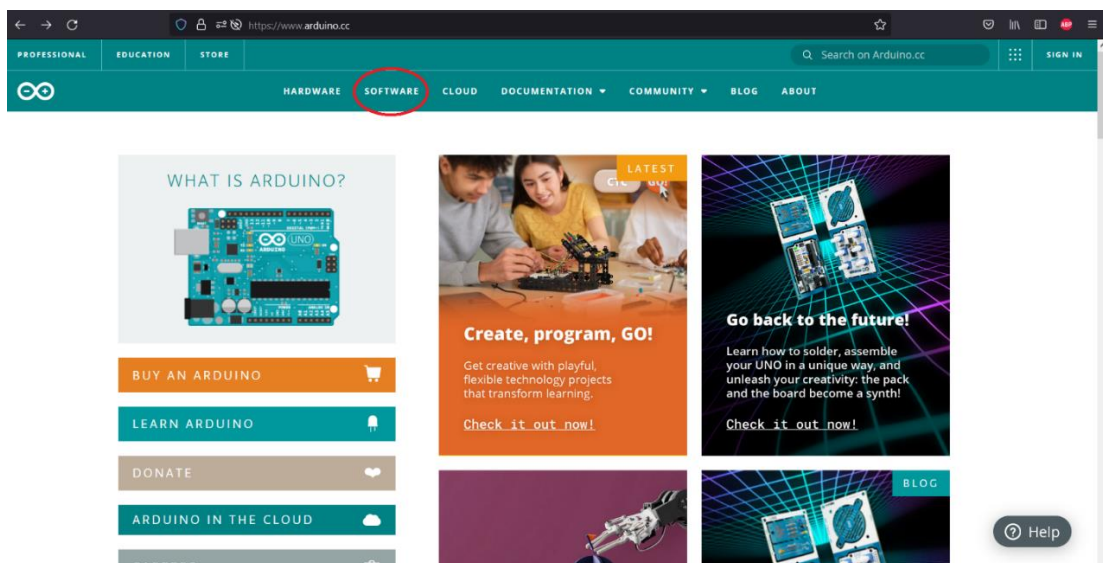
Para programar el Arduino se utiliza el Software IDE, en él se escribirá el programa y una vez terminado permitirá compilar para verificar errores. Una vez corregido, podrá ser cargado en el microcontrolador.

Arduino tiene licencia de hardware libre, por ese motivo es que existen muchos tipos de Arduino UNO con distintos precios y tamaños de microcontrolador.

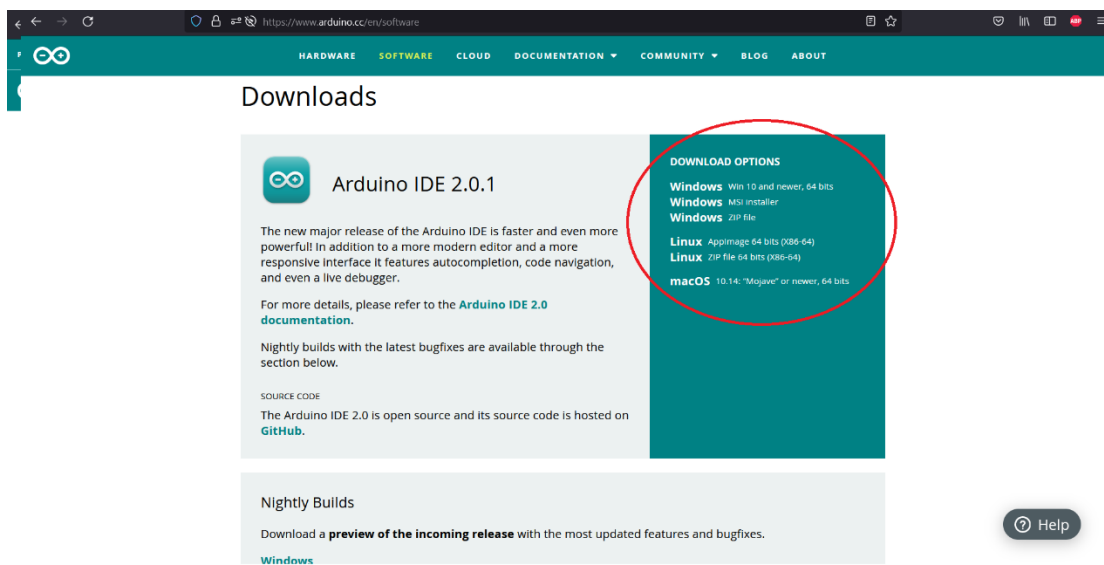
Para obtener el software IDE hay que realizar una serie de pasos detallados a continuación.

1.- En el PC o dispositivo similar se debe acceder a la página <https://www.arduino.cc>

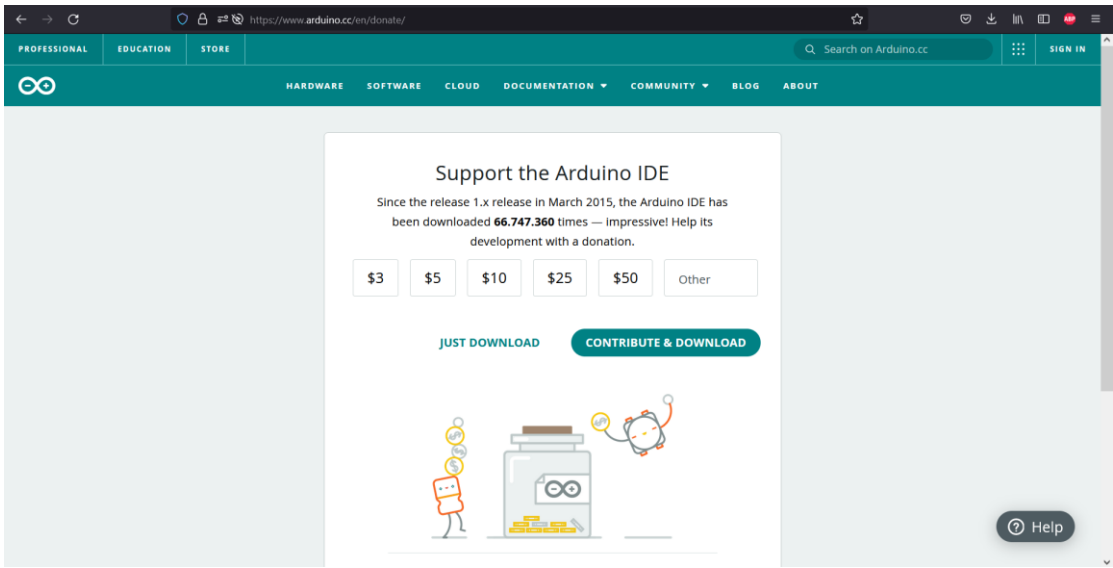
2.- Una vez en la página dar clic en el botón “SOFTWARE”.



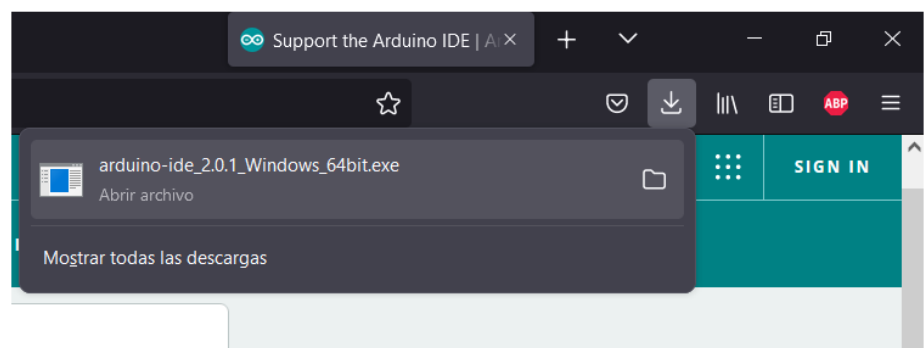
3.- En las opciones de descarga, elegir la que sea correcta el equipo de acuerdo al sistema operativo. Al descargar un archivo en formato ZIP habrá que descomprimirlo.



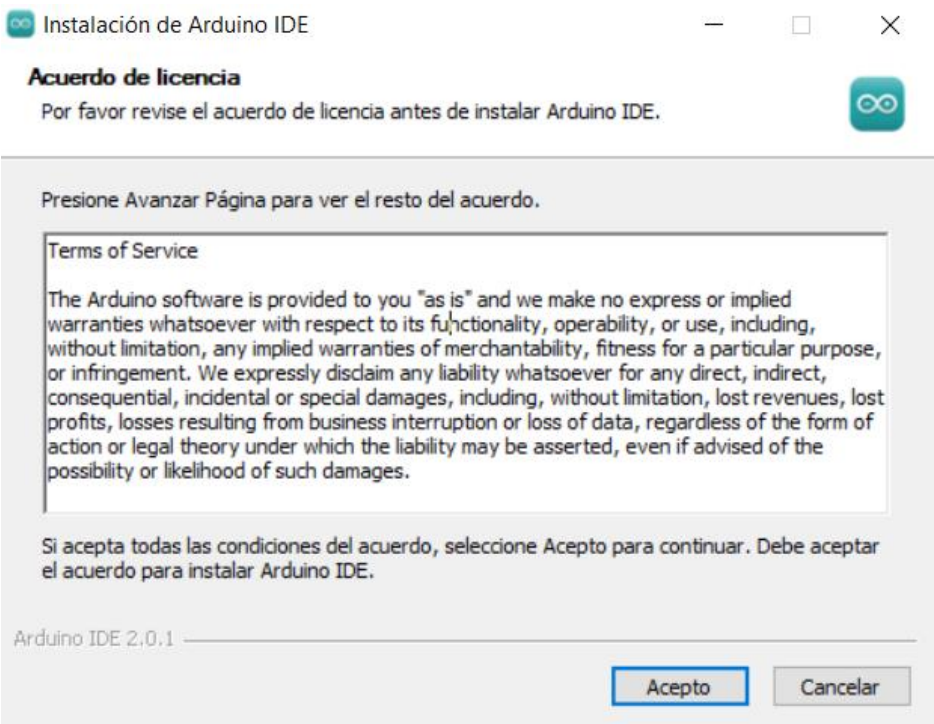
4.- Una vez seleccionado el archivo para descargar, aparecerá lo siguiente. Si se desea contribuir con un aporte para Arduino, seleccionar “CONTRIBUTE & DOWNLOAD”, por el contrario si solo se desea descargar, seleccionar “JUST DOWNLOAD”.



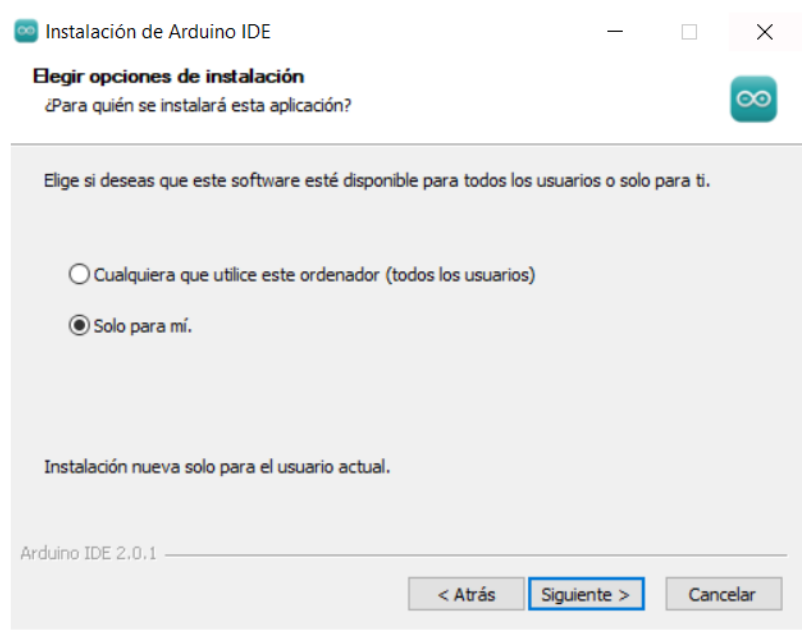
5.- Una vez descargado el archivo, ejecutar. Si es un archivo ZIP primero debe ser descomprimido para luego ejecutar.



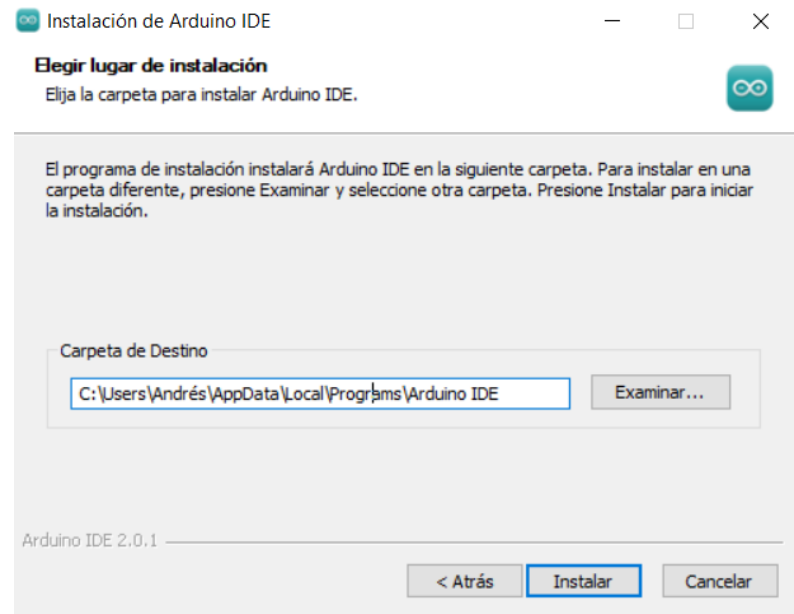
6.- Una vez ejecutado el archivo, aparecerá la siguiente ventana, presionar “Acepto” para continuar la instalación.



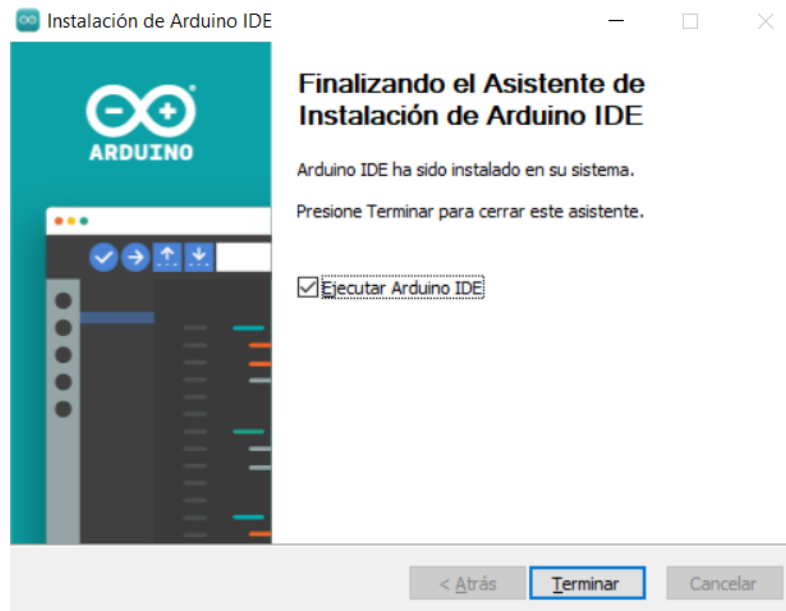
7.- Luego aparecerá la siguiente ventana, seleccionar la opción que deseada y presionar Siguiente.



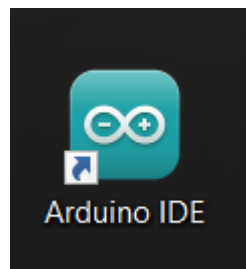
8.- La instalación pedirá elegir la carpeta en donde se instalará el programa. Si se desea instalar en una carpeta distinta presionar examinar, de lo contrario presionar instalar y luego esperar hasta que el programa termine.



9.- Una vez terminada la instalación aparecerá la siguiente ventana. Presionar terminar y luego se abrirá el software Arduino IDE.



Al terminar la instalación se debes ver un ícono como el siguiente en el escritorio.



Cuando se haya ejecutado el programa, es probable el dispositivo solicite acceso a puertos de comunicación o algunas excepciones para el Firewall de Windows. Se debe permitir el acceso para que el Software se pueda comunicar con el Arduino UNO en el momento en que se cargue el programa.

Guía 2: Sintaxis

Anteriormente se analizaron las ventajas de utilizar microcontroladores o tarjetas de desarrollo nuestro Arduino UNO, pero una de las desventajas que tienen, es que no son intuitivos y no tienen capacidad de improvisar. Todas, absolutamente todas las instrucciones deben ser claras y acorde a las reglas del lenguaje C++, es por eso que cuando al escribir el programa debe ser realizado con mucha rigurosidad.

La sintaxis es una serie de reglas que permiten realizar una comunicación más sencilla y accesible.

El programa puede ser escrito de la forma que sea más cómoda, sin embargo, cuando al momento de compilar existirán situaciones en las que el Software IDE no entenderá lo que se intenta decir y arrojará error. Para solucionar esos errores será necesario recurrir a la sintaxis, de este modo será más sencillo identificar las partes del programa en donde no han sido respetadas las reglas del lenguaje C++.

Antes de comenzar con las reglas de sintaxis, es importante definir elementos necesarios para programar:

- **Variable:** Es un valor determinado (dato) que será guardado en algún espacio de la memoria. Este valor podrá ser modificado durante la ejecución del programa gracias a que (como su nombre lo indica) es variable. Para acceder a la variable o consultar su valor será se le llamará mediante el nombre asignado al declararla. Las variables siempre son de algún tipo (ejemplo: int, bool, word), también tienen nombre y valor. Dependiendo del dato que se necesite guardar, será el tipo de variable seleccionada.
- **Compilar:** Al compilar un programa, quiere decir que el software IDE buscará errores y en caso de no encontrarlos, convertirá el programa a lenguaje de máquina para que pueda ser cargado en el Arduino UNO.
- **Constante:** A diferencia de las variables, una constante es un valor que será guardado en algún espacio de la memoria pero que no puede ser modificado.

A continuación se mencionan las reglas de sintaxis para programar en C++.

1.- Las mayúsculas y minúsculas no son lo mismo.

Ejemplo: Las variables **sensor** y **Sensor** no son lo mismo y cada una puede representar una variable distinta.

2.- Al definir una variable, la etiqueta no puede comenzar con un número.

Ejemplo: Si se define un variable como 1led el programa cuando compile arrojará error. En este caso la definición correcta de la variable podría ser **led1**

3.- Se puede definir una variable con letras del alfabeto mayúsculas, minúsculas, números y guiones bajos.

Ejemplo: **Led_1**

4.- Cuando se haga una función, todo el contenido de la misma debe estar entre llaves { }

incluso si la función tiene más de una línea de comandos.

Ejemplo: Si se tiene la función **void funcion() {contenido de la función}**

5.- Todas las líneas de comandos terminan con punto y coma ;.

6.- Si se desea escribir comentarios en una sola línea, escribir // después del punto y coma.

7.- Para escribir comentarios multilínea se comienzan escribiendo /* para abrirlo y para cerrar escribir */

8.- Para escribir decimales se debe utilizar un punto porque la coma tiene otros usos.

Ejemplo: 2.87

9.- Para llamar a una función siempre se debe escribir el nombre de la misma y terminarla entre paréntesis incluso si la función no necesita ningún parámetro dentro del argumento.

Ejemplo: para una función que no necesita parámetros: **setup()**

Ejemplo: para una función que necesita parámetros en su argumento: **escribir("UTFSM")**

10.- Cuando se tiene más de un argumento dentro de una función, instrucción o matriz se deben enumerar con comas.

Ejemplo: **digitalWrite(1,LOW)**

11.- Cuando se declara o se utiliza una variable y tiene solo un carácter, debe ser encerrada entre comillas simples.

Ejemplo: **'x'**

12.- Cuando se declara o se utiliza una variable y consiste en una cadena de caracteres, debe ser encerrada entre comillas dobles.

