Taller de Sistemas Embebidos Herramientas

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

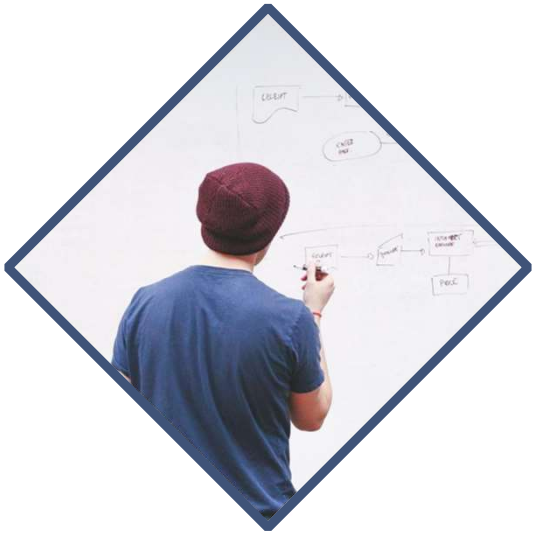
Estructura Curricular de la Carrera

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 herramientas se necesitan?

- Para el Diseño/Desarrollo/Depuración de Hardware se recurre a herramientas con prestaciones de control y análisis, en evolución permanente, sirven para sistematizar o mejoran la velocidad y precisión de los procesos/subprocesos requeridos, se agrupan en:
  - Herramientas Informáticas:
    - CAE (Computer-Aided Engineering)
    - CAD (Computer-Aided Design)
    - CAM (Computer-Aided Manufacturing)

¿Qué herramientas se necesitan?

- Módulos Electrónicos:
  - Placas de desarrollo (Development Board)
  - Placas de aplicación (Application Board)
- Instrumentos de Laboratorio:
  - Generadores de señales
  - Medidores de señales/magnitudes
  - Analizadores de señales

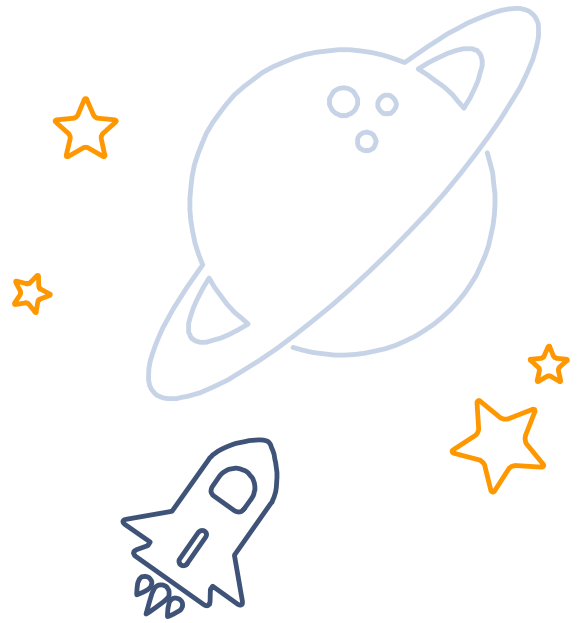
¿Qué herramientas se necesitan?

- Para el Diseño/Desarrollo/Depuración de Software se recurre a herramientas con prestaciones de control y análisis, en evolución permanente, sirven para sistematizar o mejoran la velocidad y precisión de los procesos/subprocesos requeridos, se agrupan en:
  - CASE (Computer-Aided Software Engineering)
    - Diagraming Tools
    - Modeling Tools
    - Analysis Tools
    - Central Repository

¿Qué herramientas se necesitan?

- Documentation Generators
- Code Generators
- Tools for Requirement Management
- Tools for Analysis and Design
- Tools for Database Management
- Tools for Documentation

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)

Diseño/Desarrollo/Depuración de Hardware

- En cuanto a herramientas recurriremos a:
  - Herramientas Informáticas CAE (Computer-Aided Engineering)
    - Captura esquemática, simulación, depuración, validación y verificación de circuitos electrónicos (analógicos/digitales)
    - Usadas en asignaturas previas (Introducción a la Ingeniería Electrónica, Sistemas Digitales, Análisis de Circuitos, Señales y Sistemas, etc.)
  - En cuanto a Módulos Electrónicos:
    - Placa de desarrollo: STM32 32-bits ARM Cortex MCUs, NUCLEO board
    - Dip-switchs, Buttons, Keyboards, Leds, LCD Displays, etc.
    - Instrumentos de Laboratorio

Diseño/Desarrollo/Depuración de Software

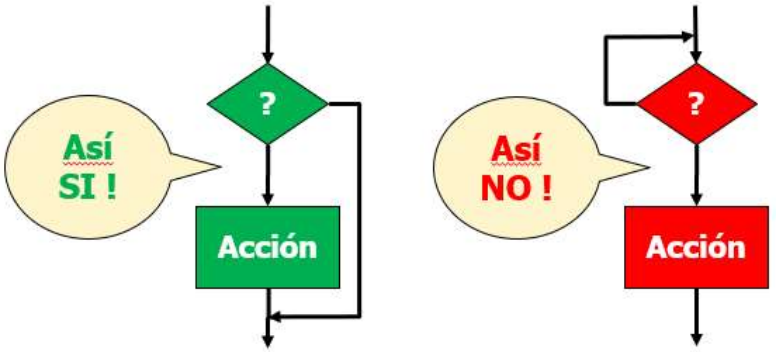
- En cuanto a herramientas recurriremos a:
  - Integrated Development Environment (IDE): STM32CubeIDE - Integrated Development Environment for STM32
  - Version Control System (VCS): Git Bash/GUI
  - Repository: GitHub
  - Software Modeling Tool: Itemis CREATE

