**INFORME:**

Juan Sanabria 30276906

Gabriel Silva 29803338

* **Microcontroladores:**

Microcontrolador 8031:

El microcontrolador 8031 es un dispositivo de la familia MCS-51 desarrollada por Intel. Con una arquitectura de Harvard y un tamaño de palabra de 8 bits, se destaca por sus periféricos integrados, como temporizadores y puertos de entrada/salida, lo que lo hace adecuado para una variedad de aplicaciones embebidas.

Microcontrolador 8051:

Perteneciente a la misma familia, el microcontrolador 8051 comparte similitudes con el 8031, pero se distingue por su versatilidad en la configuración de pines para E/S, modos de funcionamiento flexibles y capacidad para operar en varios modos, incluido el de espera para ahorrar energía. Es una elección común en sistemas industriales y dispositivos embebidos.

Microcontrolador 8052:

El 8052 es una extensión del 8051, con mayores recursos de memoria RAM y ROM, así como una mayor cantidad de pines para E/S. Este microcontrolador ofrece modos de funcionamiento adicionales, proporcionando mayor flexibilidad en el diseño de sistemas embebidos avanzados. Es ideal para aplicaciones que requieren más capacidad de procesamiento y almacenamiento.

Estos microcontroladores de la familia MCS-51 han demostrado ser fundamentales en el desarrollo de sistemas embebidos, encontrando aplicación en una amplia gama de industrias debido a su robustez, versatilidad y capacidad para adaptarse a diversas necesidades de diseño

**Algunas de sus características:**

Microcontrolador 8031:

Arquitectura: Basado en la arquitectura de Harvard.

Tamaño de palabra: 8 bits.

Periféricos: Incluye temporizadores, puertos de entrada/salida y UART para comunicación serie.

Memoria: Espacios separados para programa y datos. Puede expandirse externamente.

Instrucciones: Conjunto de instrucciones CISC.

Microcontrolador 8051:

CPU de 8 bits

Procesador Booleano con el cual puede realizar operaciones bit a bit

128 bytes de RAM interna

4 Kbytes de memoria de programa interna (ROM)

5 fuentes de interrupción con 2 niveles de prioridad

32 bits de entrada/salida direccionables bit a bit

1 puerto serie Full dúplex (UART)

2 Contadores-Temporizadores de 16 bits programables

1 oscilador para las señales de reloj

Posibilidad de direccionar hasta 64Kbytes de memoria de programa externa

Posibilidad de direccionar hasta 64Kbytes de memoria de datos externa

varias velocidades desde 12MHz

Memoria: Al igual que el 8031, tiene espacios separados para programa y datos.

Modos de funcionamiento: Puede operar en varios modos, como modo de espera para ahorrar energía.

Microcontrolador 8052:

Características adicionales: Extensión del 8051 con más recursos.

Memoria: Mayor capacidad de RAM y ROM en comparación con 8051.

Puertos de entrada/salida: Ampliación de pines para E/S.

Modos de funcionamiento: Incluye modos adicionales para mayor flexibilidad.

**Aplicaciones comunes para los tres microcontroladores:**

Sistemas embebidos en electrodomésticos.

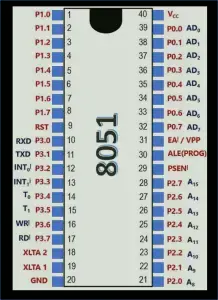
Controladores industriales.

Automatización de procesos.

