



**Universidade Federal de Santa Catarina  
Centro Tecnológico – CTC  
Departamento de Engenharia Elétrica**



CTC UFSC

**Laboratório de Comunicações e Sistemas Embarcados - LCS  
Laboratório de Integração de Software e Hardware - LISHA  
Grupo de Sistemas Embarcados - GSE**

# **“Programação de Sistemas Embarcados”**

**Prof. Eduardo Augusto Bezerra**

**Eduardo.Bezerra@ufsc.br**

**Florianópolis, junho de 2014.**

# Cursos de Sistemas Embarcados

## 1. SOFTWARE E HARDWARE EMBARCADO

### 1.1 Introdução

1.1.1 Definição de sistemas embarcados, SoCs e MPSoCs

1.1.2 Características de aplicações embarcadas

1.1.3 Desafios no projeto de sistemas embarcados

### 1.2 Projeto de sistemas embarcados

1.2.1 Requisitos

1.2.2 Especificação

1.2.3 Projeto da arquitetura software-hardware

1.2.4 Projeto dos componentes de hardware e software

1.2.5 Integração do Sistema

### 1.3 Formalismos para o projeto de sistemas embarcados

### 1.4 Exemplos de projetos reais de sistemas embarcados

# Cursos de Sistemas Embarcados

## 2. TÉCNICAS DE PROJETO DE SISTEMAS EMBARCADOS

### 2.1 Metodologias de projeto

2.1.1 Métricas: “Time-to-market”, Custo de projeto, Qualidade

2.1.2 Fluxo de projeto de sistemas embarcados

### 2.2 Análise de requisitos

2.2.1 Diferenças entre requisitos e especificação

2.2.2 Requisitos funcionais e não funcionais

### 2.3 Análise do sistema e projeto da arquitetura software-hardware

### 2.4 Qualidade no projeto de sistemas embarcados

2.4.1 Técnicas de garantia de qualidade no projeto de sistemas embarcados

2.4.2 Revisões de projeto

### 2.5 Exemplos de projetos reais

# Cursos de Sistemas Embarcados

## 3. MODELOS DE COMPUTAÇÃO

### 3.1 Diferenciação entre modelos de computação (MOC) e sistemas

3.1.1 Separação entre computação e comunicação

3.1.2 Separação entre função e arquitetura

### 3.2 Classificação de MOCs

3.2.1 MOCs para sistemas embarcados

3.2.2 Modelos Síncronos e Assíncronos

3.2.3 Modelos Temporizados e não temporizados

3.2.4 Meta modelos

3.2.5 Interfaces entre MOCs pertencentes ao mesmo domínio e a domínios diferentes

3.2.6 Integração de diferentes MOCs em um sistema embarcado

### 3.3 Relação entre modelos de computação e programação em linguagens de alto nível

### 3.4 Exemplos práticos de projeto

# Cursos de Sistemas Embarcados

## 4. SISTEMAS OPERACIONAIS EMBARCADOS

4.1 Características de sistemas operacionais embarcados

4.2 Sistemas Operacionais Embarcados

4.2.1 Escalonamento e Estados de um Processo

4.2.2 Estrutura de um Sistema Operacional Embarcado

4.2.3 Restrições Temporais em Processos

4.2.4 Comunicação Inter-processos

4.2.5 Outras Funções do Sistema Operacional

4.3 Políticas de Escalonamento em Sistemas Operacionais Embarcados

4.3.1 Escalonamento RM

4.3.2 Escalonamento EDF

4.3.3 Comparação entre RM e EDF

4.4 Mecanismos de Comunicação Inter-processos para sistemas embarcados

4.5 Customização de sistemas operacionais embarcados

4.6 Avaliação de desempenho para sistemas operacionais embarcados

4.7 Exemplos práticos de projetos de sistemas embarcados que utilizam um sistema operacional

# Cursos de Sistemas Embarcados

## 5. INFRA-ESTRUTURA DE COMUNICAÇÃO

5.1 Introdução

5.2 Arquitetura de Sistemas Embarcados Distribuídos

5.3 Infra-estrutura de Comunicação para Sistemas Embarcados

5.4 Protocolos de comunicação para sistemas embarcados

5.5 Projeto de Infra-estrutura de Comunicação

5.5.1 Análise da Comunicação

5.5.2 Avaliação de Desempenho do Sistema

5.4 Exemplos práticos de projetos

# Programação de Sistemas Embarcados

## PPGEEL

- **Objetivos:**
  - Dar continuidade aos estudos de programação de sistemas computacionais embarcados.
  - Compreender os conceitos fundamentais do paradigma de programação orientada a objetos.
  - Desenvolver a capacidade de análise de programas em C++ de complexidade média.
  - Entender o funcionamento básico de sistemas operacionais para acesso a periféricos.
  - Desenvolver programas em C++ para sistemas embarcados baseados em microprocessadores embarcados, e também sistemas do tipo System-on-a-chip (SoC).

# Programação de Sistemas Embarcados

## PPGEEL

- **Motivação:**
  - Conhecer a área de “programação de sistemas embarcados”.
  - Entender as tendências e problemas relacionados ao projeto de software para sistemas embarcados complexos.
  - Desenvolver habilidades de pesquisa na literatura e síntese de trabalhos científicos.
  - Desenvolver habilidades de apresentação de trabalhos científicos.



# Sistemas Embarcados



# Sistemas embarcados

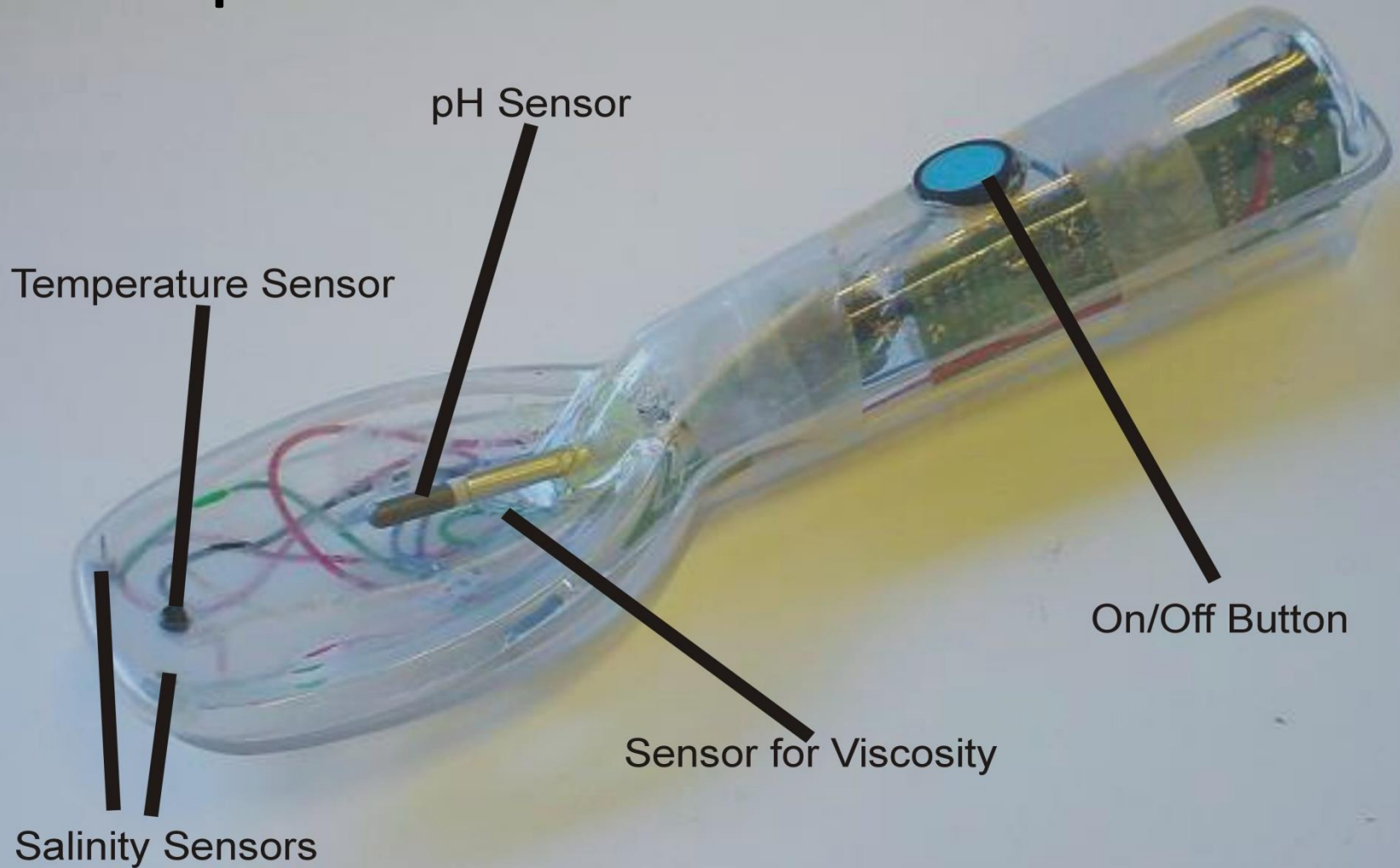
---

- Embarcados em:
  - Sistemas automotivos
  - Aviônicos
  - Brinquedos
  - Dispositivos médicos
  - Eletrodomésticos
- Bilhões de unidades



