INSTITUTO SUPERIOR

UNIVERSITARIO

sure

GUÍA GENERAL DE ESTUDIO DE LA ASIGNATURA DE MICROCONTROLADORES

x|

Guía general de estudio de la asignatura de Microcontroladores

Fabricio Manuel Tipantocta Pillajo

Gladys del Rocío Herrera Panchi

Carlos Germánico Rivera Liger

Christian Andrés Ortega Hidalgo

2024

Esta publicación ha sido sometida a revisión por pares académicos específicos por:

Silvia Alexandra Ayala Trujillo
Instituto Superior Universitario Cotopaxi

Corrección de estilo:

- Karla Jaramillo - Docente - Sucre
- Freddy Centeno - Docente - Sucre
- Ana Llumiquinga - Docente - Sucre

Diseño y diagramación:

- Ronny Chaguay - Docente - Sucre
- Diego Bonilla - Docente - Sucre

Primera Edición
Quito - Ecuador

ISBN: 978-9942-676-25-2

MISIÓN

Ser una Institución Superior Universitaria con estándares de calidad académica e innovación, reconocida a nivel nacional con proyección internacional.

VISIÓN

Formamos profesionales competentes con espíritu emprendedor, capaces de contribuir al desarrollo integral del país.

ÍNDICE

Presentación de la asignatura

La asignatura de microcontroladores es fundamental dentro del campo de la ingeniería electrónica y de sistemas, ya que proporciona a los estudiantes los conocimientos necesarios para comprender, diseñar y programar sistemas embebidos. Los microcontroladores son dispositivos electrónicos altamente integrados que incorporan una unidad de procesamiento, memoria, periféricos de entrada/salida y otros componentes, lo que los hace ideales para una amplia gama de aplicaciones en la industria, desde dispositivos domésticos hasta sistemas de control industrial avanzados. En esta asignatura, los estudiantes explorarán los principios fundamentales de los microcontroladores, aprenderán a programar en lenguaje como el C++, y desarrollarán habilidades prácticas para diseñar y prototipar sistemas electrónicos basados en microcontroladores.

Resultados del aprendizaje

Identifica la arquitectura de un microcontrolador y reconoce periféricos internos y externos para uso de sistemas de automatización.

Interpreta la lógica de manejo de electrónica digital con los sistemas micro controlados y aplica en casos de automatización industrial.

Reconoce módulos especiales internos del microcontrolador que se pueden usar en electrónica.

Aplica los conocimientos para el desarrollo de sistemas micro controlados de automatización industrial casos de control con lazo realimentado.

Identifica la arquitectura de un microcontrolador y reconoce periféricos internos y externos para uso de sistemas de automatización.

Interpreta la lógica de manejo de electrónica digital con los sistemas micro controlados y aplica en casos de automatización industrial.

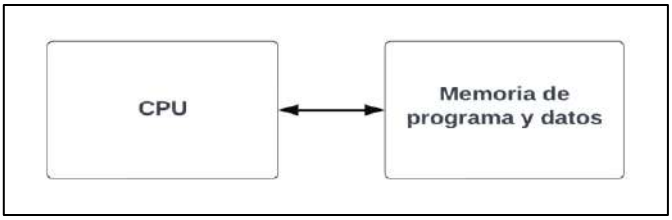
Arquitectura de los microcontroladores y tarjeta Arduino

La arquitectura de un microcontrolador es el diseño interno que define su funcionamiento y capacidades. Estos dispositivos compactos integran en un solo chip una unidad central de procesamiento (CPU), memoria, periféricos de entrada/salida y otros componentes esenciales para ejecutar programas y controlar dispositivos externos.

Arquitectura Von Newman

Esta arquitectura es un diseño que usa una memoria para almacenar instrucciones y datos. En este tipo de arquitectura los datos y las instrucciones se guardan en la misma memoria por lo que transitan por el mismo bus (Alonso, 2024), se muestra en la Figura 1.

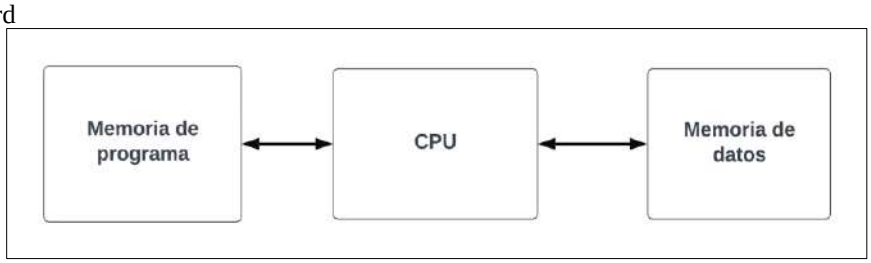
Figura 1
Arquitectura Von Newman



Arquitectura de Harvard

La arquitectura Harvard utiliza dos buses distintos debido a que los datos y las instrucciones se almacenan en dos memorias diferentes. El programa se almacena como un código numérico en la memoria. Esto permite que la CPU pueda trabajar con las dos memorias al mismo tiempo y por ende la ejecución de los programas es mucho más rápida. (Alonso, 2024), se muestra en la Figura 2.

Figura 2
Arquitectura Harvard



Para conocer a detalle los componentes a detalle que integran un microcontrolador de manera interna, se muestra la siguiente descripción.

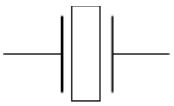
Bus de datos

Los microcontroladores utilizan buses de datos para la transferencia de información entre diferentes componentes dentro del microcontrolador. Estos buses son conjuntos de líneas bidireccionales que permiten el movimiento de datos en forma de bits (1s y 0s) entre el Microprocesador y los demás elementos del sistema, para el envió de instrucciones o datos. El número de líneas que posee es generalmente igual al número de bits que procesa la ALU. (Velarde, 2009).

Cristal (oscilador)

Los osciladores de cristal basados en el cuarzo son el componente responsable de la precisión de la frecuencia/temporización y del rendimiento en casi todos los circuitos electrónicos (Schweber, 21). En los microcontroladores el cristal actúa como resonador de reloj. Establece la frecuencia del cristal el cual determina la velocidad a la que la CPU ejecuta las instrucciones y sincroniza las operaciones del microcontrolador, el Arduino dispone de un cristal de 16 MHz.

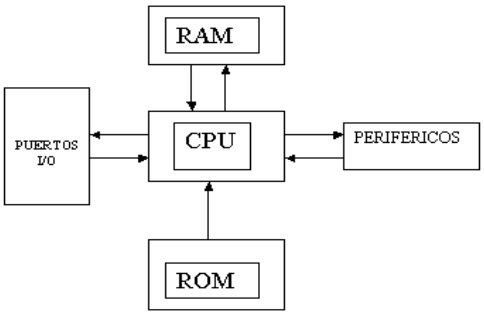
Figura 3
Oscilador



Periféricos internos

Los microcontroladores suelen incluir varios periféricos dentro del mismo chip que permiten la comunicación con el entorno externo. Los periféricos internos de un Arduino son los componentes que están integrados directamente en el microcontrolador los cuales proporcionan funciones adicionales

Figura 4
Periféricos internos



Alu

La unidad aritmética lógica es el corazón de la capacidad del procesamiento del microcontrolador, la cual, se encarga de realizar las operaciones aritméticas del procesamiento lógico, permitiéndole ejecutar los cálculos necesarios para controlar dispositivos externos o responder a entradas del usuario. Los ALU trabajan con bytes o conjuntos de 8 bits, 16 bits o 32 bits, dependiendo de la arquitectura del microcontrolador (Mariana, 2024).

Memoria Flash

Es una memoria no volátil, lo que significa que no pierde la información guardada incluso quitando tensión o apagando el microcontrolador, se utiliza para introducir el programa que se quiera ejecutar por el microcontrolador (Jecrespom, 2017). Permite la escritura y la lectura de datos de manera rápida y eficiente.

Memoria Ram

El Static Random Access Memory (SRAM), o Memoria Estática de Acceso Aleatorio es una memoria volátil lo que significa que perderá la información almacenada si se quita la alimentación o se desconecta el microcontrolador, el tipo de información que se almacena en esta memoria es variable y se puede sobrescribir en el transcurso del programa. (Jecrespom, 2017).

Memoria Eeprom

El Electrically Erasable Programmable Read-Only Memory es un tipo de memoria no volátil, en el que se puede sobrescribir contenido de un dato, esto quiere decir que se puede borrar y grabar datos una vez cargado el programa. Por lo tanto, es posible guardar datos en variables y mantener esta información intacta incluso después de haber quitado tensión al microcontrolador. (Jecrespom, 2017).

Para fines de desarrollo de la materia de microcontroladores, se ha escogido el manejo de la tarjeta Arduino Mega, que contiene el microcontrolador ATmega 2550 y se usa la plataforma embebida tanto en hardware como en software.

