**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Diseño de productos electrónicos con microcontroladores. |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 290201083 - Mejorar el funcionamiento de máquinas y procesos, buscando su eficiencia y productividad. | RESULTADOS DE APRENDIZAJE | 290201083-02. Analizar la estructura y programación de un microcontrolador acorde a parámetros establecidos. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 02 |
| NOMBRE DEL COMPONENTE FORMATIVO | Análisis estructural y funcional de un microcontrolador |
| BREVE DESCRIPCIÓN | El proceso de creación de un programa para microcontroladores incluye redactar instrucciones en lenguaje de programación, compilar el código y realizar el enlazado. Luego, el programa se simula para verificar su funcionamiento antes de quemar el *firmware* en la memoria del microcontrolador. Finalmente, se realizan pruebas en *hardware y softwar*e para asegurar su correcto desempeño. |
| PALABRAS CLAVE | Microcontrolador, compilación, simulación, *firmware,* pruebas. |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 2 - CIENCIAS NATURALES, APLICADAS Y RELACIONADAS |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**

1. Estructura básica de los microcontroladores

2. Lenguaje de programación

3. Estructura del proceso de creación de un programa para microcontroladores

1. **INTRODUCCIÓN**

El desarrollo de programas para microcontroladores es un proceso fundamental en la ingeniería electrónica y la programación de sistemas embebidos. Los microcontroladores son dispositivos que integran componentes de procesamiento, almacenamiento y comunicación en una sola unidad, lo que permite automatizar tareas en dispositivos y sistemas cotidianos. La creación de un programa funcional para estos sistemas requiere precisión y una comprensión detallada de su arquitectura.

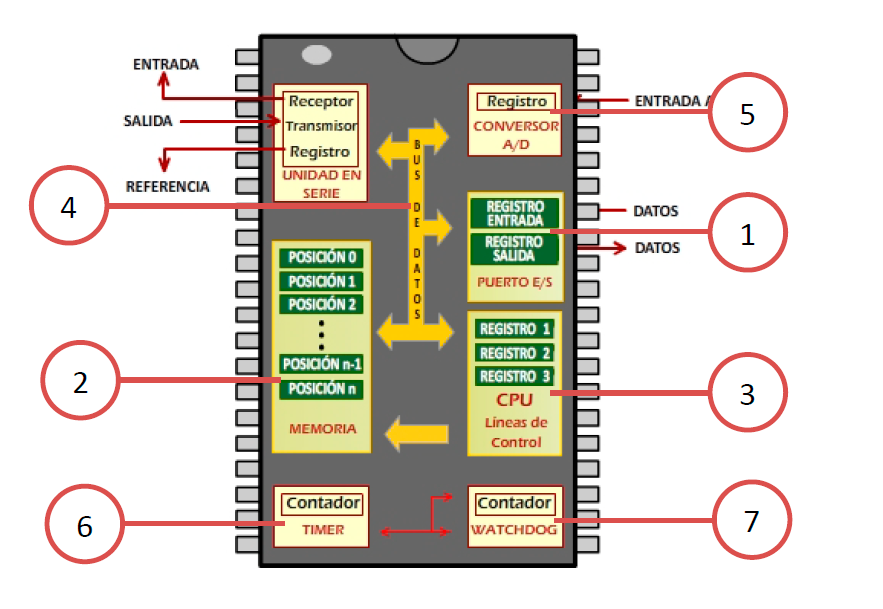
|  |  |
| --- | --- |
| El proceso de programación de un microcontrolador implica varias etapas, comenzando con la redacción de instrucciones en un lenguaje de programación específico. Posteriormente, el código fuente se compila y enlaza para traducirlo a un formato comprensible por el hardware. Esta fase convierte las instrucciones en lenguaje máquina, permitiendo que el microcontrolador las interprete de manera adecuada y eficiente. | Dibujos en color de Arduino RoMeo V2 |

Una vez compilado, el programa se somete a simulación y pruebas antes de ser "quemado" en la memoria del microcontrolador. La simulación ayuda a predecir el comportamiento del programa sin el uso físico del dispositivo, mientras que el quemado transfiere el *firmware* directamente al *hardware*. Finalmente, se realizan pruebas tanto en *hardware* como en *software* para garantizar un funcionamiento óptimo y una integración precisa del microcontrolador en su entorno operativo.