Ejemplo: **"ABC"**

13.- Para aplicar una instrucción o enviar una orden a algún módulo del Arduino, se hará mediante un punto. En estos casos habrá que incluir bibliotecas o librerías al inicio del programa, se enseñará en qué consisten durante la unidad.

Ejemplo: **Serial.print("Curso de Arduino UTFSM")**

Guía 3: Tipos de memoria

La memoria es simplemente un sistema digital capaz de almacenar datos. Existen muchos tipos de memoria con distintos funcionamientos, pero en esta guía se verán solamente las que ocupa el Arduino UNO.

1.- Memoria Flash

Aquí es en donde se almacenará el programa, si se compara con un computador, equivale al disco duro. En esta memoria se guardará el programa compilado y listo, pero no podrá ser modificado mientras se esté ejecutando, si se desea realizar modificaciones será necesario cargarlo nuevamente.

La capacidad de la memoria flash (al igual que todas las memorias) es limitada y para el caso de Arduino UNO es de 32K bytes, es por este motivo que el programa debe ser redactado de la forma más eficiente posible.

2.- Memoria EEPROM

Es una memoria diseñada para mantener los datos después de aplicar un reset, independiente si se realiza mediante el programa o presionando el botón en la placa, por lo tanto es una memoria no volátil. El tamaño de esta memoria en Arduino UNO es de 1K byte.

3.- Memoria SRAM

Equivale a la memoria RAM en un computador. Se utiliza para almacenar datos temporales, por ejemplo una lectura de pin o cadenas de caracteres. Es una memoria volátil, por lo tanto si el Arduino se queda sin energía se perderá la información. El tamaño de esta memoria es de 2K bytes en Arduino UNO.

Al ser un recurso limitado debe ser gestionada con mucho cuidado, especialmente al almacenar cadenas de caracteres o strings. Si se excede la cantidad de información y no alcanza la memoria para la cantidad de datos necesarios, el Arduino se quedará sin memoria SRAM disponible, a pesar de eso, al compilar y cargar el programa no se presentará ningún problema, pero es probable que en pleno funcionamiento se produzcan errores imprevistos.

Guía 4: Estructura de un programa

Para programar el Arduino UNO, es muy importante destacar que el programa tiene una estructura secuencial, esto significa que las instrucciones se ejecutan en orden.

El orden correcto para una estructura de programa es el que aparece en la figura 4-74.

Incluir Librerías

Declaración de variables

```
int variable1=0;
bool variable2=1;
```

```
void setup {...
```

```
....
}
```

```
void loop {...
```

```
....
}
```



Fuente: Elaboración propia

Figura 4-74 Estructura de un programa Arduino.

Primero se incluyen las librerías, se pueden obtener descargándolas o incluyendo las que viene en el software IDE.

Después se declaran las variables, dependiendo del tipo de dato es la denominación que se utilizará. En el ejemplo de la figura 1 se declara la “variable1” como un tipo de dato entero de valor 0 y la “variable2” como un tipo de dato booleano de valor 1.

Luego está la función void setup(), en ella se configuran los pines del Arduino como entradas o salidas digitales o análogas.

Para terminar está la función void loop() donde ocurrirá todo el proceso que se desea realizar, para ello se utilizan los operadores como if(), while(), for(), digitalWrite(), etc.

Para tener una idea más sencilla, se presenta un ejemplo de un programa que enciende y apaga el led de la placa Arduino UNO.

```

1  #include <Ethernet.h>
2  #include <Arduino_BuiltIn.h>      //ejemplos para incluir librerías
3
4  /*Programa ejemplo de sintaxis para guía temática de Arduino UTFSM
5  Ejemplo de comentario multilínea
6  */
7
8  void setup() { ;                  //Escribe tu código aquí para que se ejecute una sola vez
9
10 pinMode(LED_BUILTIN,OUTPUT);      //configuramos que conecta el led del Arduino como salida
11
12 }
13
14 void loop() { ;                  //Escribe tu programa principal aquí para que se ejecute repetitivamente
15
16     digitalWrite(LED_BUILTIN,HIGH); //encendemos el led
17     delay(500);                    //esperamos 500 mili segundos
18     digitalWrite(LED_BUILTIN,LOW);  //encendemos el led
19     delay(500);                    //esperamos 500 mili segundos
20
21 }
22

```

Fuente: Elaboración propia

Figura 4-75 Ejemplo de estructura de programa

En la figura 4-75 se presenta un ejemplo con los comentarios. Se aplica todo lo aprendido en la guía 2 de sintaxis.

A continuación se tienen algunas reglas muy importantes para evitar errores muy comunes.

- 1.- Antes de utilizar alguna variable siempre debe haber sido declarada primero, de lo contrario el programa arrojará error al compilar.
- 2.- Si se desea utilizar una biblioteca primero debe ser descargada y guardada, luego podrá ser incluida en el programa.
- 3.- En el caso de las funciones podrán ser llamadas después o incluso antes de declararlas, a pesar de que el programa se ejecuta en forma secuencial, las funciones se compilan de forma independiente, por este motivo pueden ser llamadas en cualquier parte del programa.

Guía 5: Configuración de pines (Entradas y Salidas)

Los pines son los terminales del microcontrolador. En el caso de Arduino UNO no existe acceso directo a ellos, pero sí se pueden conectar cables en los conectores hembra que hay a los lados.

Antes de utilizar los pines, el Arduino debe saber primero cómo serán utilizados, esto implica si funcionarán como entradas o salidas.

Para configurar los pines se utiliza la función `pinMode()`, que permitirá definir uno por uno si los pines funcionarán como entradas o salidas.

La función `pinMode()` casi siempre se utiliza en la función `setup()` ya que es poco probable que sea necesario reconfigurar los pines en la función `loop()`.

- Sintaxis de la función `pinMode()`

`pinMode(Número de pin, MODALIDAD);`

En el número de pin, simplemente se escribe el número de pin a configurar. Luego en “MODALIDAD” se escribe la forma en que funcionará, pueden ser `INPUT` (entrada), `OUTPUT` (salida) o `INPUT_PULLUP` (al pin se le agrega una resistencia interna de pull-up y queda configurado como entrada).

- Ejemplo

```
8 void setup() { ; //Escribe tu código aquí para que se ejecute una sola vez
9
10 pinMode(0, INPUT);
11 pinMode(1, OUTPUT);
12
13 }
```

Fuente: Elaboración propia

Figura 4-76 Ejemplo de configuración de pines 0 y 1

Como ejemplo se muestra la función `pinMode()` en la figura 4-76 configurando el pin 0 como entrada y el pin 1 como salida.

- Información útil

Al configurar los pines, siempre se debe considerar el número de pin desde el 0 hasta el último. Una forma para saber qué función tiene el pin que se va a configurar es ver en qué puerto se encuentra.

Puertos de nuestro Arduino

-PORT B: pines 8, 9, 10, 11, 12 y 13 (6 bits)

-PORT C: pines 14, 15, 16, 17, 18 y 19 (6 bits)

PORT C es un puerto analógico y sus pines pueden ser utilizados como entradas y salidas digitales, pero de forma analógica solo como entradas.

-PORT D: pines 0, 1, 2, 3, 4, 5, 6 y 7 (8 bits)

En algunos pines se puede ver el símbolo (~). Esto quiere decir que ese pin puede ser configurado como salida del tipo PWM, se enseñará en que consiste esto más adelante.

Guía 6: Funciones

Una función son líneas de programa que se encuentran separadas del código principal `loop()`. Su objetivo es realizar una tarea específica y al terminar, regresa a la siguiente línea de programa desde donde fue llamada.

A pesar de que un programa de Arduino está compuesto principalmente por dos funciones `void setup()` y `void loop()`, después de ellas se puede ir declarando todas las funciones que sean necesarias para ejecutar el programa.

```

1  void setup(){
2
3  //Función setup para configurar pines
4
5  }
6
7
8  void loop() {
9
10 //función loop donde ejecutará nuestro programa periódicamente
11
12  funcion_ejemplo() //llamamos a la función ejemplo
13
14
15 }
16
17
18 void funcion_ejemplo(){
19
20 //la función realizará la tarea que necesitemos y luego volverá a la función loop
21
22 }
23

```



The diagram illustrates the execution flow of a void function. A curved arrow starts from the call to `funcion_ejemplo()` on line 12, goes down to the start of the `funcion_ejemplo()` function definition on line 18, and then curves back up to the line of code immediately following the function call on line 13. This visualizes how the program jumps to the function's code and then returns to the point of the call.

Fuente: Elaboración propia

Figura 4-77 Ejemplo de cómo actúa una función tipo void

En la figura 4-77 se ve cómo actúa una función, primero se le llama en el programa `loop()`. Luego el programa se dirigirá a la función y ejecutará la tarea que haya sido programada. Una vez haya terminado volverá a la siguiente línea de código.

- Ventajas en el uso de funciones

1.- Ayudan a tener el programa mucho más organizado.

2.- Hacen que el programa tenga muchas menos líneas de código porque se puede llamar a una función más de una vez, de este modo la función puede ser reutilizada en lugar de tener que volver a escribir las líneas de código.

3.- Una función permite tener una acción concreta en un solo lugar del programa, de este modo si se llega a tener algún error, solo se debe corregir la función en lugar de buscar muchos errores como si se hubiera escrito muchas veces el mismo código en el programa.

4.- Las funciones hacen que las probabilidades de tener errores se reduzcan considerablemente en caso de tener que modificar el programa.

5.- Hacen más sencilla la reutilización del código. Si se requiere realizar la misma operación en otro programa, simplemente se puede copiar y pegar.

- Declaración de una función y sintaxis

Existen dos tipos de funciones, las que no entregan ningún valor puesto que solo ejecutan alguna tarea (tipo void) y las que devuelven algún valor.

1.- Funciones tipo void

Estas funciones no entregan ningún valor y solo ejecutan alguna tarea.

Se puede llamar una función en cualquier parte del programa, pero para declararla se debe hacer siempre fuera de cualquier otra función, eso incluye las funciones `loop()` y `setup()`.

- Sintaxis

`void función_declarada(parámetros) { ;`

contenido de la función;

`}`

```

8  void setup() {           //Función setup para configurar pines
9      pinMode(0,OUTPUT);   //configuramos pin 0 como salida
10     pinMode(1,OUTPUT);   //configuramos pin 1 como salida
11 }
12
13 void loop() {             //función loop donde se ejecutará nuestro programa periódicamente
14     parpadear_led1();     //llamamos a parpadear led 1
15     parpadear_led2();     //llamamos a parpadear led 2
16     parpadear_led1();     //llamamos a parpadear led 1
17     parpadear_led2();     //llamamos a parpadear led 2
18 }
19
20 void parpadear_led1(){    //parpadear led pin 0 dos veces
21     digitalWrite(0,HIGH);
22     delay(250);
23     digitalWrite(0,LOW);
24     delay(250);
25     digitalWrite(0,HIGH);
26     delay(250);
27     digitalWrite(0,LOW);
28     delay(250);
29 }
30
31 void parpadear_led2(){    //parpadear led pin 1 dos veces
32     digitalWrite(1,HIGH);
33     delay(250);
34     digitalWrite(1,LOW);
35     delay(250);
36     digitalWrite(1,HIGH);
37     delay(250);
38     digitalWrite(1,LOW);
39     delay(250);
40 }

```

Fuente: Elaboración propia

Figura 4-78 Ejemplo de funciones tipo void

En el ejemplo de la figura 4-78 se está suponiendo que hay dos leds conectados a los pines 0 y 1 y en `setup()` se han configurado como salidas, se necesita hacer que los leds parpadeen, dos veces el led del pin 0 y dos veces el led del pin 1, luego se vuelve a repetir la secuencia. Para no escribir dos veces el mismo código, se escriben dos funciones en donde parpadee dos veces cada led.

2.- Funciones que devuelven un tipo de dato

Funcionan de forma muy similar a las funciones tipo void y las reglas son las mismas: siempre se deben declarar fuera de cualquier otra función.

- Sintaxis para llamar la función

Tipo_de_dato variable_con_el_mismo_tipo_de_dato=**nombre_de_la_función**(parámetros);

- Sintaxis para declarar la función

Tipo_de_dato **nombre_de_la_función**(**Tipo_de_dato** parámetros){

Declarar variables locales;

Instrucciones;

Return variable_con_el_mismo_tipo_de_dato;

}

Significado de cada uno de los conceptos.

-**Tipo_de_dato**: Es el tipo de dato con el que trabajará la función.

Ejemplos: bool, char, byte, float

-**Variable con el mismo tipo de dato**: Es la variable que devolverá la función una vez que haya calculado su valor, es importante destacar que se debe declarar dentro de la función.

-**Nombre de la función**: Es el nombre que se le dará a la función, generalmente el nombre está muy relacionado con lo que hace la función.

-**Parámetros**: Son los datos con los que trabajará la función, generalmente son números.

- Ejemplo

Se necesita hacer que el Arduino calcule el 20% de algún valor para luego mostrarlo en un display., para ello será creada una función que calcule el 20% de 500.

```

1  //Aquí se declaran las variables globales
2
3  int valor=500;
4
5  void setup() {
6      // Aquí se configuran los pines como entradas y salidas:
7
8  }
9
10 void loop() {
11     // Aquí se escribe el programa principal, para que se repita periódicamente
12
13     //llamamos a nuestra función que nos calculará el 20% de 500
14     int respuesta=veinte_por_ciento(valor);
15
16     /*Una vez que nuestra función haya hecho los calculos
17     escribimos las instrucciones para mostrar en el display */
18 }
19
20 //Declaramos nuestra función que nos calcula el 20% de 500
21 int veinte_por_ciento(int valor){
22
23     int respuesta;          //declaramos "respuesta" como una variable local (solo válida en la función)
24
25     respuesta=valor*0.2;    //cálculo
26
27     return respuesta;      //Nuestra función devuelve el resultado al programa principal
28
29 };
30

```

Fuente: Elaboración propia

Figura 4-79 Ejemplo de función que devuelve un dato

En la figura 4-79 se presenta un ejemplo usando una función que devuelve un dato, en este caso realiza un cálculo en base a un dato dado por el usuario, pero puede aplicarse a datos que el Arduino tome del exterior a través de algún sensor conectado en algún pin configurado como entrada.

Guía 7: Entradas Analógicas y Salidas PWM

Así como el Arduino tiene la capacidad de trabajar con entradas y salidas digitales, también tiene algunos pines capaces de entregar y recibir señales analógicas.

Es importante recordar que las señales digitales son las que toman valores de 0 lógico para 0 volts y 1 lógico para 5 volts, pero en señales analógicas una señal puede tener cualquier valor de voltaje. Como Arduino trabaja con un máximo de 5 volts, se trabajará con señales analógicas que varíen sus valores entre 0 y 5 volts.

- Entradas analógicas

Como los procesadores solamente entienden ceros y unos, para utilizar las entradas analógicas existe un módulo interno llamado conversor análogo-digital y su trabajo es convertir señales analógicas en digitales como su nombre lo indica.

El conversor análogo digital, tiene una característica muy importante llamada resolución que para el caso de Arduino es de 10 bits (0-1023). Significa que al leer la entrada analógica, el dato que leído puede tomar valores entre 0 como mínimo para 0 volts y 1023 como máximo para 5 volts. Al realizar la división entre 5 volts y 1023 se puede notar que el Arduino puede detectar variaciones incluso menores a 5 mili volts.

- Sintaxis para leer una entrada analógica

```

Int Variable      ;      //Se declara la variable donde se guardará la lectura

Void setup(){     ;      //Función setup donde se configuran los pines
}

Void loop(){      ;      //Función loop donde se ejecuta el programa periodicamente

Variable=analogRead(número_de_pin); /* La lectura se guardará en la "Variable" */
}

```

```
1  int lectura=0 ; //declaramos la variable lectura como dato entero y le asignamos que su valor inicial sea 0
2
3  void setup() {
4      // put your setup code here, to run once:
5  }
6
7
8  void loop() {
9      // put your main code here, to run repeatedly:
10
11     lectura=analogread(A0); //Cuando leamos la entrada analógica A0 le asignará ese valor a la variable lectura
12
13 }
```

Fuente: Elaboración propia

Figura 4-80 Ejemplo de cómo usar la instrucción de lectura analógica

En el Arduino UNO los pines desde el A0 al A5 son el puerto de entradas analógicas y es posible verlo indicado en la placa.



Fuente: Elaboración propia

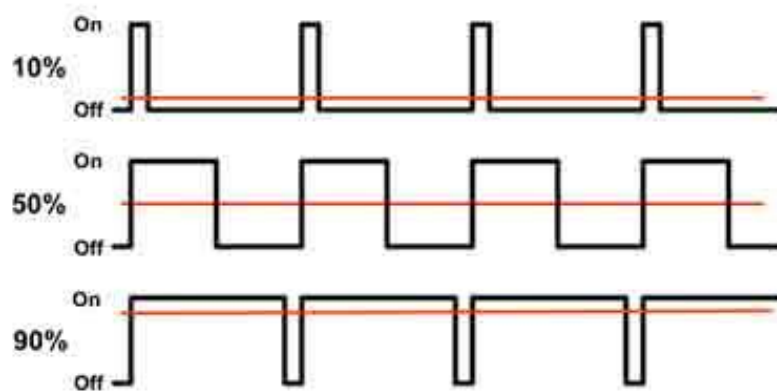
Figura 4-81 Puerto analógico en nuestro Arduino UNO

Como recordatorio: las entradas analógicas que se encuentran indicadas en la figura 4-81 también pueden ser usadas como entradas digitales.

- Salidas PWM

En realidad Arduino no tiene salidas analógicas, pero una forma de poder emularlas es la técnica PWM.

PWM (Pulse Wide Modulation) significa Modulación por Ancho de Pulso y se basa en una señal cuadrada de frecuencia constante a la que se le modifica el ciclo de trabajo (tiempo en que permanece en estado 1).



Fuente: Elaboración propia

Figura 4-82 Comportamiento de tres señales PWM con distintos ciclos de trabajo

También en el caso de salidas PWM existe el parámetro de resolución, pero (a diferencia de las entradas analógicas) es desde 0 a 255 (8 bits), siendo 0 el valor mínimo de salida (0 volts) y 255 el valor máximo (5 volts).

- Sintaxis para activar una salida PWM

```
analogWrite(pin,resolución);
```

En el parámetro pin se pone el pin de salida en el que será activada la señal PWM.

En resolución se escribe el ciclo de trabajo que tendrá la señal.

Al enviar un 0 en la resolución, en la salida habrá un 0 lógico.

Al enviar un 255 en la resolución, habrá 5 volts en la salida (1 lógico).

```
1  int led=3 //Declaramos que la variable led es nuestro pin 3
2
3
4  void setup() {
5      // put your setup code here, to run once:
6
7      pinMode(3,OUTPUT); //configuramos el pin 3 como salida
8
9  }
10
11 void loop() {
12     // put your main code here, to run repeatedly:
13
14     analogWrite(led,128); //enviamos una señal PWM al pin 3 trabajando al 50%
15
16 }
```

Fuente: Elaboración propia

Figura 4-83 Ejemplo de cómo activar una señal PWM en el pin 3

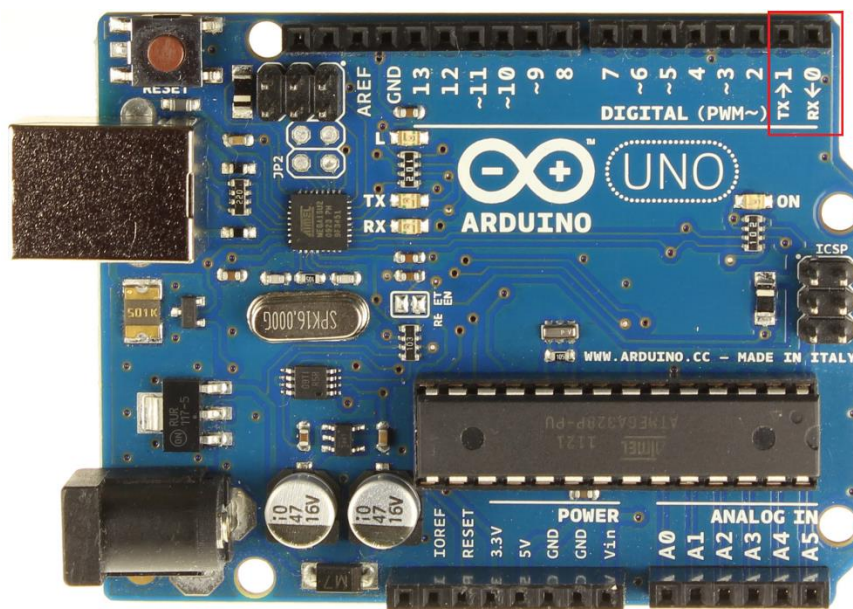
No existe un puerto específico en el Arduino de salidas PWM, sin embargo, se puede observar en la placa que algunos terminales tienen el símbolo ~, esto indica que esas salidas pueden ser utilizadas como salidas digitales y también como PWM.