Programación de microcontroladores en c++

C++ es un lenguaje de programación con el fin de poder realizar múltiples tareas de propósito general, mayormente conocido por su eficiencia y potencia, este mismo logra combinar la programación

orientada a objetos con la programación procedural y genérica. La utilización de C++ en microcontroladores abre la puerta a un desarrollo más estructurado y modular, facilitando la reutilización de código, el mantenimiento y la escalabilidad del sistema (Oliag, 1995).

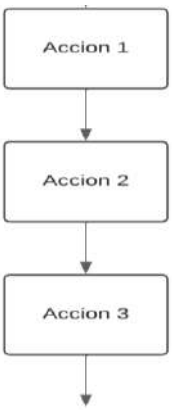
Programación estructurada

Esta forma de programar esta derivada de un famoso teorema, desarrollado por Edsger Dijkstra, la cual tiene el fin de demostrar que todo programa puede escribirse utilizando únicamente 3 sentencias básicas de control las cuales son las siguientes:

Sentencia secuencial

Es el bloque de instrucciones de forma secuencial, lo que significa que las instrucciones que sé que configura se ejecutan una tras otra en el orden en el cual se configura el código, para lograr esto se utiliza declaraciones simples y secuenciales como se puede observar a continuación:

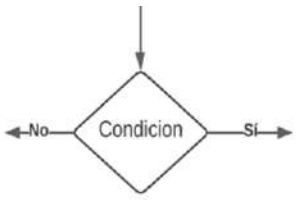
Figura 5
Diagrama de flujo secuencial



Sentencia condicional

Es una instrucción que permite que el programa tome decisiones las cuales están basadas en las condiciones específicas que configure el usuario, esto se logra mediante la instrucción ´if´, la cual ejecuta un bloque de código si se cumple una condición, también se utiliza la instrucción ´if-else´ la cual ejecuta un bloque de código si la condición en verdadera y otro si la información es falsa.

Figura 6
Diagrama de flujo condicional

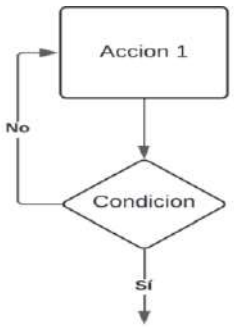


Sentencia de repetición

Es una condicional que permite que un bloque de código se ejecute repetidamente mientras esta misma cumpla una condición específica, esto se logra mediante el bucle ´while´

Figura 7

Diagrama de flujo de repetición



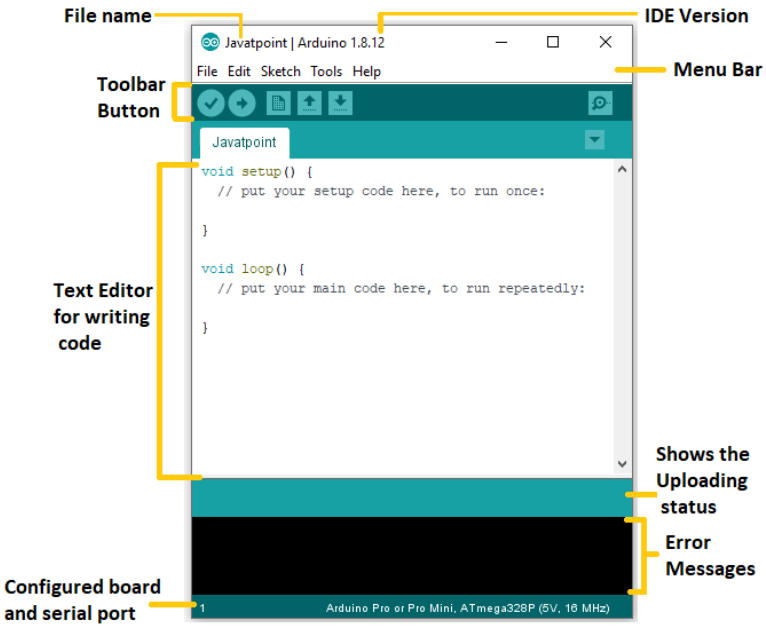
Al utilizar sentencias de control, el programador puede expresar un código claro, legible y fácil de entender lo cual facilita el mantenimiento y la depuración del software. (Mheducation, 2013)

Entorno de programación Ide Arduino

Es un conjunto de herramientas y software desarrollado con el fin de programar placas y microcontroladores de Arduino, el IDE arduino ofrece un entorno de desarrollo completo con el fin de escribir, compilar y cargar un código en placas Arduino. (Ruiz Guti, 2007).

Figura 8

IDE Arduino



Simulación con software especializado

Arduino y software especializado son herramientas indispensables para la simulación y desarrollo de proyectos electrónicos. Con software especializado es posible diseñar y simular circuitos electrónicos de manera intuitiva y eficiente, mientras que Arduino ofrece una plataforma versátil para programar y controlar dispositivos físicos. Al combinar ambas plataformas, los usuarios pueden crear y probar prototipos de forma virtual antes de implementarlos en el mundo real.

En el siguiente enlace podrá visualizar el esquema de conexión y el código a utilizar del ejemplo de un semáforo:

https://github.com/carreraElectronica/Unidad1/tree/1f1a6b9b113534a85e624f2b6d05641c3f0bfac0/Semaforo

Manejo de periféricos de entrada y salida digitales

El microcontrolador de Arduino cuenta con pines de entrada y salida para así comunicarse con el exterior. Estos diversos pines tienen la característica especial que los hace excelentes para una u otra tarea que estén en función del tipo pin, estos mismos pueden encontrarse en diferentes tipos como lo son: entradas y salidas digitales, entradas y salidas analógicas, entradas y salidas para la comunicación serial. (Diosdado, 2014)

Los pines asignados a entradas y salidas digitales, son aquellos pines que trabajan con dos estados HIGH que significa alto y dependiendo el Arduino que usemos toma el estado como 5V o 3,3V, y LOW que significa bajo y normalmente está asociado a 0V.

Configuración de entrada digital

Para realizar la configuración de los pines como entrada digital se usa la función pinMode, donde se le asigna el pin a usar y declararlo como entrada (INPUT), o también se le puede asignar como entrada con resistencia externa (INPUT_PULLUP), la cual ayudará a conectar componentes que envíen señales digitales sin necesidad de una resistencia externa.

- pinMode(#pin,INPUT), o
- pinMode(#pin,INPUT_PULLUP)

Configuración de salida digital

Para realizar la configuración de los pines como entrada digital se usa pinMode, donde se asigna el pin a usar y se declara como salida (OUTPUT), para el caso de las salidas digitales no se puede asignar una resistencia interna, pero se puede también declarar los puertos con DDRx.

- pinMode (#pin, OUTPUT)

Lecturas de entradas digitales (digitalRead)

Al realizar la lectura de una entrada digital, vamos a leer el valor que el pin tiene y se almacenará como HIGH o como LOW, la siguiente línea de código se escribirá la función setup (Diosdado, 2014)

- digitalRead (#pin)

Escritura de salidas digitales (digitalWrite)

Cuando se realiza la escritura en algún pin digital se envía el estado lógico HIGH o LOW al pin que previamente se ha asignado, en el caso de usar el valor HIGH se envía 5V al pin asignado, y LOW envía 0V. (Diosdado, 2014)

- digitalWrite(#pin,HIGH)
- digitalWrite(#pin,LOW)

A continuación, se muestra un ejemplo de entradas y salidas digitales:

```
void setup () {
  pinMode (8, OUTPUT);
  pinMode (9, INPUT_PULLUP);
}
void loop () {
  int p1= digitalRead (9);
  digitalWrite (8, HIGH);
}
```

Manejo de elementos de electrónica básica con Arduino

El manejo de elementos de electrónica básica con Arduino es fundamental para desarrollar una amplia variedad de proyectos, en este tema se repasará cómo manejar los elementos.

Elementos de electrónica básica

Led

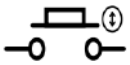
Es un dispositivo semiconductor que emite luz cuando una corriente eléctrica lo atraviesa en la dirección adecuada.

Figura 9
Led

Pulsador

Es un dispositivo simple utilizado para controlar el flujo de corriente eléctrica en un circuito.

Figura 10
Pulsador



Dipswitch

Es un componente electrónico que se utiliza para configurar o seleccionar diferentes opciones de funcionamiento en dispositivos electrónicos.

Figura 11
Dipswitch

Resistencia

Es un componente electrónico pasivo diseñado para limitar o controlar el flujo de corriente eléctrica en un circuito.

Figura 12
Resistencia

Aplicaciones digitales con eventos discretos

Ejercicio de aplicación blink

Para el caso de la aplicación con blink se usa 4 leds, los cuales permanecerán prendidos por 1 segundo y luego apagados por 1 segundo.