1. **DESARROLLO DE CONTENIDOS:** 
   * + 1. **Estructura básica de los microcontroladores**

Los microcontroladores están constituidos, fundamentalmente, por los siguientes componentes:

**Figura 1.** Microcontroladores

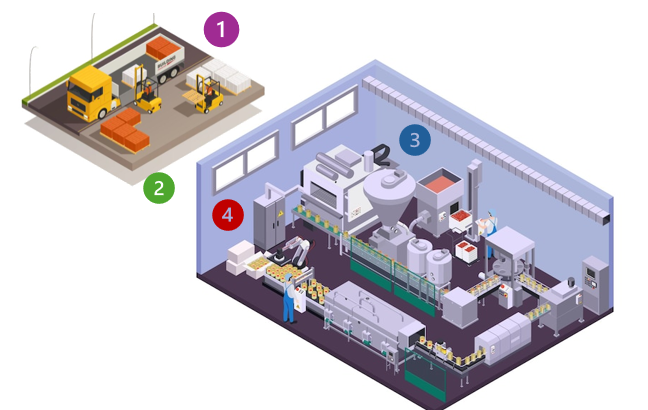


|  |
| --- |
| Acordeón  CF02\_1\_Estructura básica de los microcontroladores |

**Analogía con una planta procesadora**

La estructura de un microcontrolador puede asimilarse a una planta procesadora de papas fritas:

**Figura 2.** Empresa papas fritas



|  |
| --- |
| Acordeón o imagen interactiva  CF02\_1\_Analogía con una planta procesadora |

* + - 1. **Lenguaje de programación**

|  |  |
| --- | --- |
| Cada lenguaje de programación posee características propias en cuanto a gramática y sintaxis. A diferencia del lenguaje humano, los lenguajes de programación transmiten cuatro ideas fundamentales y requieren precisión, claridad y ausencia de ambigüedad. Existen dos niveles principales de lenguajes de programación: | LLM, AI Large Language Model concept. Businessman working on laptop and holding LLM icons on screen. A language model distinguished by its general-purpose language generation capability. Chat AI. |

**Lenguaje Assembly (ASM)**

|  |  |
| --- | --- |
| Cada lenguaje tiene características propias en cuanto a su gramática y sintaxis. A diferencia del lenguaje humano que permite transmitir múltiples ideas a la vez, los lenguajes de programación no requieren que comuniquen gran variedad de pensamientos, es suficiente que permitan la transmisión de solo cuatro pensamientos básicos. Además, la comunicación a través de estos lenguajes tiene que ser precisa, clara y sin ambigüedades. | Asm Icon |

**Directivas en lenguaje Assembly**

Las directivas en Assembly son similares a las instrucciones pero no dependen del modelo específico del microcontrolador; representan una característica propia del lenguaje ASM y se utilizan para asignar significados o propósitos a través de variables o registros.

**Directivas de control**

Las directivas de control en Assembly son comandos que gestionan aspectos específicos del programa, como la inclusión de archivos, la definición de constantes y variables, y la configuración de memoria y *bits.*

|  |
| --- |
| SLIDE  CF02\_2\_Directivas de control |

**Directivas de configuración**

Las directivas de configuración en Assembly permiten ajustar parámetros esenciales del programa, como la configuración de *bits,* el modelo de microcontrolador, y la definición de bloques de datos o condiciones de ejecución, optimizando el control y funcionalidad del código.

|  |
| --- |
| TARJETAS  CF02\_2\_Directivas de configuración |

**Instrucciones de condición**

Las instrucciones de condición en Assembly permiten controlar el flujo de ejecución del programa según se cumplan o no ciertas condiciones, proporcionando flexibilidad en la toma de decisiones dentro del código.

|  |
| --- |
| PESTAÑAS  CF02\_2\_Instrucciones de condición |

**Operadores matemáticos**

Los operadores matemáticos en Assembly permiten realizar cálculos aritméticos y lógicos en el programa, proporcionando una variedad de operaciones como suma, multiplicación, desplazamientos de *bits* y comparaciones, esenciales para manipular y evaluar datos de manera precisa en el código.