# Sistemas embarcados

## Espátula eletrônica





# Sistemas embarcados

---

## Espátula eletrônica

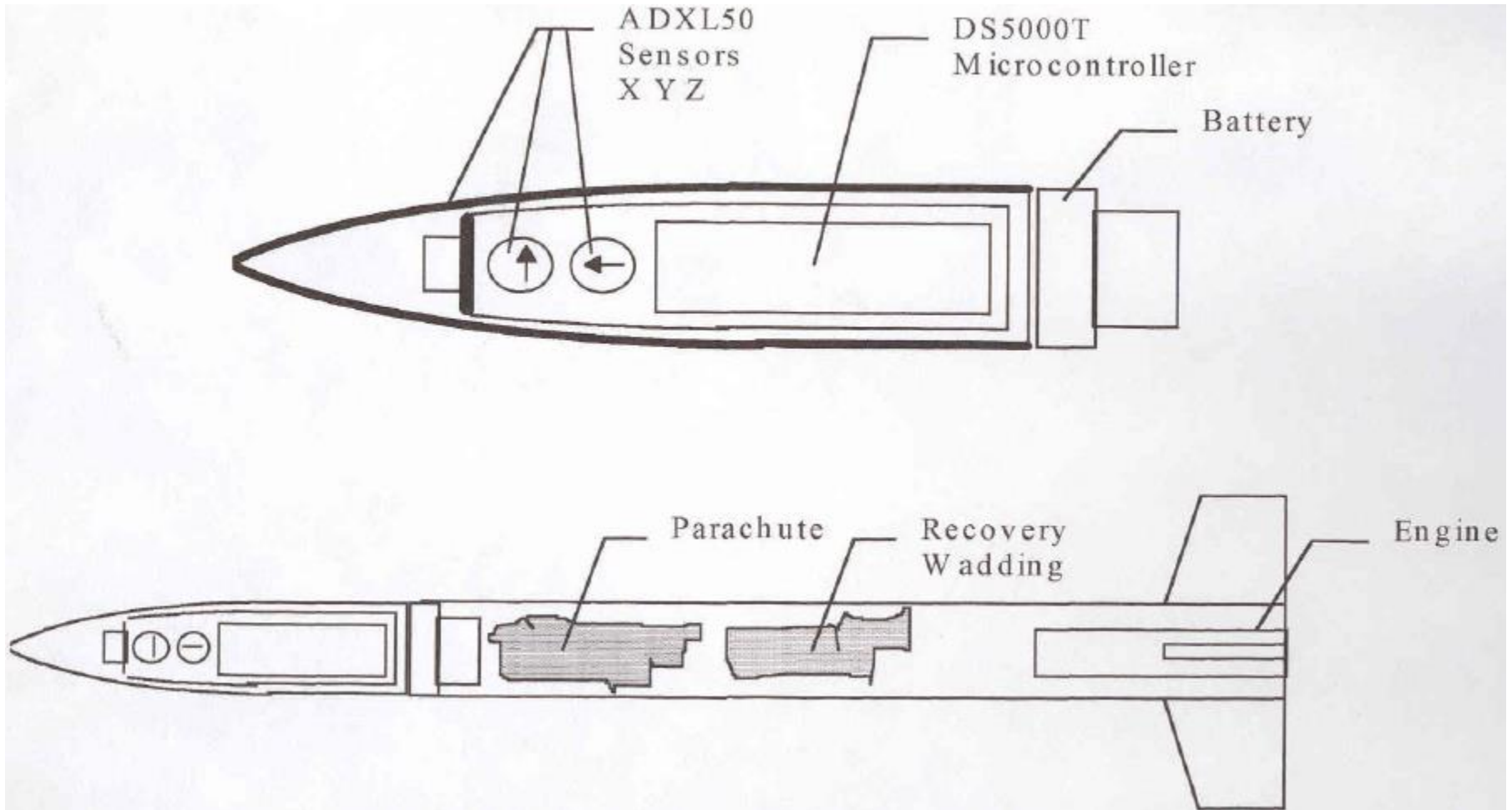


# Sistemas embarcados

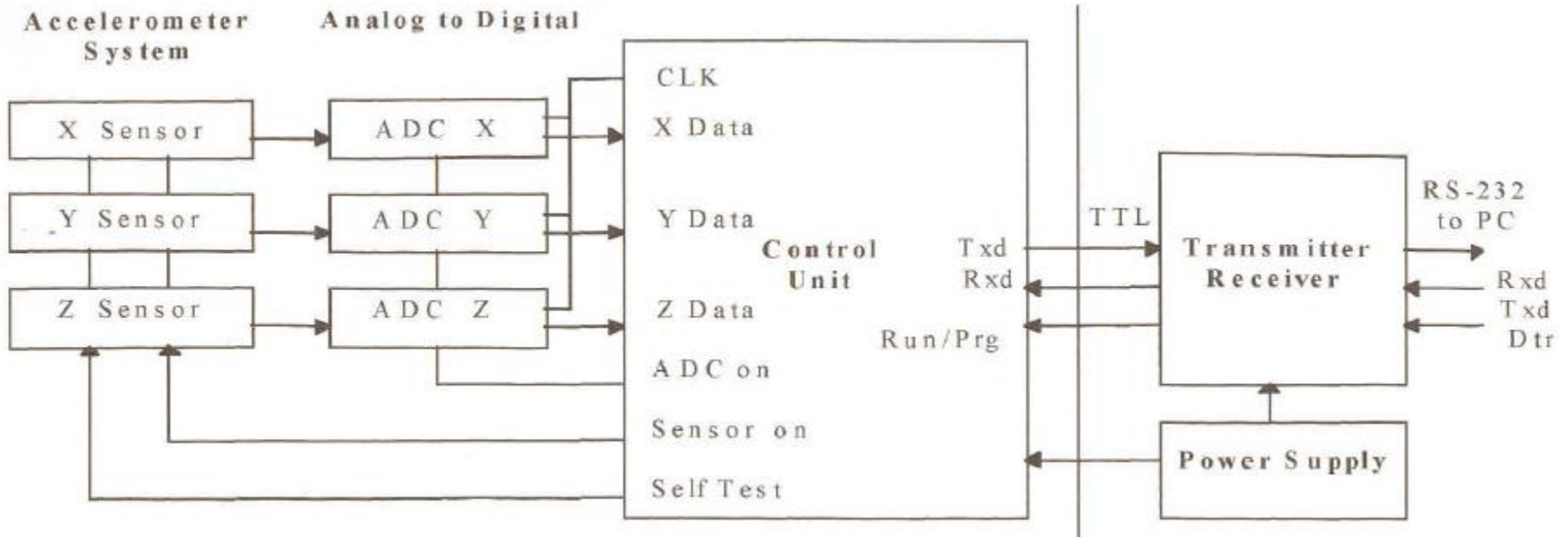
---



# Sistemas embarcados



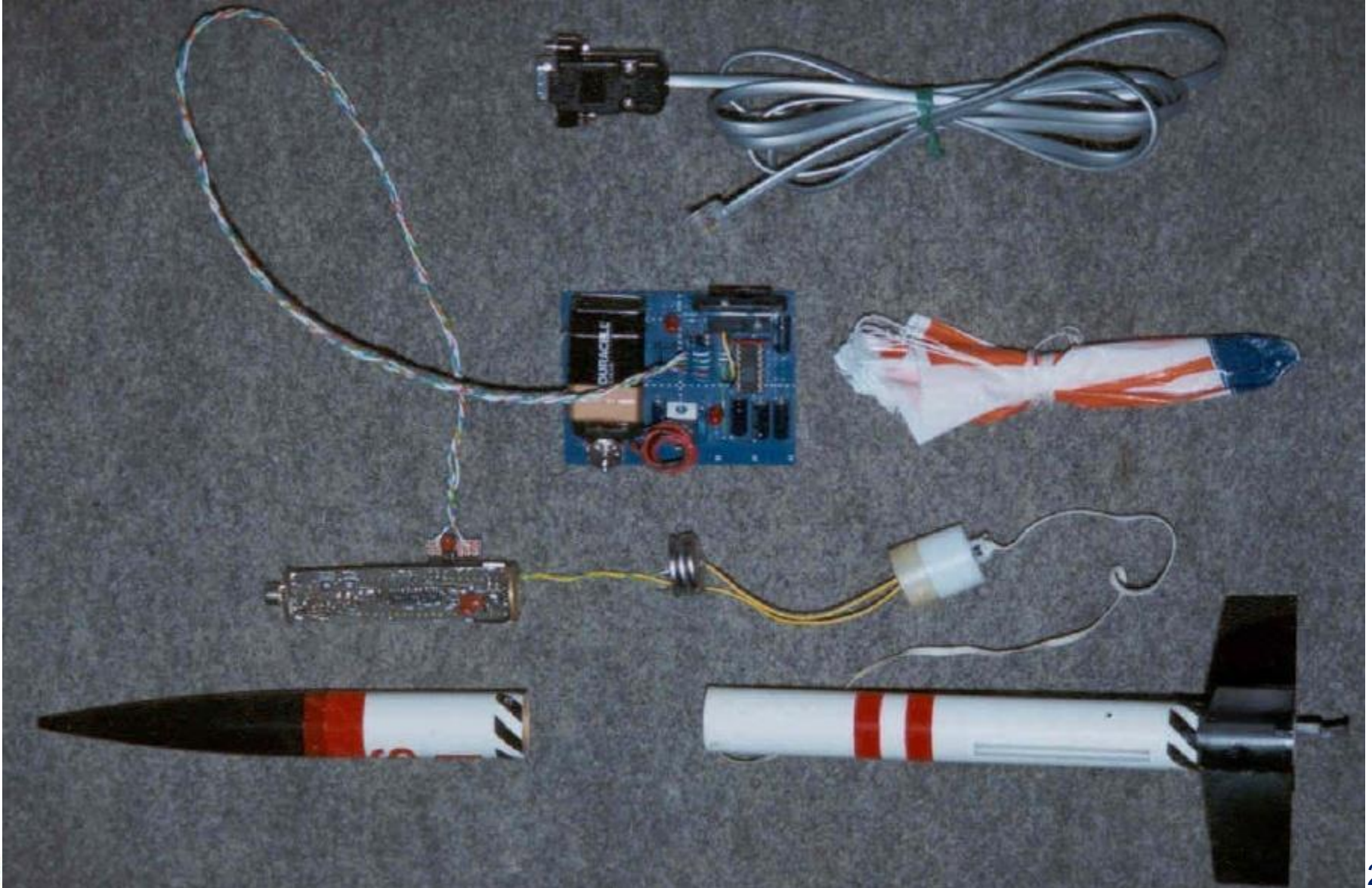
# Sistemas embarcados





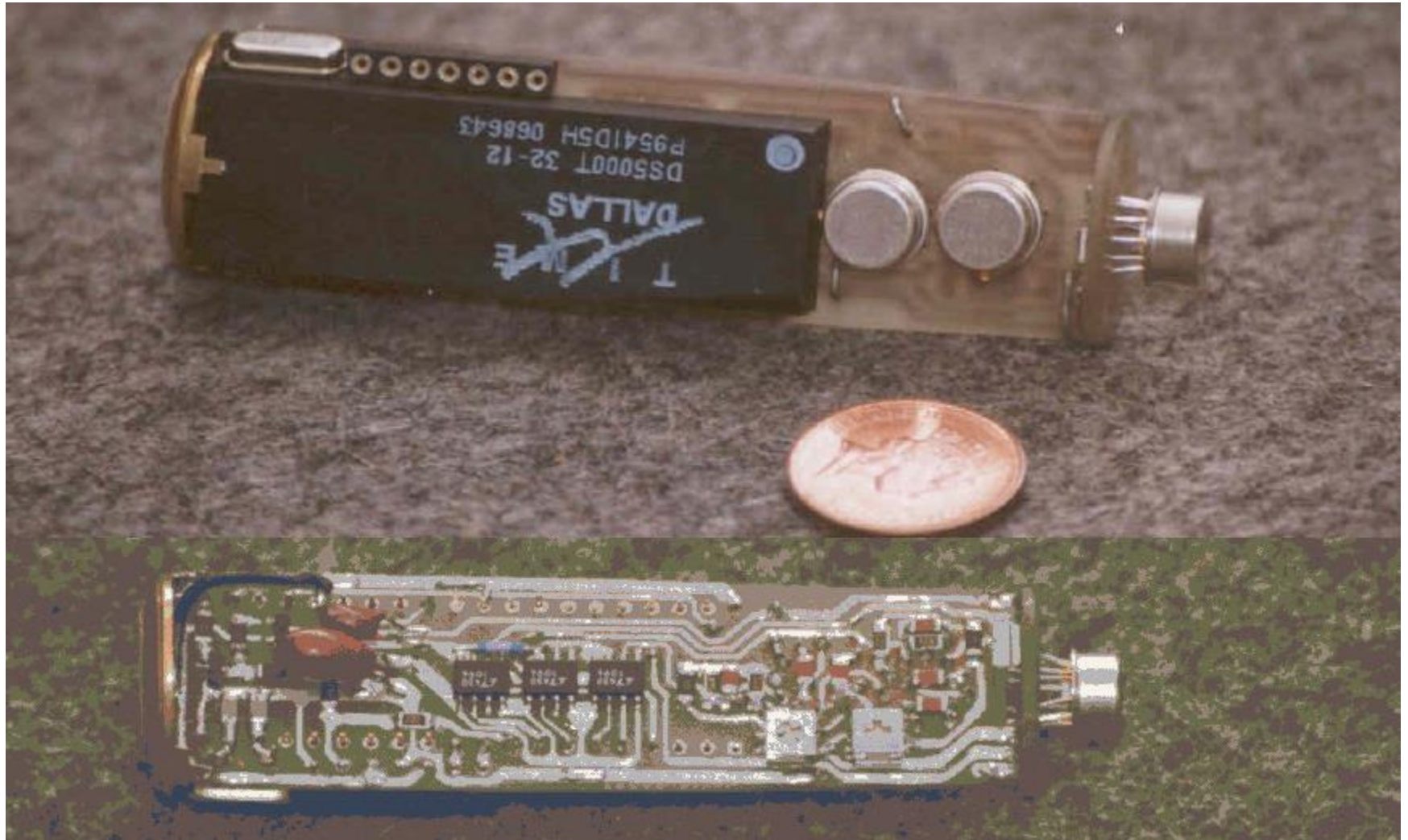
# Sistemas embarcados

---



# Sistemas embarcados

---





# Sistemas embarcados

---



# Sistemas embarcados

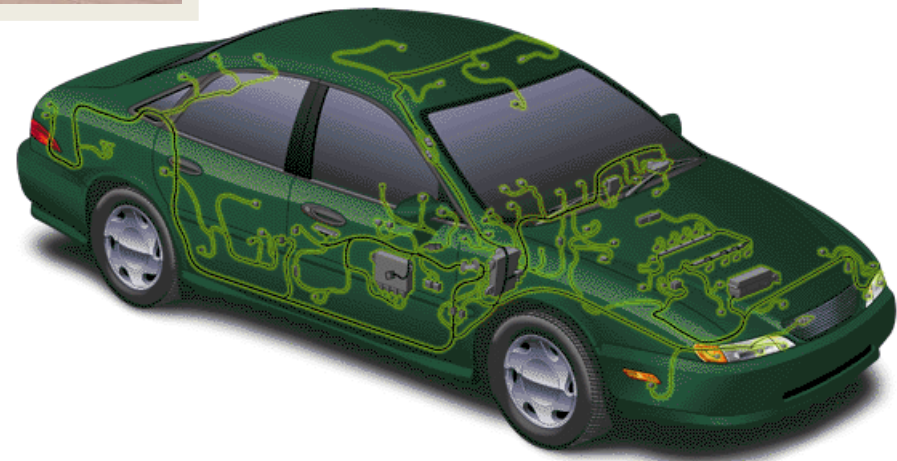
---



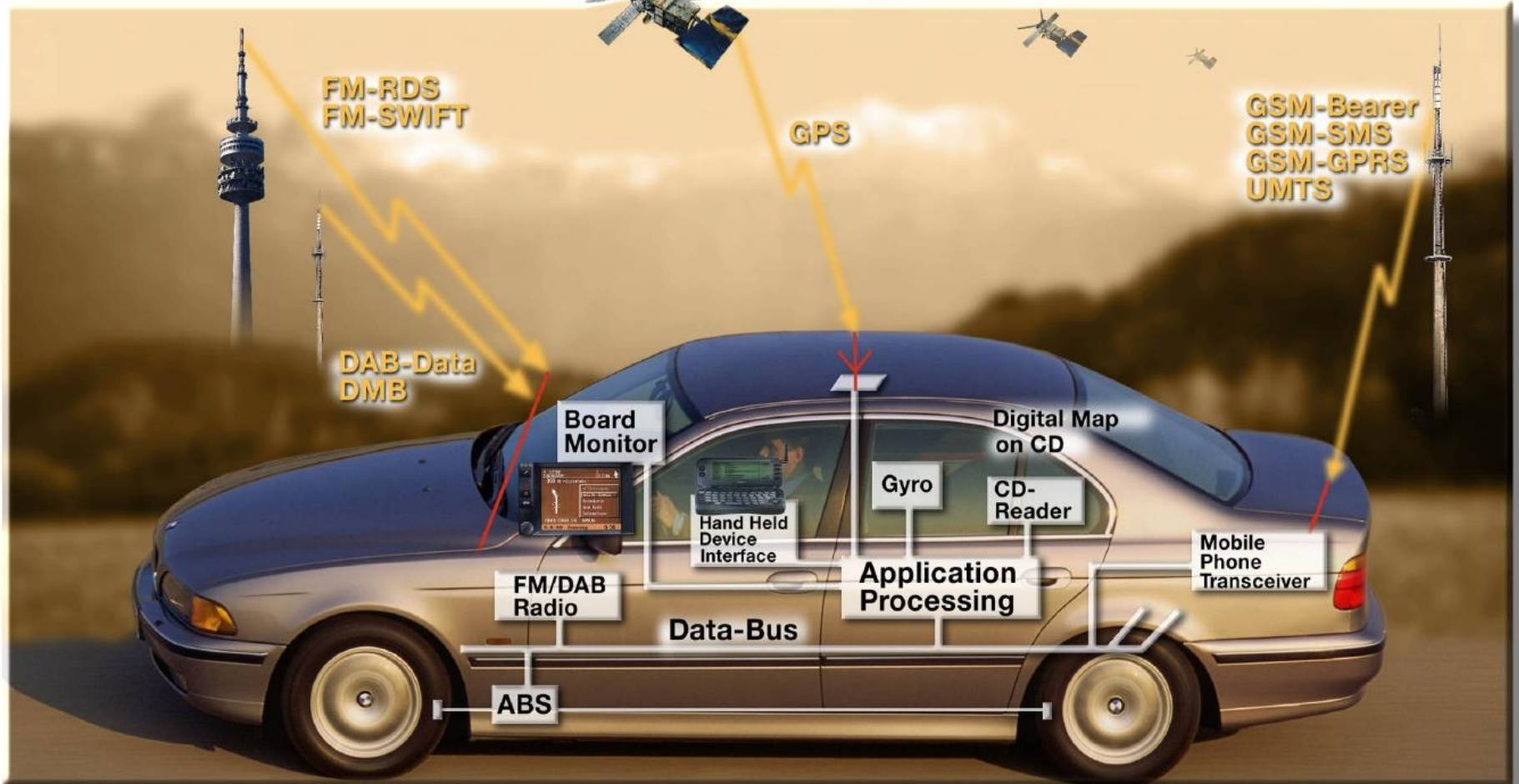


# Sistemas embarcados

---

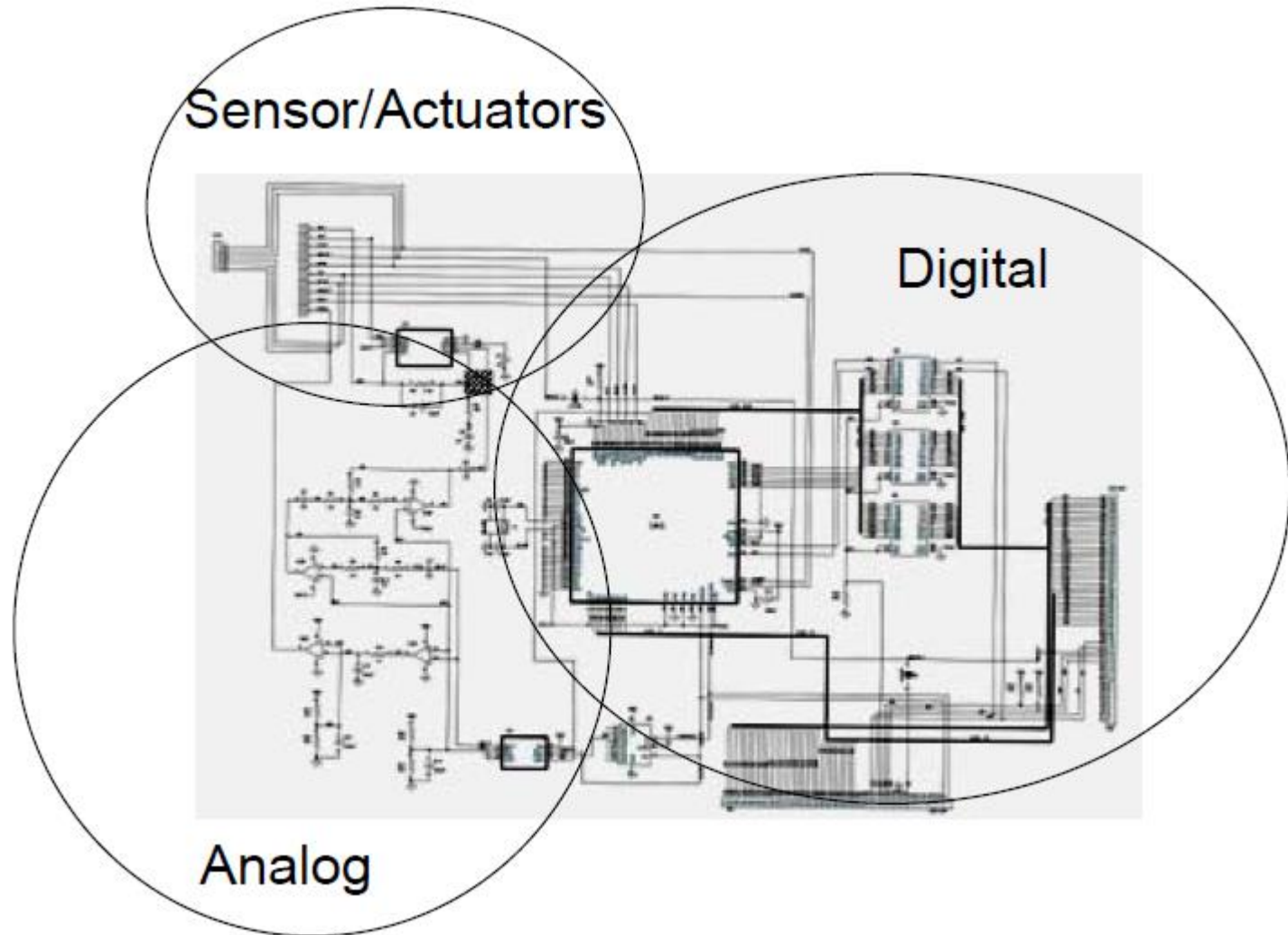


# Sistemas embarcados

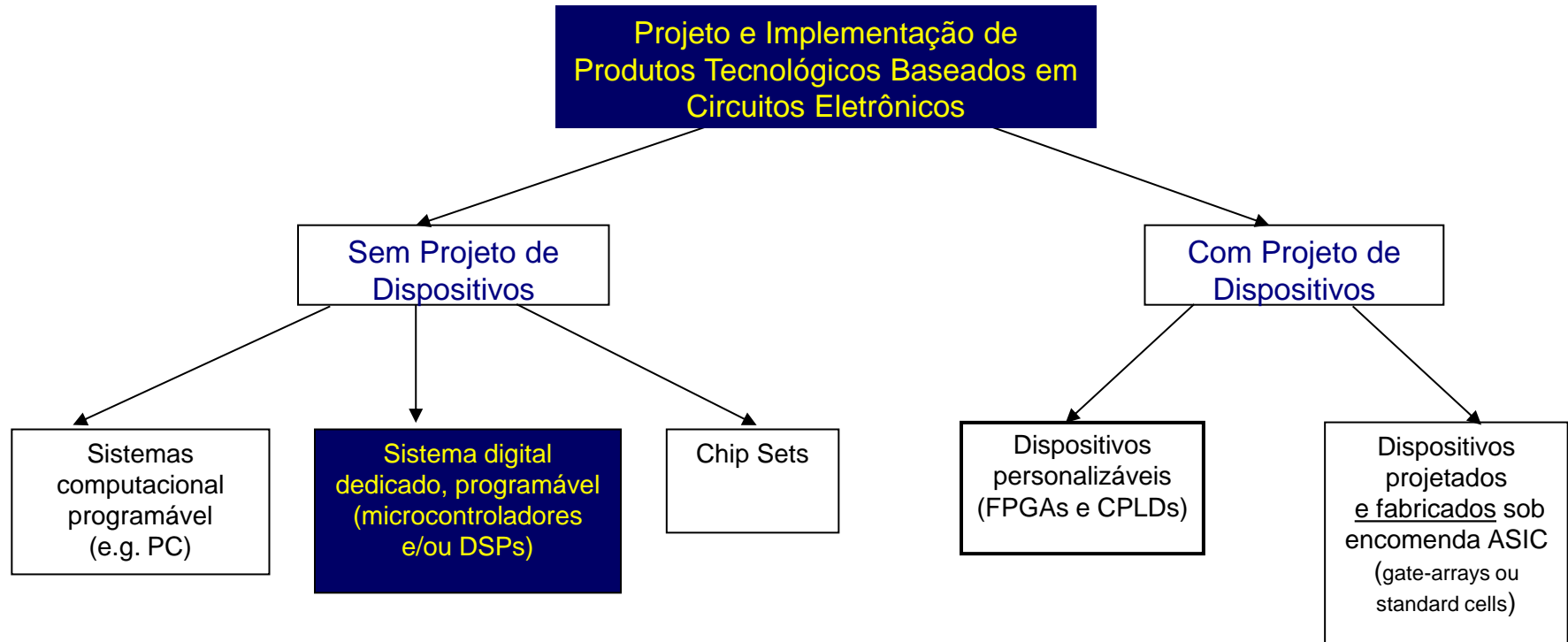


# Sistemas embarcados

---



# Sistemas embarcados



→ Aumento de desempenho (maior velocidade e menor potência dissipada), sigilo de projeto, custo de desenvolvimento

← Diminuição da complexidade de projeto, flexibilidade para alterações



# Sistemas embarcados

## Escolhas de projeto



| Tecnologia           | Desempenho/<br>Custo | Tempo para<br>obter sistema<br>funcionando | Tempo para<br>atingir alto<br>desempenho | Tempo para<br>alteração na<br>funcionalidade<br>do código |
|----------------------|----------------------|--|--|---|
| <b>ASIC</b>          | Muito alto           | Muito longo                                | Muito longo                              | Impossível  |
| <b>FPGA</b>          | Médio/Alto           | Médio                                      | Longo                                    | Médio   |
| <b>ASIP/<br/>DSP</b> | Médio/Alto           | Longo                                      | Longo                                    | Médio   |