El código y el esquema a utilizar en esta aplicación lo podrán visualizar en el siguiente enlace:

https://github.com/carreraElectronica/Unidad1/tree/1f1a6b9b113534a85e624f2b6d05641c3f0bfac0/Aplicacion-Blink

Ejercicio de aplicación de un juego de luces

En el caso de los juegos de luces se usará el puerto A, se realizará un juego de luces que encienda los leds de los extremos al centro y de regreso hacia afuera.

El código y ele esquema a utilizar para este circuito se lo puede visualizar en el siguiente enlace:

https://github.com/carreraElectronica/Unidad1/tree/835e60f4e4a2d24cfaf92698fa9f8ccc1facffba/juego-de-luces

Ejercicio de aplicación con pulsadores

Para la siguiente aplicación se usará 3 pulsadores, al presionar cada pulsador se debe mostrar alguna frase en el monitor serial.

El código y el esquema a utilizar para este circuito se lo puede visualizar en el siguiente enlace:

https://github.com/carreraElectronica/Unidad1/tree/35b911ab7b6a594eac23a0bef68381dff21008b8/Aplicacion-pulsadores

Ejercicio de aplicación con Dipswitch

Para la siguiente aplicación se usará un dipswitch de 4 interruptores, al cambiar el estado de cada interruptor se mostrará una frase en el puerto serial.

El código y el esquema a utilizar para este circuito se lo puede visualizar en el siguiente enlace:

https://github.com/carreraElectronica/Unidad1/tree/35b911ab7b6a594eac23a0bef68381dff21008b8/aplicaion-Dipswitch

Para la aplicación de sensores digitales se simula con pulsadores o dipswitch que cumplen una función similar.

Ejercicio de aplicaciones combinadas de entradas y salidas digitales

Para la siguiente aplicación se pretende realizar diversos juegos de luces con 8 diodos leds, 2 pulsadores y 4 dipswitch. Con un switch comienza el programa, con el resto de dipswitch se realiza diferentes

juegos de luces, al presionar un pulsador en debe realizar un parpadeo (blink) y con el otro pulsador un mando alternado.

El código y el esquema a utilizar para este circuito se lo puede visualizar en el siguiente enlace:

https://github.com/carreraElectronica/Unidad1/tree/1c3b33c92726c812b2b727f964930416c5ae15b8/Entradas-salidas-digitales

Actividades propuestas para la unidad

1. Semáforo Sencillo: Utiliza LEDs rojo, amarillo y verde para simular un semáforo.

2. Control de LED con Pulsador*: Enciende y apaga un LED con un pulsador.

3. LEDs con Dos Pulsadores: Enciende un LED con un pulsador y apágalo con otro.

4.Secuencia de LEDs: Crea una secuencia de encendido y apagado con varios LEDs.

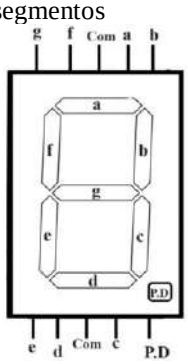
5.LED Intermitente: Programa un LED para que parpadee a intervalos regulares.

UNIDAD 2 TÉCNICAS DE BARRIDO Y MANEJO DE DISPLAY

Manejo y barrido de display de 7 segmentos

Según W. Yánez (2012), el display de 7 segmentos permite representar números de un solo dígito y ciertas letras, encendiendo o apagando segmentos específicos. Cada uno de los segmentos del display están compuestos internamente por diodos LED que se encuentran ubicados estratégicamente para formar un número ocho. También dispone de un LED para el punto decimal. Los segmentos se denominan en orden alfabético, es decir, a, b, c, d, e, f, g y P.D (punto decimal). En la Figura 13 se muestra la distribución de los siete segmentos del display.

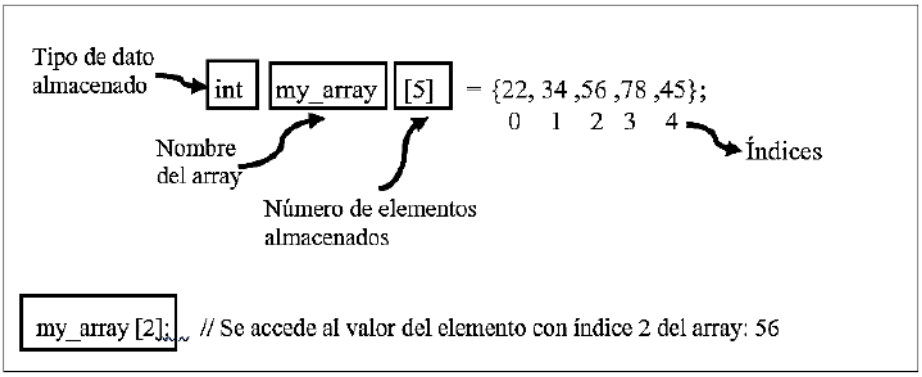
Figura 13
Denominación de los segmentos del display 7 segmentos



Manejo de tablas en c

Un array o tabla en el lenguaje C, como indica L. Yánez (2014), es una agrupación de datos o elementos del mismo tipo, pueden ser datos de tipo int, float, char, string, etc. Los arrays son elementos, que además de tener un nombre, cuentan con una especificación del tipo de datos que almacena, el tamaño (el número de elementos que conforman el array), con índices y una o varias dimensiones. En otras palabras, la sintaxis para la escritura de un array es *tipo_de_dato nombre_array[tamaño]= {elemento1, elemento2, ...}.* Cada elemento ocupa una posición o índice dentro del array. Los índices comienzan desde el 0 y se incrementan de uno en uno. Para acceder a un elemento específico dentro de la tabla, se utiliza el nombre de la tabla y el índice o posición de dicho elemento entre corchetes. La Figura 14 muestra ejemplos del manejo de arrays.

Figura 14
Ejemplo de creación y manejo de arrays o tablas

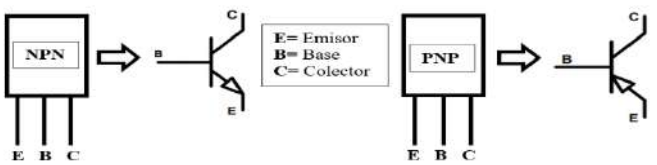


Según Estrada (2021), la clasificación de los displays de siete segmentos se basa en la conexión interna de los terminales de los LED que componen los segmentos del display.

Barrido

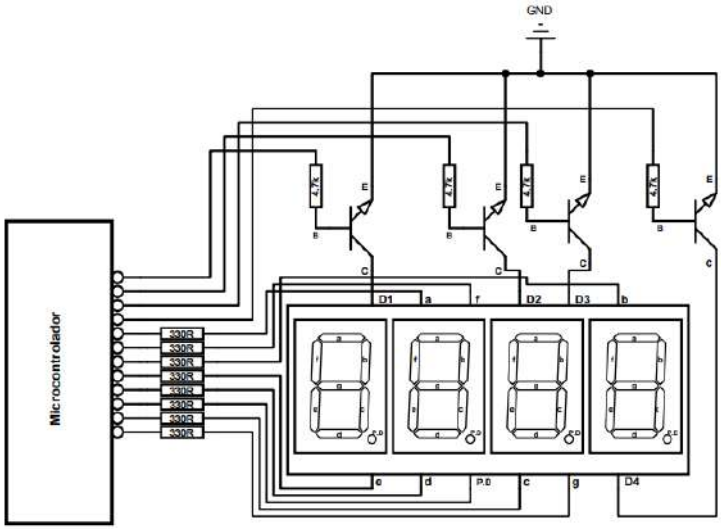
En varios proyectos electrónicos generalmente se requiere presentar números de más de un dígito, por lo mismo, frecuentemente se trabaja con más de un display de 7 segmentos. Como indica Castaño (2015), para controlar solo un display, tanto de ánodo común como de cátodo común, mediante un microcontrolador (la placa de Arduino, por ejemplo) son necesarios 7 pines. Una alternativa más eficiente para hacerlo es la técnica llamada barrido. El barrido consiste en usar solo 7 pines del microcontrolador para el primer, segundo, tercer o cuarto display. Al encender los display de uno en uno mediante pines adicionales del microcontrolador (un pin por cada display usado) a una alta velocidad utilizando un transistor, se genera la ilusión óptica de que los números se están mostrando simultáneamente. Como solución a dicho problema, se utilizan transistores configurados como switches. Para el display de ánodo común se utiliza el transistor PNP (2N3906) y el NPN (2N3904) para el de cátodo común. En la Figura 15 se muestra la disposición de los pines de los transistores y sus respectivos símbolos.

Figura 15
Transistor NPN (2N3904) y PNP (2N3906)



En la Figura 16 se muestra las conexiones para manejar el barrido de displays.