Las salidas PWM son los pines 3, 5, 6, 9, 10 y 11.

Guía 8: Comunicación Serial

Arduino contiene un microcontrolador equipado con un módulo transmisor receptor serie de tipo TTL UART (Transmisor Receptor Asíncrono Universal), mediante él es posible establecer comunicación con otro Arduino, con el PC o cualquier dispositivos que esté equipado con un módulo similar.

UART es un protocolo de comunicación que consta de dos puertos, uno para transmitir información y uno para recibir información. Es posible identificar estos puertos mirando la placa de Arduino, el pin 0 es el puerto receptor y el pin 1 es el puerto transmisor.



Fuente: Elaboración propia

Figura 4-84 Placa de Arduino UNO con puertos de transmisión y recepción destacados en color rojo

Al conectar un Arduino UNO junto con otro Arduino UNO, deben tener las mismas configuraciones, es por eso que el protocolo UART tiene los siguientes parámetros de configuración.

- Bits de datos
- Bits de parada
- Bits de paridad
- Bits de inicio
- Velocidad en bits por segundo, lo que se conoce como baudio

Utilizar el puerto serie en general es una de las formas más sencillas de enviar información desde un dispositivo a otro, porque permite enviar, letras, números y hasta frases completas.

En el caso de Arduino UNO los pines TX y RX están conectados internamente al puerto USB, gracias a eso se puede utilizar el puerto serie a través del conector USB y establecer comunicación con el PC.

- Sintaxis para habilitar el puerto serial y enviar información

Dentro de la función void setup() se debe el siguiente código para inicializar el puerto serial, la velocidad en baudios debe ser la misma en el dispositivo que transmite y en el que recibe, esto es para que estén sincronizados, en caso de que estén configurados a velocidades diferentes, la comunicación de los dispositivos no va a estar sincronizada y por lo tanto se recibirá información incorrecta.

```
Serial.begin(velocidad_en_baudios);
```

Luego dentro de la función void loop() se inserta lo que se desea transmitir mediante el Arduino con los siguientes códigos

```
Serial.print("Información \n");
```

Los símbolos \n cumplen la función de la tecla enter, es para que al enviar la siguiente información esta aparezca en distintas líneas.

También existe el siguiente comando

```
Serial.println("Información");
```

Este comando sirve para que al enviar la información, aparezca inmediatamente en una nueva línea.

El comando `serial.print()` o `serial.println()` también se puede utilizar para enviar números en distintos sistemas, es decir, se pone el número en sistema decimal y luego se le dice al Arduino que lo envíe en el sistema que se desee y automáticamente el Arduino convertirá el dato al sistema deseado y lo transmitirá, la sintaxis es la siguiente

`Serial.print(datos,base);`

`Serial.println(datos,base);`

En el parámetro `datos` se escribe el número que se desea enviar (en sistema decimal).

En el parámetro `base` se pone la base del sistema numérico en que se desea enviar los datos, el formato para escribirlo es HEX para hexadecimal, DEC para decimal, OCT para octal y BIN para binario.

```
1 void setup() {  
2  
3   Serial.begin(9600);  
4 }  
5  
6 void loop() {  
7  
8   Serial.print("Puerto Serial \n");  
9   Serial.println("Probando envio de datos");  
10  Serial.println("Guia tematica de Arduino UNO");  
11  delay(3000);  
12 }
```

Fuente: Elaboración propia

Figura 4-85 Ejemplo de código para enviar información

En el ejemplo de la figura 4-85 se está enviando información a través del puerto serie hacia el PC, para ver la información se presiona el botón “Serial monitor” en el Software IDE, si el procedimiento ha sido realizado de forma correcta, se visualizará lo que aparece en la imagen de la figura 4-86.

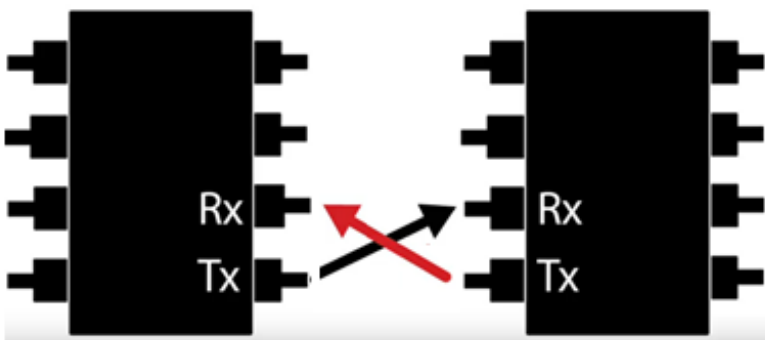


Fuente: Elaboración propia

Figura 4-86 Arduino UNO enviando información desde el puerto serie hacia el PC.

Se le pone un delay de 3000 para que el Arduino envíe los mensajes cada 3 segundos. En el ejemplo se inicia el puerto serie a 9600 baudios por segundo y el monitor serie se configura para visualizar los mensajes a la misma velocidad.

En el caso de que se desee establecer comunicación entre Arduinos, habrá que conectarlos como en la figura 4-87.



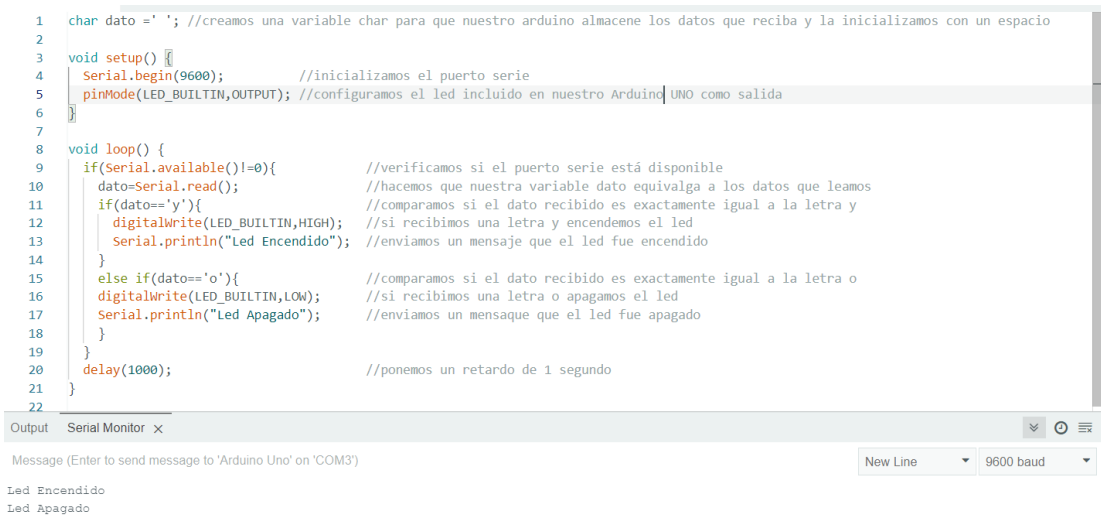
Fuente: Elaboración propia

Figura 4-87 Conexión de puertos serie entre Arduinos

Cada uno de los circuitos integrados representa un Arduino y las flechas representan la forma en que deben estar conectados los puertos para establecer comunicación.

- Ejemplo de recepción de datos

Se realizará una pequeña programación, se le enviarán datos al Arduino mediante el PC para que tome una decisión con los datos que reciba.



Fuente: Elaboración propia

Figura 4-88 Ejemplo de pequeño programa para recibir datos

El programa está diseñado para que cuando al enviar la letra “y” (sin comillas) el Arduino prenda el led que viene incorporado en la placa, por el contrario si recibe la letra “o” (sin comillas) apagará el led.

Al encender el led, Arduino enviará un mensaje de que ha sido encendido, y lo mismo hará cuando sea apagado.

Los comandos if y else if sirven para verificar si se cumple o no alguna condición, se profundizará en ellos más adelante.

Guía 9: Uso de Librerías

Las librerías son archivos que contienen códigos que permiten extender las funciones de Arduino. Existen librerías básicas que vienen incluidas en el software IDE y también se puede añadir otras externamente.

Existen tres tipos de librerías

- Librerías de núcleo
- Librerías estándar
- Contribuciones

Todas las librerías sin importar el tipo, se componen de dos archivos tipo *.h (encabezado) y *.cpp (programa).

El archivo de encabezado (*.h) será en donde se definan las variables y básicamente la configuración de la librería.

El archivo de programa (*.cpp) va a decir las instrucciones, es decir, qué va a hacer la librería.

- Sintaxis para agregar una librería

```
#include <librería.h>
```

Esta instrucción de incluir la o las librerías debe ir en la primera parte del programa, incluso antes de la declaración de variables. De este modo se puede vincular todo el código de la librería al programa.

- Librerías núcleo

Son las que determinan las instrucciones y funciones básicas de Arduino, gracias a ellas se puede programar de una forma mucho más intuitiva.

Es importante recordar que el lenguaje más profundo de programación es Assembler, pero gracias a las librerías junto con sus instrucciones y códigos es posible programar a un nivel no tan profundo y no tan cercano al lenguaje de máquina.

En ellas viene definido por ejemplo las funciones `void loop()`, `void setup()`, `digitalWrite()` y todas las demás.

- Librerías Estándar

Son librerías que no son estrictamente necesarias para el funcionamiento principal de Arduino, pero son muy utilizadas por toda la comunidad, es por eso que muchas de ellas vienen también pre instaladas junto con el software IDE. En todas las librerías las ideas son las mismas (incluir instrucciones y ampliar las funcionalidades de Arduino).

Algunos ejemplos de librerías estándar son

```
#include <LiquidCrystal.h>    //    Para controlar una pantalla LCD
#include <EEPROM.h>           //    Para controlar la memoria EEPROM
```

- Librerías de contribuciones por la comunidad

Como Arduino es una plataforma de desarrollo de hardware libre y software abierto, también es posible (si así lo desea el usuario) desarrollar librerías propias para fines específicos (controlar algún sensor, GPS, comunicación Wi Fi, etc).

Las librerías de contribuciones son aquellas que han sido creadas por personas de la comunidad.

Es probable que si en algún momento es necesario utilizar algún circuito integrado y no se sepa cómo funciona, o ya se sepa pero en el software de Arduino no haya sido desarrollada ninguna instrucción, haya que investigar y eventualmente se presente la necesidad de conseguir alguna librería desarrollada por personas de la comunidad.

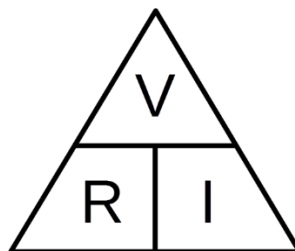
UNIDAD 5: APUNTOS Y CONCEPTOS ÚTILES

Guía1: ¿Qué es un cortocircuito?

La definición de cortocircuito es una falla de un aparato o línea eléctrica en donde la corriente pasa directamente desde la fase al neutro sin encontrar ningún tipo de resistencia, pero como en este curso solo se trabajará con voltaje y corriente continua, un cortocircuito será una falla en donde la corriente pase directamente desde el terminal positivo al negativo o a tierra sin ninguna resistencia de por medio.

En el caso de Arduino, un cortocircuito se producirá cuando un pin que se encuentre en 5 volts tenga contacto con otro pin que esté en estado lógico 0 y no haya ninguna resistencia de por medio. Si esto llega a ocurrir, es muy probable que el Arduino se queme y si está energizado a través de un PC es probable que también sufra daños, es por eso que es necesario tener especial cuidado en el armado de los circuitos.

- Explicación desde el punto de vista de Ley de Ohm



Fuente: Elaboración propia

Figura 4-89 Ley de Ohm

Teniendo en cuenta el triángulo de la figura 4-89, al despejar la corriente, quedará lo siguiente

$$I = \frac{V}{R}$$

Cuando se produce un cortocircuito, la resistencia tiende a 0, esto produce que la corriente tienda a infinito, por lo tanto, se produce un exceso de corriente y esto dañará los dispositivos.

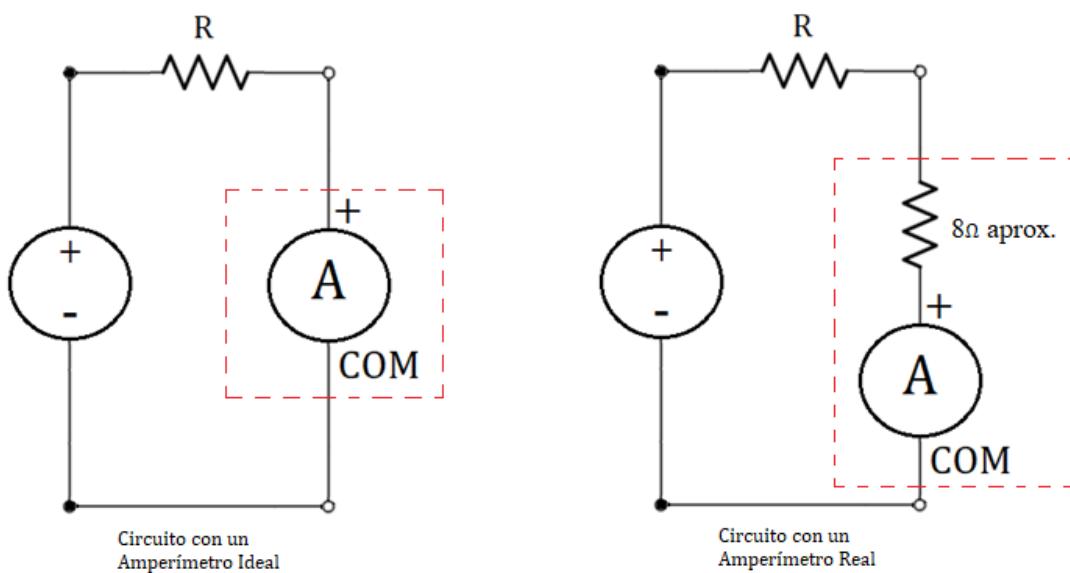
Guía 2: Efecto de Carga

El efecto de carga es el error en la medición al utilizar algún instrumento o dispositivo para medir, este error es producido por el mismo instrumento debido a sus impedancias internas.

- Ejemplo de efecto de carga al medir corriente

Recordatorio: al medir corriente siempre se debe conectar el amperímetro en serie.

Un amperímetro ideal tiene resistencia interna 0 ohm, pero un instrumento real tiene una resistencia aproximada de 8 ohm, si se desea medir una corriente pequeña en un circuito con resistencias muy grandes, se puede considerar el error de medición como algo despreciable, pero si el circuito tiene resistencias pequeñas, como por ejemplo midiendo la corriente en algún parlante, o en algún dispositivo que genere calor, el error será mayor.



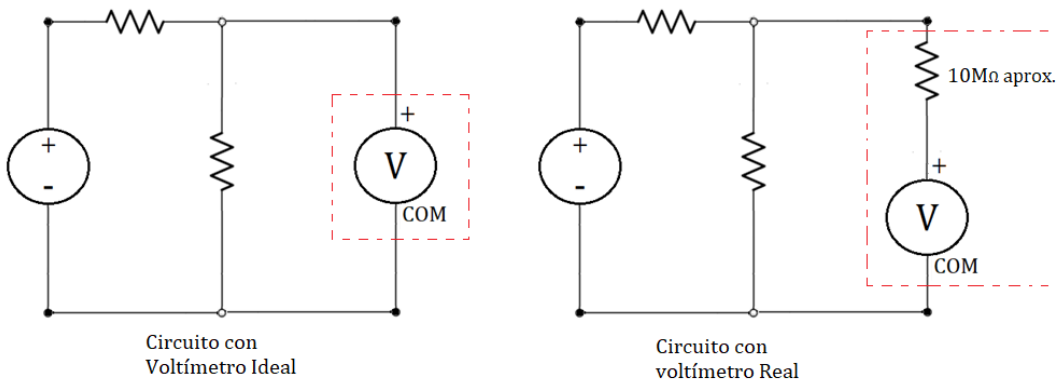
Fuente: Elaboración propia

Figura 4-90 Comparación de un circuito con un amperímetro ideal y uno real

En la figura 4-90 se puede observar la comparación de ambos circuitos, en caso de que la resistencia R sea de un valor muy bajo, la resistencia interna de 8 ohm generará un error en la medición.

- Ejemplo de efecto de carga al medir voltaje

En el caso de las mediciones de voltaje, se producirá un error en la medición al medir la tensión y el circuito tenga resistencias con valores muy altos.



Fuente: Elaboración propia

Figura 4-91 Comparación de un circuito con un voltímetro ideal y uno real

Si en los ejemplos de la figura 4-91 hubieran resistencias pequeñas, se podría considerar el error de lectura por efecto de carga como un valor despreciable, por el contrario, si las resistencias fueran superiores a 1M ohm, habría que tener en consideración el error.

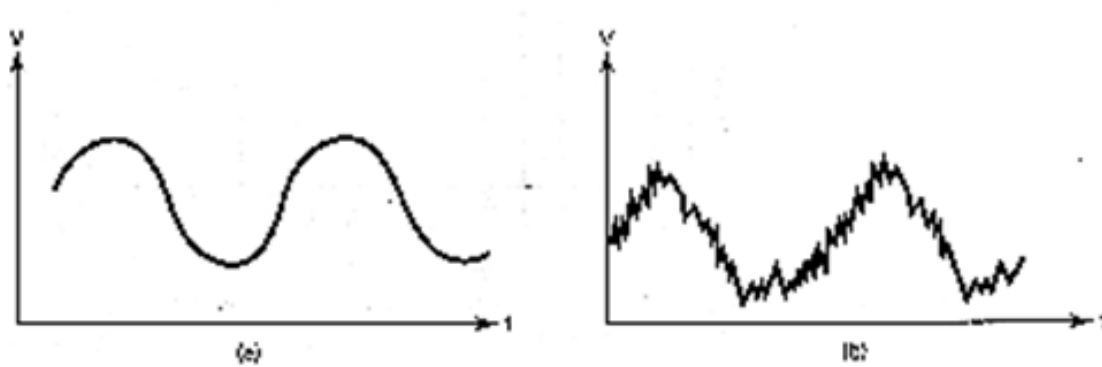
Guía 3: Ruido

El ruido eléctrico son señales aleatorias no deseadas que aparecen en los circuitos produciendo interferencia. Estas señales pueden producir efectos inesperados, algunos de ellos son fallas de comunicación o errores en la interpretación de señales.

El problema del ruido es muy común en circuitos de electrónica analógica, sin embargo en los circuitos digitales mientras el ruido no sea capaz de alterar un estado lógico, no habrá consecuencias.

Un ejemplo muy común de problema de ruido son las señales de tipo AM, al escuchar la radio en AM siempre se escuchan las señales distorsionadas, con sonidos extraños o con un sonido no fluido.

Las causas del ruido son variadas, pero las más comunes son las señales electromagnéticas en el aire, la red de 220v, equipos eléctricos con dispositivos de inducción (motores eléctricos o máquinas soldadoras) y ruido generado por otros dispositivos electrónicos.



Fuente: Elaboración propia

Figura 4-92 Ejemplo de señal normal y señal con ruido

En la figura 4-92 hay dos señales, una completamente normal y otra con un caso extremo de ruido en que la señal está totalmente distorsionada.

La forma más efectiva de atenuar o eliminar el ruido es con filtros de señales que pueden ser activos o pasivos. Otra forma muy común es el enmallado de cables para bloquear el paso de señales electromagnéticas.