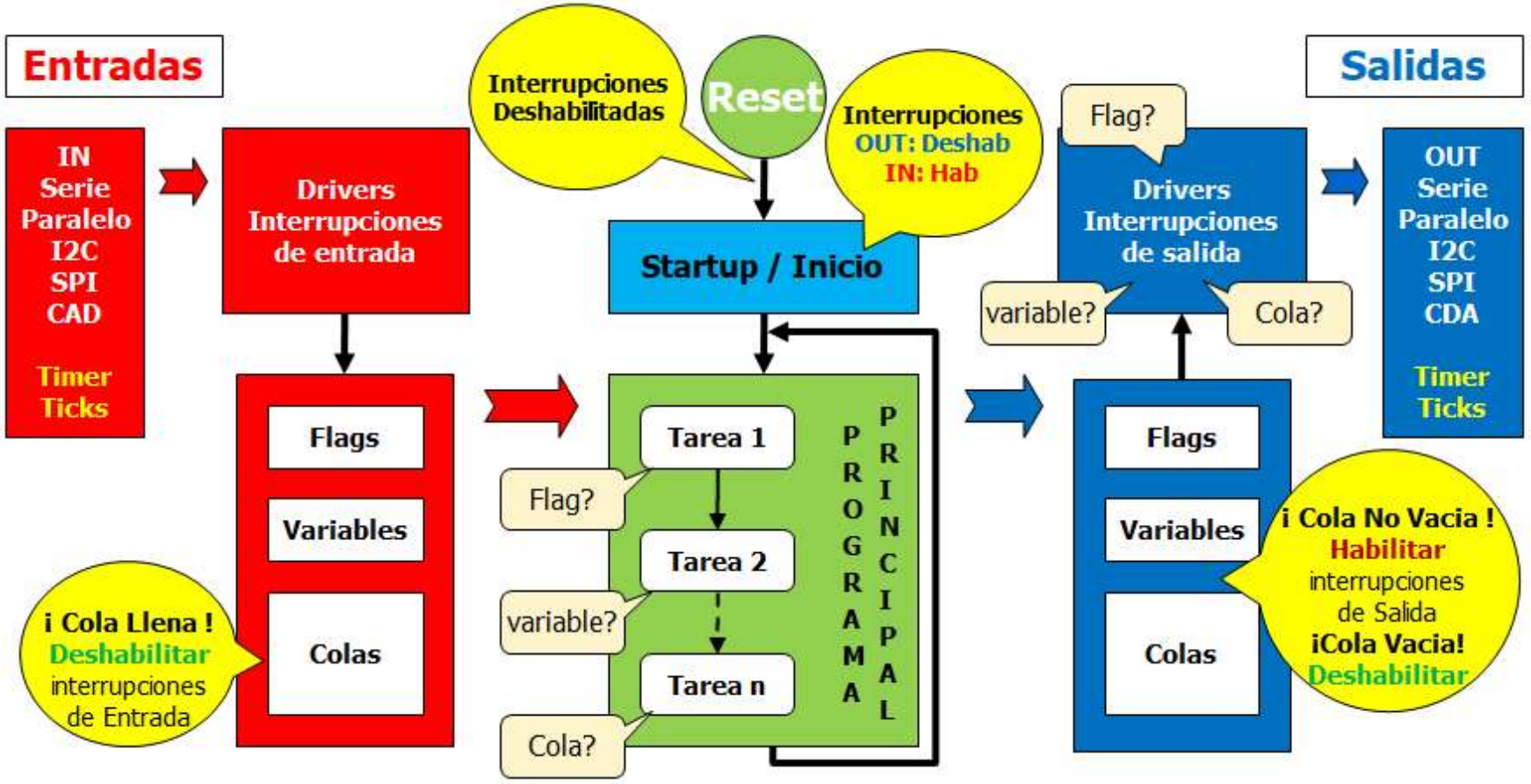
Diseño/Desarrollo/Depuración de Software

- Codificaremos en C
  - "Nothing better than C", Linus Torvalds
- Aplicando el estándar de codificación
  - Embebidos Embedded C Coding Standard by Michael Barr
- Codificaremos soluciones del tipo:
  - Estructurada, Modular
    - Escrutar, Procesar y Actuar
  - Bare Metal (sin Sistema Operativo, con Patrones de Diseño de Software)
    - Super-Loop (polling & Interrupts), con modelado de tareas (diagramas de estado)
    - Event-Triggered Systems

