Universidad de Concepción

Facultad de Ingeniería
Facultad de Ciencias Físicas y Matemáticas
Sociedad de Robótica RAS IEEE UdeC

Taller Básico de Microcontroladores

Introducción a los Microcontroladores

16 de junio de 2025

Índice

Objetivos del Taller

- Comprender la arquitectura y funcionamiento básico de un microcontrolador.
- Identificar las diferencias entre microcontroladores y microprocesadores.
- Familiarizarse con los principales bloques que conforman un microcontrolador: CPU, memorias, periféricos, etc.
- Conocer diferentes familias y plataformas, como Arduino, ESP32 y STM32.
- Aplicar buenas prácticas de programación, conexión y pruebas en el desarrollo con microcontroladores.
- Explorar proyectos motivadores que permitan aprender haciendo.

1. Conceptos Fundamentales

1.1. Microcontrolador

Es un circuito integrado programable que contiene, en un solo chip, todos los componentes necesarios para ejecutar tareas de control automático en sistemas embebidos. En términos simples, actúa como un pequeño computador autónomo diseñado para realizar funciones específicas en dispositivos electrónicos.

- Un **sistema embebido** es una combinación de hardware y software que opera de forma autónoma, generalmente con recursos limitados, y está embebido (integrado) dentro de un producto o sistema mayor. Son generalmente no reconfigurables por el usuario final

1.2. Microprocesador

Es el núcleo de un sistema de cómputo de propósito general. Se trata de un chip electrónico que interpreta y ejecuta instrucciones, coordinando el funcionamiento de los demás componentes mediante buses de datos, direcciones y control. Es el cerebro de los sistemas computacionales de propósito general. A diferencia de un microcontrolador, no tiene todo integrado, pero es mucho más potente y versátil, ideal para ejecutar múltiples programas y gestionar grandes volúmenes de datos.

2. Diferencias entre Microprocesador y Microcontrolador

Los microprocesadores y microcontroladores son unidades de procesamiento digital que comparten similitudes en su estructura fundamental, pero están diseñados para cumplir funciones distintas dentro de los sistemas electrónicos.

Un microprocesador es una unidad central de procesamiento diseñada para sistemas de propósito general, como computadoras. Requiere componentes externos (RAM, ROM, puertos, etc.) para funcionar, lo que lo hace más complejo, costoso y con mayor consumo energético, pero con alta capacidad de procesamiento.

En cambio, un microcontrolador es un chip todo-en-uno que integra CPU, memoria y periféricos. Está pensado para tareas específicas en sistemas embebidos, como electrodomésticos o dispositivos IoT (El Internet de las Cosas). Su bajo consumo, simplicidad y costo reducido lo hacen ideal para aplicaciones de control automático.

En resumen, mientras que el microprocesador es ideal para sistemas complejos y versátiles, el microcontrolador está diseñado para aplicaciones dedicadas y embebidas que requieren simplicidad, confiabilidad y eficiencia energética. De manera más gráfica se puede observar:

2.1. Tabla comparativa:

Característica	Microprocesador	Microcontrolador
Componentes integrados	Solo CPU	CPU + memoria + I/O + periféricos
Uso	Computación general	Sistemas embebidos / control dedicado
Complejidad del sistema	Alta	Baja
Coste	Alto	Económico
Velocidad y potencia	Alta	Moderada
Consumo de energía	Alto	Bajo

3. 03. Arquitectura básica de un Microcontrolador

3.1. CPU

La Unidad Central de Procesamiento (CPU) ejecuta las instrucciones del programa, y se encarga de realizar operaciones aritméticas, lógicas y de control. Es el çerebro"del microcontrolador.

3.2. Memorias

Un microcontrolador contiene distintos tipos de memoria:

- *ROM/FLASH*: Almacenamiento permanente del programa.
- *RAM*: Memoria de acceso aleatorio para almacenamiento temporal de datos.
- *EEPROM*: Memoria no volátil programable, adecuada para guardar configuraciones o datos persistentes.

3.3. Periféricos de Entrada/Salida (I/O)

Los pines de entrada/salida permiten que el microcontrolador se comunique con el mundo exterior, leyendo sensores o activando actuadores como luces y motores.

