Taller de Sistemas Embebidos
Sistemas Embebidos

Información relevante

Taller de Sistemas Embebidos

Asignatura correspondiente a la **actualización 2023** del Plan de Estudios 2020 y resoluciones modificatorias, de Ingeniería Electrónica de FIUBA

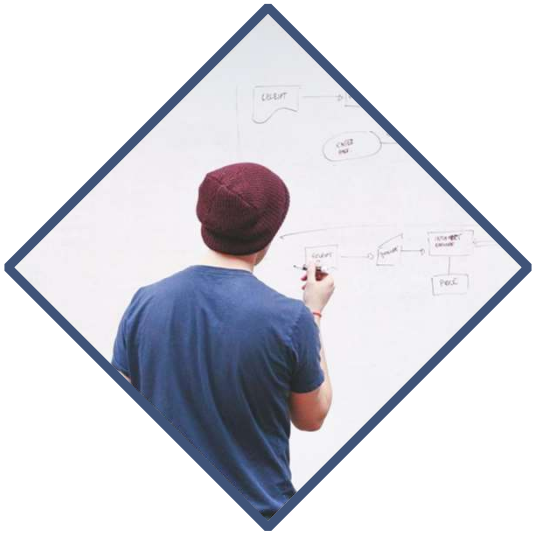
Estructura Curricular de la Carrera

El **Proyecto Intermedio** se desarrolla en la asignatura **Taller de Sistemas Embebidos**, la cual tiene un enfoque centrado en la **práctica propia de la carrera** más que en el desarrollo teórico disciplinar, con eje en la **participación de las y los estudiantes**

Más información . . .

. . . sobre la **actualización 2023** . . . https://www.fi.uba.ar/grado/carreras/ingenieria-electronica/plan-de-estudios

. . . sobre el **Taller de Sistemas Embebidos** . . . https://campusgrado.fi.uba.ar/course/view.php?id=1217



¡Hola!

Soy Juan Manuel Cruz
Taller de Sistemas Embebidos
Consultas a: jcruz@fi.uba.ar

1

Introducción

Actualización 2023 del Plan de Estudios 2020 y resoluciones . . .

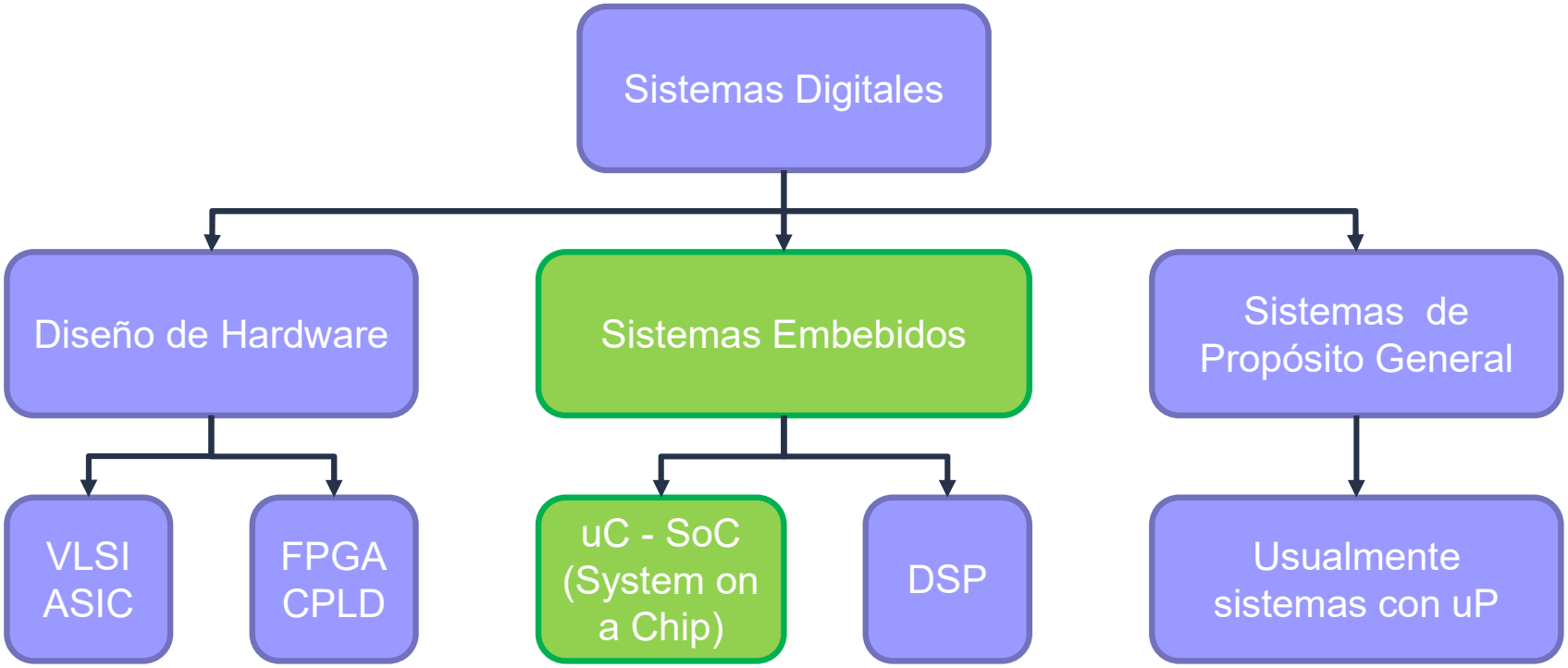
¿Qué es un Sistema Embebido?