Diseño/Desarrollo/Depuración de Software

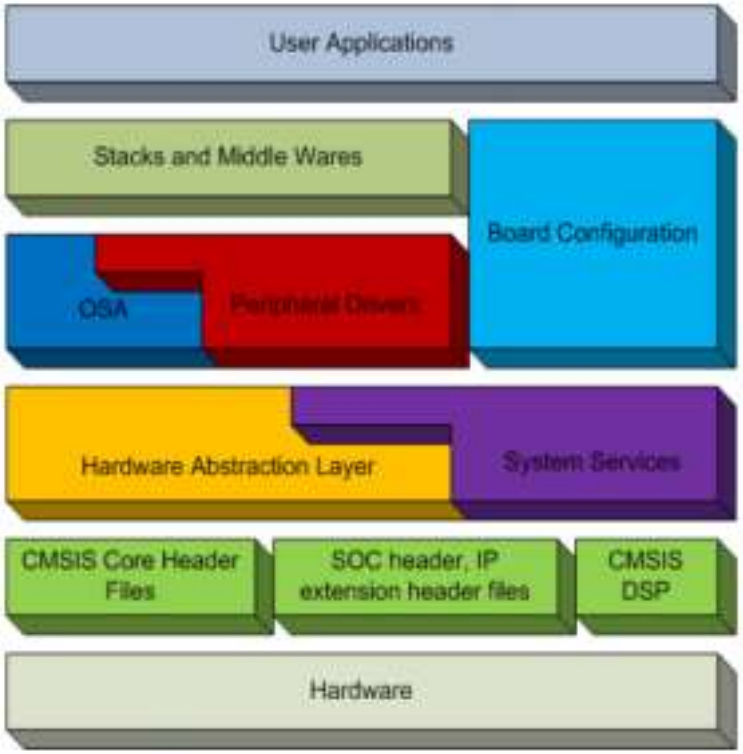
- Estructura de la Aplicación (escrutar, procesar, actuar):
  - Startup (Inicio) => Inicializaciones básicas del MCU
  - Main (Programa Principal) => Iteración perpetua de Tareas (algoritmos)
  - Drivers (Manejadores) de Entrada / Salida => Interacción con el mundo exterior (atención de eventos y sincronismos, con Callbacks)
  - Los módulos se comunican / sincronizan mediante:
    - Flags o Variables o Colas
  - Garantizar comportamiento comunitario (que ningún módulo se apropie de la CPU), el código bloqueante es inaceptable




Entradas
IN
Serie
Paralelo
I2C
SPI
CAD
Timer
Ticks
Drivers
Interrupciones
de entrada
Flags
Variables
Colas
¡ Cola Llena !
Deshabilitar
interrupciones
de Entrada
Interrupciones
Deshabilitadas
Reset
Interrupciones
OUT: Deshab
IN: Hab
Startup / Inicio
Tarea 1
Tarea 2
Tarea n
PROGRAMA PRINCIPAL
Flag?
variable?
Cola?
Flag?
Drivers
Interrupciones
de salida
variable?
Cola?
Flags
Variables
Colas
¡ Cola No Vacia !
Habilitar
interrupciones
de Salida
¡Cola Vacia!
Deshabilitar
Salidas
OUT
Serie
Paralelo
I2C
SPI
CDA
Timer
Ticks

Diseño/Desarrollo/Depuración de Software

- Cortex Microcontroller Software Interface Standard (CMSIS)
- System on a Chip (SoC)
- Hardware Abstraction Layer (HAL)
- Operating System Abstraction (OSA) capa p/sistema operativo (MQX, FreeRTOS, uCos, etc.)
- Stacks and Middleware incluye USB stack, TCP/IP stack, Audio, Graphics, Boot Loader
- Board Support Packages (BSP) => Board Configuration



Diseño/Desarrollo/Depuración de Software

- Regla de pulgar del principiante puesto a programar en assembly, asegurar que:
  - $T_{cpu\text{-}pp} \leq 2\ T_{fmax}$ Entrada detectar/Salida generar (1)
    $\leq 1000\ T_{instrucción}$ (2)
    $\leq Tick_{min}$ (2)
    $\leq 60\sim70\% \sum (T_{cpu\text{-}driver/tarea})$ (3)
  - $T_{cpu\text{-}driver/tarea} \leq T_{cpu\text{-}pp} / 10$ (2)

(1) Nyquist-Shannon (Teorema Muestreo)

(2) JMC (Experiencia de desarrollador)

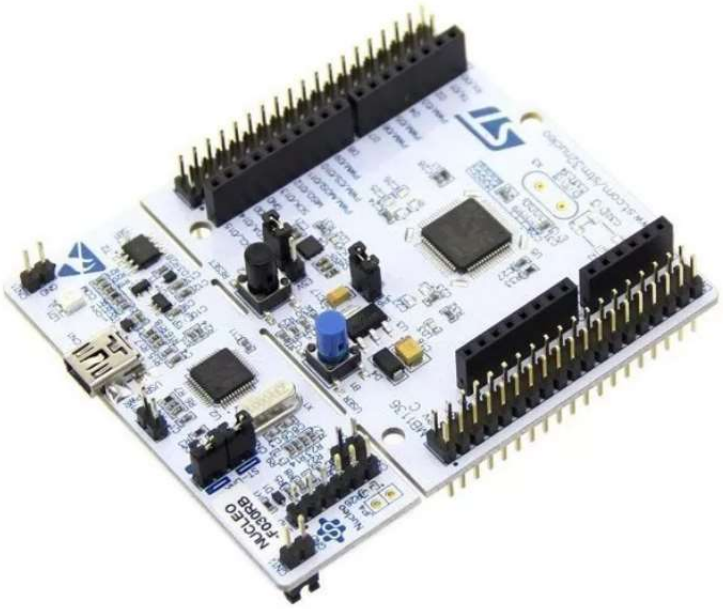
(3) Mamá de JMC (Experiencia de modista)