3.4. Temporizadores y Contadores

Componentes que permiten medir intervalos de tiempo, generar señales de modulación por ancho de pulso (PWM) o contar eventos, esenciales en tareas de control y sincronización.

Tip rápido

Antes de alimentar un circuito, revisa conexiones y polaridad con un multímetro. ¡Evita cortos!

3.5. Conversores A/D y D/A (ADC/DAC)

Los conversores analógico/digital y digital/analógico permiten al microcontrolador interactuar con señales del mundo real, convirtiéndolas entre formatos analógico y digital.

3.6. Sistema de Interrupciones

Permite que el microcontrolador interrumpa la ejecución normal del programa para atender eventos urgentes, como una señal de un sensor o un cambio en una entrada.

3.7. Módulos de Comunicación

Facilitan la transmisión y recepción de datos mediante protocolos como UART, SPI, I2C, así como conectividad inalámbrica (WiFi, Bluetooth, USB), según el modelo del microcontrolador.

4. 04. Tipos y Familias de Microcontroladores

4.1. AVR (Arduino – Atmel/Microchip)

Microcontroladores de 8 bits basados en arquitectura RISC. Usan frecuencias de reloj de hasta 16 MHz. Programables en Arduino IDE, lenguaje C/C++ simplificado. Muy usados en educación y prototipado.

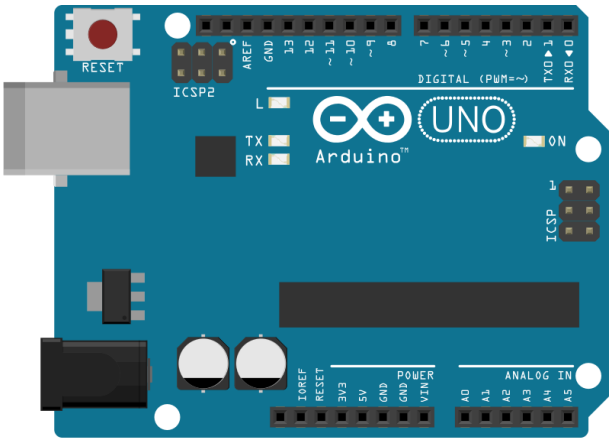

Figura 1: Placa Arduino Uno R3

4.2. ESP32 (Espressif Systems)

Microcontrolador de 32 bits con WiFi y Bluetooth integrados. Doble núcleo, hasta 240 MHz. Compatible con Arduino IDE, PlatformIO, MicroPython y ESP-IDF. Ideal para proyectos IoT.

Figura 2: Placa ESP32

4.3. STM32 (STMicroelectronics)

Familia de 32 bits basada en ARM Cortex-M. Potentes y eficientes energéticamente, orientados a robótica, procesamiento de señales y dispositivos portátiles.

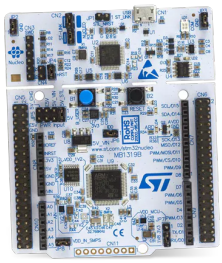

Figura 3: Placa STM32

4.4. PIC (Microchip)

Microcontroladores de 8 a 32 bits, robustos y confiables en la industria. Se programan en MPLAB X IDE, con control preciso de temporizadores y PWM.

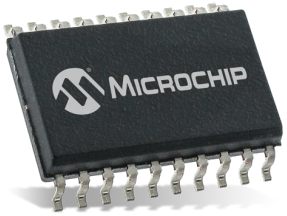

Figura 4: Microchip

5. Buenas practicas

5.1. Lectura del Datasheet

Antes de comenzar a trabajar con cualquier microcontrolador, es fundamental consultar su hoja de datos o *datasheet*. Este documento proporciona información detallada sobre las especificaciones eléctricas, características de los pines, capacidades de memoria, módulos disponibles (como timers, ADC, DAC, UART, etc.) y protocolos de comunicación soportados. Ignorar esta información puede conducir a errores como alimentar el dispositivo con un voltaje incorrecto, configurar mal un pin o usar de forma inadecuada un periférico.
Una buena práctica es tener siempre el datasheet a mano durante el diseño del circuito y la programación. Además, muchas hojas de datos incluyen diagramas internos y secuencias de inicialización que permiten un mejor entendimiento del funcionamiento del chip.