**Tabla 1.** Operadores matemáticos

|  |  |  |
| --- | --- | --- |
| **Símbolo** | **Descripción** | **Ejemplo** |
| $ | Estado actual del contador del programa | goto $ + 3 |
| ( | Corchete izquierdo | 1 + (d \* 4) |
| ) | Corchete derecho | (Length + 1) \* 256 |
| ! | Complemento lógico | if ! (a = b) |
| ~ | Complemento | flags = ~flags |
| - | Negación (segundo complemento) | -1 \* Length |
| high | Devuelve el byte más alto | movlw high CTR\_Table |
| low | Devuelve el byte más inferior | movlw low CTR\_Table |
| \* | Multiplicador | a = b \* c |
| / | Divisor | a = b / c |
| % | Divisor por módulo | entry\_len = tot\_len % 15 |
| + | Sumando | tot\_len =entry\_len \* 3 + 1 |
| - | Restando | entry\_len = (tot - 1) / 8 |
| << | Moviendo a la izquierda | val = flags << 1 |
| >> | Moviendo a la derecha | val = flags >> 1 |
| >= | Mayor que o igual | if entry\_idx >= num\_entries |
| > | Mayor que | if entry\_idx > num\_entries |
| < | Menor que | if entry\_idx < num\_entries |
| <= | Menor que, o igual | if entry\_idx <= num\_entries |
| == | Igual | if entry\_idx == num\_entries |
| != | No igual | if entry\_idx != num\_entries |
| & | Operación AND en los bits | flags = flags & ERROR\_BIT |
| ^ | Exclusivo OR en los bits | flags = flags ^ ERROR\_BIT |
| ` | ` | Lógica OR sobre bits |
| && | Lógica OR sobre bits (if) | if (len ==512) && (b ==c) |
| = | Igual | entry\_index = 0 |
| += | Añadir y asignar | entry\_index +=1 |
| -= | Restar y asignar | entry\_index -=1 |
| \*= | Multiplicar y asignar | entry\_index \*= entry\_length |
| /= | Dividir y asignar | entry\_index /= entry\_length |
| %= | Divide el módulo y asignar | entry\_index %=1 |
| <<= | Mueve a la izquierda y asigna | flags << = 3 |
| >>= | Mueve a la derecha y asigna | flags >> = 3 |
| &= | Lógica AND y asigna | flags &= ERROR\_FLAG |
| ` | =` | Lógica OR y asigna |
| ^= | Exclusivo OR y asigna | flags ^= ERROR\_FLAG |
| ++ | Incrementa por uno | 1++ |
| -- | Reduce por uno | 1- |

Ejemplo de lenguaje Assembly: "Hola Mundo" en x86 bajo DOS:

|  |
| --- |
| .model small  .stack  .data  Cadena1 DB 'Hola Mundo.$'  .code  programa:  mov ax, @data  mov ds, ax  mov dx, offset Cadena1  mov ah, 9  int 21h  end programa |

Este ejemplo imprime "Hola Mundo" en DOS usando lenguaje Assembly.

* + - 1. **Estructura del proceso de creación de un programa para microcontroladores**

|  |  |
| --- | --- |
| En esta fase, se redactan las instrucciones del programa. El programa, escrito en un lenguaje de programación comprensible para el ser humano (frecuentemente utilizando lenguajes formales descritos por gramáticas independientes del contexto), no es ejecutado inmediatamente en una computadora. | Primer plano de la placa de circuito electrónico con CPU microchip componentes electrónicos de fondo |

La opción más común es compilar el programa, aunque también puede ejecutarse mediante un intérprete. El código fuente debe someterse a un proceso de transformación para convertirse en lenguaje máquina, interpretable por el procesador, conocido como **compilación**. La creación de un programa ejecutable (como un archivo .exe en Microsoft Windows) generalmente implica dos pasos:

**Simulación del programa**

|  |  |
| --- | --- |
| Para probar el programa sin emplear físicamente un recurso electrónico, se pueden utilizar herramientas tecnológicas, ya sean de *software o hardware*, que permitan verificar si la aplicación funciona correctamente; estas herramientas se denominan **simuladores**. | Resumen Tecnología moderna del desarrollador de pantalla de código de programación C Lenguaje de programación de secuencias de comandos de computadora |

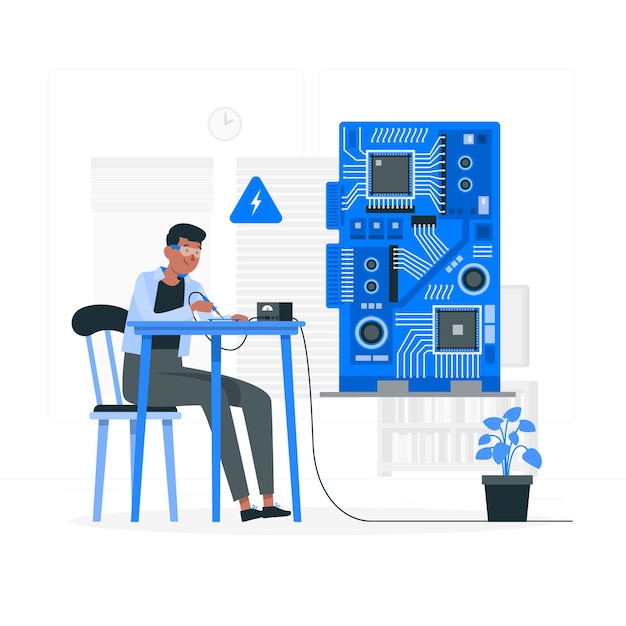
Un simulador es un *software* capaz de reproducir virtualmente el comportamiento real de un microcontrolador a partir del código .asm creado. Existen diferentes simuladores en el mercado que permiten desarrollar diseños complejos para programar microcontroladores, como los siguientes:

**Quemado del programa**  
Una vez simulado el programa y verificado su funcionamiento, se genera el archivo .asm con las instrucciones, conocido como ***firmware***o **programación en firme**. Este *firmware* se descarga o **quema** en la memoria del microcontrolador en un proceso denominado quemado.



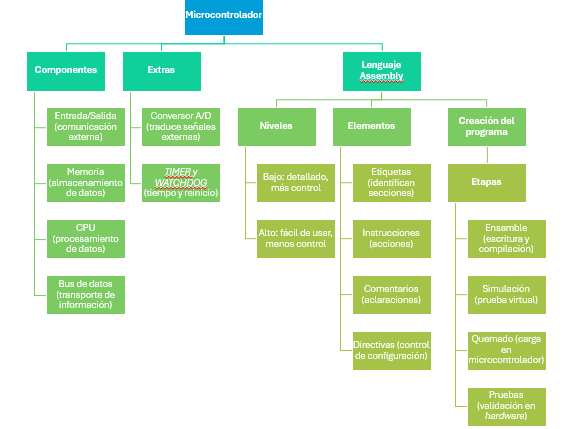
El *firmware* funciona como un bloque de instrucciones con las órdenes específicas de lo que se quiere ejecutar (por ejemplo, la activación de una alarma sonora). Este bloque establece la lógica de bajo nivel que controla los circuitos electrónicos del dispositivo. Funcionalmente, el *firmware* es el intermediario entre las órdenes externas que recibe el dispositivo y su electrónica, permitiendo la correcta ejecución de estas órdenes.

**Pruebas del programa**  
Las pruebas permiten verificar si el código .asm funciona correctamente y si el microcontrolador está en buen estado. Para ello, es recomendable seguir un procedimiento básico que incluya tanto el *hardware* como el *software* del microcontrolador.



1. **SÍNTESIS**

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo.