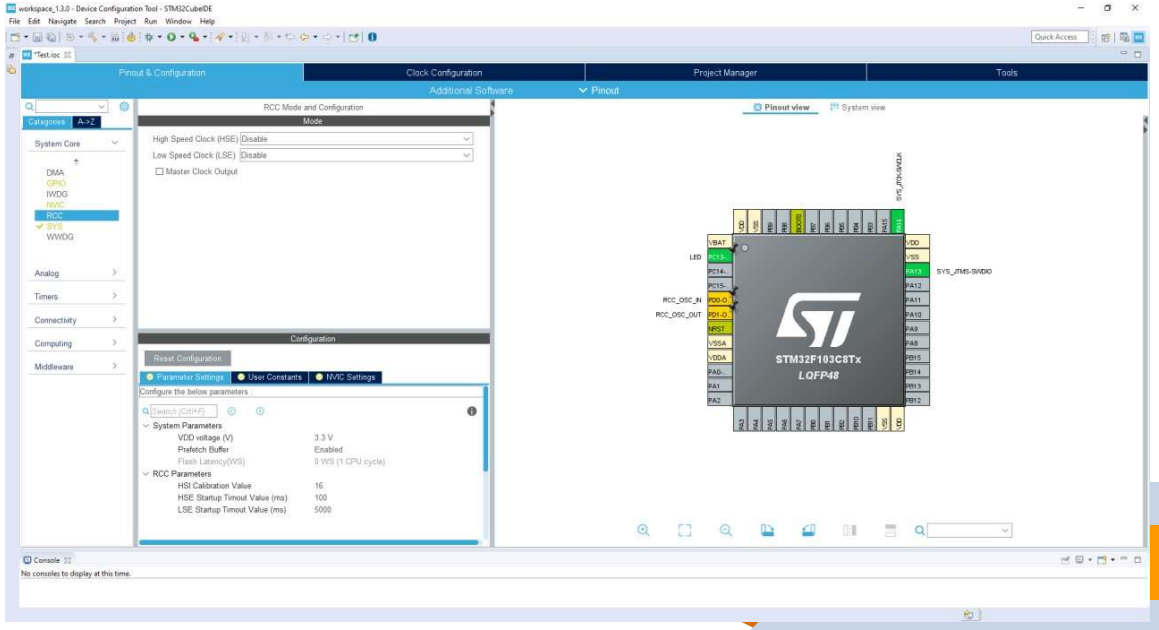
Diseño/Desarrollo/Depuración de Software

- MCS-51 ejecuta ~ 1MIPS c/F_{clock} = 1MHz o más (CPU de la familia Intel original o derivados) => $T_{instrucción} \approx 1\mu S$
  - $T_{cpu\text{-}pp} \leq 1mS$ / $Tick_{min} \geq 1mS$ / $T_{cpu\text{-}driver/tarea} \leq 100\mu S$
  - Bus de Datos , 1 Acumulador y Operaciones Básicas de 8 bits
  - 1 Puntero a Memoria (favor de comparar con Atmega128)
- ARM CORTEX M3 ejecuta ~ 120MIPS c/F_{clock} = 100MHz o más (CPU de la familia NXP LPC17xx) => $T_{instrucción} \approx 10nS$
  - Bus de Datos, 16 Acumuladores/Punteros y Operaciones Complejas de 32 bits
  - Manteniendo constantes los T enumerado puedo ejecutar 100 veces mayor cantidad de más poderosas instrucciones nos permite resolver problemas más complejos con soluciones de software más abstractas, portables y elaboradas

Diseño/Desarrollo/Depuración

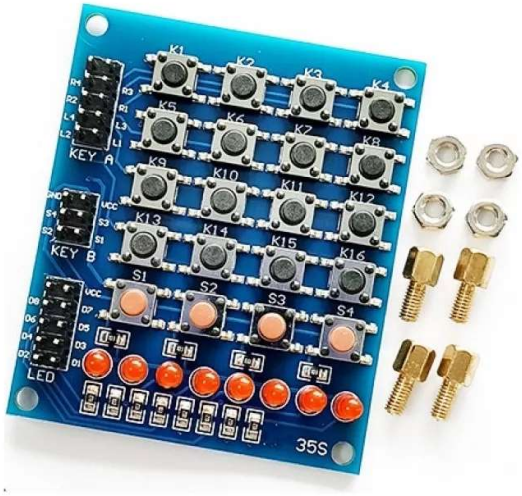
- STM32 32-bits ARM Cortex MCUs, NUCLEO board (STMicroelectronics)
  - NUCLEO-F103RB, placa de desarrollo con STM103RB MCU (Cortex M3)
  - STM32IDECube, herramienta gratuita basada en el IDE Eclipse y las herramientas GCC