| <b>Genérico</b>      | Baixo/Médio          | Muito curto                                | Não atingível                            | Muito curto   |

# Sistemas embarcados

Hardware dedicado

Software executando em hardware genérico

| Implementação                                       | Custo de projeto                       | Custo unitário         | Upgrades, correções de bugs | Tamanho                  | Consumo            | Velocidade          |
|---|--|------------------------|-----------------------------|--------------------------|--------------------|---------------------|
| Lógica discreta                                     | Baixo                                  | Médio                  | Difícil                     | Grande                   | ?                  | Muito rápido        |
| ASIC  | Alto<br>\$500K/<br>conjunto<br>máscara | Muito baixo            | Difícil                     | Minúsculo<br>1 die       | Baixo              | Rapidíssimo         |
| Lógica programável –<br>FPGA, PLD                   | Baixo                                  | Médio                  | Fácil                       | Pequeno                  | Médio<br>para alto | Muito rápido        |
| Microprocessador<br>+ memória +<br>periféricos      | Baixo<br>para<br>médio                 | Médio                  | Fácil                       | Pequeno<br>para<br>médio | Médio              | Moderado            |
| Microcontrolador<br>(int. memória e<br>periféricos) | Baixo                                  | Médio<br>para<br>baixo | Fácil                       | Pequeno                  | Médio              | Lento a<br>moderado |