- Un Sistema es un conjunto de partes/componentes interrelacionados que están diseñados/desarrollados para realizar tareas comunes o para realizar algún trabajo específico para el cual ha sido creado
- Embebido significa incluir algo con otra cosa por una razón. O simplemente podemos decir algo que está integrado o adjunto a otra cosa
- Ahora, después de comprender lo que significan las palabras Sistema y Embebido, podemos comprender fácilmente qué se pretende decir con Sistemas Embebidos

Sistema Embebido

Sistema electrónico de cómputo, compuesto por hardware y software, que incluye procesamiento de datos para realizar un función específica

¿En qué nos concentraremos?



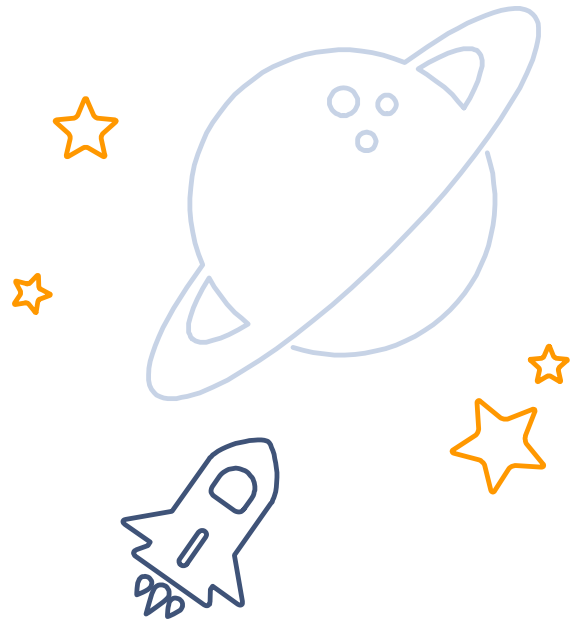
El diseño de Sistemas Embebidos se orienta a:

- Reducir: tamaño (portátil) / consumo (alimentado a batería) / costo (acorde al mercado)
- Aumentar: eficiencia & confiabilidad (siempre operativo y haciendo lo que hay que hacer) / re-usabilidad (HW & SW portables)
- Mejorar: desempeño (lo que hay que hacer)
- Asegurar: determinismo & tiempo de respuesta (igual respuesta a igual secuencia de estímulos, computo adecuado en un tiempo acotado) / seguridad (safety, si falla puede producir daños)
- Atender: mayor cantidad de tareas posibles, etc.
- Contar con: conectividad e interfaz de usuario en uso corriente
- Congeniar: recursos de HW con los requerimientos del SW

Estado del Arte

- Contamos con plataformas (uC - SoC/DSP/FPGA/ASIC/ etc.) de rendimiento y recursos en crecimiento que permitan atender el incremento del procesamiento, necesario para soportar periféricos avanzados con capacidad de atender nuevas conectividades e interfaces de usuario requeridas por el mercado (usuarios)
- Variada oferta de plataformas competitivas en costo, disponibilidad, soporte, herramientas (de HW & SW); en especial en el campo de los microcontroladores de nueva generación (ARM: 32 bits))
- Esto permite recurrir a las mejores prácticas de Ingeniería de Software, al uso de modelos, lenguajes de alto nivel, con y sin un sistema operativo de tiempo real (RTOS), empleando técnicas de programación específicas para lograr eficiencia, confiabilidad y re-usabilidad

Solución Adecuada



. . . lo más simple posible, previa determinación del objetivo de excelencia a cumplir, obviamente contando con la documentación debida y recurriendo a la metodología de trabajo adecuada

2

Solución Adecuada

1er Cuatrimestre de 2024, dictado por primera vez . . .