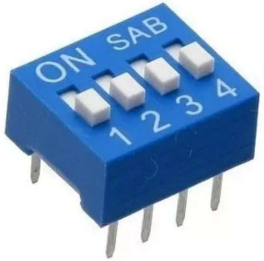




Diseño/Desarrollo/Depuración

- Periféricos & cables, para usar con la placa de desarrollo STM103RB MCU (Cortex M3)





Diseño/Desarrollo/Depuración

- Placas de desarrollo (Development Board), permiten probar nuevas ideas y crear prototipos rápidamente con el MCU para el que fue concebida (usualmente provista por el fabricante del MCU, en tal caso se la llama nativa)
  - Si pertenece a una familia de placas, comparten los mismos conectores (conectores Arduino Uno rev3, Manufacturer form, Arduino Nano, etc.), lo que hace fácil expandirla con hardware complementario para aplicaciones específicas (add-on boards)
  - Suelen integrar un depurador (debugger)/programador (programmer), eliminando la necesidad de recurrir a hardware adicional para su uso
  - Cuentan con IDE provista por el fabricante del MCU (nativo), que viene con una variedad de ejemplos, que funcionan perfectamente en una amplia gama de entornos de desarrollo

Diseño/Desarrollo/Depuración

- Un IDE (Integrated Development Environment) es un software que combina herramientas de desarrollo de uso común en una aplicación GUI (Graphical User Interface) compacta
  - Es una combinación de herramientas como un editor de código, un compilador de código y un depurador de código con una terminal (consola) integrada
  - Al integrar funciones como edición, creación, prueba y contenedor de software en una herramienta fácil de usar, los IDE ayudan a aumentar la productividad de los desarrolladores
  - Los programadores y desarrolladores de software suelen utilizar los IDE para facilitar su proceso de programación
  - Los IDE ofrecen funciones adicionales a la edición ordinaria (herramientas de desarrollo de uso frecuente), en una interfaz simple, uno puede crear aplicaciones sin necesidad de configurar e integrar manualmente el entorno de desarrollo

Diseño/Desarrollo/Depuración

- Los IDE proporcionan una amplia variedad de características, normalmente consisten en:
  - Editor: ayuda a escribir código de software, resaltando la sintaxis con indicadores visuales, autocompletado del lenguaje y comprobando errores mientras escribe
  - Compiler: traduce el código legible por humanos en código específico de la máquina, ejecutable en diferentes sistemas operativos como Linux, Windows o Mac OS. La mayoría de los IDE vienen con compiladores integrados para el lenguaje que soportan
  - Debugger: ayuda a probar y depurar aplicaciones y señalar gráficamente la ubicación de los fallos o errores, si los hubiera
  - Build-in Terminal: La terminal (consola) es una interfaz basada en texto que se puede utilizar para interactuar con el sistema operativo de la máquina. Los desarrolladores pueden ejecutar scripts o comandos dentro de un IDE con una terminal/consola integrada

Diseño/Desarrollo/Depuración

- Version Control: ayuda a aportar claridad al desarrollo del software. Algunos IDE también admiten herramientas de control de versiones como Git, a través de las cuales un usuario puede rastrear y administrar los cambios en el código del software
- Code snippets: los IDE admiten fragmentos de código que generalmente se utilizan para realizar una tarea y también pueden reducir en gran medida el trabajo redundante
- Extensions & Plugins: Las extensiones y complementos se utilizan para ampliar la funcionalidad de los IDE con respecto a lenguajes de programación específicos
- Code navegation: los IDE vienen con herramientas como code folding, navegación de clases y métodos, y herramientas de refactoring que simplifican la revisión y el análisis del código

Diseño/Desarrollo/Depuración

- ¿Por qué los desarrolladores utilizan IDE?
  - Al proporcionar un entorno único y unificado para gestionar todos los aspectos del proceso de desarrollo, los IDE pueden ayudar a mejorar la productividad, la calidad del código y la experiencia de desarrollo general del desarrollador (personalización)
- Los IDE vienen en varias formas, unos diseñados para funcionar con un lenguaje específico, otros dirigidos a una plataforma particular, como dispositivos móviles
  - Como por ejemplo IDE de escritorio, en la nube, para desarrollo de aplicaciones móviles, específicos para trabajar con bases de datos, etc.
- La elección de un IDE suele estar determinada por el tipo de proyecto en el que está trabajando, así como por diversos requisitos de entorno

Diseño/Desarrollo/Depuración

- Git es un sistema de gestión de código fuente y control de versiones distribuido ampliamente utilizado
  - Realiza un seguimiento eficaz de los cambios en el código fuente, lo que permite realizar ramificaciones (branching), fusiones (merging) y versiones (versioning) sin esfuerzo
  - Utiliza un modelo descentralizado donde cada desarrollador tiene su propia copia del repositorio y trabaja en el acto en el proyecto
  - Gestiona los proyectos con repositorios y puede clonar un proyecto para operar localmente en él
  - Con staging y committing, realiza el seguimiento y control de los cambios
  - Puede extraer (pull) el código más reciente del proyecto en la copia local y enviar (push) actualizaciones locales a los proyectos principales