| PC embarcado  | Baixo                                  | Alto                   | Fácil                       | Médio                    | Médio<br>para alto | Moderado            |

# Software embarcado

# Software embarcado

---

Artigo: Embedded Software (páginas 55-95)

Autor: Edward A. Lee, [eal@eecs.berkeley.edu](mailto:eal@eecs.berkeley.edu)

Livro Advances in Computers (ISBN: 978-0-12-012156-4)

Editor Marvin V. Zelkowitz

Academic Press, London, 2002

# Embedded Software - Edward A. Lee

---

- Its principal role is not the transformation of data, but rather the **interaction with the physical world**.
- It executes on machines that are not, first and foremost, computers. They are cars, airplanes, telephones, audio equipment, robots, appliances, toys, security systems, pacemakers, heart monitors, weapons, television sets, printers, scanners, climate control systems, manufacturing systems, and so on.
- Software with a principal role of interacting with the physical world must, of necessity, acquire some properties of the physical world. It **takes time**. It **consumes power**. It **does not terminate** (unless it fails).

# Embedded Software - Edward A. Lee

---

- Computer science has tended to view this physicality of embedded software as messy. Consequently, design of embedded software has not benefited from the richly developed abstractions of the twentieth century. Instead of using object modeling, polymorphic type systems, and automated memory management, engineers write assembly code for idiosyncratic digital signal processors (DSPs) that can do finite impulse response filtering in one (deterministic) instruction cycle per tap.
- They see Java programs stalling for one third of a second to perform garbage collection and update the user interface, and they envision airplanes falling out of the sky. The fact is that the best-of-class methods offered by computer scientists today are, for the most part, a poor match to the requirements of embedded systems.

# Embedded Software - Edward A. Lee

---

- Embedded software designers face a serious challenge. The complexity of their applications (and consequent size of their programs) is growing rapidly.
- Their devices now often sit on a network, wireless or wired.
- Even some programmable DSPs now run a TCP/IP protocol stack.
- Meanwhile, reliability standards for embedded software remain very high, unlike general-purpose software.
- At a maximum, entirely new abstractions are needed that embrace physicality and deliver robustness.

# Embedded Software - Edward A. Lee

---

- An arrogant view of embedded software is that it is just software on small computers.
- This view is naïve. **Timeliness**, **concurrency**, **liveness**, **reactivity**, and **heterogeneity** need to be an integral part of the programming abstractions.
- They are essential to the correctness of a program. It is not sufficient to realize the right mapping from input data to output data.
- Embedded software designers face a serious challenge. The complexity of their applications (and consequent size of their programs) is growing rapidly.