5.2. Configuración Correcta de Pines de Entrada y Salida

Cada pin del microcontrolador puede actuar como entrada, salida digital, salida con modulación por ancho de pulso (PWM) o incluso como entrada analógica, dependiendo de su configuración. Es esencial definir correctamente el modo de cada pin antes de utilizarlo, ya que una configuración incorrecta puede generar lecturas erróneas, daños en componentes conectados o fallos de funcionamiento.

Por ejemplo, si se configura un pin como salida y se conecta directamente a otro dispositivo que también actúa como salida, podría producirse un cortocircuito. Por ello, debe utilizarse el bloque de configuración adecuado, como ‘pinMode()‘ en el entorno Arduino o registros de control en otros entornos.

5.3. Protección contra Sobrecargas en Salidas Digitales

Los pines de salida digital de un microcontrolador están diseñados para entregar una corriente limitada, que normalmente está entre 20 y 40 mA. Si se conecta una carga que requiere más corriente, como un motor, un relé o una tira LED, sin una interfaz adecuada, se puede dañar permanentemente el microcontrolador.
Para evitar esto, es recomendable emplear dispositivos de conmutación como transistores, MOSFETs o módulos de drivers. Estos actúan como intermediarios, permitiendo al microcontrolador controlar grandes cargas sin exponerse directamente a altos niveles de corriente o tensiones externas.

¡Atención!

Nunca conectes directamente motores o cargas grandes al microcontrolador. ¡Usa transistores o relés!

5.4. Documentación Clara y Comentarios en el Código

Un código bien documentado no solo facilita el trabajo en equipo, sino que también mejora la mantenibilidad del proyecto a lo largo del tiempo. Es muy común que, tras semanas o meses, incluso el propio autor olvide la función de una sección de código mal comentada.
Algunas recomendaciones incluyen:

- Utilizar nombres de variables descriptivos (por ejemplo, ‘temperaturaSensor‘ en lugar de ‘t‘).
- Dividir el código en funciones con tareas bien definidas.
- Agregar comentarios explicando el propósito de bloques importantes.
- Especificar los rangos esperados para entradas y salidas.

Esto no solo ayuda a otros desarrolladores que revisen el código, sino que también es esencial para depurar errores y realizar mejoras futuras.

5.5. Pruebas Incrementales y Verificación de Hardware

Una buena práctica en el desarrollo de sistemas con microcontroladores es realizar pruebas por etapas. En lugar de implementar todo el sistema de una vez, se recomienda probar primero cada componente de forma aislada (por ejemplo, el sensor, el actuador, la comunicación serial), y luego integrarlos progresivamente.
Asimismo, es crucial verificar las conexiones físicas antes de energizar el circuito, utilizando un multímetro para comprobar continuidad, valores de resistencia y posibles cortocircuitos. Esta precaución previene daños tanto en el microcontrolador como en los periféricos conectados.

5.6. Manejo de Energía y Modos de Bajo Consumo

En muchas aplicaciones, especialmente en dispositivos portátiles o IoT, el consumo energético es un factor crítico. La mayoría de los microcontroladores modernos ofrecen diferentes modos

de bajo consumo, como *sleep*, *deep sleep*, o *power-down*, que permiten desactivar partes del sistema cuando no se están utilizando.
Implementar correctamente estas funcionalidades puede extender significativamente la autonomía del dispositivo, reduciendo la necesidad de baterías grandes o recargas frecuentes. Es recomendable estudiar los registros de control energético y planificar el software de manera que aproveche estos modos sin afectar el rendimiento de la aplicación.

Consejo

Nunca subestimes el valor de una buena organización del proyecto. Usa control de versiones (como `Git`) para mantener un historial claro de cambios, y respalda regularmente tu código y esquemas eléctricos. Esto no solo te salvará en caso de errores, sino que facilitará el trabajo colaborativo.

6. Proyectos interesantes

Los siguientes proyectos pueden desarrollarse utilizando plataformas como Arduino, las cuales facilitan el acceso a la programación y prototipado rápido gracias a su entorno amigable y a la gran comunidad de soporte. Estos proyectos permiten poner en práctica conceptos clave de microcontroladores, sensores, actuadores, control y comunicación.