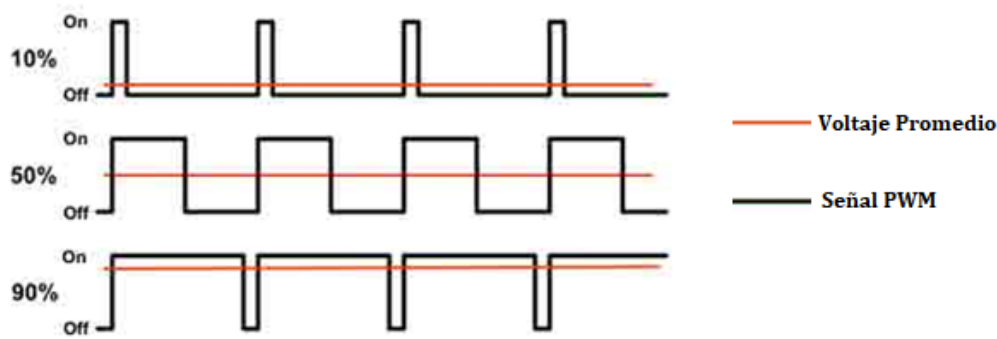
Guía 4: Voltaje Average

El voltaje average es un concepto que se aplica al activar salidas del tipo PWM y corresponde al voltaje promedio de una señal. Aunque todas las señales tienen un voltaje promedio, aquí solo se verá este concepto para la señal que entrega el Arduino.

Como recordatorio: una salida PWM entrega una señal cuadrada de frecuencia constante (puede ser de 490Hz o 980Hz), a la que se le modificará el ciclo de trabajo; al modificarlo se está variando el voltaje average.

El ciclo de trabajo también se expresa con las abreviaturas DT (Duty Cycle).

En la figura 4-93 se presentan 3 ejemplos de señales PWM con distintos ciclos de trabajo junto a su voltaje promedio.



Fuente: Elaboración propia

Figura 4-93 Ejemplos de señal PWM con su respectivo voltaje promedio

Como es posible apreciar en la figura 4-93, mientras mayor sea el ciclo de trabajo, más alto será el voltaje promedio.

Para calcular el voltaje promedio de una señal PWM se tiene la ecuación de la figura 4-94

$$V_{av} = V_{max} \bullet \frac{\text{Ciclo de Trabajo (\%)}}{100}$$

Fuente: Elaboración propia

Figura 4-94 Ecuación para calcular el Voltaje promedio de una señal PWM.

Donde:

V_{av} = Voltaje promedio de una señal PWM

V_{max} = Voltaje máximo de la señal, en el caso de Arduino serán 5 volts

Ciclo de trabajo = Corresponde al porcentaje de tiempo en un periodo (T) en que la señal se encuentra en estado lógico 1.

Al utilizar una salida PWM y seleccionar la resolución que se aplicará, el número 255 representa un ciclo de trabajo de 100%, es decir, la señal estaría en estado lógico 1 todo el tiempo.

La escala de 0-255 en Arduino equivale a la escala 0-100 para ciclo de trabajo. Para calcular el ciclo de trabajo deseado en la salida se puede realizar mediante una regla de 3.

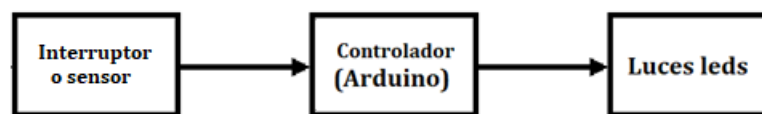
Guía 5: Diagrama en Bloques

A través de un diagrama en bloques es posible representar prácticamente cualquier sistema en el que se esté trabajando, desde un pequeño circuito hasta un gran proceso de producción.

Los diagramas en bloques son muy utilizados en procesos que utilizan control automático. Dado que con Arduino también se puede automatizar pequeños procesos, estos diagramas servirán para poder hacer más sencillo el análisis y diseño de los circuitos.

La idea principal de un diagrama en bloques es separar un circuito, sistema o proceso de producción desde un bloque grande a bloques más pequeños.

Por ejemplo, algo muy común en el uso de Arduino son sistemas de luces leds, esto se podría representar como en la figura 4-95



Fuente: Elaboración propia

Figura 4-95 Control de luces leds

En el ejemplo de la figura 4-95 se tiene un diagrama en bloques que representa un circuito para controlar el encendido y apagado de luces led.

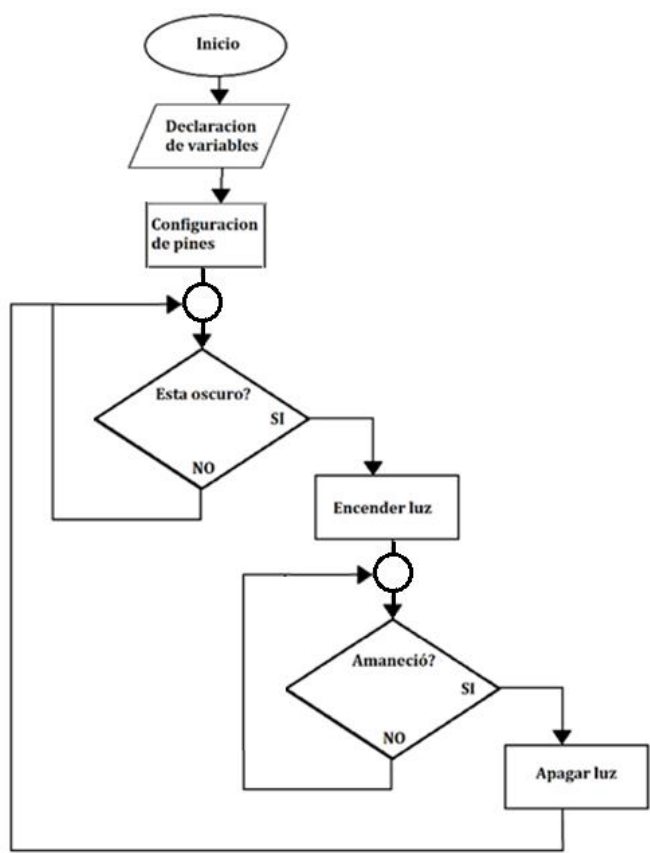
Guía 6: Diagrama de Flujo

Al igual que los diagramas de bloque permiten hacer diseños de circuitos y hardware, existen los diagramas de flujo que permiten describir cómo funciona un programa.

Los diagramas de flujo utilizan figuras geométricas como rectángulos, rombos, óvalos y otras figuras que servirán para que quede determinado lo que hace el bloque, estas figuras se emplean para hacer más sencilla la comprensión del programa. Además de figuras, los diagramas tienen flechas que unirán los bloques, de este modo se forma una secuencia en que el Arduino va tomando decisiones según lo programado.

- Ejemplo de un diagrama de flujo

En el ejemplo de la figura 4-96 se supone que se tiene un sensor que detecta la luz solar, cuando se oscurece debe encender una luz y cuando amanezca debe apagar la luz.

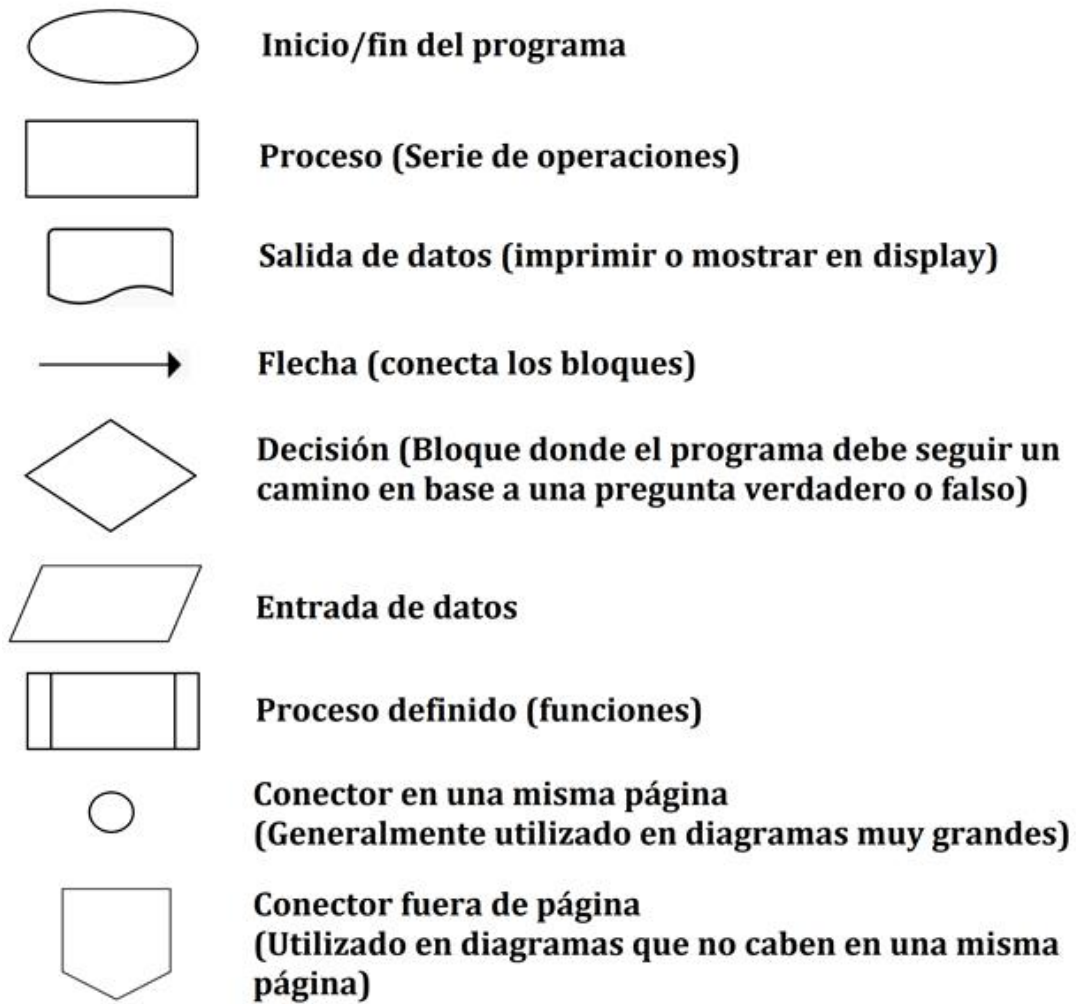


Fuente: Elaboración propia

Figura 4-96 Ejemplo de diagrama de flujo

La programación es secuencial, es decir, Arduino UNO ejecutará el programa en el mismo orden del diagrama de flujo diseñado.

- Símbolos para dibujar un diagrama de flujo



Fuente: Elaboración propia

Figura 4-97 Símbolos utilizados en un diagrama de flujo y su descripción

Además de la informática y la programación, los diagramas de flujo se utilizan en otras situaciones, como la economía, control de calidad y áreas administrativas.

Guía 7: Aislar Salidas

Arduino al ser un dispositivo pequeño, no puede entregar corrientes elevadas, por lo tanto, si se desea controlar un dispositivo más grande (como un motor, algún dispositivo que genere calor o que tenga un consumo de corriente mayor a 40 mili Amperes) es necesario utilizar buffers o adaptadores de impedancias como amplificadores operacionales u optoacopladores para separar el circuito de control (Arduino) de la etapa de potencia.

Los buffers pueden ser de corriente o de voltaje y su función es proteger el circuito de control.

- Límites de corriente de salida: para Arduino UNO

-Si es alimentado por USB: un total de **500 mA**

-Si es alimentado por fuente externa o batería: un total de **1 A**

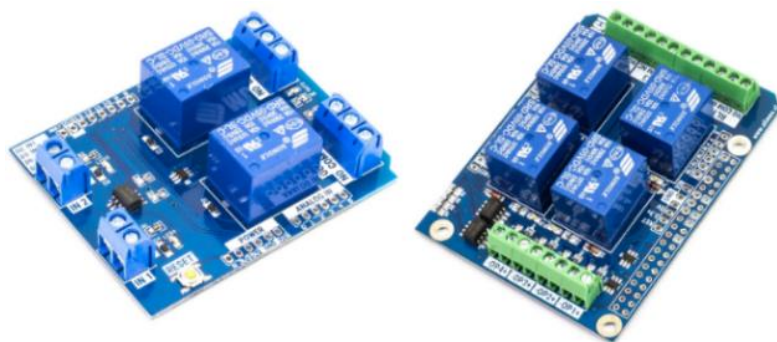
-Máximo individual por pin de Entrada/Salida: **40 mA**

-Suma de todas las Entradas/Salidas combinadas

(SIN incluir el pin de "5V"): **200 mA**

Teniendo en cuenta los límites de corriente de Arduino UNO, se debe elegir el buffer que más se adapte a la aplicación desarrollada.

1.- Kits de relés



Fuente: Elaboración propia

Figura 4-98 Kits de relé para Arduino UNO de dos y cuatro canales.

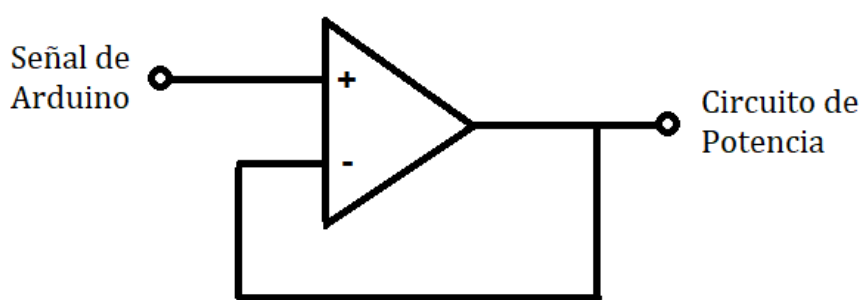
Al conectar los kits de relé de la figura 4-98 con Arduino UNO se podrá diseñar programas para controlar el encendido y apagado de dispositivos que consuman hasta 10 Amperes cada uno.

2.- Seguidor de tensión

El circuito seguidor de tensión está hecho en base a un amplificador operacional en el que se conecta una señal en el terminal de entrada no inversor (+) y luego se conecta el terminal inversor (-) a la salida del operacional. Es uno de los adaptadores de impedancia más comunes, sin embargo, al elegir este buffer, se debe tener en cuenta algunos parámetros importantes

-Slew Rate: Corresponde a la variación de voltaje por microsegundos del amplificador operacional, si este parámetro es muy bajo (menos de 1 volt por microsegundo) y el circuito trabajará con frecuencias altas (sobre 10Khz) el buffer no funcionará correctamente.

-Voltaje de polarización: Se debe tener en cuenta que el voltaje de polarización del amplificador operacional siempre sea mayor que el voltaje de la señal proveniente del Arduino, de lo contrario habrá pérdidas de tensión en la salida del buffer.



Fuente: Elaboración propia

Figura 4-99 Amplificador Operacional configurado como seguidor de tensión

3.- Optoacopladores

Los optoacopladores son un tipo de dispositivo en que la señal se transmite de forma óptica desde un emisor a un receptor, generalmente tienen internamente un led o un fotodiodo en el emisor, mientras que en el receptor tienen un fototransistor o un fototriac.

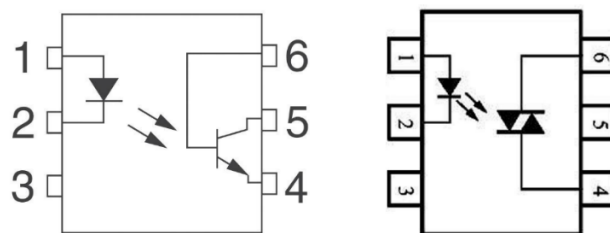
El principio de funcionamiento es que la señal de control se transmite de forma óptica, de este modo el circuito de control queda físicamente aislado de la etapa de potencia.



Fuente: Elaboración propia

Figura 4-100 Imagen referencial de un optoacoplador

El encapsulado de la figura 4-100 es el más común, aunque también existen en color negro y con 8 pines.



Fuente: Elaboración propia

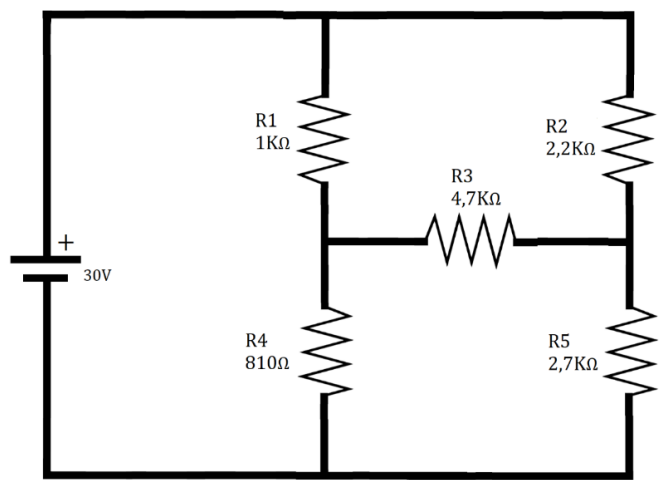
Figura 4-101 Esquemáticos referenciales de un optoacoplador con fototransistor (izquierda) y uno con fototriac (derecha).

En los esquemáticos de la figura 4-101 se puede ver que la transmisión de la señal se produce de forma óptica.

En caso de utilizar optoacopladores, se debe revisar el datasheet para saber cómo conectarlo.

Guía 8: Teorema de Kenelly

Existen casos especiales en los que las resistencias (o cualquier otro componente) no están conectados en serie y tampoco en paralelo, por lo tanto no es posible utilizar ninguna de las ecuaciones anteriores para realizar el análisis. Afortunadamente para estos casos especiales, existe el teorema de Kennelly.



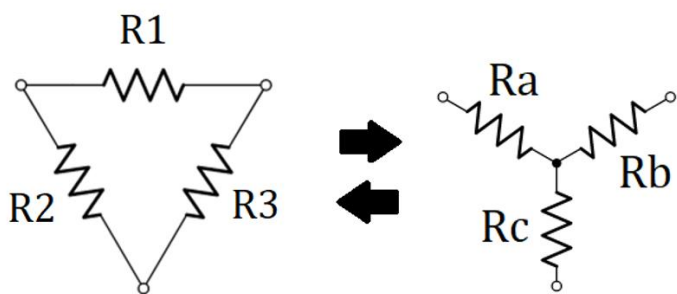
Fuente: Elaboración propia

Figura 4-102 Circuito para resolver con Teorema de Kennelly

Al intentar resolver el circuito de la figura 4-102 con lo aprendido hasta ahora, no es posible solucionarlo, por eso se presenta el teorema de Kennelly.

- Teorema de Kennelly

El teorema de Kennelly o más conocido en jerga electrónica como Conversión Estrella a Triángulo, son una serie de ecuaciones que permiten convertir una conexión de resistencias conectadas en Triángulo (Delta Δ) a Conexión Estrella (Y) y viceversa.



Fuente: Elaboración propia

Figura 4-103 Esquema de conexiones Triangulo (Δ) y Estrella (Y)

- Ecuaciones para convertir Triangulo a Estrella ($\Delta \rightarrow Y$)

Se utilizan las mismas etiquetas de las resistencias que aparecen en la figura 4-103:

$$R_a = \frac{R_1 \cdot R_2}{R_1 + R_2 + R_3}$$

$$R_b = \frac{R_1 \cdot R_3}{R_1 + R_2 + R_3}$$

$$R_c = \frac{R_2 \cdot R_3}{R_1 + R_2 + R_3}$$

- Ecuaciones para convertir de Estrella a Triangulo ($Y \rightarrow \Delta$):

$$R_1 = \frac{R_a \cdot R_b + R_b \cdot R_c + R_c \cdot R_a}{R_c}$$

$$R_2 = \frac{R_a \cdot R_b + R_b \cdot R_c + R_c \cdot R_a}{R_b}$$

$$R_3 = \frac{R_a \cdot R_b + R_b \cdot R_c + R_c \cdot R_a}{R_a}$$

- Información importante

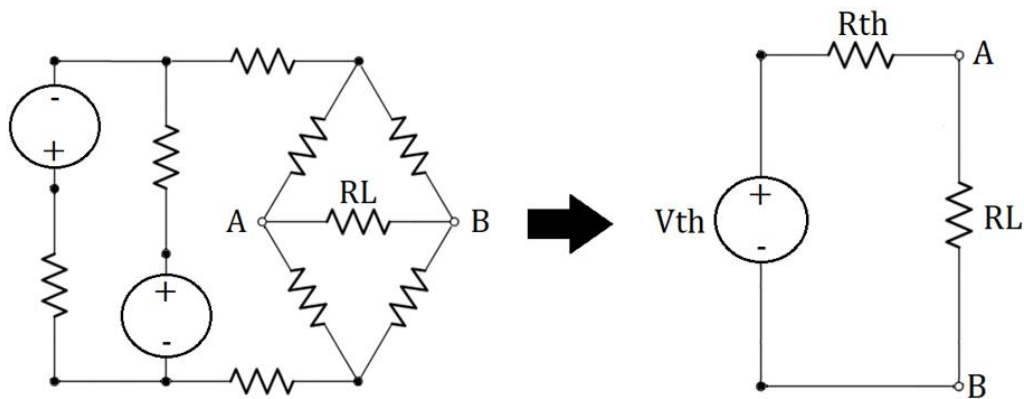
Generalmente este tipo de conexiones se utilizan en circuitos con puente de Wheatstone, muy utilizados en circuitos de instrumentación (con sensores).