Figura 16
Esquema de conexión para el display de ánodo común.



Para el display de ánodo común, como indica Castaño (2015), los segmentos se encienden con 0's lógicos y los de cátodo común con 1's lógicos. Al finalizar, se obtendrá un valor binario de 8 bits. Para optimizar la programación del microcontrolador, aquel número binario se transforma a números decimales.El array resultante almacenará diez elementos o datos de tipo entero (int) y dispondrá de 9 índices. Para el display de ánodo común, *int numerosDisplayA[10]= {192, 249, 164, 176, 153, 146, 131, 248, 128, 152};*

Una manera sencilla para comprender la programación a nivel de puertos es creando un programa para el parpadeo de 8 leds. Con el siguiente enlace de GitHub se puede acceder al programa:

https://github.com/carreraElectronica/Unidad2/blob/c8e1c72bee263c2eabc55a3779eb98716fc56433/Blink_Leds_Puerto.ino

Aplicaciones: contador de pulsos de 0 a 9999

El programa está conformado por cuatro partes importantes: la declaración de variables, la configuración de puertos y pines en void setup (), la creación de dos funciones y el sketch principal en void loop (). En el siguiente enlace de GitHub podrá visualizar el esquema de conexión y el código a utilizar:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicaciones_Contador_0_9999

Manejo de display LCD

Las pantallas de cristal líquido o mejor conocidas como LCD son una tecnología fundamental en la industria de la visualización, debido a que ofrecen una combinación de eficiencia energética, delgadez, y capacidad, las cuales ayudan a mostrar imágenes nítidas y claras en una gran variedad de dispositivos electrónicos.

La LCD (Liquid Crystal Display) es una pequeña pantalla que se puede conectar a un microcontrolador, como Arduino, la cual permitirá mostrar textos de una manera sencilla a través de una matriz de puntos luminosos. Existen varios módulos de LCD los cuales se encuentran en diferentes presentaciones como: 16x2, 20x2, 20x4, etc. (Bolaños, 2012).

Figura 17
Display LCD 2x16 y configuración de pines

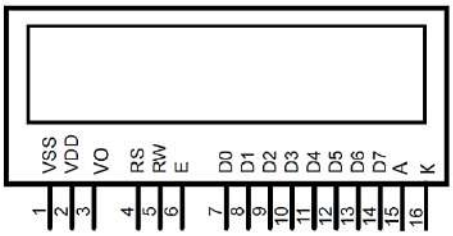


Tabla 1
Display LCD 2x16 y configuración de pines

SÍMBOLOS	FUNCIÓN
VSS	pin conectado a tierra (GND)
VDD	pin conectado a 5 voltios
VO	pin de contraste de LCD
RS	pin selector de registro
RW	pin de lectura o escritura
E	pin que habilita o deshabilita la LCD
D0 a D7	pines de transferencia de datos
A	pin del LED (ánodo) ilumina la LCD (5V)
K	pin del LED (cátodo) ilumina la LCD

Librería liquid crystal de Arduino

Para controlar el LCD, se instala la librería "LiquidCrystal" que facilita la comunicación entre la placa Arduino y la pantalla LCD. En la Tabla 2 se detallan las instrucciones que se usan para el manejo de la LCD.

Tabla 2
Instrucciones de la librería LiquidCrystal

INSTRUCCIÓN	DESCRIPCIÓN
lcd.begin():	Inicia el Display LCD y se especifica las dimensiones de la pantalla, se especifica el número de columnas (cols) y de filas (rows) del LCD.
lcd.clear():	Borra todo el contenido de la pantalla LCD.
lcd.setCursor():	Posiciona la ubicación en la que se mostrará el texto escrito para la pantalla LCD.
lcd.print():	Escribe un texto o mensaje en el LCD.

Diseño de menús con presentación en LCD

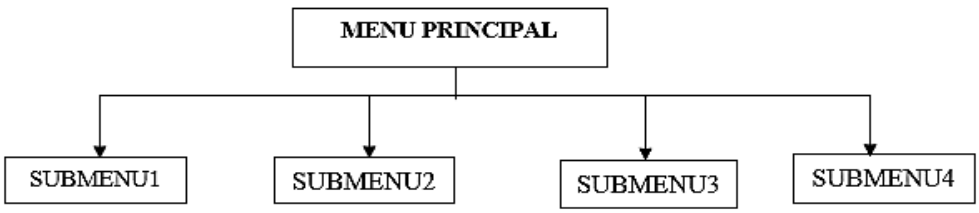
¿Qué es un menú?

En el ámbito de la informática, un menú es una lista de opciones clasificadas que posee diferentes alternativas, las cuales permiten el acceso a determinadas operaciones.

¿Qué es un submenú?

Un submenú es un menú secundario que se encuentra dentro de un menú principal. El submenú es una lista de opciones que se despliega desde un elemento del menú principal. Como se lo puede observar en la Figura 18.

Figura 18
Despliegue de submenús.



Estructura "while"

Es una sentencia de control de flujo que permite ejecutar repetidamente un bloque de código mientras una condición específica se evalúe como verdadera.

La sintaxis

```
while(condición){
  // instrucciones
}
```

Estructura switch...case

La sentencia “switch” es similar a la sentencia “if else”, la sentencia controla el programa especificando el código que se va a ejecutar bajo diferentes condiciones. Por lo tanto, se utiliza para elegir entre varias condiciones discretas, dependiendo del valor de una expresión.

La sintaxis es:

```
switch (expresión) {
  case 1:
    //ejecuta algo cuando la expresión es 1
  break;  //sale del switch
  case 2:
      //ejecuta algo cuando la expresión es 2
  break;  //sale del switch
}
```

Aplicación: Área de figuras geométricas con menús y sub menús con la sentencia Switch

El programa que permite navegar entre el área de un círculo y el área de un rectángulo, mediante menús y submenús utilizando la sentencia Switch, se puede visualizar en el siguiente link:

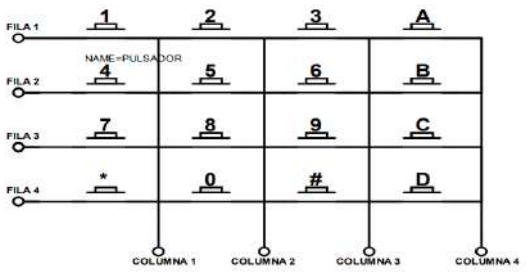
https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_Menus_submenus

Manejo y barrido de teclados matriciales

Teclado matricial 4x4

El teclado matricial, como indica González et al. (2019), es un arreglo de interruptores en cuatro filas y cuatro columnas. Por lo tanto, 16 interruptores conforman un teclado matricial. Su funcionamiento se basa en la unión de filas y columnas cuando un interruptor es presionado. Por ejemplo, al presionar el pulsador marcado como 1 (ver la Figura 19) la fila 1 se une con la columna 1. Asimismo, si se unen la fila 3 con la columna 2, la única tecla que pudo ser oprimida es el 8. Los interruptores denominados con letras mayúsculas (A, B, C, D) pueden ser configurados con otro valor por el programador.

Figura 19
Arreglo de interruptores en el teclado matricial 4x4



Barrido de teclado

El teclado matricial requiere de ocho pines del microcontrolador. Cuatro pines serán asignados para las filas y las otras, para las columnas. Los pines usados por las filas, además de configurarlos a nivel de puerto, se debe configurarlos individualmente mediante el pinMode (pin, modo) para posteriormente leer por separado cada pin. Con los puertos y pines configurados, el barrido del teclado se realiza con dos funciones: la primera denominada barridoTeclado, la cual se encarga de habilitar las columnas y asignar un valor a una variable de acuerdo al interruptor presionado. Por otro lado, la segunda función denominada leerFilas se encarga de monitorear cada una de las cuatro filas para detectar un cambio de estado, es decir, un interruptor accionado. El programa para el barrido se puede ver en el siguiente enlace:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Barrido_de_tecladoMatricial

Aplicaciones con display y teclado

Aplicación: teclado y display 7 segmentos

El programa para mostrar el número de cuatro dígitos en un display de 7 segmentos que se introduce en un teclado matricial, se puede observar en el siguiente link de GitHub: https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_teclado_display

Aplicación: teclado y display LCD

El programa para mostrar dos números de cuatro dígitos en una pantalla LCD, uno en cada fila, que se introduce en un teclado matricial y eliminar el contenido de la pantalla, lo podrá visualizar el

siguiente link de GitHub:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_teclado_display_LCD

Aplicación con: leds, relés, pulsadores

Este programa te permite controlar un juego de luces con pulsadores y teclado matricial. Ajusta la velocidad, define las repeticiones, visualiza el tiempo en displays de 7 segmentos y las repeticiones en un LCD, y enciende/apaga un foco con un pulsador y un relé. En el siguiente link podrá visualizar el esquema de conexión y el código a utilizar:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_Leds_reles.