1. **ACTIVIDADES DIDÁCTICAS (Se debe incorporar mínimo 1, máximo 2)**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Introducción a los microcontroladores y lenguaje Assembly |
| Objetivo de la actividad | Comprender los componentes fundamentales de un microcontrolador, su funcionamiento y el uso básico del lenguaje assembly para el desarrollo de programas, mediante el análisis de sus etapas, elementos y funciones esenciales en sistemas electrónicos. |
| Tipo de actividad sugerida | Cuestionario |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | *CF02\_Actividad didactica* |

1. **MATERIAL COMPLEMENTARIO:**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| Estructura básica de los microcontroladores | Delgado Miguel Angel. (2020). Estructura interna del microcontrolador. [Archivo de video] Youtube. | Video | <https://www.youtube.com/watch?v=glLbD1aqJKo&ab_channel=DelgadoMiguelAngel> |
| Lenguaje de programación | Coding Academy Perú. (2020). Assembler - Lo básico del lenguaje. [Archivo de video] Youtube. | Video | <https://www.youtube.com/watch?v=eHA0WHFUinQ&ab_channel=CodingAcademyPer%C3%BA> |
| Lenguaje de programación | Sergie Arizandieta (2022). Aprende Programación en Assembler. [Archivo de video] Youtube. | Video | <https://www.youtube.com/watch?v=cFhUA7DnIVo&list=PLON3-BoIoWiV0Te8sxsvXw8u2k3DmBZun&ab_channel=SergieArizandieta> |
| Estructura del proceso de creación de un programa para microcontroladores | Electrónica y Circuitos. (2020). BIENVENIDA AL CURSO DE PROGRAMACIÓN DE MICROCONTROLADORES PIC EN LENGUAJE C. [Archivo de video] Youtube. | Video | <https://www.youtube.com/watch?v=SXl-VD5WU-s&list=PLONPO-iVba9nbY_KTCHt9GGj9dGSj1qYo&ab_channel=Electr%C3%B3nicayCircuitos> |
| Estructura del proceso de creación de un programa para microcontroladores | Xataka (2020). ARDUINO: QUÉ ES, CÓMO FUNCIONA y PARA QUÉ se puede UTILIZAR. [Archivo de video] Youtube. | Video | <https://www.youtube.com/watch?v=Zs9MZosVuqo&ab_channel=Xataka> |

1. **GLOSARIO:**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| Código fuente: | texto escrito en lenguaje de programación, que luego se compila para su ejecución en el hardware. |
| Compilación: | proceso de traducción del código fuente en un lenguaje de bajo nivel para que el *hardware* pueda interpretarlo. |
| Enlazado: | paso en el cual se une el código de bajo nivel con las bibliotecas necesarias para crear un programa ejecutable. |
| *Firmware:* | conjunto de instrucciones específicas que controlan los circuitos electrónicos de un dispositivo. |
| Interprete: | programa que ejecuta el código fuente línea por línea sin necesidad de compilarlo. |
| Lenguaje de programación: | sistema formal de símbolos y reglas usado para escribir instrucciones entendibles por el microcontrolador. |
| Microcontrolador: | dispositivo electrónico que integra procesamiento, almacenamiento y comunicación en una sola unidad. |
| Pruebas de *hardware:* | evaluaciones que aseguran el funcionamiento correcto de los componentes físicos del microcontrolador. |
| Quemado: | proceso de transferencia del *firmware* a la memoria del microcontrolador. |
| Simulador: | *software* que permite verificar virtualmente el funcionamiento de un microcontrolador a partir del código programado. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

VanSickle, T. (2001). Programming microcontrollers in C. Academic Press.

Hyde, R. (2010). The art of assembly language (2nd ed.). No Starch Press.

Barnett, R. H., Cox, S., & O’Cull, L. (2006). Embedded C programming and the Atmel AVR. Delmar Cengage Learning.

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia  *(Para el SENA indicar Regional y Centro de Formación)* | Fecha |
|  | Paola Alexandra Moya | Evaluadora instruccional | Regional Antioquia - Centro de Servicios de Salud | 2024 |
|  | Olga Constanza Bermúdez Jaimes | Responsable Línea de Producción Antioquia | Regional Antioquia - Centro de Servicios de Salud | 2024 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
| Autor (es) |  |  |  |  |  |