También es muy utilizada en circuitos eléctricos trifásicos con tensiones de 380V o superiores.

Guía 9: Teorema de Thevenin

El teorema de Thevenin indica que si en un circuito hay una carga conectada entre dos puntos A y B; si se desconecta la carga, es posible cambiar el circuito por uno equivalente con una fuente de voltaje en serie con una resistencia. Al volver a conectar la carga entre los terminales A y B del circuito equivalente, la tensión en la carga y la corriente que circula a través de ella serán las mismas tanto en el circuito original como en el circuito equivalente.

Cuando se haya realizado la conversión del circuito original al equivalente, la nueva fuente de voltaje se llamará “Voltaje Thevenin” y lo se abreviará como V_{th} . La resistencia que irá en serie con la nueva fuente se llamara “Resistencia Thevenin” y se abreviará como R_{th} .



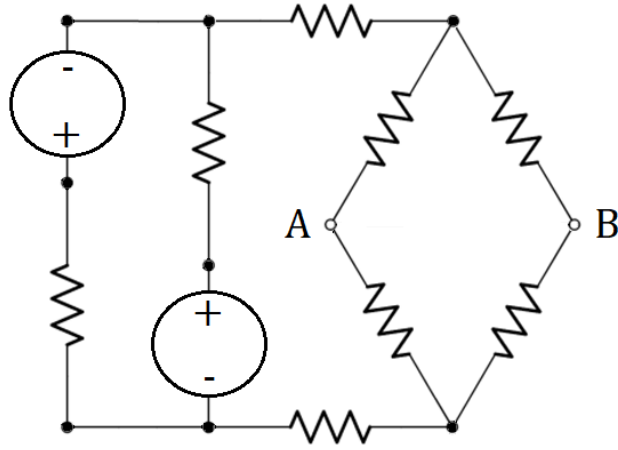
Fuente: Elaboración propia

Figura 4-104 Circuito con una carga “ R_L ” y su circuito equivalente Thevenin con la misma carga

Tal como se muestra en la figura 4-104, al realizar esta conversión, la corriente y el voltaje en R_L deben ser los mismos que en el circuito original.

- Pasos para convertir un circuito en su equivalente Thevenin, se utilizará el circuito de la figura 4-104 como ejemplo para el análisis.

1. Desconexión de la carga:



Fuente: Elaboración propia

Figura 4-105 Circuito original (sin carga RL).

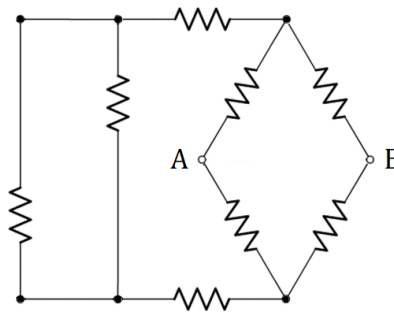
2. Teniendo el circuito sin carga, se calcula el voltaje entre los puntos A y B que será el Voltaje Thevenin (V_{th}), se puede usar el método que sea más comodo. Para obtenerlo primero se debe calcular V_A y luego V_B e introducirlos en la siguiente ecuación.

$$V_{AB} = V_A - V_B$$

$$V_{AB} = V_{th}$$

3. Cálculo de la Resistencia Thevenin (R_{th}):

Para calcular la resistencia Thevenin (R_{th}) se debe reemplazar las fuentes de voltaje por cortocircuitos (esto solo en el análisis en papel, nunca en forma práctica) y las fuentes de corriente por circuitos abiertos.



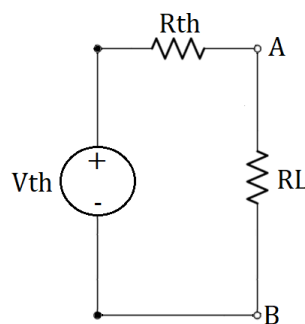
Fuente: Elaboración propia

Figura 4-106 Circuito con las fuentes de voltaje reemplazadas por cortocircuitos

Teniendo el circuito de la figura 4-106, se calcula la resistencia total del circuito entre los puntos A y B, en caso de que resulte complicado, se puede volver a dibujar el circuito de otra forma para que sea más sencillo.

Una vez obtenida la resistencia total entre los puntos A y B (que será la Resistencia Thevenin R_{th}), estarán los dos parámetros necesarios para dibujar el circuito equivalente.

4. Circuito equivalente y conexión de la carga



Fuente: Elaboración propia

Figura 4-107 Circuito Equivalente Thevenin con la carga conectada

La tensión y la corriente en R_L deben ser las mismas en el circuito equivalente Thevenin (figura 4-107) que en el circuito original.

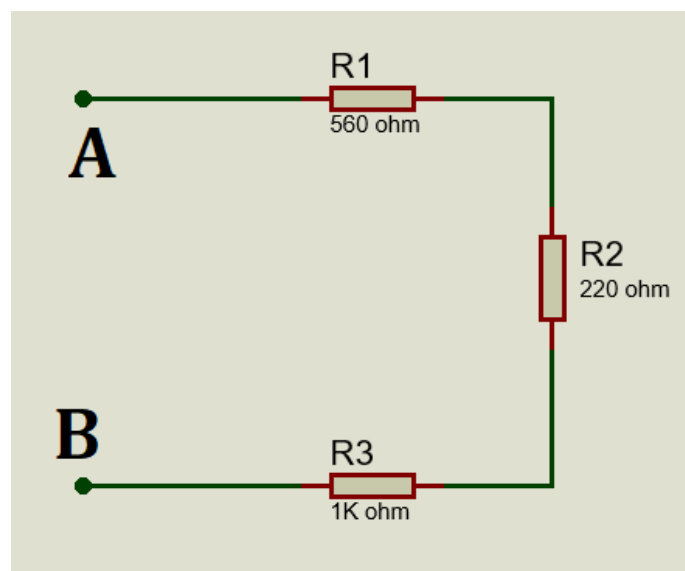
UNIDAD 6: EXPERIENCIAS DE ARDUINO

Experiencia 1: Uso de multímetro de ley de Ohm

Objetivos:

- Realizar prácticas con el protoboard, multímetro, resistencias y potenciómetro.
- Aplicar conocimientos de las unidades 1 y 2.

1.- Teniendo el circuito de la figura 4-108

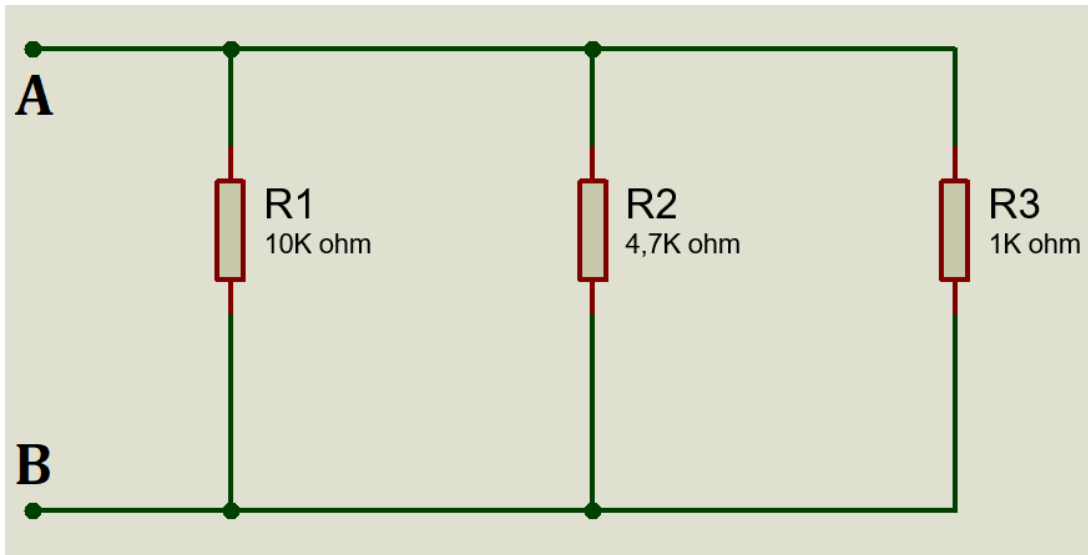


Fuente: Elaboración propia

Figura 4-108 Circuito Serie

- Calcular la resistencia total entre los puntos A y B.
- Conectar las resistencias en el protoboard. Luego medir con el multímetro la resistencia total entre los puntos A y B. Comentar resultados.
- Energizar el circuito conectando el terminal positivo de la fuente o batería en el punto A y el negativo en el punto B. Luego medir la corriente total del circuito y el voltaje en cada resistencia. Comentar los resultados.

2.- Teniendo el circuito de la figura 4-109

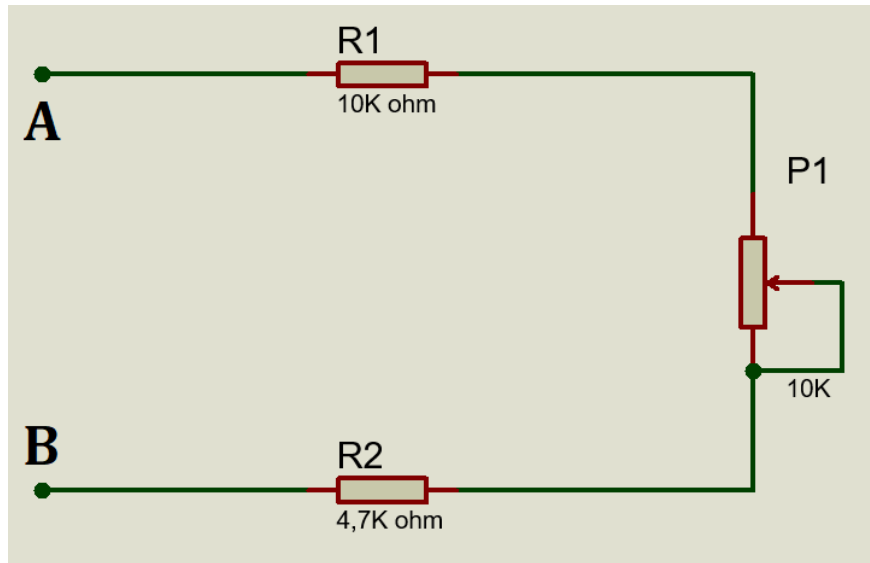


Fuente: Elaboración propia

Figura 4-109 Circuito Paralelo

- Calcular la resistencia total entre los puntos A y B.
- Conectar las resistencias en el protoboard. Luego medir con el multímetro la resistencia total entre los puntos A y B. Comentar los resultados.
- Energizar el circuito conectando el terminal positivo de la fuente o batería en el punto A y el negativo en el punto B. Luego medir el voltaje y la corriente en cada resistencia. Comentar los resultados.

3.- Teniendo el circuito de la figura 4-110



Fuente: Elaboración propia

Figura 4-110 Circuito serie con potenciómetro

- Conectar las resistencias y el potenciómetro en el protoboard. Luego medir con el multímetro la resistencia total entre los puntos A y B. Teniendo conectado el multímetro, mover el cursor del potenciómetro. Comentar lo observado.
- Energizar el circuito conectando el terminal positivo de la fuente o batería en el punto A y el negativo en el punto B, Al realizar la medición de voltaje en el potenciómetro, mover el cursor. Comentar lo observado.

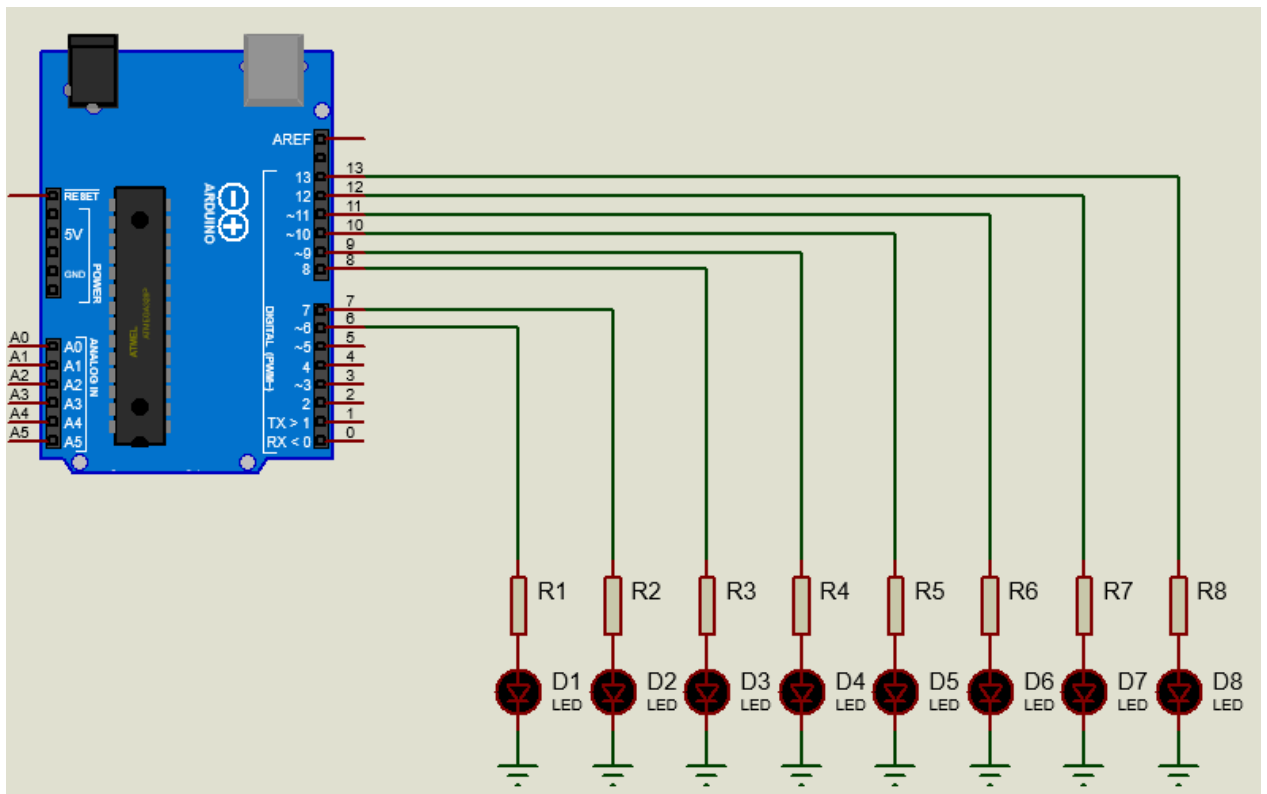
Experiencia 2: Secuencias con Leds

Objetivos:

- Aprender a conectar leds con resistencias.
- Aplicar función delay y digitalWrite.
- Encender y apagar leds.

La función delay() trabaja con milisegundos (milésima parte de un segundo), al llamar esta función, se debe escribir dentro de los paréntesis la cantidad de tiempo en milisegundos que el microcontrolador estará en espera hasta realizar la siguiente instrucción. Por ejemplo, si se desea que el Arduino espere un segundo: delay (1000).

- Teniendo el circuito de la figura 4-111:



Fuente: Elaboración propia

Figura 4-111 Circuito experiencia 2

- 1.- Calcular el valor de las resistencias para conectar los leds al Arduino.
- 2.- Implementar el circuito en el protoboard. Luego conectar el Arduino a tu PC.
- 3.- Abrir el software IDE y crear un programa en que todos los leds se enciendan y se apaguen al mismo tiempo, considerar un segundo de tiempo para los leds encendidos y un segundo para los leds apagados.
- 4.- Escribir un programa para que los leds se enciendan y se apaguen de forma intercalada (de modo que los pines conectados a los leds estén en estados lógicos 1 0 1 0 1 0 1 0 y luego 0 1 0 1 0 1 0 1). Determinar el tiempo en que los leds permanecerán encendidos y apagados para poder visualizar la secuencia.
- 5.- Escribir un programa en que se cree un efecto visual de desplazamiento a la izquierda y luego a la derecha creando el efecto de un metrónomo.
- 6.- Escribir otro programa con una secuencia a elección propia distinta a las anteriores, Determinar los tiempos de delay necesarios para poder visualizar.

Experiencia 3: Instrucción if

Objetivos:

- Utilizar funciones if, else if y else.
- Introducción

Con la instrucción if() se puede hacer que el Arduino tome una decisión, equivale a la figura de un rombo en los diagramas de flujo. Esta instrucción debe ir dentro de la función void loop() o también puede ir dentro de otra función.

- Sintaxis de la instrucción if

```
if(condición) {  
  instrucciones si la condición es verdadera;  
}
```

Con la instrucción if existen 6 tipos de condiciones o comparaciones posibles:

- > Mayor que
- < Menor que
- >= Mayor o igual que
- <= Menor o igual que
- == Igual a
- != Diferente a

En caso de que la condición propuesta se cumpla, el Arduino detectará esa condición como verdadera y ejecutará lo que esté dentro de llaves {}, en caso de que la condición no se cumpla, el Arduino detectará esa condición como falsa y se saltará el código entre llaves.

Al realizar comparaciones, se puede hacer con números o también con variables. Otra opción es que se podrá proponer una condición en la que se detecte el estado en que se encuentre algún pin de entrada y dependiendo si esta en 0 o 1 tomar una decisión.

- Ejemplo:

```
if(digitalRead(0) == 0){ // La condición es: si el pin 0 se encuentra en estado lógico 0
    digitalWrite(1,HIGH); /* Si se cumple la condición se pondrá en estado lógico 1 el pin 1. */
}
```

- Sintaxis de las instrucciones if y else

```
if(condición) {
    instrucciones si la condición es verdadera;
}
else {
    instrucciones si la condición es falsa;
}
```

En este caso se realiza un conjunto de instrucciones si la condición planteada se cumple, en caso contrario se podrá tomar otra decisión mediante else.

- Ejemplo

```
if(digitalRead(0) == 0) { /* La condición es: si el pin 0 se encuentra en estado lógico 0 */
    digitalWrite(LED_BUILTIN,HIGH); /* Si la condición se cumple, se enciende el led
    incluido en el Arduino */
}
else {
    digitalWrite(LED_BUILTIN,LOW); /* Si la condición no se cumple, se apaga el led
    incluido en el Arduino */
}
```

- Sintaxis de las funciones if, else y else if

```
if(condicion1){
```

```
Instrucciones en caso de que se cumpla la condicion 1;
```

```
}
```

```
else if(condicion 2){
```

```
Instrucciones en caso de que se cumpla la condicion 2;
```

```
}
```

```
else{
```

```
Instrucciones en caso de que no se cumpla ninguna condición;
```

```
}
```

La ventaja al usar estas tres instrucciones es que permite tener un mejor control de nuestro código al poder agrupar varias condiciones, de este modo se puede tomar una o más decisiones.

Es muy probable que cuando las condiciones de comparación sean booleanas solamente se utilice if o else, por el contrario si las condiciones incluyen variables analógicas, caracteres u otra variable como int o word, haya que utilizar if, else y else if.