Diseño/Desarrollo/Depuración de Hardware y Software: Circuito eléctrico (electrónica analógica/digital/ alimentación) – Circuito Impreso – Producto – Manufactura – Programas (Firmware/Middleware/Software)

En la toma de decisiones influyen:

- El tipo y campo de aplicación, requerimiento del cliente y del medio (regulaciones)
- La idiosincrasia, conocimiento, experiencia y habilidad del profesional
- El contexto:
  - La provisión de herramientas de diseño/desarrollo/depuración
  - La provisión de insumos (conectores/componentes electrónicos/módulos electrónicos/cables de interconexión/gabinete/embalaje/etc.)
  - La provisión de manufactura de circuito impreso/módulo electrónico/cable de interconexión/etc. (montaje y soldadura de componentes/control de calidad/prueba y puesta en marcha/montaje de módulos electrónicos y cables de interconexión en gabinete/embalaje/marcado/etc.)

Diseño/Desarrollo/Depuración de Hardware

- Circuito Eléctrico
  - Diagrama de Bloques de funciones principales, muestra:
    - Cómo se divide el sistema en funciones principales
    - Cómo se interconectan las funciones principales entre sí
  - Diagrama de c/función principal, muestra:
    - Cómo se interconectan los componentes/módulos electrónicos entre sí
      - Adecuar cada componente/módulo electrónico al que usará
      - Cumplir 100% la hoja de datos del componente/módulo electrónico
  - Cumplir Reglas de Interconexión (Familias Lógicas/ . . .)

Diseño/Desarrollo/Depuración de Software

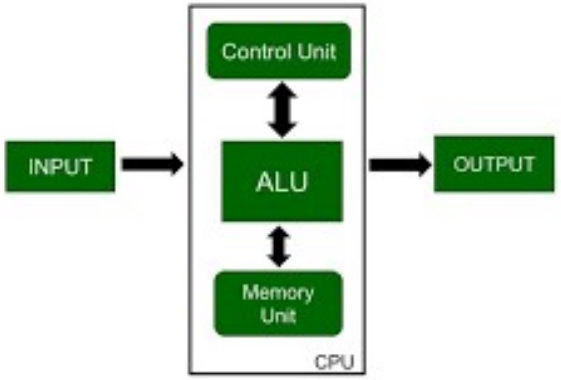
- Nos preocupa tanto el determinismo como el tiempo de respuesta del Sistema
- Sistemas Duros (Hard Real Time): restricciones de tiempo rigurosas
  - Sistemas de control en los cuales se debe escrutar, procesar y actuar en un plazo perentorio (Estrictos), …
  - …, si esto no se cumple el Sistema queda a lazo abierto, pudiendo tener consecuencias catastróficas
- Sistemas Blandos (Soft Real Time): restricciones de tiempo menos rigurosas
  - Sistemas de adquisición de datos o multimedia (Flexibles o Firmes), en …
  - …, si esporádicamente se degrada o se pierde la respuesta, puede ser inconveniente/incómodo pero sin consecuencias catastróficas

Diseño/Desarrollo/Depuración de Software

- Un lenguaje de programación es un lenguaje formal que especifica un conjunto de instrucciones para que una computadora realice tareas específicas
- Se utiliza para escribir programas y aplicaciones de software y para controlar y manipular sistemas informáticos
- Existen muchos lenguajes de programación diferentes, cada uno con su propia sintaxis, estructura y conjunto de comandos. Algunos de los lenguajes de programación más utilizados incluyen Java, Python, C++, JavaScript y C#
- La elección del lenguaje de programación depende de los requisitos específicos de un proyecto, incluida la plataforma que se utiliza, la audiencia prevista y el resultado deseado
- Los lenguajes de programación continúan evolucionando y cambiando, con el tiempo se desarrollan nuevos lenguajes y se actualizan los más antiguos para satisfacer las necesidades cambiantes

Diseño/Desarrollo/Depuración de Software

- Una computadora es un dispositivo que puede aceptar instrucciones humanas, las procesa y responde a ellas; o una computadora es un dispositivo informático que se utiliza para procesar datos bajo el control de un programa de computadora
- Un programa es una secuencia de instrucciones junto con datos
- Los componentes básicos de una computadora son:
  - Unidad de Entrada
  - Unidad Central de Procesamiento (CPU), que se divide en:
    - Unidad de Memoria
    - Unidad de Control
    - Unidad Aritmética Lógica
  - Unidad de Salida



Diseño/Desarrollo/Depuración de Software

- A la CPU se le llama el cerebro de nuestra computadora porque acepta datos, le proporciona espacio de memoria temporal hasta que se almacena (guarda), realiza operaciones lógicas en él, por lo tanto procesa (aquí también significa convierte) datos (en información)
- Una computadora se compone de hardware y software
- El software es un conjunto de programas que realizan múltiples tareas juntas
- Un sistema operativo también es software (software del sistema) que ayuda a los humanos a interactuar con el sistema informático
- Estos programas informáticos están escritos en un lenguaje de programación de alto nivel
- Los lenguajes de alto nivel son lenguajes casi humanos, más complejos que el lenguaje comprensible por la computadora, que se llama lenguaje de máquina o lenguaje de bajo nivel (solo entiende lenguaje binario (lenguaje de 0s y 1s))