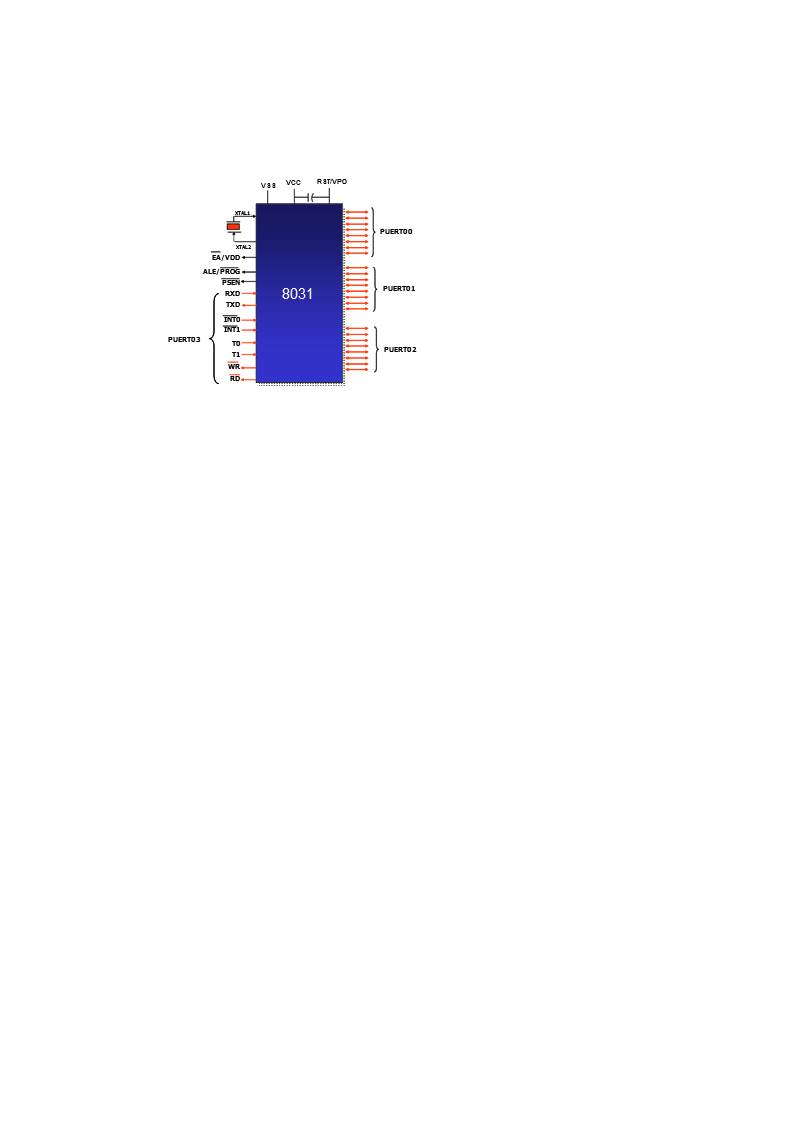
Dispositivos médicos.

Sistemas de seguridad

**Diagrama de pines:**



| **Número de PIN** | **Nombre** | **Función** |
| --- | --- | --- |
| 1 | VCC | Fuente de Energía |
| 2 | GND | Polo a Tierra |
| 3-10 | P0.0-P0.7 | Puerto de E/S 0 |
| 11-18 | P1.0-P1.7 | Puerto de E/S 1 |
| 19-26 | P2.0-P2.7 | Puerto de E/S 2 |
| 27-34 | P3.0-P3.7 | Puerto de E/S 3 |
| 35 | RST | Reanudar |
| 36 | DE | Habilitar bloqueo de dirección |
| 37 | PSEN | Habilitación del almacén de programas |
| 38 | EA | Acceso externo |
| 39 | XTAL1 | Entrada del oscilador de cristal |
| 40 | XTAL2 | Salida del oscilador de cristal |



**Funciones alternas de los puertos del 8051:**

Cada uno de los puertos de E/S tiene una función alternativa que le permite mayor

versatilidad al 8051.

El Puerto 0 sirve para multiplexar la parte baja de direcciones y al bus de datos

durante el acceso a código y memoria de datos externos.

El Puerto 1 tiene funciones alternativas solo en el 8052. Los bits 0 y 1 sirven como

T2 y T2EX respectivamente.

El Puerto 2 emite el byte de la parte alta de direcciones durante el acceso a

memoria, de código o datos, externa que utiliza direcciones de 16 bits.

El puerto 3 tiene las funciones listadas en seguida:

P3.0 RXD (Entrada de puerto serie)

P3.1 TXD (Salida de puerto serie)

P3.2 INT0 (Interrupción externa 0)

P3.3 INT1(Interrupción externa 1)

P3.4 T0 (Entrada para temporizador 0)

P3.5 T1 (Entrada para temporizador 1)

P3.6 WR (pulso de escritura para memoria ext.)