Aplicación: calculadora con display LCD, display de 7 segmentos y teclados

El programa de una calculadora que permite realizar operaciones matemáticas básicas y muestra el resultado en una pantalla LCD y display de 7 segmentos se puede apreciar en el siguiente link:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_Calculadora_LCD

Aplicación: manejo de menú con display LCD y teclado

El programa para gestionar menús en la pantalla LCD a través de un teclado matricial, junto con el esquema de conexiones correspondiente, está disponible en el siguiente enlace de GitHub: https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_Manejo_menu_LCD

Aplicación: sistema de seguridad

El programa para establecer un sistema de seguridad utilizando un teclado matricial y una pantalla LCD está diseñado para permitir la configuración de una contraseña mediante la tecla 'D', la verificación de su corrección con la tecla '=' y la reactivación del sistema con la tecla 'C'. En el siguiente enlace de GitHub, se encuentra disponible el código y el esquema de conexiones para este ejercicio:

https://github.com/carreraElectronica/Unidad2/tree/536017dbd189338aa7be81894ea0169f636eb50b/Aplicacion_Sistema_seguridad

Actividades propuestas para la unidad

1.- Crear un contador simple que se incrementa o decrementa usando un teclado matricial, y muestra el valor en un display de 7 segmentos.

2.- Mostrar un mensaje en una pantalla LCD controlado por la entrada de un teclado matricial.

3.- Crear una calculadora básica que muestre resultados en una pantalla LCD y use un teclado matricial para la entrada.

4.- Usar un teclado matricial para encender y apagar un LED, mostrando el estado actual en una pantalla LCD.

5.- Crear un medidor de temperatura que muestre la lectura en una pantalla LCD y un display de 7 segmentos. El teclado matricial permite cambiar entre Celsius y Fahrenheit.

UNIDAD 3 MANEJO DE MÓDULOS ESPECIALES

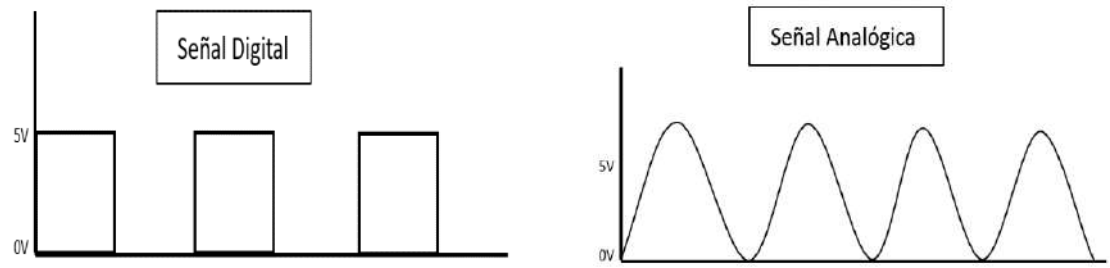
Los módulos especiales son aquellos componentes electrónicos los cuales están destinados a funciones específicas, sea la medición de temperatura, la detección del movimiento de objetos, su uso en trabajos con el microcontrolador Arduino brinda a expertos y a personas nuevas en el tema la capacidad de crear sistemas más complejos y personalizados, adaptados a las diferentes necesidades de cada aplicación.

Conversor analógico-digital y digital-analógico

Entradas/salidas digitales y analógicas

Las entradas digitales detectan señales en estado binario, es decir, leen la información como 1 (que equivale a 5V) o 0 (que equivale a 0V), como se observa en la Figura 20. Las salidas digitales controlan dispositivos externos (como luces, motores o relés) enviando valores lógicos de 1 (5V) o 0 (0V). Por otro lado, las entradas y salidas analógicas permiten medir voltajes de 0 a 5 voltios como valores de 0 a 1023, respectivamente. Esta característica permite leer variables como la luz, el sonido o la temperatura, para ser procesados por el microcontrolador. Las salidas analógicas generan voltajes variables en un rango de 0 a 5 voltios los cuales no ayudan a controlar con precisión el brillo de una luz o la velocidad de un motor.

Figura 20
Señal digital y analógica



Los dispositivos Analógicos-Digital convierten un nivel de voltaje analógico en un valor digital correspondiente mediante un ADC (Analog to Digital Converter). El conversor Análogo-Digital maneja 10 bit lo que se puede representar en 2^{10} eso quiere decir que tiene 1024 caracteres que trabaja en un rango de 0 a 1023 y estos valores representan 0V (2^0) y 5V (2^{10}), respectivamente. La función analogRead (pin) permite leer el valor analógico y convertirlo en un valor digital dentro del rango de 0

a 1023En el siguiente link se observa el código y circuito utilizado para la lectura de la conversión análoga-digital.

https://github.com/carreraElectronica/Unidad3/blob/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/3_Conversor_A_D_de_10_bits/3_Conversor_Anologo_a_Digital_de_10_bits.ino

Linealización de señal de análoga

La linealización de señales analógicas consiste en transformar los valores de la lectura analógica, que comúnmente son los valores que se encuentran en el rango de 0 a 1023, en valores de voltaje de 0V a 5V. Para realizar la linealización en el void loop () se toma el valor de un pin analógico y se multiplica por 5 y se divide por 1023. El resultado obtenido se muestra a través del puerto serial.

En el siguiente link de GitHub se encuentra el código y el esquema de conexiones usado para la linealización de la señal analógica:

https://github.com/carreraElectronica/Unidad3/blob/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/4_Linealizaci_n_de_se_al_de_an_loga/4_Linealizaci_n_de_se_al_de_an_loga.ino

Conversor digital-analógico con R2R

Un conversor Digital-Analógico es aquella que transfiere información expresada en forma digital o en forma binaria (0 y 1) a una forma analógica (Alfredo & Perez, 2015).

El Arduino no cuenta con un DAC (Digital to Analog Converter) integrado, sin embargo, el Arduino puede simular un DAC mediante el uso del ancho de pulso, la cual permitirá emular una señal analógica variando el ciclo de trabajo de una señal cuadrada. Esto se puede lograr mediante la función analogWrite (pin, value), donde el valor vario de 0 que equivale a 0V y 255 que equivale 5V.

El conversor DAC con R2R es un método muy popular y a la vez efectivo el circuito es una configuración de resistencia que toman peso binario o bits como entrada y los convierte en voltaje analógico que se envía a un pin analógico (Ramesh, 2021).

En el siguiente link se puede visualizar el circuito de conexión y el código utilizados para el conversor Digital-Analógica:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/6_Conversor_digital_an_logo_con_R2R

Aplicación: voltímetro con potenciómetro

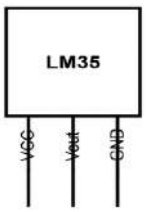
Para la implementación del siguiente ejercicio se utiliza un potenciómetro, el cual mediante programación enviara datos de 0V a 5V. Los datos se visualizan en un Display LCD. El diagrama de conexión y el código utilizado para la aplicación se puede observar en el siguiente link:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/7_Volt_metro_con_potenci_metro

Aplicación: sensor de temperatura

El LM35 es un sensor que ayuda con la medición de la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. Su rango de medición es de -55℃ (-550mV) a 150℃ (1500mV), (Luis, 2015).

Figura 21
Terminales del LM35



La combinación del LM35 con el Arduino para visualizar la temperatura de un lugar se puede ver en el siguiente link:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/8_Sensor_de_Temperatura

Interrupciones y temporizadores

Interrupciones externas

El microcontrolador Arduino incorpora el concepto de interrupciones, el cual es un mecanismo que permite asociar una función a la ocurrencia de un determinado evento, a esta función se denomina ISR (Interruption Service Rutine) (Llamas, 2016). Cuando ocurre el evento el microcontrolador sale inmediatamente del flujo normal del programa y ejecuta la función ISR que se le asocio, la cual ignora cualquier otra tarea que se esté realizando, al finalizar el ISR el programa regresa a su flujo normal en el mismo punto donde se interrumpió. (Llamas, 2016).

Para declarar una interrupción se usa la siguiente sintaxis en Arduino: attachInterrupt(digitalPinToInterrupt(pin), ISR, mode).

Un ejemplo sencillo con dos leds y la interrupción INT3 para entender el funcionamiento de las interrupciones, se puede observar en el siguiente enlace de GitHub: *https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/9_Interrupciones_con_Arduino*

Interrupción por timer

El Timer o temporizadores con Arduino está fuertemente relacionado con las interrupciones. El evento de interrupción se ejecuta cuando se cumple un cierto tiempo definido por el programador. Este tipo de interrupciones es usado comúnmente para la lectura y el control de sensores, la creación de señales PWM. Para la utilización de los timer en Arduino, es necesario descargar e incluir librerías, por ejemplo: #include <TimerOne.h>(CASTAÑOSERGIO, 2019). Mientras que para declarar una interrupción por timer se utilizan las siguientes sintaxis: Timer1.initialize(tiempoMicrosegundos); y Timer1.attachInterrupt(funcionAejecutarse).