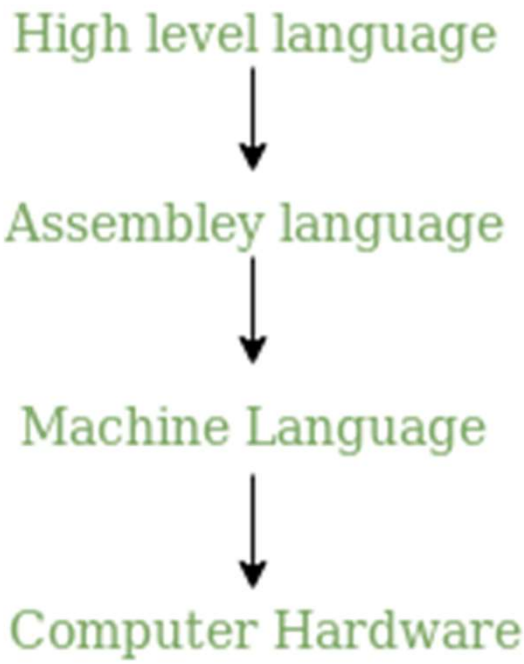
Diseño/Desarrollo/Depuración de Software

Jerarquía de los Lenguajes de Programación

Entre el lenguaje de alto nivel y el de máquina, existen lenguajes ensambladores (assembly, código de máquina simbólico).

Los lenguajes ensambladores (assembly) son específicos de la arquitectura de la computadora. Se requiere de un programa de aplicación (Assembler) para convertir código ensamblador (assembly) en código de máquina ejecutable

El lenguaje de alto nivel es portable pero requiere interpretación o compilación para convertirlo en lenguaje de máquina comprensible por la computadora

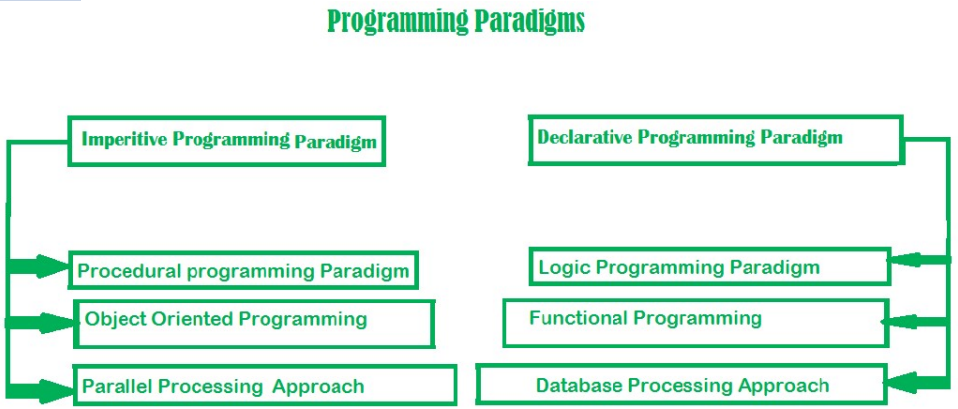


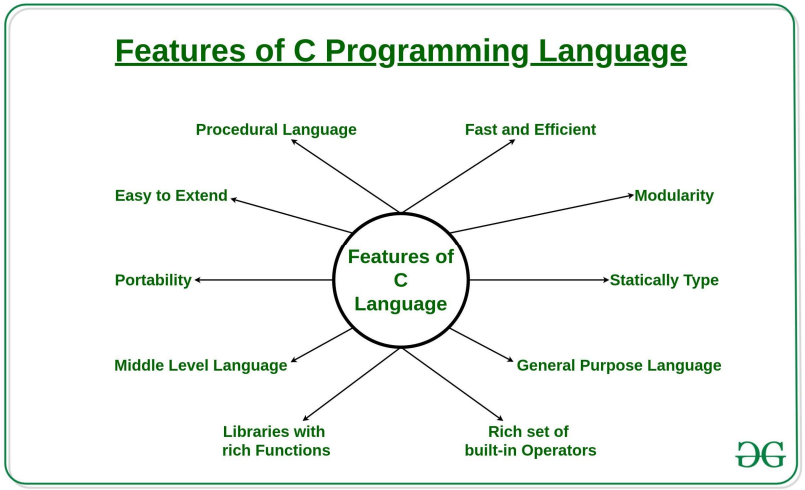
Diseño/Desarrollo/Depuración de Software

C es un lenguaje de programación procedimental. Desarrollado inicialmente por Dennis Ritchie en 1972