A continuación se muestra un ejemplo en que se lee una entrada analógica, es importante recordar que al leerla arrojará un valor proporcional entre 0-1023 cuando haya voltajes entre 0 y 5v.

Una vez configurados los pines como entradas y salidas todas las variables hayan sido declaradas, se tendrá el siguiente programa dentro de la función void loop().

- Ejemplo

```
void loop{

  if(analogRead(A0)<=205){
    digitalWrite(0,HIGH);
    digitalWrite(1,LOW);
    digitalWrite(2,LOW);
  }

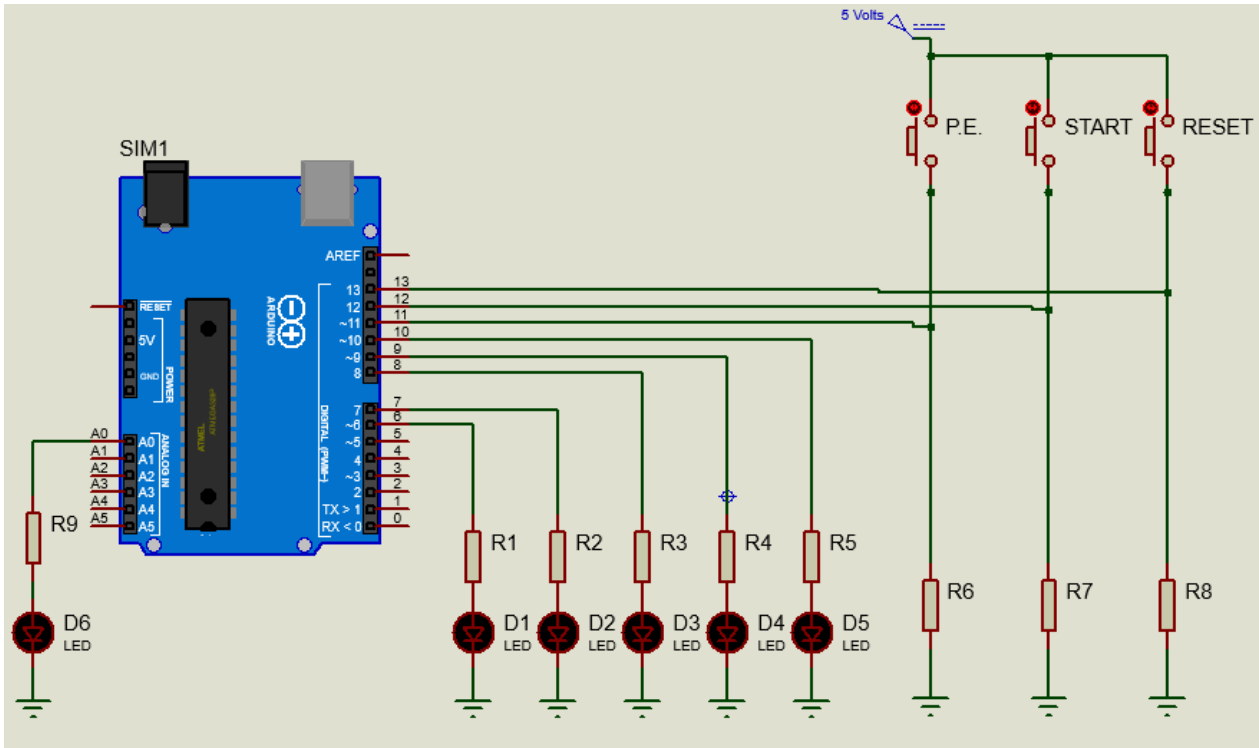
  else if(analogRead(A0)>=818){

    digitalWrite(1,HIGH);
    digitalWrite(0,LOW);
    digitalWrite(2,LOW);
  }

  else{
    digitalWrite(0,LOW);
    digitalWrite(1,LOW);
    digitalWrite(2,HIGH);
    delay(200);
    digitalWrite(2,LOW);
    delay(200);
  }
}
```

Se realiza la lectura de una entrada analógica, cuando el valor sea menor o igual a 1 volt, se encenderá el led conectado al pin 0. Cuando el valor sea mayor o igual 4 volts, se encenderá el led conectado al pin 1, y cuando el valor de la entrada analógica esté entre 1 y 4 volts, parpadeará un led conectado al pin 2.

- Teniendo el circuito de la figura 4-112:



Fuente: Elaboración propia

Figura 4-112 Circuito Experiencia 3

El circuito representa un sistema que controla un proceso donde cada componente tiene su equivalencia:

Leds D1, D2, D3, D4 y D5 representan un motor cada uno.

Led D6 representa una baliza.

Pulsador P.E. representa una parada de emergencia.

Pulsador Reset representa una botonera normalmente abierta que borra la falla.

Pulsador Start representa una botonera normalmente abierta que pone en marcha el proceso.

- Reglas de funcionamiento

Al presionar el pulsador Start deben encenderse los cinco motores.

Al presionar el pulsador P.E. deben detenerse los 5 motores, debe encenderse la baliza para indicar que hay una emergencia. Mientras la baliza se encuentre encendida, el sistema debe quedar bloqueado, de modo que si presionamos el pulsador Start el sistema no arranque nuevamente.

Para arrancar nuevamente el sistema, se debe presionar el botón Reset, al ser presionado se apagará la baliza y el sistema quedará desbloqueado, esta vez al presionar el pulsador Start, el sistema debe arrancar.

1.- Calcular el valor de las resistencias para conectar los leds al Arduino. Para las resistencias de los pulsadores, utilizar resistencias del valor más elevado posible.

2.- Implementar el circuito en el protoboard. Luego conectar el Arduino a tu PC.

3.- Crear un diagrama de flujo con las reglas de funcionamiento.

4.- En el software IDE diseñar un programa para que el Arduino controle el proceso según las reglas especificadas.

Experiencia 4: Bucle for

Objetivos:

- Aprender a utilizar la instrucción FOR.
- Introduccion

La instrucción for se utiliza para repetir un bloque de instrucciones encerradas entre llaves, para determinar la cantidad de veces que se repetirá el ciclo se configurará una condición en donde irá la cantidad de veces que se repetirá el bucle. Para que el Arduino sepa cuando debe terminar las repeticiones se usará un contador de incremento o decremento, en el que vaya contando cada vez que el ciclo se repita.

Esta instrucción es útil para cualquier operación repetitiva, generalmente se utiliza para matrices o configuración de pines.

- Sintaxis

```
for(inicio; condición; incremento o decremento){  
  Instrucciones;  
}
```

- Parámetros de la función for

-Inicio: es lo que ocurre primero y solamente una vez, aquí se declara el tipo de variable y el valor de inicio de la misma.

-Condición: es la prueba que hará el Arduino en cada repetición del bucle for, primero el Arduino realizará las instrucciones entre llaves y al terminar se ejecutará el incremento o decremento, luego se vuelve a probar la condición y mientras sea verdadera, el bucle volverá a ejecutarse. El bucle for terminará cuando la condición se vuelva falsa.

-Incremento o decremento: se ejecuta cada vez que el ciclo se repite a través del bucle for cuando la condición es verdadera. Cuando este parámetro torne la variable en un valor mayor o menor (dependiendo si es incremento o decremento) al de la condición, hará que la condición se vuelva falsa y con ello el bucle no se volverá a repetir.

- Ejemplo:

Se configurarán los pines mediante un bucle for, esto ahorrará líneas de código y por lo tanto hará más sencilla la redacción y comprensión del programa.

Recordatorio: la configuración de pines se debe hacer en la función void setup().

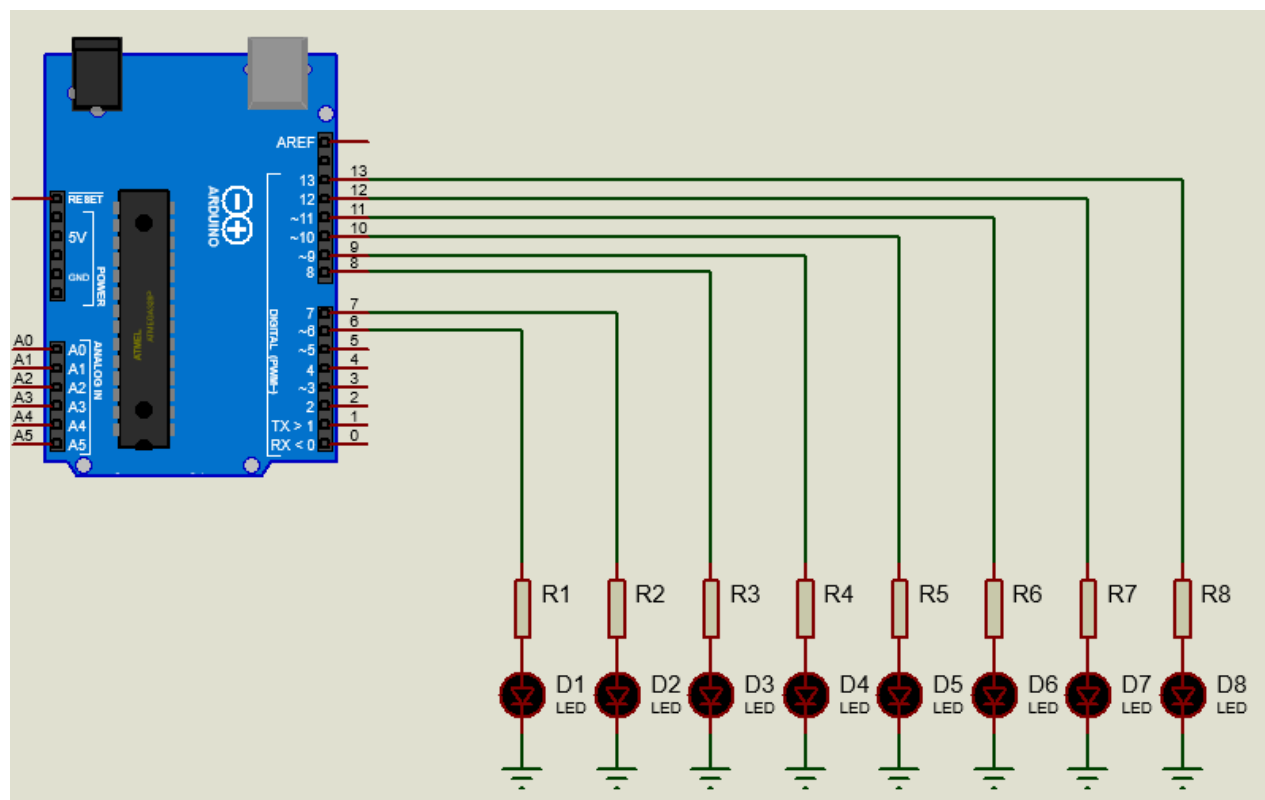
```
void setup(){
  for(int pinIn = 0; pinIn<=7; pinIn++){
    pinMode(pinIn,INPUT);
  }
  for(int pinOut=8; pinOut<=13; pinOut++){
    pinMode(pinOut,OUTPUT);
  }
}
```

En el primer bucle for se configura la variable pinIn como dato entero y se inicia con el valor 0. La condición será que mientras la variable pinIn sea menor o igual que 7, el bucle se repita. En el incremento se hace que cada vez que el bucle se repita le sume 1 a la variable pinIn.

En las instrucciones solo se utiliza la función pinMode y en sus parámetros en lugar de poner el número de pin, se escribe la variable pinIn, de este modo se configurarán los pines desde el 0 al 7 como entradas.

En el segundo bucle se realiza la misma operación, solo se cambia el nombre de la variable a pinOut y se inicia con un valor 8 y en la condición se configura que el bucle se repita mientras la variable pinOut tenga un valor menor o igual a 13.

- Teniendo el circuito de la figura 4-113:



Fuente: Elaboración propia

Figura 4-113 Circuito experiencia 4

- 1.- Calcular las resistencias que van en serie a los leds. Luego implementar el circuito.
- 2.- En el software IDE, configurar los pines utilizando la función for.
- 3.- Crear un programa con la función for en donde se vea un efecto visual de desplazamiento de leds desde el led conectado al pin 0 hast el led del pin 7.
- 4.- Con la función for, crear un programa en que se vea un efecto visual inverso al del punto anterior, es decir, que se vea un efecto de desplazamiento comenzando desde el pin 7 hasta el pin 0.
- 5.- Crear un programa con la función for, en donde se vea un efecto visual de movimiento de los leds, de modo que el led encendido se “desplace” de un lado a otro como efecto metrónomo, esto se puede lograr combinando los programas de los puntos 3 y 4.

Experiencia 5: Bucle while

Objetivos:

- Aprender a utilizar la instrucción WHILE.
- Introducción

La función while es otra forma en la que se puede realizar un bucle y su modo de operación es básicamente repetir las instrucciones entre llaves mientras la condición sea verdadera.

La condición se evalúa cada vez al inicio del bucle, esto quiere decir que si la condición cambia durante la ejecución de las instrucciones entre llaves, el programa no dejará de ejecutarlas hasta que evalúe nuevamente la condición y sea falsa.

Es importante destacar que su funcionamiento es muy similar a la función for, dado que también usa un incremento o decremento, pero la diferencia es que este parámetro va dentro de las instrucciones entre llaves. La variable que elegida para el incremento o decremento (a diferencia de la función for) debe ser declarada al inicio del programa.

- Sintaxis

```
while (condición) {  
    instrucciones;  
}
```

- Parámetros

Condición: la condición a evaluar determinará el inicio y el fin del bucle, estas condiciones se escriben de la misma forma en que se escriben las funciones if o for:

> Mayor que

< Menor que

>= Mayor o igual que

<= Menor o igual que

== Igual a

!= Diferente a

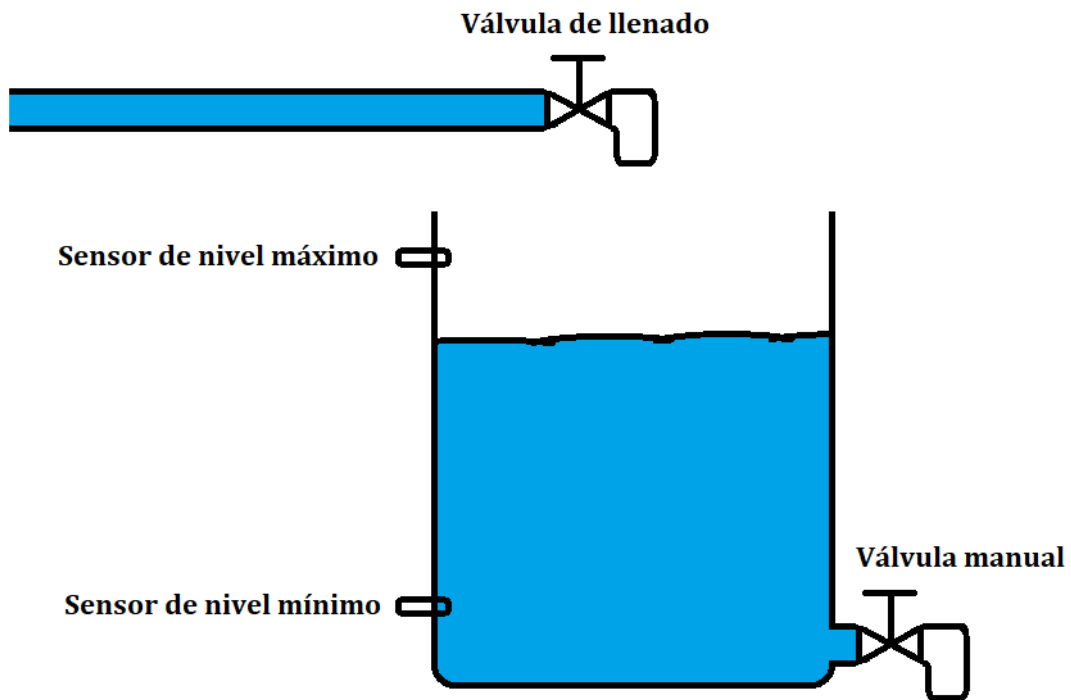
- Ejemplo

Se simulará un sensor final de carrera en la puerta de un refrigerador conectado al pin 0, cuando el sensor detecte que la puerta está abierta enviará un 1, al detectar esto, el Arduino enviará un 1 al pin 1 para encender una luz.

```
void loop(){  
  while(digitalRead(0)==0){  
    digitalWrite(0,LOW);  
  }  
  while(digitalRead(0)==1){  
    digitalWrite(1,HIGH);  
  }  
}
```

Mientras la condición evaluada por la función while sea verdadera, el bucle no se romperá. En el instante en que el Arduino detecte un 1 en el pin 0, la condición se volverá falsa, saliendo del bucle.

- Teniendo el proceso de la figura 4-114:



Fuente: Elaboración propia

Figura 4-114 Sistema de control de nivel en un estanque

Reglas de funcionamiento

La imagen representa un estanque con dos sensores de nivel, uno en la parte inferior indicando que el estanque se encuentra al mínimo y otro en la parte superior indicando que el estanque está lleno.

La válvula de llenado debe activarse cuando el sensor de nivel mínimo lo indique y debe detenerse en el instante en que haya detección del sensor de nivel máximo.

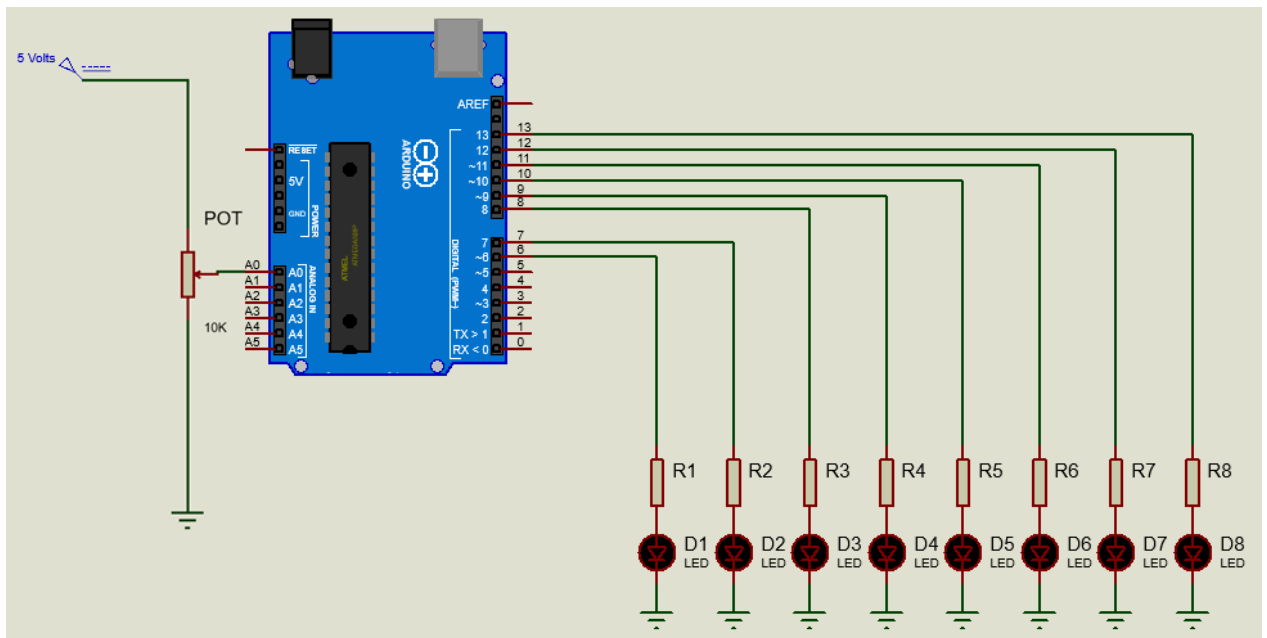
La válvula manual representa una llave común que será operada por personas a medida que vayan sacando agua del estanque, el Arduino no controla esta válvula.

- 1.- Diseñar un circuito con el Arduino para controlar el proceso.
- 2.- Diseñar el diagrama de flujo.
- 3.- Escribir un programa que controle el proceso utilizando funciones while.