Diseño/Desarrollo/Depuración

- ¿Por qué utilizar Git?
  - Debes saber que alrededor del 70% de los desarrolladores de todo el mundo utilizan Git para el desarrollo
  - Algunas de las razones destacadas para usar Git son:
    - Los desarrolladores pueden trabajar juntos desde cualquier lugar
    - Los desarrolladores pueden ver el historial completo y comparar los cambios anteriores y nuevos del proyecto
    - Los desarrolladores pueden volver a versiones anteriores de un proyecto

Diseño/Desarrollo/Depuración

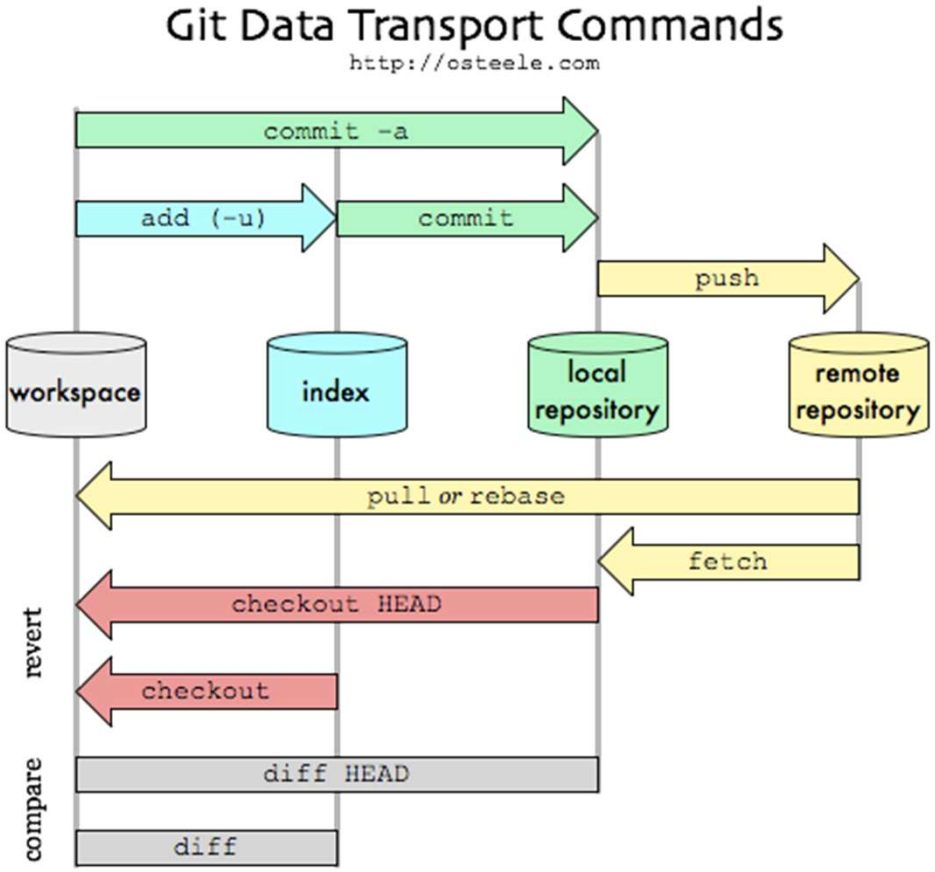
Trabajando con Git

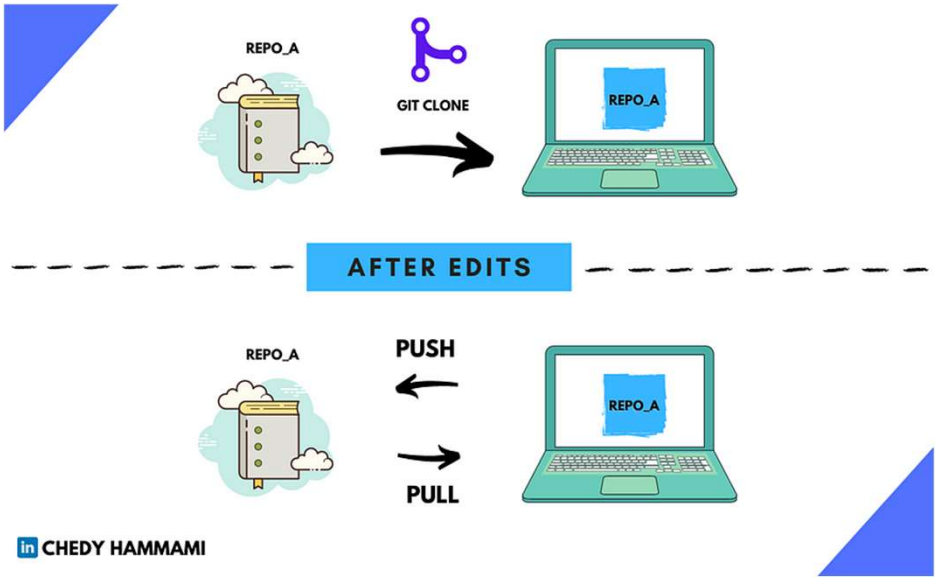
- Cuando una carpeta se inicializa con Git, se convierte en un repositorio: una ubicación especial donde Git registra todos los cambios realizados en una carpeta oculta
- En esa carpeta, cada vez que cambias (change), agregas (add) o eliminas (remove) un archivo, Git toma nota del cambio y marca el archivo como "modified"
- Puedes elegir qué archivos modificados quieres guardar preparándolos (staging)
- Considere staging como preparación de los cambios para una instantánea en particular que desea conservar
- Una vez que los cambios preparados (staged) sean de su agrado, confírmelos y Git mantendrá una copia permanente de esos archivos en su historial
- Git mantiene un registro completo de cada “commit" que realizas, permitiéndote verlos