P3.7 RD (pulso de lectura para memoria ext)

**Que necesito para programar uno de estos microcontroladores?**

Para programar microcontroladores de la familia MCS-51, como el 8031, 8051 o 8052, en lenguaje C, necesitarás algunos elementos adicionales. Aquí hay una lista general de lo que necesitarías:

Placa de Desarrollo o Hardware de Programación:

Puedes utilizar una placa de desarrollo específica para el microcontrolador que estás utilizando. Estas placas suelen incluir un zócalo para el microcontrolador, así como interfaces de programación y depuración.

Otra opción es un programador independiente compatible con el microcontrolador MCS-51.

Programador y Cable de Conexión:

Un programador compatible con MCS-51, como el popular "USBASP" o "STC-ISP", que te permitirá cargar el código en el microcontrolador.

El cable de conexión adecuado para conectar el programador a la placa de desarrollo.

Entorno de Desarrollo Integrado (IDE):

Un entorno de desarrollo que soporte la programación en lenguaje C para microcontroladores MCS-51. Puedes usar herramientas como Keil C51, SDCC (Small Device C Compiler), o cualquier IDE que sea compatible con tu microcontrolador específico.

Compilador C para MCS-51:

Asegúrate de tener un compilador C compatible con la arquitectura MCS-51. Algunas opciones populares son el compilador de C de Keil y el compilador SDCC.

Conocimientos de Programación Embebida:

Familiarízate con la arquitectura y conjunto de instrucciones específicos del microcontrolador MCS-51.

Entiende cómo trabajar con registros de periféricos, configuración de pines y operaciones de temporización.

Documentación Técnica:

Tener acceso a las hojas de datos y manuales del microcontrolador específico para referencia y configuración.

Hardware de Prueba:

Siempre es útil tener herramientas para la depuración, como un osciloscopio o herramientas de depuración proporcionadas por el entorno de desarrollo.

**Ejemplo manejo de puertos y temporizadores:**

#include <8051.h>

// Definir el LED a controlar

#define LED\_PIN P1\_0

// Declaraciones de funciones

void delay\_ms(unsigned int milliseconds);

void main() {

// Configurar el puerto P1 como salida para el LED

LED\_PIN = 0; // Inicializar el LED apagado

P1M1 = 0x00; // Configurar P1 como salida

P1M0 = 0xFF;

// Configurar el temporizador 1 en modo 1 (16 bits)

TMOD = 0x10;

while (1) {

// Encender el LED

LED\_PIN = 1;

// Esperar 500 ms utilizando la función de retardo

delay\_ms(500);

// Apagar el LED

LED\_PIN = 0;

// Esperar 500 ms nuevamente

delay\_ms(500);

}

}

// Implementación de la función de retardo en milisegundos

void delay\_ms(unsigned int milliseconds) {

unsigned int i, j;

for (i = 0; i < milliseconds; i++) {

for (j = 0; j < 125; j++) {

// Ajusta estos valores según la frecuencia del microcontrolador

TH1 = 0xFC; // Carga inicial para un retardo de 1 ms

TL1 = 0x18;

TR1 = 1; // Inicia el temporizador

while (!TF1); // Espera a que se desborde el temporizador

TR1 = 0; // Detiene el temporizador

TF1 = 0; // Restablece el indicador de desbordamiento

}

}

}

Este código parpadea el LED conectado al pin P1.0 con un ciclo de encendido y apagado de 500 ms utilizando un temporizador para generar retardos precisos. Ajusta los valores de TH1 y TL1 según la frecuencia de tu microcontrolador para obtener el retardo deseado. Este ejemplo también utiliza la función delay\_ms para simplificar la creación de retardos en milisegundos.

* **Arduino:**

Arduino es una plataforma de electrónica de código abierto que se basa en hardware y software fácil de usar. Consiste en una placa de circuito física programable (a menudo llamada microcontrolador) y un entorno de desarrollo integrado (IDE) utilizado para escribir y cargar código de computadora en la placa. Arduino es popular entre aficionados, estudiantes y profesionales para crear proyectos interactivos y prototipos.

Los componentes clave de la plataforma Arduino incluyen:

Placa de Arduino: La placa de circuito física que alberga el microcontrolador, que es el cerebro de Arduino. Hay varios modelos de placas Arduino con diferentes especificaciones.

Microcontrolador: El microcontrolador es el corazón de la placa de Arduino y se encarga de ejecutar el programa o código cargado en él. El microcontrolador más común utilizado en las placas Arduino es la serie Atmel AVR, pero también hay placas Arduino basadas en otros microcontroladores, como ARM.