En el siguiente enlace de GitHub, se puede observar un ejemplo básico del funcionamiento de los timers: *https://github.com/carreraElectronica/Unidad3/blob/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/11_Interrupci_n_por_timer/11_Interrupci_n_por_timer.ino*

Aplicación de interrupciones: contador de personas

Para la realización del contador de personas se utiliza el barrido de Display y las interrupciones. El programa cuenta con dos pulsadores: uno de incremento y otro de decremento. El contador tendrá un rango de 0 a 99 y se visualizará en los Display de 7 segmentos.

En el siguiente enlace se visualiza el esquema y el código utilizado en el circuito del contador de personas:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/12_Contador_de_personas

Aplicación de interrupciones: reloj digital.

El reloj digital se realiza con interrupciones con timers para representar los segundos, minutos y horas. Esta información se mostrará en un Display LCD. En el siguiente enlace se puede visualizar el esquema de conexión y el código utilizado en el circuito del reloj digital.

En el siguiente enlace se visualiza el esquema y el código utilizado en la aplicación.

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/13_Reloj_Digital

Modulación de ancho de pulso PWM

La modulación de ancho de pulso (PWM) se utiliza para variar la cantidad de corriente que se le puede introducir a un dispositivo electrónico como un motor o LED, en el cual se puede alternar el ancho relativo de los pulsos de la señal. La señal proporcionada tendrá una frecuencia fija, y se alterna entre el encendido y apagado para generar la señal cuadrada (Correa Eras & Remache Ortega, 2006).

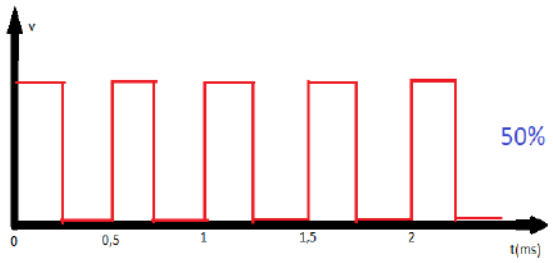
Cálculo de voltaje promedio con PWM

Para calcular el voltaje promedio se requiere el ciclo de trabajo de una señal que es uno de los conceptos primordiales del PWM. El ciclo de trabajo se expresa con la letra D y se menciona como la relación entre tiempo en alto (el cual se expresa como TON) y el periodo que ofrece la señal (que se expresa como T). El resultado se multiplica por 100, debido a que se expresa como porcentaje. Esto representa el tiempo en cual la señal está en estado alto durante un ciclo.

$$D = \frac{TON}{T} * 100\%$$

Por ejemplo, un ciclo que tenga 50% de trabajo significa que la señal está en estado alto por la mitad de tiempo y baja la otra mitad del tiempo. Si el ciclo de trabajo es del 25% en alto significa que el otro 75% es bajo por contraste si el 75% del ciclo de trabajo es alto el otro 25% es bajo, ver la Figura 22

Figura 22
Señal de ciclo de trabajo del 50%



Con el ciclo de trabajo se controla el voltaje promedio, es decir poder cambiar el voltaje de salida, usando la siguiente expresión calculamos el voltaje promedio (Jose, Guerra, 2022):

$$Vprom = (VH - VL) * \frac{D}{100}$$

Donde:

- VH es el voltaje alto (1)
- VL es el voltaje bajo (0)

La placa Arduino suele tener el voltaje bajo de 0V, la formula se puede simplificar:

$$Vprom = Vcc * \frac{D}{100}$$

Donde:

- VCC es el voltaje que ofrecen los pines digitales

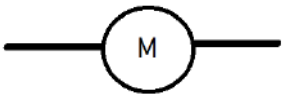
Configuración de PWM en Arduino

La señal PWM se usa comúnmente para configurar circuitos analógicos, con el periodo y la frecuencia del tren de impulsos se configura la potencia suministrada al circuito. Este tipo de señal se suele utilizar para iniciar motores de CC con el fin de modular su velocidad, también se utiliza para poder modular la intensidad del brillo del led, etc. En la placa Arduino, se puede controlar la señal de salida del PWM con la función analogWrite (pin, valorPWM). Los pines con la función de PWM van desde el 2 al 13 (Ruiz Guti, 2007). Por ejemplo, para un ciclo de trabajo del 100% será: analogWrite (2,255).

Aplicación de PWM: modulación de velocidad de un motor

Un motor es una máquina que tiene como objetivo transformar la energía eléctrica en energía mecánica, para lograr esto solo se necesita aplicar una tensión eléctrica en sus bornes (ver Figura 23).

Figura 23
Símbolo del motor



El PWM ayuda a modular la tensión suministrada al motor, en consecuencia, permite el control de la velocidad del motor. Con un valor PWM de 0, el motor está en reposo; con 128, el motor funciona a la mitad de su velocidad máxima y con 255, trabaja al 100% de su velocidad. De esta manera, se controla la velocidad gradualmente del motor (Correa Eras & Remache Ortega, 2006).

En el siguiente enlace se puede observar el programa y el esquema de conexiones para el control de velocidad de un motor

https://github.com/carreraElectronica/Unidad3/tree/1a0d8737051c05c0ea372d6440519a7546b01c86/Control%20de%20velocidad%20motor

Aplicaciones con módulos especiales

Aplicación: conversión analógica a digital

D-DA

Mediante un sensor de temperatura LM35 se mide la temperatura (la cual es una señal analógica) y se usa el Arduino para transformar el valor analógico obtenido en un valor digital para posteriormente representarla en grados centígrados.

En el siguiente enlace se puede visualizar el esquema de conexión y el código utilizado para el sensor de temperatura LM35:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/14_Leer_sensor_temperatura_lm35

Aplicación: interrupciones

Con la utilización de interrupciones, se realiza un contador binario mediante leds, usando la configuración de puerto, que incrementa o decrementa al accionar dos pulsadores. En el sketch principal se ejecuta un blink sencillo.

En el siguiente enlace se pude visualizar el esquema de conexión y el código utilizado para este ejemplo:

https://github.com/carreraElectronica/Unidad3/tree/483d90867d835feb6701962602122b6b9cfc01ca/15_Aplicaci_n_Interrupciones

Aplicación: PWM

Con la ayuda del PWM se puede generar una señal analógica variable para controlar la luminosidad de un led, el hardware a utilizar y el código se visualizan en el siguiente enlace de GitHub:

https://github.com/carreraElectronica/Unidad3/tree/949a0f90f86aedf4381eb9cdd43dee7cf5944e74/16_Aplicaci_n_PWM

Actividades propuestas para la unidad

1.- Ajustar el brillo de un LED usando un potenciómetro y mostrar el valor ADC en el monitor serial. Usar interrupciones para cambiar entre diferentes modos de brillo.

2.- Usar un LDR (resistor dependiente de la luz) para controlar la velocidad de un motor mediante PWM. Utilizar una interrupción para detener el motor cuando un botón sea presionado.

3.- Usar un sensor de temperatura para activar una alarma (LED parpadeando) cuando la temperatura excede un umbral. Usar una interrupción para silenciar la alarma con un botón.

UNIDAD 4 COMUNICACIÓN Y SISTEMAS DE CONTROL

Comunicación serial

En el mundo de la electrónica existen numerosos componentes y sistemas de control que permiten la comunicación entre dispositivos en esta unidad se observara la forma de comunión que tiene el microcontrolador y su aplicación con LabVIEW.

¿Qué es la comunicación?

La comunicación es un sistema el cual es utilizado en la electrónica para la transferencia de información entre diversos dispositivos, en este caso al usar el microcontrolador Arduino se podrá realizar una comunicación con la computadora a la que está conectado mediante el puerto serial.

¿Qué son los sistemas de control?

Los sistemas de control son fundamentales en numerosas aplicaciones de la electrónica y en la automatización, pueden ser usados en la gestión de procesos industriales hasta el control de dispositivos usados en para uso doméstico, estos sistemas se utilizan para el monitoreo de algún sistema o proceso (Santos, 2024)

¿Qué es la comunicación serial en Arduino y cómo se utiliza?