Diseño/Desarrollo/Depuración

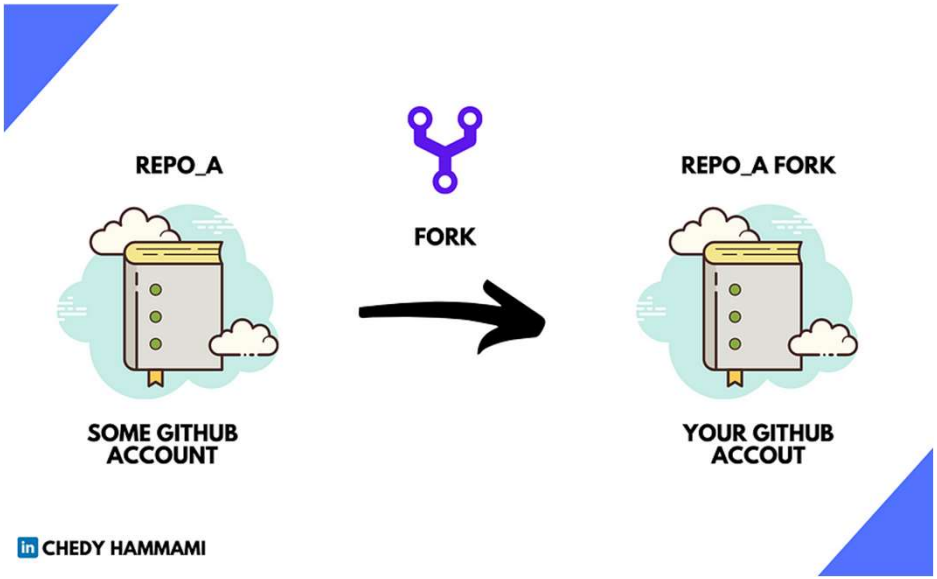
- ¿Qué es GitHub?
  - GitHub es un servicio de alojamiento (hosting) para repositorios de Git
  - Si tiene un proyecto alojado en GitHub, puede acceder y descargar ese proyecto con comandos en cualquier computadora a la que tenga acceso y realizar cambios y enviar la última versión a GitHub
  - GitHub te permite almacenar tu repositorio en su plataforma
  - También viene con GitHub, capacidad de colaborar con otros desarrolladores desde cualquier ubicación