Experiencia 6: Entradas Analógicas

Objetivos:

- Utilizar el puerto analógico e interpretar la información.
- Teniendo el circuito de la figura 4-115:



Fuente: Elaboración propia

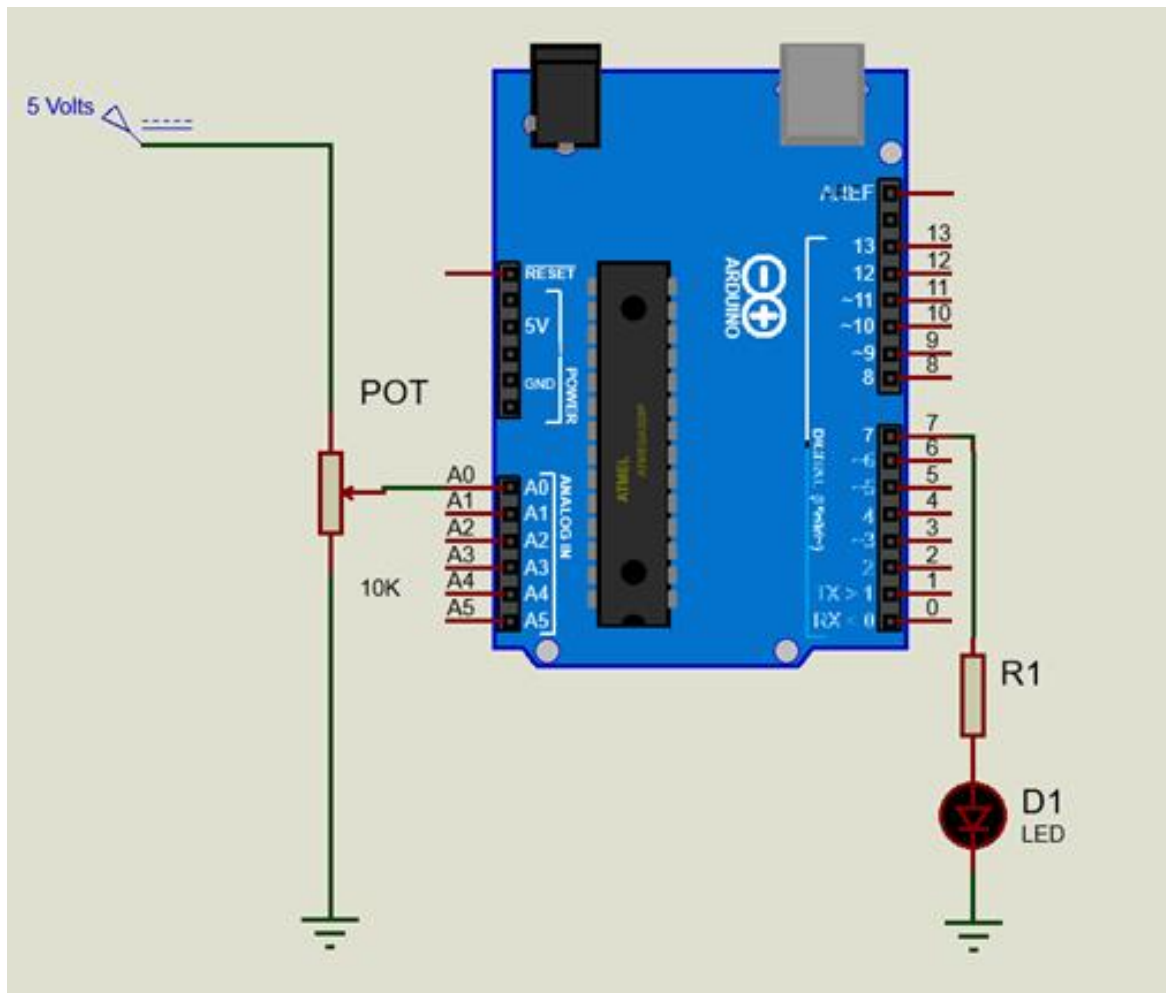
Figura 4-115 Circuito 1 experiencia 6

1.- Calcular las resistencias para cada led.

2.- Implementar el circuito.

3.- En el software IDE, diseñar un programa que opere detectando el voltaje del potenciómetro en la entrada analógica y vaya encendiendo un led por cada 625mV a medida que el voltaje aumenta. El programa también debe apagar los leds a medida que el voltaje disminuye. En caso de ser necesario, utilizar diagramas de flujo antes de comenzar a escribir el programa.

- Teniendo el circuito de la figura 4-116:



Fuente: Elaboración propia

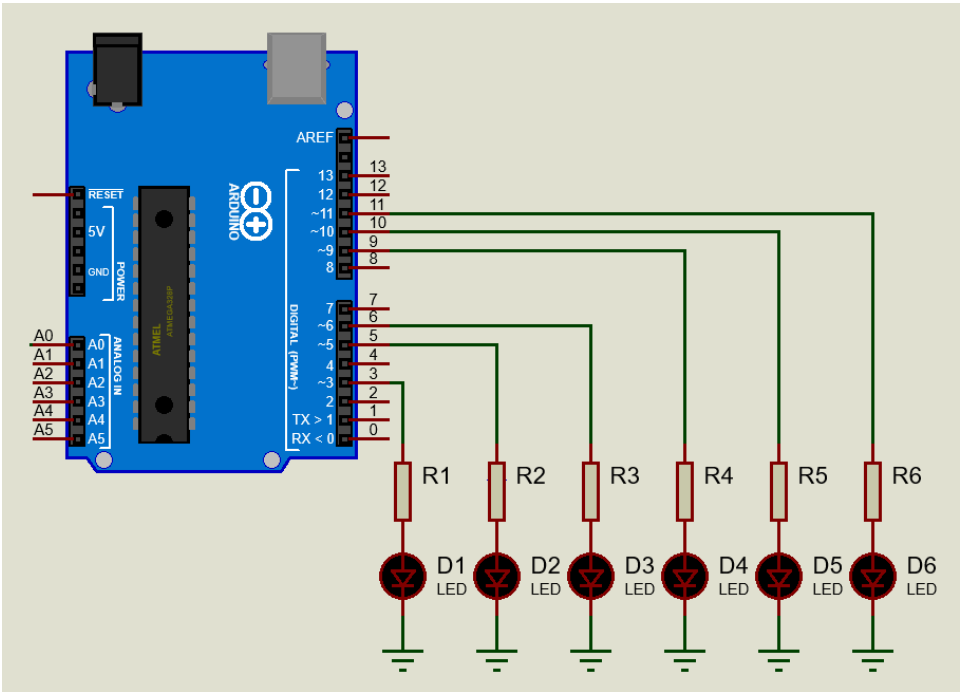
Figura 4-116 Circuito 2 experiencia 6

- 1.- Calcular la resistencia para el led.
- 2.- Implementar el circuito.
- 3.- En el software IDE, diseñar un programa que detecte el voltaje en la entrada analógica. El led debe encenderse y apagarse a frecuencias en que sea posible visualizar el encendido y apagado. La frecuencia de encendido y apagado del led debe variar en función del voltaje detectado en la entrada analógica. Considerar que si en la entrada analógica hay 0v el led debe estar apagado y si se detectan 5v el led debe estar encendido.

Experiencia 7: PWM

Objetivos:

- Utilizar salidas PWM.
- Aplicar conocimientos de experiencias anteriores.
- Teniendo el circuito de la figura 4-117:

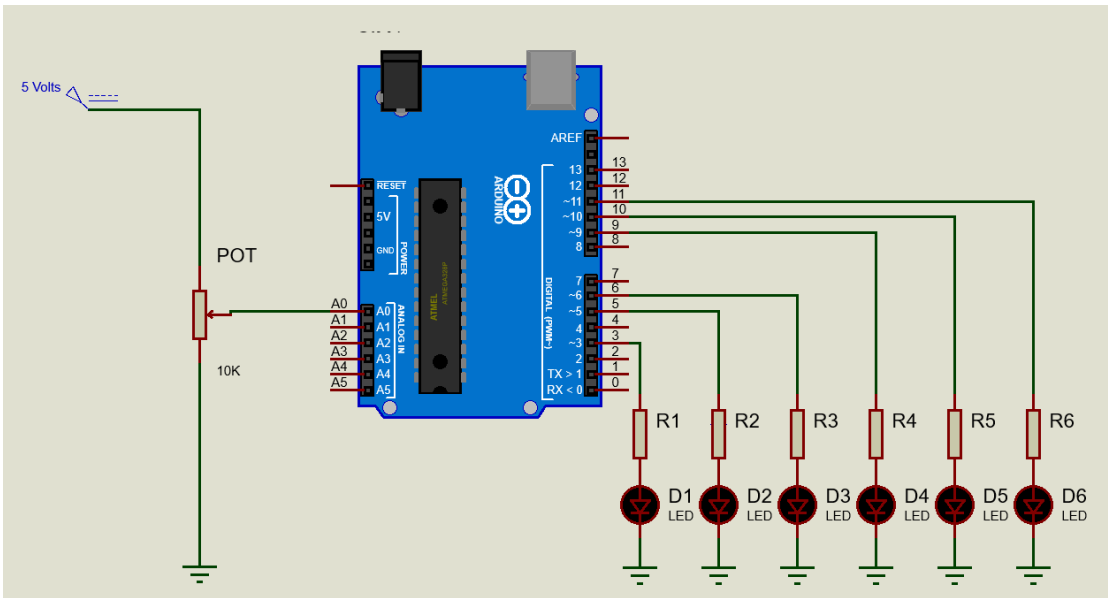


Fuente: Elaboración propia

Figura 4-117 Circuito 1 experiencia 7

- 1.- Calcular las resistencias para cada led.
- 2.- Implementar el circuito.
- 3.- En el software IDE, diseñar un programa en que los leds se enciendan todos al mismo tiempo, pero que su luminosidad aumente de forma gradual tardando dos segundos en encenderse totalmente y que luego se apaguen al mismo tiempo también de forma gradual tardando 2 segundos en apagarse totalmente, el ciclo debe repetirse de forma periódica.

- Teniendo el circuito de la figura 4-118:



Fuente: Elaboración propia

Figura 4-118 Circuito 2 experiencia 7

1.- Diseñar un programa en que la luminosidad de los leds varíe según el voltaje detectado en la entrada analógica, cuando el Arduino detecte 0v los leds deben estar apagados y cuando detecte 5v deben estar encendidos.

Experiencia 8: Control de Servomotor

Objetivos:

- Aprender a controlar un servomotor.
- Introducción

El servomotor es un motor que permite tener un movimiento muy preciso y con un gran torque. Internamente está compuesto por un motor de corriente continua, un circuito de control que entrega 3 cables y una caja de engranajes que reducirá la velocidad y aumentará el torque. A través de los cables se puede conectar el terminal positivo, tierra y la señal de información para el movimiento del motor. Todo este sistema estará ensamblado dentro de una caja plástica, lo que es visible desde el exterior son el eje del motor y los 3 cables para alimentación y control.

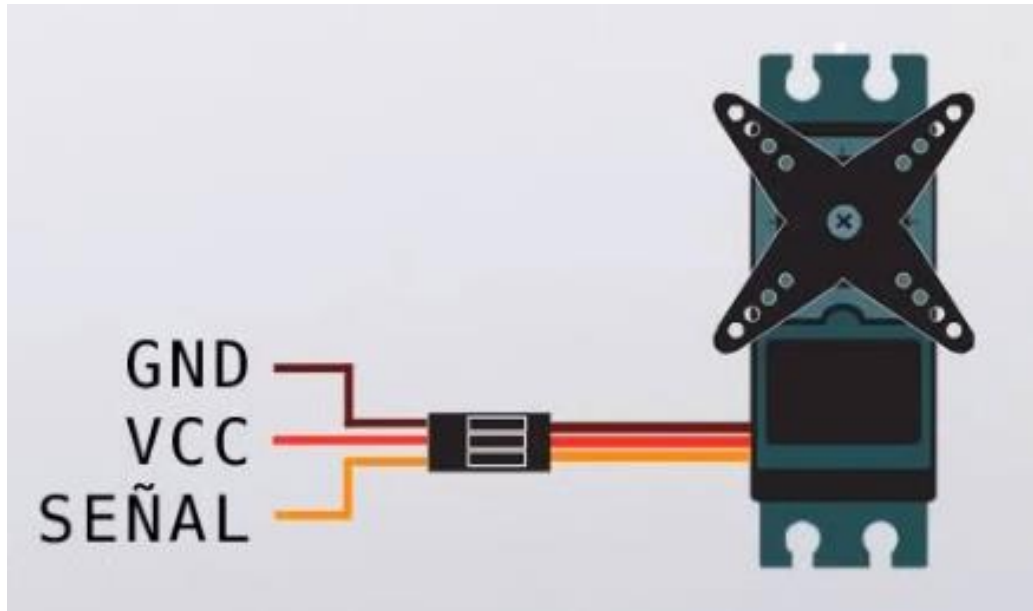


Fuente: <https://fidiasrodriguez.com/que-es-un-servomotor/>

Figura 4-119 Imagen Referencial de un Servomotor Desarmado

- Conexión del servomotor

Como se ha mencionado, para conectarlo se tienen 3 cables. Generalmente el cable negro es para conectar la tierra y el cable rojo para la alimentación, aunque los colores pueden variar dependiendo del fabricante.

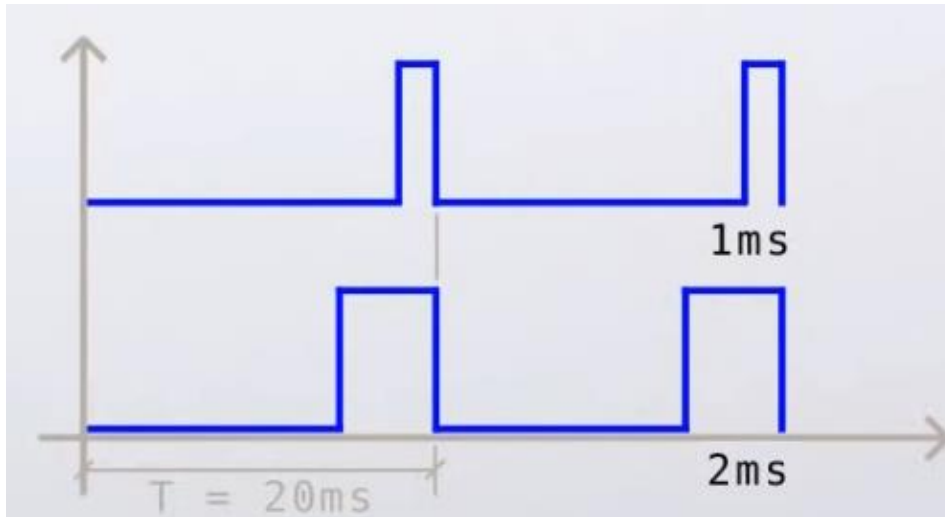


Fuente: <https://descubrearduino.com/servomotores/>

Figura 4-120 Función de cada cable del servomotor

Existen más combinaciones de colores para los cables del servomotor, en caso de que existan dudas sobre cómo conectarlo, se deberá consultar en la página del fabricante con el modelo del servo.

- Señal de control para el servo



Fuente: <https://saraih1008.wixsite.com/roboticaeducativa/copia-de-4-2-led-rgb>

Figura 4-121 Señal 1ms para posicionar el servo en 0° y señal de 2ms para 180°

Las señales de la figura 2-3 son para que el servo se mueva a sus límites mínimos y máximos, para ajustarlo en alguna posición intermedia habrá que variar el ancho del pulso entre 1 y 2 mili segundos. Estas señales son del tipo PWM.

- Pasos necesarios en software IDE para operar el servomotor

Paso 1: Importar la librería dedicada a servomotores, para ello se debe escribir:

```
#include <Servo.h>
```

Paso 2: Dar nombre al motor servo para poder referenciarlo más adelante en el programa.

```
Servo nombre_del_servo;
```

Este paso crea un objeto del tipo servo y lo definirá con el nombre que se escriba en el parámetro "nombre_del_servo"..

Paso 3: Dentro de la función `void setup()` se utilizará la siguiente función:

```
nombre_del_servo.attach(pin_servo, mínimo, máximo);
```

El nombre del servo será el mismo que haya sido asignado en el paso 2.

El parámetro `pin_servo` se refiere al pin del arduino en que está conectada la señal de control del servo.

Los parámetros mínimo y máximo se refieren a la duración de los pulsos de la señal de control para posicionar el servo en 0° y 180° respectivamente. En estos parámetros se debe escribir 1000 para el mínimo y 2000 para el máximo ya que esta función trabaja con microsegundos.

Paso 4: Dentro de la función `void loop()` se utilizará la siguiente instrucción para mover el servo a la posición deseada.

`Nombre_del_servo.write(posición);`

En el parámetro posición se debe escribir el ángulo al que se desea mover el servo, debe ser un valor entre 0 y 180.

Paso 5 (adicional): Función necesaria para convertir valores

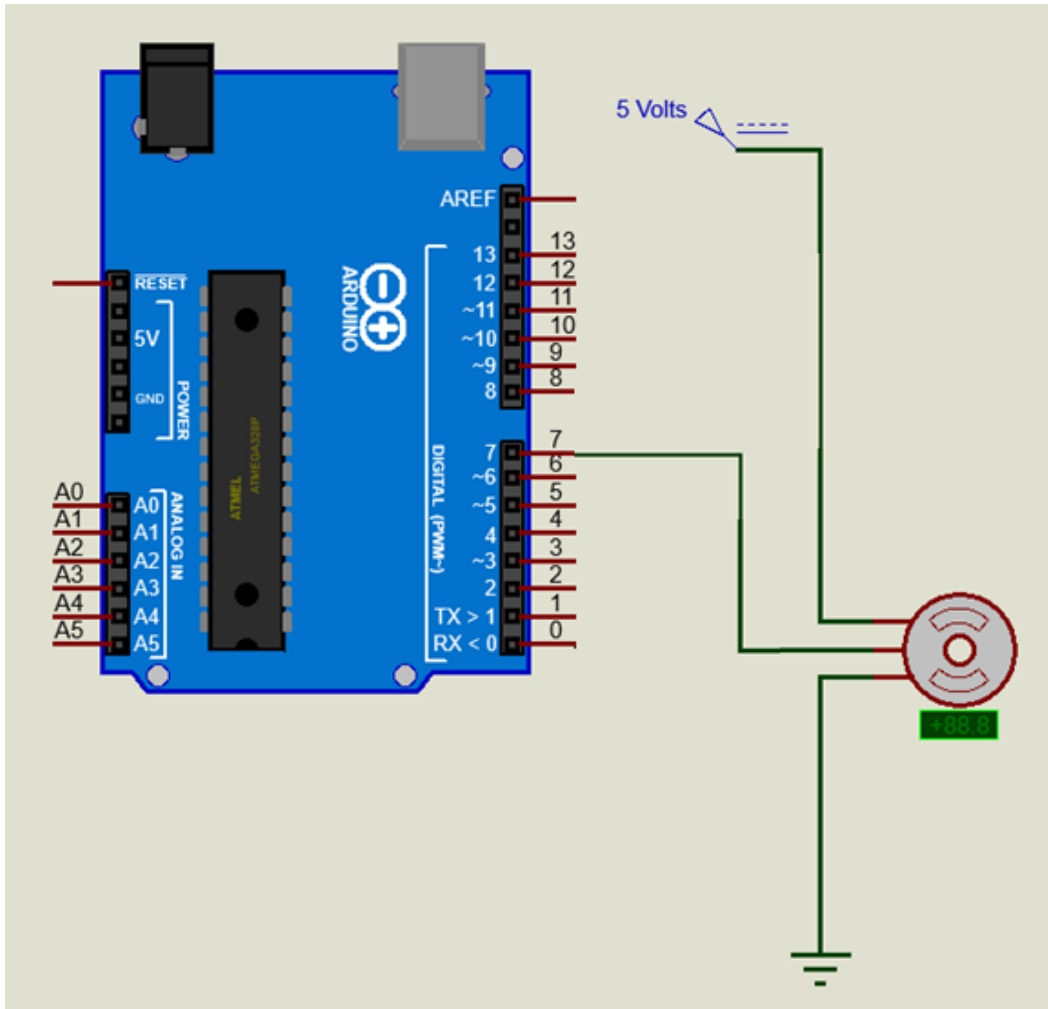
`map(variable, valor_mínimo1, valor_máximo1, valor_mínimo2, valor_máximo2);`

Esta función sirve para escalar valores de una variable que tengan mínimos y máximos a otra nueva escala. Es muy utilizada cuando se trabaja con valores analógicos y convertirlos.