La comunicación serial en Arduino es una forma de comunicación entre dispositivos en la que se envía y recibe datos de un bit a la vez a través de un canal de comunicación. En el caso del microcontrolador de Arduino la comunicación serial es la forma más efectiva para enviar y recibir datos entre el ordenador y la placa de Arduino, también a través de esta comunicación se podrá mandar diferentes órdenes al microcontrolador Arduino para así automatizar diversos procesos o también recibir información y así verla en la pantalla del ordenador.(Castaño, 2019)

Comunicación serial Arduino PC

Para utilizar la comunicación serial en Arduino, primero se iniciará la comunicación serial con la línea de código la cual toca establecer la velocidad de trasmisión en baudios y se utiliza para indicar que la placa esta lista para enviar y recibir datos, para lo cual se usa Serial.begin (9600). Para saber si el Arduino está

trasmitiendo correctamente, se utiliza el monitor serial de Arduino el cual viene incluido en el software de Arduino.

Interfaz Rs232

La comunicación en seria RS232 es un protocolo utilizado para trasmitir datos entre diferentes dispositivos, esto no ayuda a que los dispositivos se comuniquen a largas distancias usando una interfaz sencilla, la interconexión de Arduino con la interfaz RS232 permite realizar una excelente comunicación serial entre dispositivos, en el caso del siguiente ejercicio se utilizara El módulo MAX232, el cual ayuda como un convertidor de nivel de voltaje, cerrando la brecha entre los niveles lógicos.

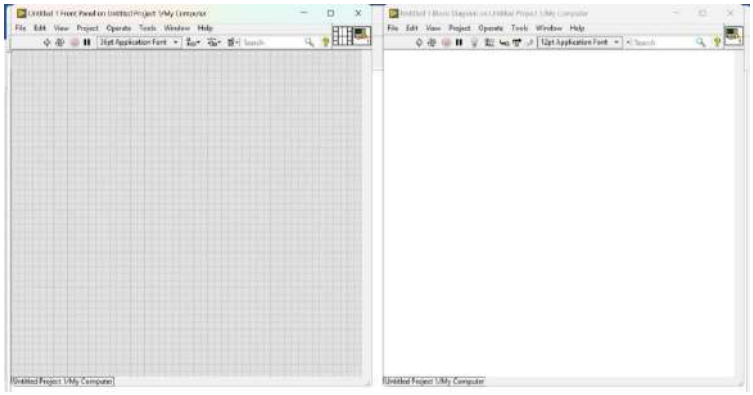
En el siguiente ejemplo al escribir "ON" al terminal serie, el LED se encenderá y al escribir "OFF" al terminal serie, el LED se apagará. Se usa el convertidor RS232 a TTL y el convertidor DB9 a USB para la comunicación entre Arduino y PC a través del protocolo de comunicación RS232 (Bhuiyan, 2023). En el siguiente enlace de GitHub se disponible el hardware y software necesario para ejecutar el circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino3

LabVIEW

Es un ambiente de programación basado en programación gráfica y no en texto como lo son lenguajes de programación tradicionales como C, C++ o java. LabVIEW es un desarrollo de programación interactivo que usa el lenguaje de programación grafico el cual cuenta con diversos instrumentos virtuales. Los ficheros generados con LabVIEW se llaman Instrumentos Virtuales los cuales se componen de el panel frontal (front panel) o interfaz con el usuario donde se observa cierta información y ejecute diversas acciones y el diagrama de bloques (block diagram) que es el código fuente donde se programa el funcionamiento de la aplicación (Facultad, 2010). En la Figura 24 se puede observar lo antes mencionado.

Figura 24
Diagrama de bloque y panel frontal

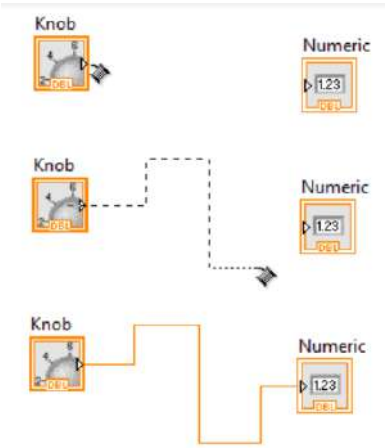


LabVIEW tiene los menús básicos de cualquier programa como “File” (Archivo) donde se puede abrir un nuevo archivo, cerrarlo, guardarlo, imprimirlo, “Edit” (Editar) donde se puede copiar, cortar, pegar, “Windows” (Ventana) para trabajar con las ventanas o Help (ayuda), más otros menús que son propios del programa como “Operate” (Operar) donde se selecciona si correr o detener el programa, si se corre paso a paso, etc. Abajo de los menús se encuentra una barra de herramientas que varía en algunos botones entre la del panel frontal y la del diagrama de bloques (Facultad, 2010)

Interconexión entre elementos del diagrama de bloques

Los bloques se interconectan entre si mediante alambres, al acercar el cursor a alguna entrada o salida de algún bloque este se convierte automáticamente en un carrete de hilo, al dar clic se crea el alambre, sin necesidad de mantener presionado el mouse se puede arrastrar hasta donde se desea conectar, como se puede ver en la Figura 25.

Figura 25
Interconexión en LabVIEW



LabVIEW y Arduino

En este tema se observará cómo realizar la comunicación serial entre LabVIEW y Arduino para así poder controlar una salida digital.

Las diferentes funciones que se utilizarán para la comunicación con Arduino serán:

- Serial.begin(), //inicia la conexión serial
- Serial.available(), //devuelve el número de bytes disponibles para ser leídos por el puerto serie.
- Serial.read(); //Devuelve el primer byte del buffer de lectura quitándolo de éste.
- Serial.write(); //Escribe datos binarios en el puerto serie.

En el caso del envío de datos desde Arduino a LabVIEW se utilizará como referencia la imagen que se encuentra el siguiente enlace, la cual muestra cómo debe armarse el diagrama de bloques al entrar la información del Arduino, en el caso del ejemplo la lectura de diversos sensores y un dipswitch y mediante un código en Arduino enviamos la información de los sensores y el dipswitch. En el siguiente enlace de GitHub se disponible el hardware y software necesario para ejecutar el circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino6

En este caso se leerá el puerto C y se enviara al LabVIEW mediante la línea de código, Serial.print(). En el caso contrario para él envió de datos desde Arduino a LabVIEW se dividirá la información en bloques. En el siguiente enlace de GitHub se encuentra disponible el hardware y software necesario para ejecutar el circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino5

Sistemas de control

En palabras de Gutiérrez Hinestroza & Iturralde Kure (2017), un controlador se encarga de corregir la desviación del valor de salida medido en un proceso con respecto a un valor de referencia predefinido por el usuario, es decir iguala el valor de salida a un valor prefijado. Su función es comparar el valor de salida con el de referencia para determinar si existe una diferencia considerable entre ellas. El dispositivo produce una señal de control que activa un actuador, o un elemento final de control, para disminuir dicha diferencia a cero o a un valor despreciable. Existen varios tipos de controladores. Su

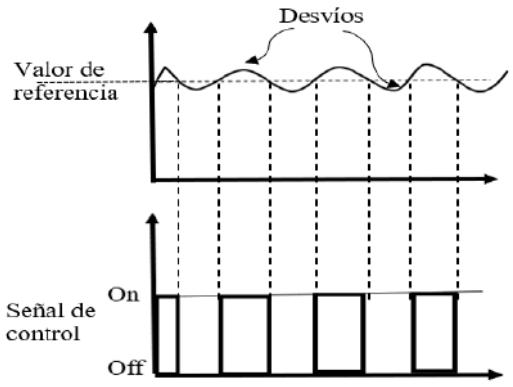
clasificación se basa en la manera en que generan la señal de control, entre ellos están el controlador ON/OFF, controlador P, controlador PI y el controlador PID.

Figura 26
Diagrama de funcionamiento del controlador

Control on-off

El control On-Off (o también conocido como control de dos posiciones) permite, según Villajulca (2019), inspeccionar si el valor de salida medido en un proceso se encuentra por encima o por debajo del valor preestablecido. Para controlar la desviación, la señal de control acciona el elemento de control final que tiene solo dos posiciones: completamente encendido y apagado. Por lo que se trata de un control simple. Sin embargo, el control obtenido es bastante impreciso. Un claro ejemplo de su uso es en los ventiladores automáticos, que se encienden si la temperatura es mayor a la referencia establecida y se apaga cuando es menor a ella.