Se desarrolló principalmente como un lenguaje de programación de sistemas para escribir un sistema operativo

Las características principales del lenguaje C incluyen acceso de bajo nivel a la memoria, un conjunto simple de palabras clave y un estilo limpio; que lo hacen adecuado para la programación de sistemas como un sistema operativo o el desarrollo de compiladores





Diseño/Desarrollo/Depuración de Software

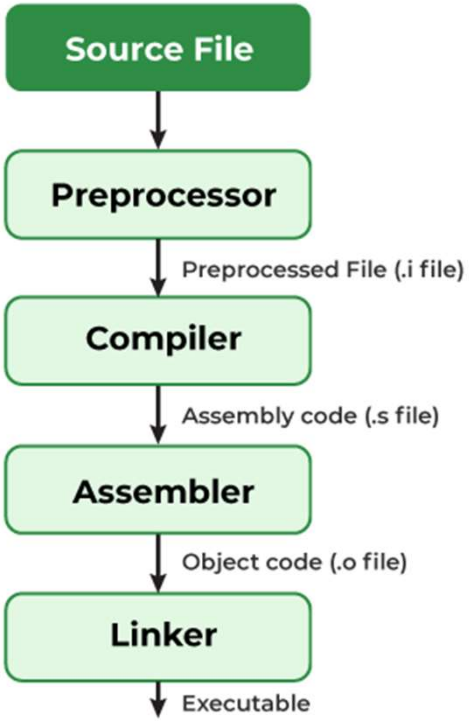
El código fuente debe pasar por varios pasos antes de convertirse en un programa ejecutable y poder correr en la CPU

Preprocessor: busca errores de sintaxis en el código fuente

Compiler/Assembler: traduce el lenguaje C/Assembly a código objeto (un código de máquina que no está listo para ejecutarse)

Linker: vincula el código objeto con funciones de biblioteca precompiladas, creando un programa ejecutable

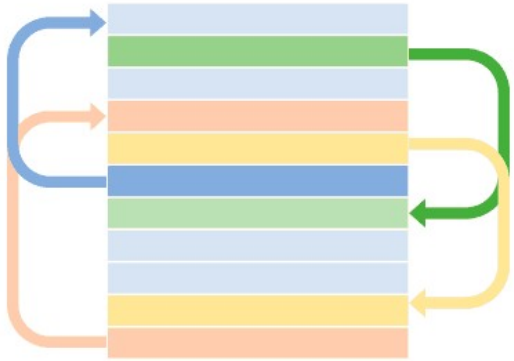
Se requiere un Loader para carga el programa ejecutable en memoria, para su posterior ejecución



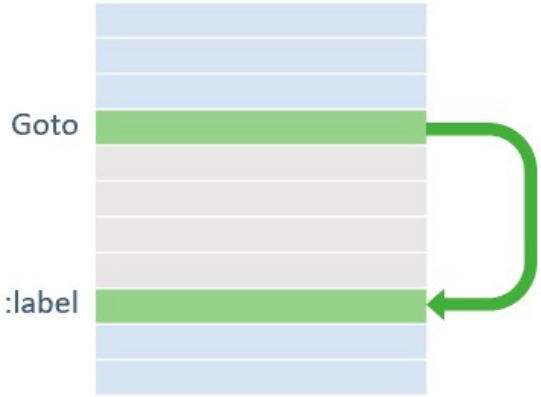
> *Es usual programar Sistemas Embebidos en C y utilizar bibliotecas de programas/macros tanto en Assembly, como en C++ (por sus ventajas)*
>
> *"Nothing better than C", Linus Torvalds*
> *https://www.youtube.com/watch?v=CYvJPra7Ebk*

> *Un estándar de codificación en C, es un conjunto de reglas para el código fuente que adopta un equipo de programadores que trabajan juntos en un proyecto, diseño de un Sistemas Embebidos*
>
> *Embedded C Coding Standard by Michael Barr*
>
> *https://barrgroup.com/embedded-systems/books/embedded-c-coding-standard*