6.1. Robot Autoequilibrado

Este proyecto simula el funcionamiento de un segway o robot bípedo. Utiliza sensores inerciales para medir la inclinación del robot. A través de un algoritmo de control PID, el sistema ajusta continuamente los motores para mantener el equilibrio. Es una excelente forma de introducirse en el control en lazo cerrado, el procesamiento de señales y el diseño mecánico básico.

6.2. Torreta Controlada por Joystick

Una torreta motorizada con dos grados de libertad que puede ser manipulada por un joystick analógico. Este proyecto permite aprender sobre control de servomotores, lectura de entradas analógicas, y estructuras de control en tiempo real. Puede escalarse fácilmente a control remoto inalámbrico o sistemas automatizados de seguimiento.

6.3. Máquina de Burbujas Automatizada

Consiste en un mecanismo que acciona una rueda sumergida en jabón y otro que genera flujo de aire para formar burbujas. Es ideal para trabajar conceptos de automatización básica, temporización y control de motores, además de fomentar la creatividad en el diseño de estructuras físicas.

6.4. Girasol Robótico (Seguidor de Luz)

Este proyecto simula una planta que sigue la luz solar. A través de sensores de luz, el sistema detecta la dirección de mayor iluminación y ajusta su orientación mediante un servomotor. Es útil para introducirse en sensores analógicos, condicionales y automatización basada en estímulos del entorno.

6.5. Cubo LED 4x4x4

Un proyecto visualmente impresionante que utiliza un arreglo tridimensional de LEDs para mostrar animaciones en el espacio. Involucra conceptos de multiplexado, programación estructurada, sincronización y diseño electrónico detallado. También fomenta la precisión en la construcción.

6.6. Robot WALL-E con Microservos

Inspirado en el famoso personaje animado, este robot emplea microservos para simular movimientos expresivos y ruedas para desplazarse. Además de ser una aplicación entretenida, permite trabajar con control de movimientos, diseño mecánico y lectura de sensores para evitar obstáculos.

6.7. Cerradura Secreta por Golpes (Secret Knock)

Una cerradura inteligente que se activa al reconocer una secuencia de golpes específica. Este proyecto desarrolla habilidades en adquisición de señales, comparación de patrones, y sistemas de seguridad básicos. Fomenta la creatividad en el diseño de entradas no convencionales.

6.8. Ventilador con Control de Temperatura

Este sistema adapta la velocidad de un ventilador según la temperatura ambiente. Utiliza sensores y modulación por ancho de pulso (PWM), y ejemplifica la aplicación de lazos de control en sistemas de climatización. Es muy útil para entender cómo se automatizan tareas comunes del entorno.

6.9. Robot Controlado por Gestos

Aprovechando sensores inerciales ubicados en la mano del usuario, este robot responde a movimientos como inclinaciones o giros. Es una introducción potente al diseño de interfaces naturales, control inalámbrico, y fusión de sensores para interpretación de gestos.

6.10. Otros Proyectos Sugeridos

- Lámpara RGB con control por potenciómetro o aplicación móvil
- Termómetro digital con visualización en pantalla
- Sistema de riego automático que responde a la humedad del suelo
- Mini estación meteorológica con múltiples sensores ambientales
- Alarma de seguridad activada por movimiento o proximidad

Estos proyectos pueden abordarse progresivamente, adaptándose a diferentes niveles de dificultad, y constituyen una excelente base para seguir aprendiendo.

Recursos Adicionales

- Sitio oficial de Arduino: `https://www.arduino.cc/en/Guide`
- Simulador y editor de circuitos Tinkercad: `https://www.tinkercad.com`

Bibliografía

- Valdés-Pérez, R. (2005). *Sistemas Embebidos con Microcontroladores*. Alfaomega Grupo Editor.

Créditos

Este documento fue elaborado por:

Antonia I. Morales Gutiérrez
Presidenta RAS IEEE UdeC
antonia.mgut7@gmail.com
GitHub: @Antonia-mgut

Universidad de Concepción — Junio 2025

Redes Sociales y Contacto de RAS IEEE UdeC:
ras.udec@gmail.com
Instagram: @ras.udec

Parte de este material fue desarrollado con apoyo de la herramienta ChatGPT (OpenAI), utilizada como asistente de redacción, edición técnica y organización de contenidos.

Términos de uso y licencia