# **Componentes básicos e fluxo de desenvolvimento**

# Componentes básicos

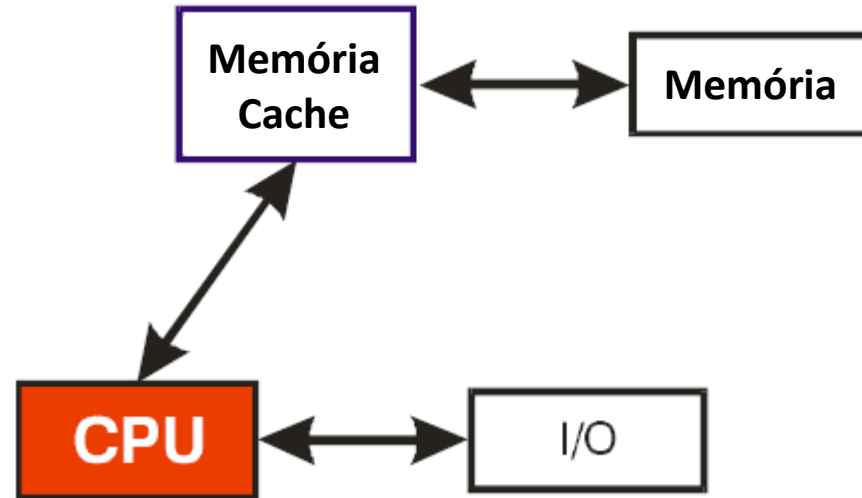
---

## Características de sistemas embarcados: visão do projetista na seleção da tecnologia e ferramentas de desenvolvimento

- Desenvolvimento simultâneo de hardware e software (*hardware/software codesign*)
- Variedade de microprocessadores/microcontroladores
- Variedade de sistemas operacionais, grande parte de tempo real (RTOS)
  - Muitas vezes sem serviços de SOs tais como 'printf'
- Quantidade reduzida de recursos ao se comparar com aplicações desktop
- Necessidade de ferramentas especiais para desenvolvimento
- Grande dificuldade para depuração
- Hardware e software precisam ser extremamente robustos