Los valores mínimo 1 y máximo 1 corresponden a la escala en que trabaja la variable.

Los valores mínimo 2 y máximo 2 corresponden a la nueva escala a la que será convertido el valor de la variable.

- Teniendo el circuito de la figura 2-4

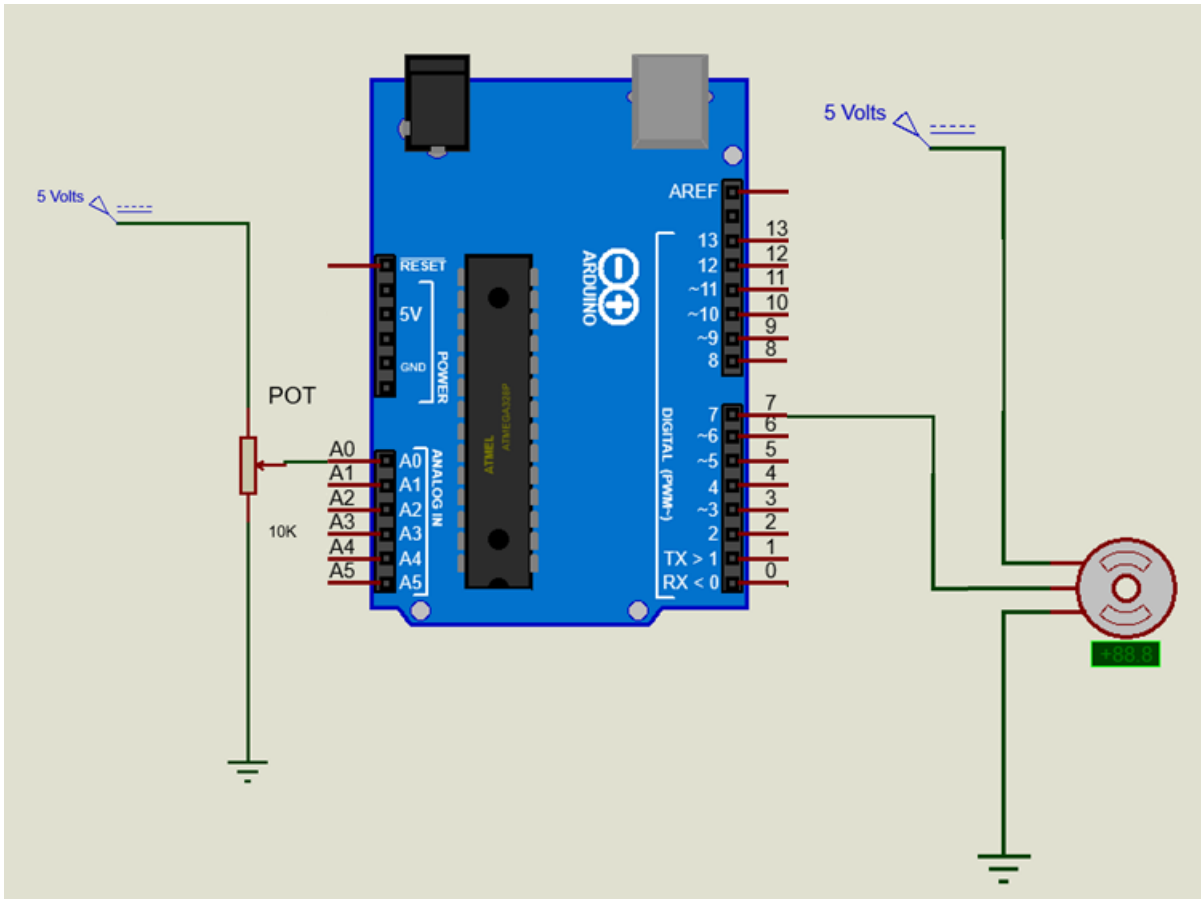


Fuente: Elaboración propia

Figura 4-122 Circuito para controlar un servomotor con Arduino

- 1.- Implementar el circuito y conectar el arduino a tu PC.
- 2.- Diseñar un programa que mueva el servo a 0°, 90° y 180°. El servo deberá cambiar de posición cada 3 segundos, la secuencia será: 0° - 90° - 180° - 90°. El arduino deberá repetir la secuencia periódicamente.

- Teniendo el circuito de la figura 2-5



Fuente: Elaboración propia

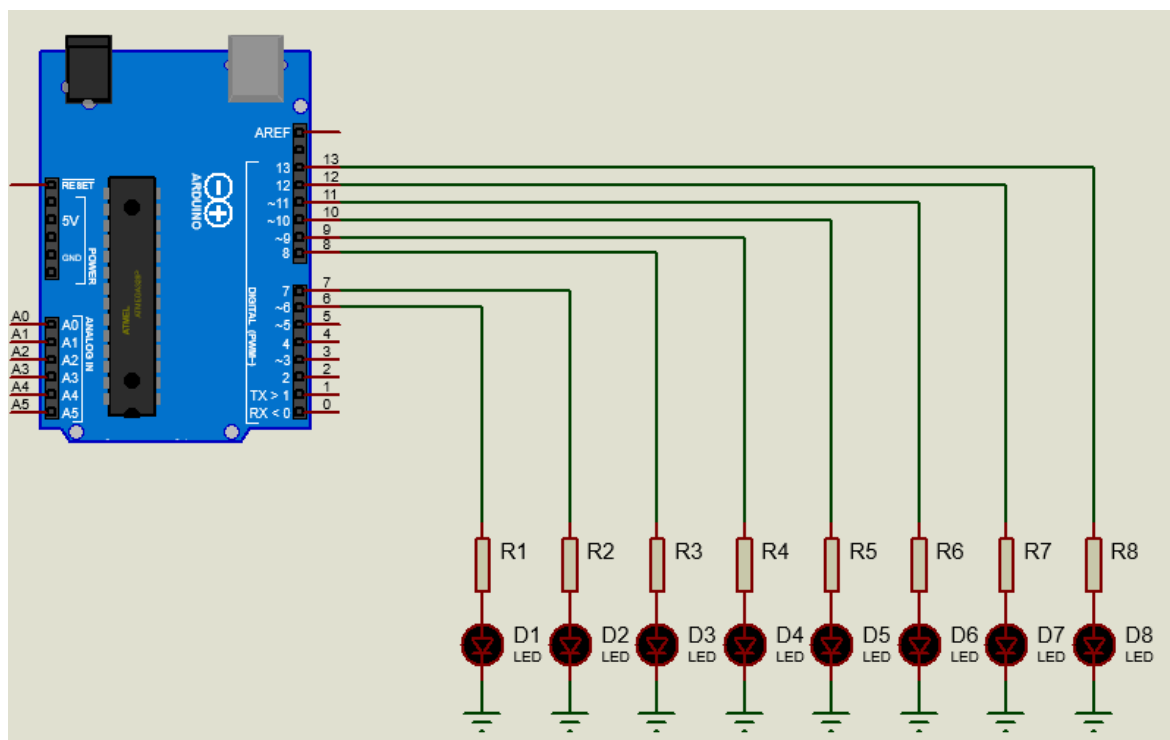
Figura 4-123 Circuito para controlar servomotor con Arduino mediante un potenciómetro

- 1.- Implementar el circuito y conectar el arduino a tu PC.
- 2.- Diseñar un programa en que el servomotor se mueva según la variación de voltaje generada por el potenciómetro. Los límites mínimos y máximos serán $0v = 0^\circ$ y $5v = 180^\circ$ respectivamente.

Experiencia 9: Comunicación Serial

Objetivo:

- Aplicar conocimientos de comunicación serial.
- Teniendo el circuito de la figura 4-124:



Fuente: Elaboración propia

Figura 4-124 Circuito 1 experiencia 9

1. Implementar el circuito
- 2.- Diseñar un programa en que el Arduino reciba datos en el puerto serial provenientes del teclado del PC y dependiendo de lo recibido encienda secuencias de leds:

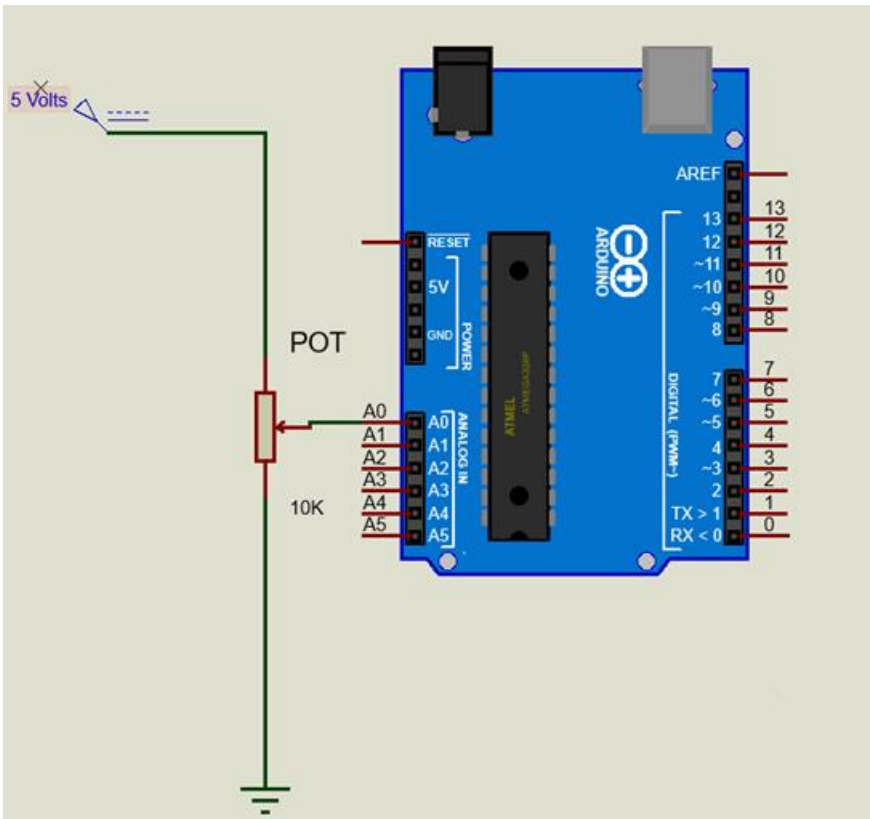
Si recibe una letra a: Secuencia 1

Si recibe una letra b: Secuencia 2

Si recibe una letra c: Secuencia 3

Si recibe una letra d: Secuencia 4

- Teniendo el circuito de la figura 4-125:



Fuente: Elaboración propia

Figura 4-125 Circuito 2 experiencia 9

- 1.- Implementar el circuito
- 2.- Diseñar un programa que envíe el voltaje leído en la entrada analógica a través del puerto serial. Usar el monitor serial del programa IDE para visualizar la información. El programa debe enviar la lectura instantánea.

Experiencia 10: Interrupciones

Objetivos:

- Aprender a programar interrupciones.
- Introducción

Una interrupción es un sistema dentro del Arduino que asocia una función a la ocurrencia de un evento externo específico, a esta función se le conoce como ISR (Interruption Service Routine) y solamente se ejecuta cuando ocurre el determinado evento externo vinculado a la interrupción. Al ocurrir el evento, se pausa la ejecución del programa normal y el Arduino ejecuta el código de la rutina ISR, una vez terminada esta rutina continua la ejecución del código principal desde donde se produjo la pausa.

Gracias a las interrupciones se dispone de un sistema que será de gran ayuda para detectar inmediatamente un evento externo sin la necesidad de estar leyendo todo el tiempo el pin donde se reciba la señal asociada.

Generalmente cuando se espera algún evento externo se utilizan instrucciones como if o while, pero el problema que tienen es que mantienen ocupado al microcontrolador, mientras que en el caso de las interrupciones esto no ocurre y el Arduino puede ejecutar libremente el código principal.

- Pines de interrupciones

En los Arduinos existen ciertos pines que además de sus funciones normales (como entradas analógicas o PWM) están dedicados para interrupciones. En el caso de Arduino UNO son los pines 2 y 3.

- Instrucciones para configurar una interrupción

Para configurar una interrupción se utiliza la siguiente instrucción:

```
attachInterrupt(pin, función, modo_de _activación);
```

La instrucción attachInterrupt debe ir dentro de la función `void setup()`

- Parámetros de la función attachInterrupt()

-Parámetro pin: Este parámetro debe ser configurado cuidadosamente ya que configura el pin que será utilizado para detectar un evento externo y activar la interrupción.

Existen dos formas de configurar este parámetro:

1.-En la primera forma no se debe escribir el número de pin normal, se debe escribir el número ordinal. Esto significa que como el arduino tiene los pines 2 y 3 dedicados a interrupciones, si se desea utilizar el pin 2 se escribe 0 y si se desea utilizar el pin 3 se escribe 1.

Por ejemplo, si se desea utilizar el pin 2 como interrupción se debe escribir:

```
attachInterrupt(0, función, modo_de _activación);
```

2.- En la segunda forma, en lugar de poner el número de pin ordinal se puede la función `digitalPinToInterrupt(pin)`, en esta función al escribir el número de pin 2 o 3 en su parámetro, le indicará al arduino el número ordinal de interrupción (0 o 1).

Como ejemplo si se desea utilizar el pin 3 para interrupción se escribe

```
attachInterrupt(digitalPinToInterrupt(3), función, modo_de _activación);
```

-Parámetro función: es la función que se llamará cuando se produzca la interrupción, puede ser nombrada como lo desee el usuario. El detalle más importante es que la función que llamada no admite parámetros y por lo tanto no devuelve ningún valor.

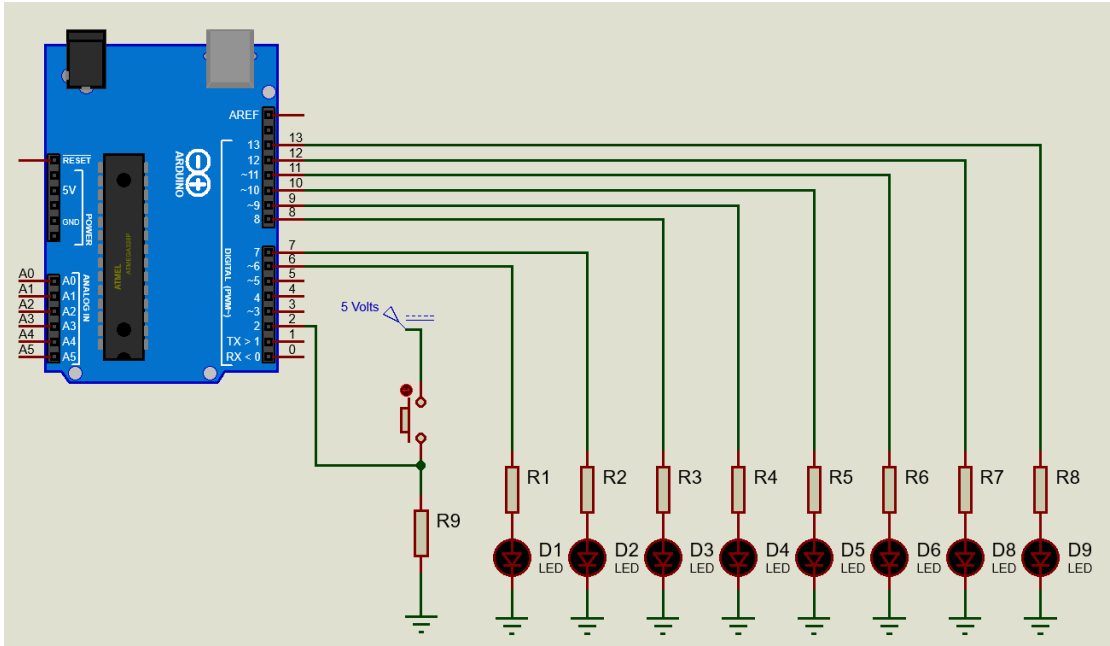
-Modo de activación: configura la forma en que se activa la interrupción, existen cuatro formas de activación

- 1.- RISING: se activa la interrupción cuando el pin cambie de 0 a 1 (canto de subida).
- 2.- FALLING: se activa la interrupción cuando el pin cambie de 1 a 0 (canto de bajada).
- 3.-CHANGE: se activa la interrupción cuando el pin cambie de estado, puede ser de 0 a 1 o de 1 a 0 (se activa con canto de subida y/o bajada).
- 4.- LOW: se activa la interrupción cuando el pin esté en 0.

Ejemplo de una interrupción que se activará cuando el pin 2 cambie de 0 a 1

```
attachInterrupt(0, función, RISING);
```

- Teniendo el circuito de la figura 4-126:

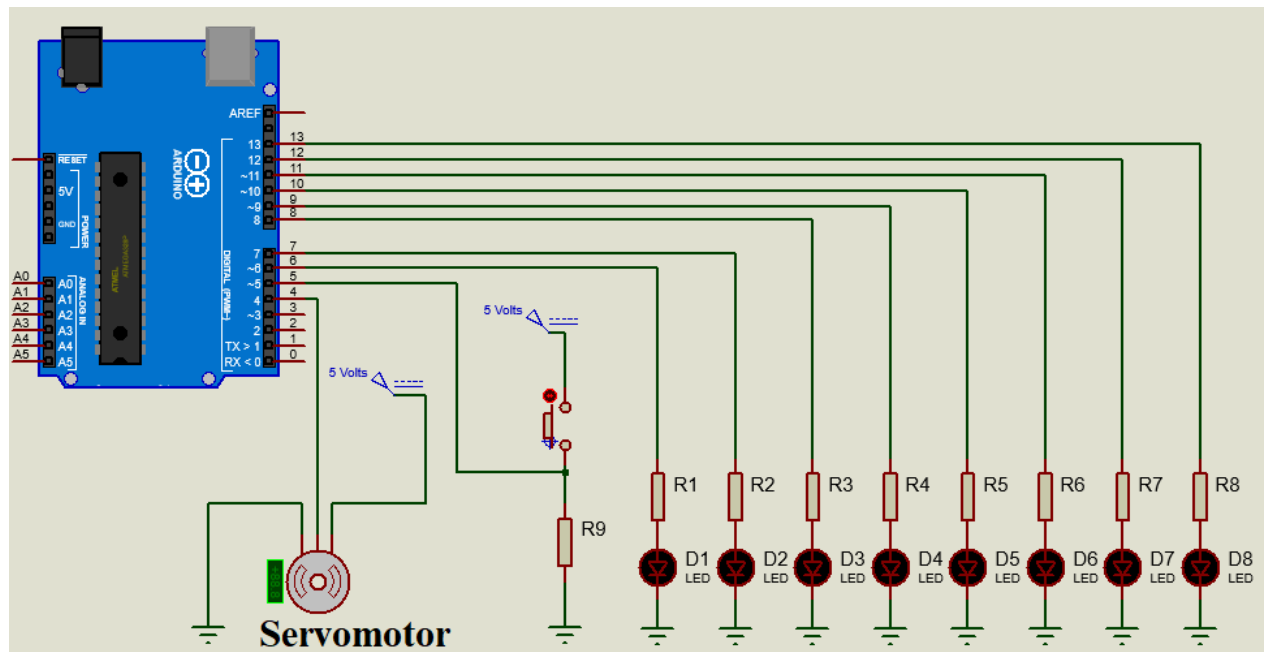


Fuente: Elaboración propia

Figura 4-126 Circuito 1 experiencia 10

- 1.- Calcular los valores de las resistencias.
- 2.- Implementar el circuito.
- 3.- Diseñar en un programa en que el arduino haga una secuencia de leds, en el momento en que se presione el pulsador deberán quedar los leds encendidos durante 10 segundos a través de una interrupción.

- Teniendo el circuito de la figura 4-127:



Fuente: Elaboración propia

Figura 4-127 Circuito 2 experiencia 10

- 1.- Calcular los valores de las resistencias.
- 2.- Implementar el circuito.
- 3.- Diseñar en un programa en que el Arduino haga una secuencia de leds, en el momento en que se presione el pulsador mediante una interrupción los leds deberán apagarse y el servomotor deberá moverse dos veces de entre 0° y 180°.