IDE de Arduino (Entorno de Desarrollo Integrado): El software utilizado para escribir, compilar y cargar código en la placa de Arduino. Proporciona una interfaz amigable para programar y depurar proyectos Arduino.

Lenguaje de Programación: La programación de Arduino se realiza utilizando una versión simplificada de C++ con funciones y bibliotecas adicionales específicas.

**Pasos generales para poder programar en arduino:**

Instalar el IDE de Arduino:

Descarga e instala el Arduino IDE desde el sitio web oficial: Arduino Software.

Abre el IDE después de la instalación.

Conectar Arduino al Computador:

Conecta tu placa de Arduino a tu computadora mediante un cable USB.

Seleccionar el Tipo de Placa:

Ve a "Herramientas" en el menú del Arduino IDE.

Selecciona el modelo de placa que estás utilizando desde la opción "Placa".

Seleccionar el Puerto:

En el mismo menú "Herramientas", elige el puerto al que está conectado tu Arduino.

Escribir y Cargar el Programa:

En el IDE, abre un nuevo sketch (Archivo -> Nuevo).

Escribe tu código en el área de edición. El código de Arduino está basado en C/C++ simplificado.

Verifica tu código haciendo clic en el ícono de "Verificar" (checkmark) en la esquina superior izquierda.

Carga el código en tu Arduino haciendo clic en el ícono de "Cargar" (flecha hacia la derecha).

Observar la Consola:

Abre la consola del IDE (Ventana -> Consola) para ver mensajes de depuración y errores.

Observar el LED de la Placa:

Muchas placas Arduino tienen un LED incorporado que parpadea durante la carga del programa. Observa si parpadea para verificar que la carga ha sido exitosa.

Modificar y Experimentar:

Haz cambios en el código según sea necesario y repite el proceso de carga para ver cómo afectan tus modificaciones al comportamiento del programa.

Conectar Componentes:

Si estás trabajando con sensores, actuadores u otros componentes, asegúrate de conectarlos correctamente a la placa según las especificaciones del fabricante y tu código.

Depurar si es necesario:

Usa la consola, las luces LED en la placa y otros métodos para depurar si encuentras problemas en tu código.

**Ejemplo básico:**

// Definir el número del pin donde está conectado el LED

const int ledPin = 13;

// El código en la función setup() se ejecuta una vez al inicio

void setup() {

// Inicializar el pin del LED como una salida

pinMode(ledPin, OUTPUT);

}

// El código en la función loop() se ejecuta repetidamente

void loop() {

// Encender el LED

digitalWrite(ledPin, HIGH);

// Esperar 1 segundo

delay(1000);

// Apagar el LED

digitalWrite(ledPin, LOW);

// Esperar 1 segundo

delay(1000);

}

**Costo para programar en arduino:**

Programar en Arduino desde cero puede ser bastante asequible, ya que la plataforma Arduino está diseñada para ser accesible tanto en términos de costos como de habilidades necesarias. Aquí hay algunos puntos a considerar:

Costo del Hardware:

Las placas Arduino en sí mismas son relativamente asequibles. Puedes adquirir modelos básicos por menos de $10 a $20, y hay opciones más avanzadas disponibles a precios más altos.

Además de la placa Arduino, es posible que necesites componentes adicionales como cables, resistencias, LEDs y sensores. Estos componentes suelen ser económicos y están disponibles en kits para principiantes.

Software:

El entorno de desarrollo integrado (IDE) de Arduino es gratuito y está disponible para Windows, Mac y Linux. No hay costos asociados con la adquisición del software.

Recursos de Aprendizaje:

Hay una abundancia de recursos en línea gratuitos para aprender a programar en Arduino, como tutoriales, documentación oficial de Arduino y comunidades en línea.

Puedes encontrar libros y cursos pagos, pero hay muchas fuentes gratuitas y de alta calidad para comenzar.

Herramientas de Programación:

Para empezar, solo necesitas un ordenador con conexión a Internet y el IDE de Arduino instalado. No hay necesidad de invertir en herramientas costosas.

Habilidades Técnicas:

Arduino está diseñado para ser accesible para principiantes, incluso aquellos que no tienen experiencia previa en programación. Muchos tutoriales y recursos están destinados a personas que comienzan desde cero.