Figura 27
Diagrama de funcionamiento de un controlador de On/Off

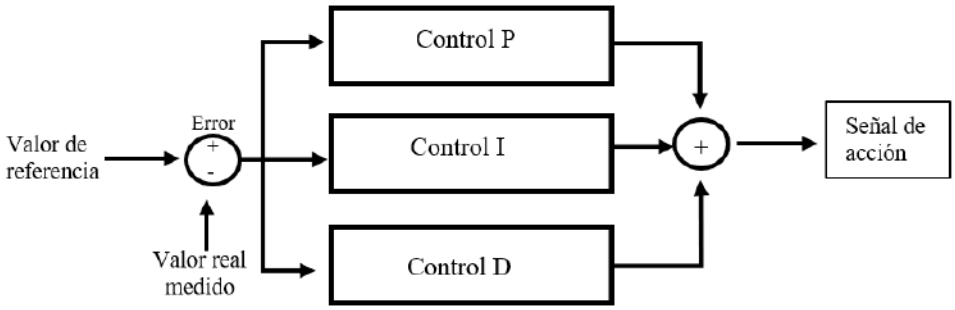


Control PID

Según Boghossian, Ardemis & Brown, James & Zak (2022), el control PID aprovecha los beneficios de cada uno de los tres controles básicos: control proporcional, control integral y el control derivativo. A los dos controles anteriores se suma el control D. El control derivativo analiza la velocidad de cambio del error. Si existe un error que está variando demasiado rápido, el control D responde para eliminar esas oscilaciones y estabilizar el sistema. Por lo tanto, todo en conjunto, ante una variación lenta de la

señal del error entran en acción el control proporcional e integral. En cambio, si el error varía rápidamente actúa el control derivativo. En otras palabras, el control PID permite una respuesta y corrección rápidas ante un desvío del valor real medido de un valor de referencia.

Figura 28
Esquema de un controlador PID



Aplicaciones con comunicación serial y sistemas de control

Aplicación: sistema de control LabVIEW y Arduino

Este proyecto permite recoger información del valor de un potenciómetro, la temperatura y el nivel de agua de un tanque, y visualizar estos datos en LabVIEW. Además, se monitorea el estado de cuatro interruptores para proporcionar una visión completa del sistema en tiempo real.

A continuación, se muestra el código y esquema del circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino

Aplicaciones con comunicación serial y sistemas de control

Aplicación: control de P – PI

Proyecto para controlar la temperatura a 30 grados Celsius utilizando un sensor LM35 para medir la temperatura y un ventilador para regularla. En este caso, el Arduino utilizará un controlador PI para mantener la temperatura cerca del valor de referencia (30 grados Celsius).

A continuación, se muestra el código y el esquema del circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino2

Aplicación: control de velocidad de un motor DC

El proyecto para controlar la velocidad de un motor DC mediante LabVIEW y Arduino requiere el uso de un transistor NPN adicional para gestionar la potencia del motor. Además, se necesita desarrollar un

panel de control en LabVIEW que permita al usuario manipular la velocidad del motor de manera intuitiva y precisa.

A continuación, se muestra el código y esquema del circuito:

https://github.com/carreraElectronica/Unidad4/tree/main/labVIEW_y_Arduino4

Autoevaluación

En el siguiente apartado se muestra la evaluación de la materia, escoja la respuesta correcta según corresponda.

1. ¿Cuál de los siguientes no es un microcontrolador comúnmente utilizado en placas Arduino?

   a) ATmega328P, b) ESP8266, c) Raspberry Pi, d) ATmega2560

2. ¿Cuál de las siguientes placas Arduino es conocida por su tamaño reducido y bajo costo?

   a) Arduino Uno, b) Arduino Nano, c) Arduino Mega, d) Arduino Due

3. ¿Qué lenguaje de programación se utiliza comúnmente para programar placas Arduino?

   a) Python, b) C++, c) JavaScript, d) Java

4. ¿Qué tipo de sensor se utilizaría para medir la temperatura con Arduino?

   a) Sensor de ultrasonido, b) Sensor de luz, c) Sensor de temperatura DS18B20, d) Sensor de humedad

5. ¿Cuál de las siguientes afirmaciones describe mejor la función de un servo motor?

   a) Genera sonido, b) Convierte energía eléctrica en energía mecánica, c) Realiza movimientos precisos basados en señales de control, d) Mide la temperatura del ambiente

Referencias Bibliográficas

Alfredo, J., & Perez, M. (2015). *Convertidor Ingeniería en Sistemas Electrónicos Industriales.*

Alonso, R. (2024). *Von Neumann, la arquitectura común de todos los procesadores. HardZone.* https://hardzone.es/tutoriales/rendimiento/von-neumann-limitaciones/

Artero, Ó. T. (2016). *El mundo genuino Arduino. RC libros.*

Boghossian, Ardemis & Brown, James & Zak, S. (2022). *Control de P, I, D, PI, PD y PID.*https://espanol.libretexts.org/Ingenieria/Ingeniería_Industrial_y_de_Sistemas/Libro%3A_Dinámica_y_Controles_de_Procesos_Químicos_(Woolf)/09%3A_Control_proporcional-integral-derivado_(PID)/9.02%3A_Control_de_P%2C_I%2C_D%2C_PI%2C_PD_y_PID

Bolaños, D. (2012). *Manejo De Display LCD. En Manejo De Display Lcd (págs. 1-9). C++,92.*

Castaño, S. (2015). *Multiplexación Display de 7 Segmentos con PIC.* https://controlautomaticoeducacion.com/microcontroladores-pic/multiplexacion-display-7-segmentos/

Castaño, S. (2019). *Control Automático Educación.* https://controlautomaticoeducacion.com/arduino/comunicacion-serial-con-arduino/#Que_es_la_comunicacion_serial_en_Arduino_y_como_se_utiliza

Charlena. (2022). *Control proporcional.* https://techlib.net/techedu/control- proporcional/

Diosdado, R. (2014). *Manual Arduino. Saudi Med J, 33, 3–8.*

Estrada, R. (2021). *Display 7 Segmentos ánodo y cátodo común.* https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/

Facultad, I. (2010). Tema : *Introducción a Labview Material y Equipo*. 1–19.

Goilav, N., & Geoffrey, L. O. I. (2016). *Arduino: Aprender a desarrollar para crear objetos inteligentes.* Ediciones ENI.

Gutiérrez Hinestroza, M., & Iturralde Kure, S. A. (2017). *Fundamentos Básicos de Instrumentacion y Control.* https://repositorio.upse.edu.ec/bitstream/46000/4228/1/Fundamentos de Instrumentacion y Control.pdf

Hernández, L. del V. (2021). *Pines digitales de Arduino pinMode, digitalWrite y digitalRead.*

Jecrespom. (2017). *Memoria Flash, SRAM y EEPROM. Aprendiendo Arduino.* https://aprendiendoarduino.wordpress.com/2017/06/21/memoria-flash-sram-y- eeprom-3/

Llamas, L. (2016). *Qué son y cómo usar interrupciones en Arduino.*

Mariana. (2024, 26 enero). *ALU. Unidad aritmético lógica. Fisicotrónica.* http://fisicotronica.com/alu-unidad-aritmetico-logica/

Mheducation. (2013). *Programación en lenguajes estructurados. 414.* https://www.mheducation.es/bcv/guide/capitulo/8448148703.pdf

Rodriguez, F. (2018). *Universidad Nacional Autonoma De Mexico Facultad De Estudios Superiores Cuautitlán. 6.*

Muñoz, A. M., & Córcoles, S. C. (2018). *Arduino. Edición 2018 Curso práctico.* Ra-Ma Editorial.

Oliag, S. T. (1995). *Curso de programación en C++.*

Ramesh, A. (2021). *Blogs de rushi.*

Rodriguez, F. (2019). *Creación y uso de funciones en Arduino.* https://fidiasrodriguez.com/creacion-y-uso-de-funciones-en-arduino/

Ruiz Guti, M. (2007). *Manual de Programación Arduino Arduino : Manual de Programación. Arduino Notebook, 1, 3–70.*

Santos, M. (2024). *Polaridad.es.* https://polaridad.es/elementos-de-un-sistema-de-control/

Schweber, B. (2021). *DIGIKEY. Obtenido de Comprender los parámetros del oscilador de cristal para optimizar la selección de componentes:* https://www.digikey.com.mx/es/articles/understand-crystal-oscillator-parameters-to-optimize-component-selection#:~:text=Los%20osciladores%20de%20cristal%20proporcionan,maestra%20en%20sintonizadores%20y%20sintetizadores.

Velarde, J. E. (2009). *Estructura de Buses Compartidos en Microcomputadoras [Diapositivas]. SlideShare.* https://es.slideshare.net/jevelarde/estructura-de-buses-compartidos-en-microcomputadoras

Villajulca, J. (2019). *Control ON/OFF o Todo/Nada.* https://instrumentacionycontrol.net/control-on-off-o-todo-nada/

Villajulca, J. (2022*). El control proporcional: definiciones prácticas y precisas.* https://instrumentacionycontrol.net/el-control-proporcional-definiciones-practicas-y-precisas/

Yánez, L. (2014). *Fundamentos de la programación. In Facultad de Informática.*

Yánez, W. (2012). *CONSTRUCCIÓN DE UN PROTOTIPO PARA EL CONTROL DE TURNOS ELECTRÓNICO CON 6 PUESTOS DIFERENTES DE LLAMADA Y 99 TURNOS.*

ISBN: 978-9942-676-25-2