# Componentes básicos

---

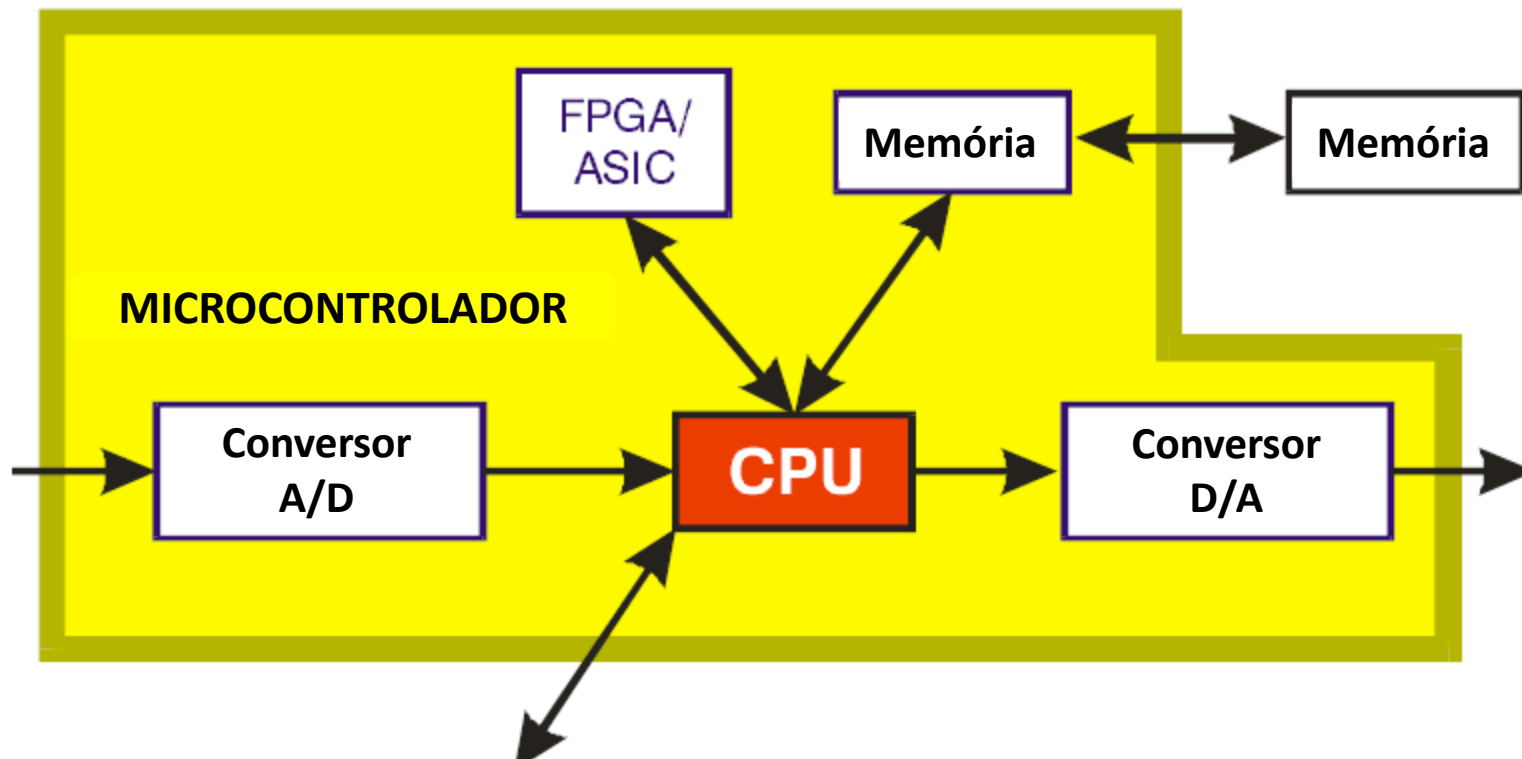


## Componentes básicos de sistemas embarcados:

- CPU
- Memória de dados e programa
- Sistema de entrada/saída

# Componentes básicos

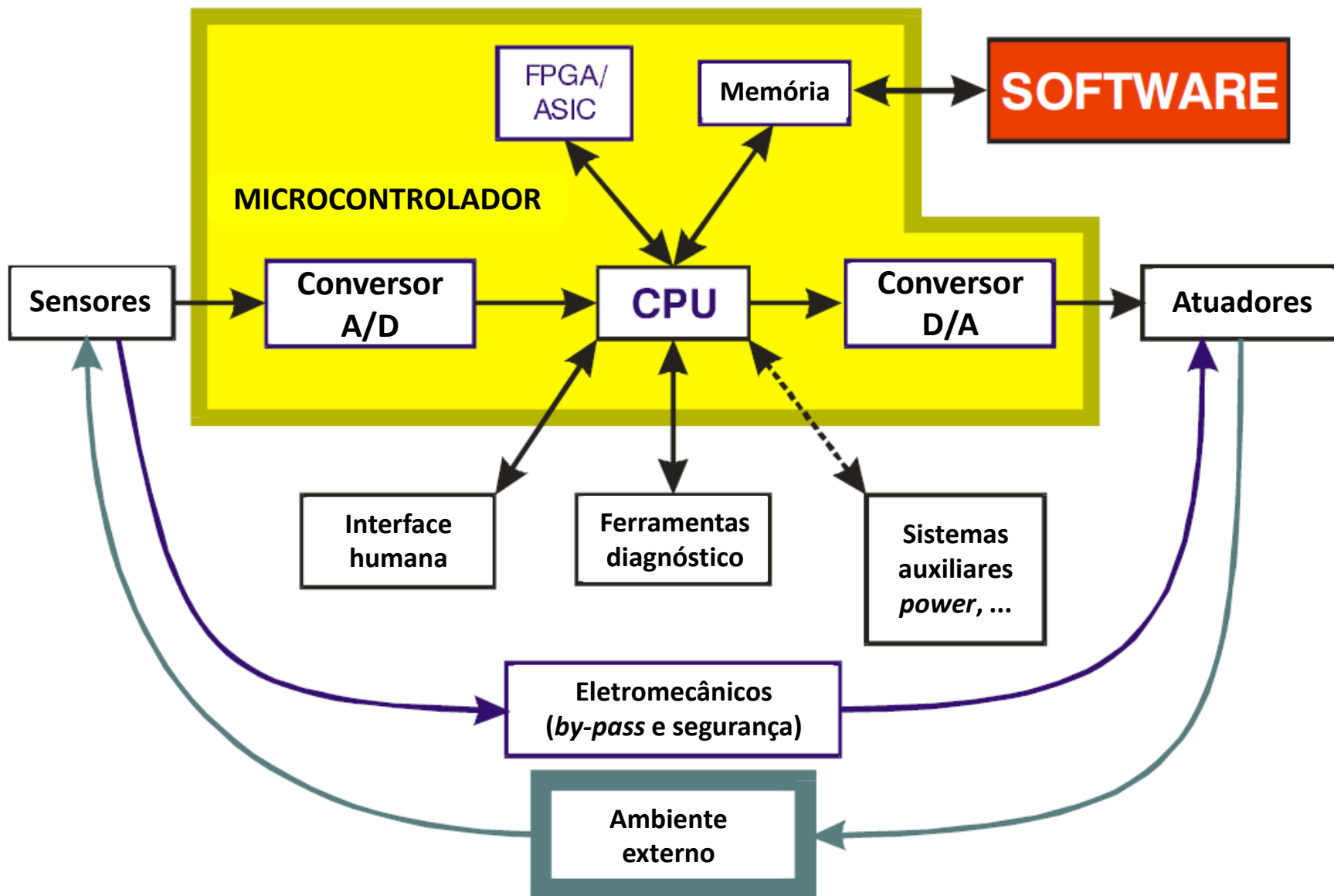
---



## Microcontroladores são computadores em um único chip

- Os periféricos estão embarcados no mesmo chip da CPU
- Algumas características, tamanho e custo reduzidos, alto desempenho com baixo consumo de energia, uso eficiente de espaço no PCB, baixo clock, endereçamento bit-a-bit