Diseño/Desarrollo/Depuración de Software

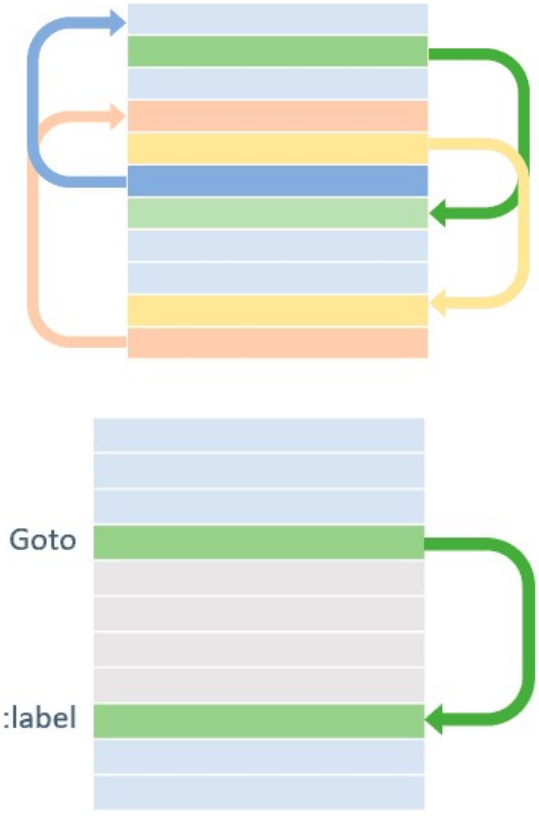


- Código Spaghetti, término de uso generalizado para código no estructurado y difícil de leer
  - Tal código en cualquier código-base grande puede crear sus problemas, si no se resuelven a tiempo. Puede provocar gran desperdicio de recursos importantes, como tiempo y energía, encontrar errores/corregirlos ya que el código no tiene estructura

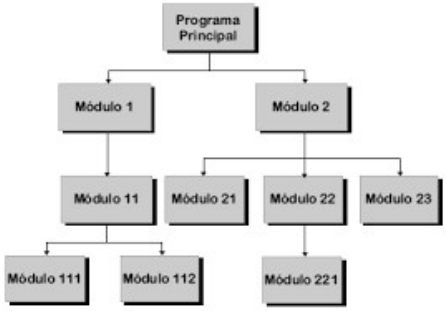


Diseño/Desarrollo/Depuración de Software

- Mientras que, en el enfoque de Programación Estructurada, se puede definir como un enfoque de programación en el que el programa se crea como una estructura única
  - Significa que el código ejecutará instrucción por instrucción una tras otra. No admite la posibilidad de saltar de una instrucción a otra con la ayuda de cualquier declaración como GOTO, etc. Las instrucciones se ejecutarán de forma serializada y estructurada
  - Ventajas: Más fácil de leer/entender/usar/desarrollar/depurar/mantener. Basado en problemas en lugar de estar basado en máquinas. Requiere menos esfuerzo y tiempo. Mayormente, independiente de la máquina



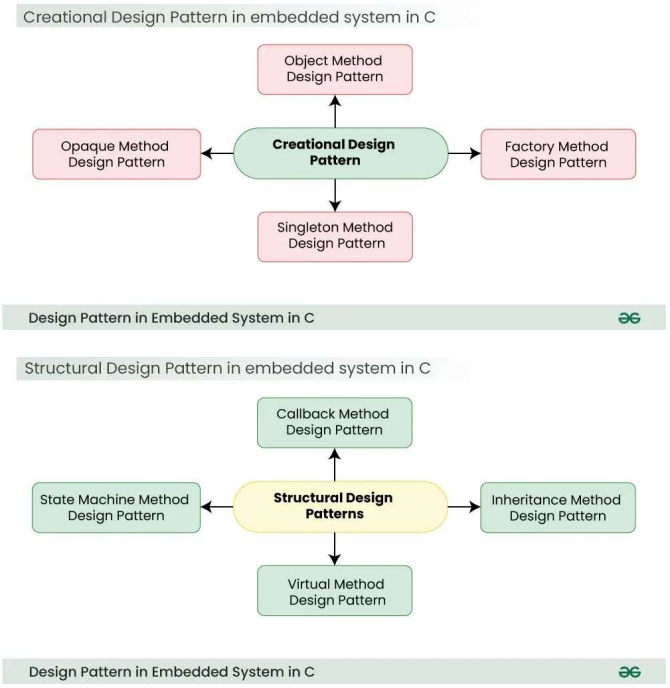
Diseño/Desarrollo/Depuración de Software



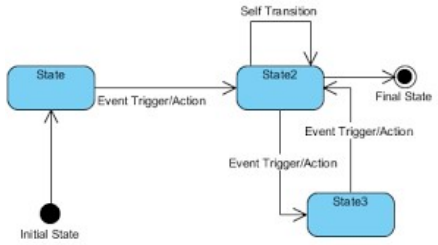
- La programación Modular es el proceso de subdividir un programa en subprogramas separados. Un módulo es un componente de software independiente. A menudo se puede utilizar en una variedad de aplicaciones y funciones con otros componentes del sistema
  - Se deben decidir las limitaciones de todos y cada uno de los módulos. De qué manera se debe dividir un programa en diferentes módulos. Comunicación entre diferentes módulos del código para la correcta ejecución de todo el programa
  - Facilidad de uso: este enfoque permite la simplicidad, podemos acceder a él en forma de módulos. Esto facilita la depuración del código y es propenso a menos errores
  - Reusabilidad: permite al usuario reutilizar la funcionalidad con una interfaz diferente sin tener que volver a escribir todo el programa
  - Facilidad de mantenimiento: ayuda a reducir las colisiones al momento de trabajar en módulos (equipo que trabajar en una aplicación grande)