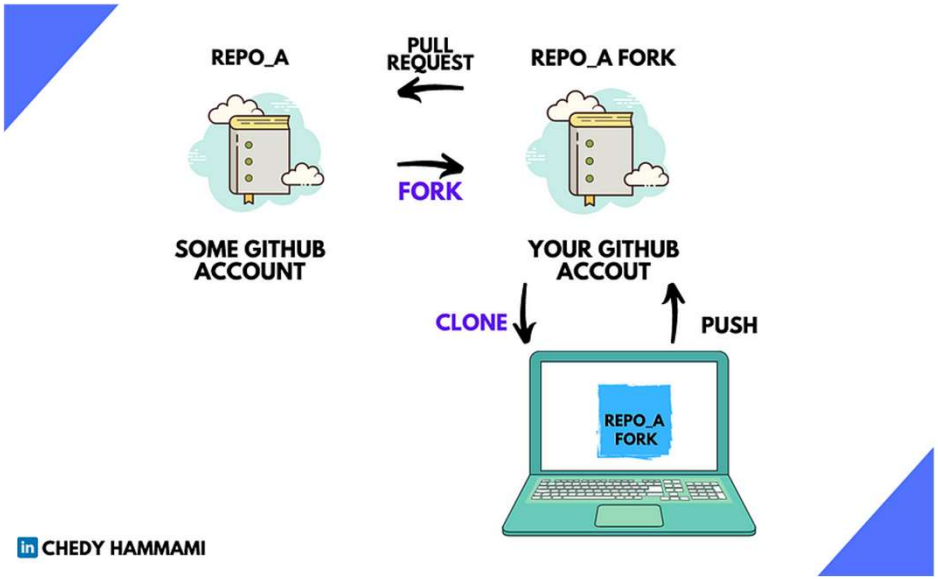
Diseño/Desarrollo/Depuración





Diseño/Desarrollo/Depuración

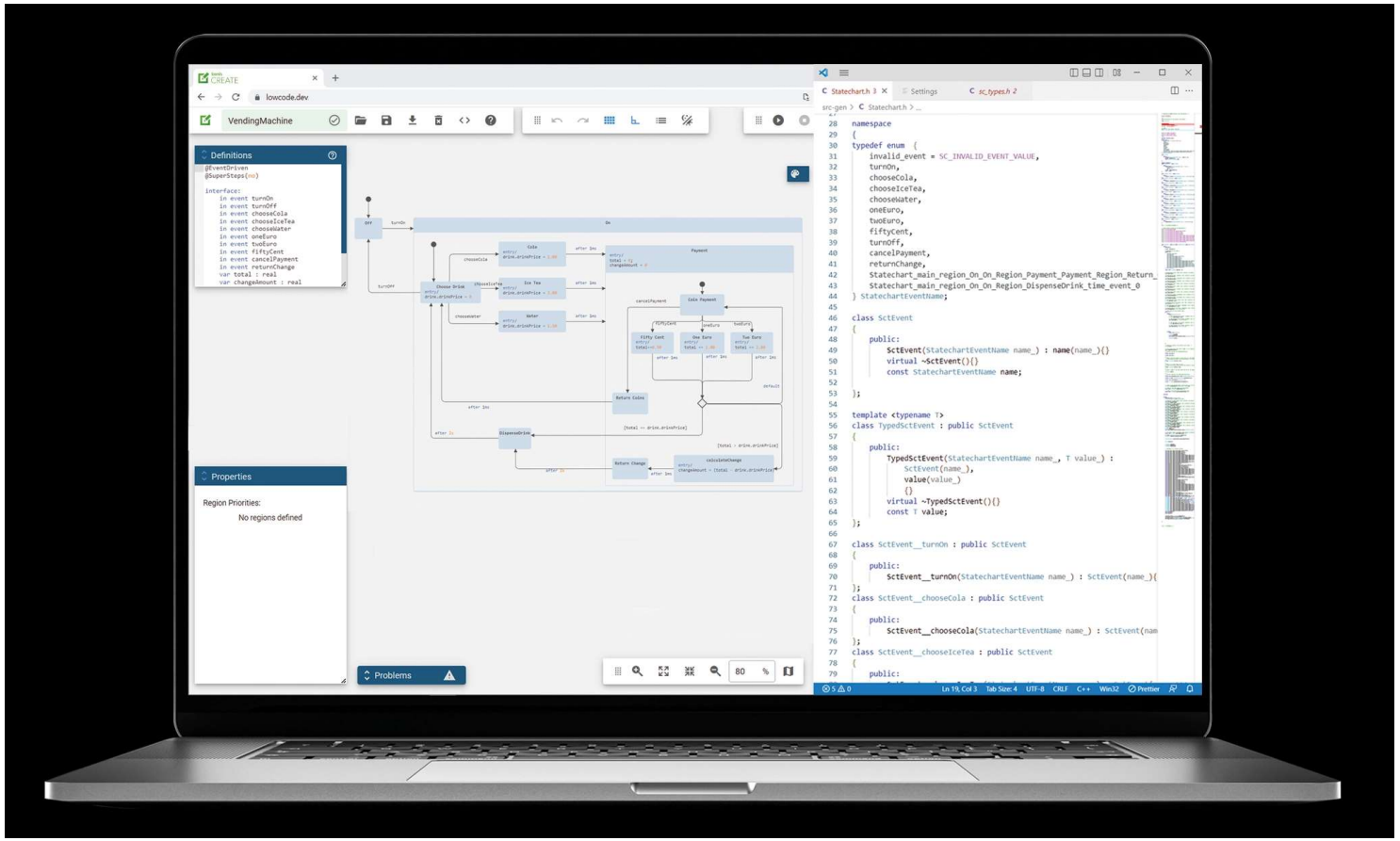




Diseño/Desarrollo/Depuración

- Itemis CREATE
  - Simplifique sus desarrollos, use Máquinas de Estado (Diagramas de Estado)
  - Acelere el desarrollo de productos, reduzca el tiempo de comercialización y los costos aumentando la eficiencia de su proceso de desarrollo de productos
  - Evite errores costosos, afronte situaciones excepcionales que de otro modo podrían pasarse por alto y reduzca la posibilidad de cometer errores
  - Lo complejo se vuelve fácil, el uso de modelos visuales hace que la generación de código sea fácil de entender para todas las partes involucradas, e incluso para quienes no son desarrolladores
  - Evite la duplicación de esfuerzos, evite el desarrollo duplicado y reutilice modelos de diagramas de estado gráficos (Diagramas de Estado) en desarrollos futuros

Diseño/Desarrollo/Depuración



Referencias

- CAD vs CAE vs CAM (Electronic manufacturing industry) - https://www.pcbaaa.com/cad-vs-cae-vs-cam/
- Computer Aided Software Engineering (CASE) - https://www.geeksforgeeks.org/
- What is an Arm processor? - https://www.techtarget.com/whatis/definition/ARM-processor
- STM32 32-bit Arm Cortex MCUs - https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html
- STM32 Nucleo Boards - https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html
- STM32 Nucleo expansion boards - https://www.st.com/en/evaluation-tools/stm32-nucleo-expansion-boards.html
- STM32CubeIDE - Integrated Development Environment for STM32 - https://www.st.com/en/development-tools/stm32cubeide.html

Referencias

- Git - https://git-scm.com/
- Git Tutorial - https://www.geeksforgeeks.org/
- GitHub - https://github.com/
- Git Clone vs Fork in GITHUB - https://chedyhammami.medium.com/git-clone-vs-fork-in-github-610f158d61e3
- What is an IDE? - https://www.geeksforgeeks.org/
- Itemis CREATE - https://www.itemis.com/en/products/create/
- "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

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