Diseño/Desarrollo/Depuración de Software

- Patrones de Diseño de Software para Sistemas Embebidos en C
  - Se definen como una solución general reusable, a problemas comunes que ocurren durante el diseño y desarrollo de software embebido
  - Proporcionan una plantilla para resolver ciertos tipos de problemas y ayudan a los desarrolladores a estructurar el código de una manera que lo haga más modular, mantenible y escalable
  - Se suelen utilizar patrones de diseño creacionales, estructurales y algunos otros …

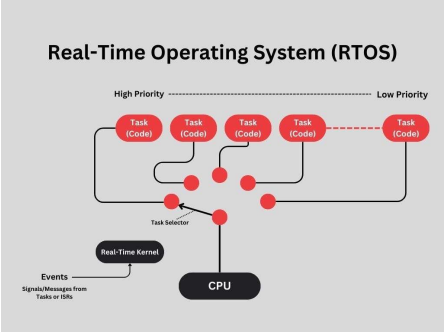


Diseño/Desarrollo/Depuración de Software



- El Unified Modeling Language (UML) es un lenguaje de modelado de propósito general
  - Su objetivo es definir una forma estándar de visualizar la forma en que se ha diseñado un sistema (similar a los planos utilizados en otros campos de la ingeniería)
  - No es un lenguaje de programación, es más bien un lenguaje visual
- State Machine Diagrams (Máquinas de Estado) | Unified Modeling Language (UML), se usan para representar la condición del sistema (o parte del mismo) en instantes de tiempo finito
  - Diagrama que representa el comportamiento mediante transiciones de estados finitos
  - Se conoce como State-Chart Diagrams (Diagrama de Estados)
  - De manera simple, se utiliza un diagrama de estado para modelar el comportamiento dinámico de una clase en respuesta al tiempo y a estímulos externos cambiantes. Podemos decir que todas y cada una de las clases tienen un estado, pero no modelamos todas las clases utilizando diagramas de estado

Diseño/Desarrollo/Depuración de Software

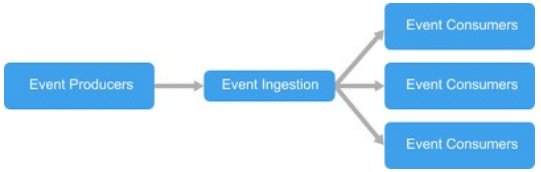


- Al desarrollar Sistemas Embebidos capaces de funcionar en Tiempo Real, una de las primeras y más importantes preguntas es si las aplicaciones deben ejecutarse bajo un Sistema Operativo de Tiempo Real (RTOS) o si se debe desarrollar una solución Bare Metal
- En la programación Bare Metal, la aplicación se escribe accediendo directamente al hardware sin utilizar una interfaz de programación externa (sin Sistema Operativo)
  - La aplicación accede directamente a los registros de hardware del microcontrolador
  - En un bucle/lazo sin fin, que ejecuta tareas hasta completar (RTC), con un tiempo de ejecución fijo
  - Esta ejecución secuencial sólo se desvía cuando ocurre un evento de interrupción
  - Este enfoque de desarrollo Bare Metal para Sistemas Embebidos se conoce como Super-Loop (polling & Interrupts), con modelado de tareas (diagramas de estado)

Diseño/Desarrollo/Depuración de Software

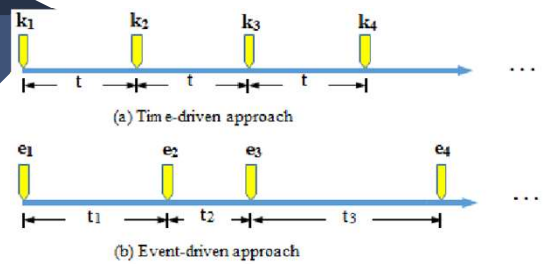
- En la programación con RTOS se utiliza un Núcleo de Sistema Operativo (Kernel) con un Programador/Planificador (Scheduler) y Controladores de Dispositivo (Device Drivers) entre el hardware y el código de la aplicación
  - Posibilita, tanto multitarea (multitasking) como multihilos (multithreading) y multiprocesamiento (multiprocessing)
  - En lugar de pocas tareas claramente definidas y definibles en términos de recursos, se pueden ejecutar muchas tareas (completa o descomponerla en hilos - threads) en uno o varios núcleos (core) de CPU (donde las tareas individuales pueden priorizarse cómodamente y agruparse según el rendimiento de los núcleos de CPU disponibles)
  - Los Sistemas Operativos de Tiempo Real con sus planificadores permiten ejecutar procesos en Concurrencia (Concurrent)/Paralelo (Parallel), de forma flexible y priorizada, asumiendo la responsabilidad de la funcionalidad del sistema. Tanto de forma Cooperativa (Cooperative) como Apropiativa (Preemptive)

Diseño/Desarrollo/Depuración de Software



- Como solución a Sistemas Embebidos orientados a Control, tenemos los Sistemas Disparados/Activados por Tiempo o por Evento (Time- and Event-Triggered Systems)
  - Un disparador/activador (trigger) es un suceso (evento/sincronismo) que provoca el inicio de alguna acción en el Sistema de Control
  - La acción puede ser la ejecución de una tarea leyendo una variable y calculando un nuevo valor de una variable de corrección, o el envío de un mensaje informando valores actuales de variables como presión o temperatura
- En el Control activado/disparado por Evento, una acción se inicia sólo si ocurre un evento significativo (condición de programa/encuesta/interrupción)
  - Por ejemplo, un sensor enviaría un mensaje solo si la temperatura ha cambiado más de 3 °C desde que se envió el último mensaje

Diseño/Desarrollo/Depuración de Software



- En el Control disparado/activado por Tiempo, todas las acciones se inician periódicamente mediante un sincronismo aportado por una Base de Tiempo (Reloj de Tiempo Real)
  - El sensor de nuestro ejemplo enviaría un mensaje en cada ciclo de reloj incluso si la temperatura permanece constante
- El control T-TS genera más comunicación y procesamiento que el control E-TS
- En un sistema distribuido con control E-TS, un componente que no recibe un mensaje de algún otro componente no sabe si no ha habido ningún evento significativo durante mucho tiempo o si el otro componente ha fallado o se ha desconectado
- En un sistema distribuido con control T-TS, los mensajes adicionales enviados actúan como latidos (heartbeats), que le dicen al receptor que el componente emisor y la conexión a ese componente todavía están activos

Diseño/Desarrollo/Depuración de Software

- Los sistemas distribuidos de tiempo real T-TS generalmente sincronizan los relojes de todos los nodos para formar un reloj global, estableciendo una base de tiempo global
  - Los eventos pueden tener una marca de tiempo con el tick del reloj de su nodo
  - La marca de tiempo permite definir un orden entre observaciones realizadas en nodos arbitrarios
  - La granularidad del reloj global debe ser lo suficientemente fina para que los eventos posteriores obtengan marcas de tiempo diferentes

Referencias

- C Programming Language Tutorial - https://www.geeksforgeeks.org/
- "Nothing better than C", Linus Torvalds - https://www.youtube.com/watch?v=CYvJPra7Ebk
- Embedded C Coding Standard by Michael Barr - www.barrgroup.com/embedded-systems/books/embedded-c-coding-standard
- Development on Bare Metal vs. RTOS - https://www.sysgo.com/professional-articles/bare-metal-vs-rtos
- Difference between Multiprogramming, multitasking, multithreading and multiprocessing - https://www.geeksforgeeks.org/
- Difference between Preemptive and Cooperative Multitasking - https://www.geeksforgeeks.org/
- Event-Triggered Systems (ETS) and Time-Triggered (TTS) - https://ebrary.net/51334/computer_science/time_event_triggered_systems
- Design Patterns for Embedded Systems in C - https://www.geeksforgeeks.org/

Referencias

- Compiling a C Program: Behind the Scenes - https://www.geeksforgeeks.org/
- Loader in C/C++ - https://www.geeksforgeeks.org/
- Spaghetti Code - https://www.geeksforgeeks.org/
- Structured Programming Approach with Advantages and Disadvantages - https://www.geeksforgeeks.org/
- Introduction of Programming Paradigms - https://www.geeksforgeeks.org/
- Differences between Procedural and Object Oriented Programming - https://www.geeksforgeeks.org/
- Modular Approach in Programming - https://www.geeksforgeeks.org/
- Unified Modeling Language (UML) Diagrams | An Introduction - https://www.geeksforgeeks.org/
- State Machine Diagrams | Unified Modeling Language (UML) -https://www.geeksforgeeks.org/unified-modeling-language-uml-state-diagrams/

Manos a la obra con el . . .

. . . Proyecto Intermedio

. . . un enfoque centrado en la **práctica propia de la carrera** más que en el desarrollo teórico disciplinar, con eje en la **participación de las y los estudiantes**


Las y los estudiantes preguntarán:
¿en qué lío nos metimos?

¡Muchas gracias!

¿Preguntas?

. . .

Consultas a: jcruz@fi.uba.ar