# Componentes básicos



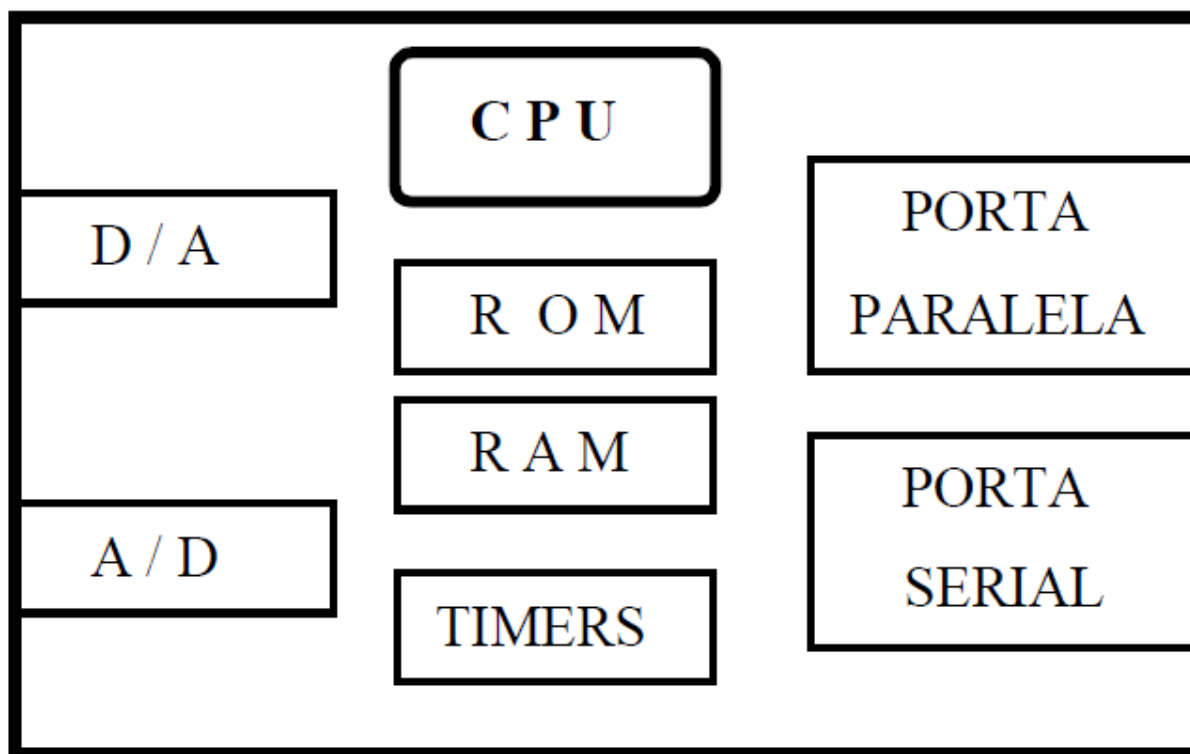
# Componentes básicos

---

## MCU – *Microcontroller Unit*

Composta por CPU e periféricos no mesmo encapsulamento

Componente central de um sistema embarcado típico

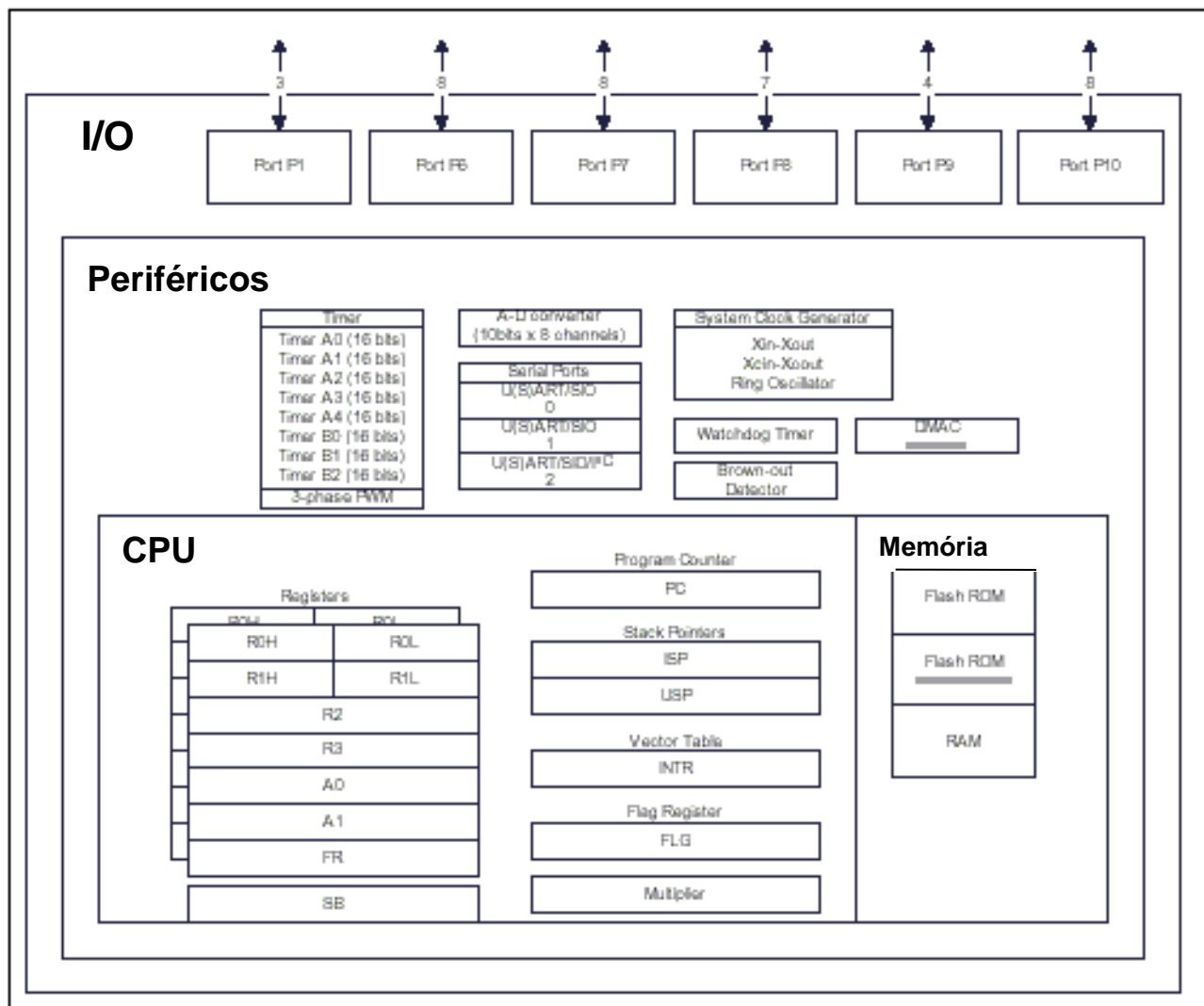


# Componentes básicos

## MCU – *Microcontroller Unit*

Composta por CPU e periféricos no mesmo encapsulamento

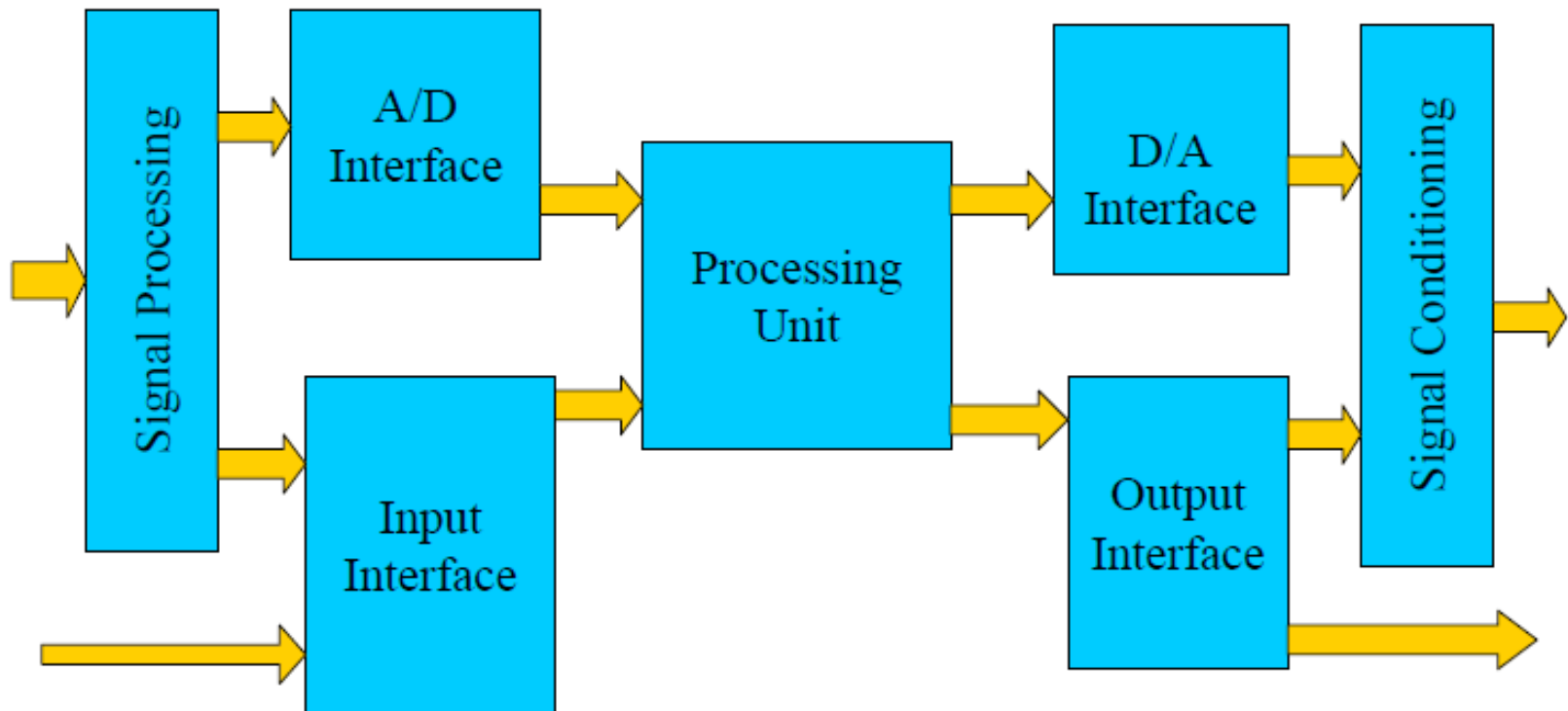
- Registradores
- RAM
- Flash
- EEPROM
- Portas digitais
- Portas Analógicas
- Timers
- Gerador de relógio
- DMA



# Componentes básicos

---

## Fluxo de dados





# Componentes básicos

---

## Diversidade de fabricantes e modelos de microcontroladores para sistemas embarcados:

- LINHA PIC (Microchip)
- LINHA AVR (Atmel)
- LINHA 8051 (Philips, Dallas, Intel, Cygnal, Texas, TDK, Siemens ... )
- Z8 Encore (Zilog)
- HC08 (Motorola)
- Renesas
- ARM (NXP)
- MSP430 (Texas)
- ...

# Componentes básicos

---

Diversidade de fabricantes e modelos de microcontroladores para sistemas embarcados:

- LINHA PIC (Microchip)
- LINHA AVR (Atmel)
- LINHA 8051 (Philips, Dallas, Intel, Cygnal, Texas, TDK, Siemens ... )
- Z8 Encore (Zilog)
- HC08 (Motorola)
- Renesas
- ARM (NXP)
- MSP430 (Texas)
- ...


## Escolha do dispositivo

- Capacidade de processamento
  - 8 bits, 16 bits, 32 bits
  - Clock, 4MHz, 40Mhz, ...
- Periféricos necessários
- Capacidade de memória
  - Programa
  - Dados
- Outros fatores
  - Ferramentas disponíveis
  - Formato físico
  - Continuidade / Reaproveitamento de projeto

# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)

---

- Reuniões com o cliente para levantamento de requisitos, funcionalidades, restrições, prazos, ...

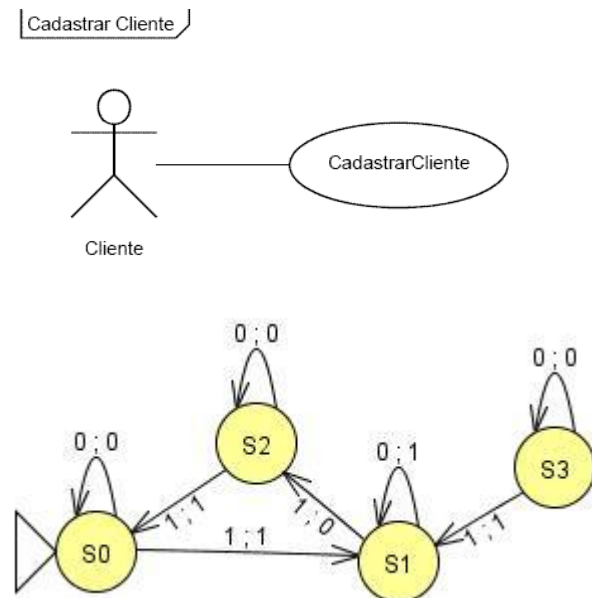
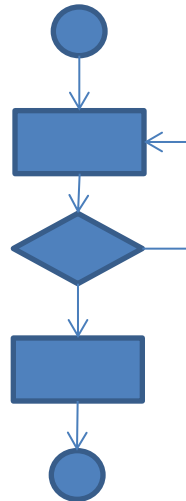


Requisitos  
Especificações  
Restrições

# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)

- Reuniões com o cliente para levantamento de requisitos, funcionalidades, restrições, prazos, ...
- **Uso de ferramentas para modelagem da solução proposta (ex. FSMs; fluxogramas; diagramas UML; entre outros) – auxilia o entendimento não apenas da equipe de software/hardware, mas também a interface com o cliente**

Requisitos  
Especificações  
Restrições



# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)

- Reuniões com o cliente para levantamento de requisitos, funcionalidades, restrições, prazos, ...
- Uso de ferramentas para modelagem da solução proposta (ex. FSMs; fluxogramas; diagramas UML; entre outros) – auxilia o entendimento não apenas da equipe de software/hardware, mas também a interface com o cliente
- **Se disponível, uso de simulador, cross-compiler e plataforma de prototipação para desenvolvimento do software e primeiros contatos com o projeto de hardware**

Requisitos  
Especificações  
Restrições

Desenvolvimento do software embarcado:  
Simulador,  
cross-compiler

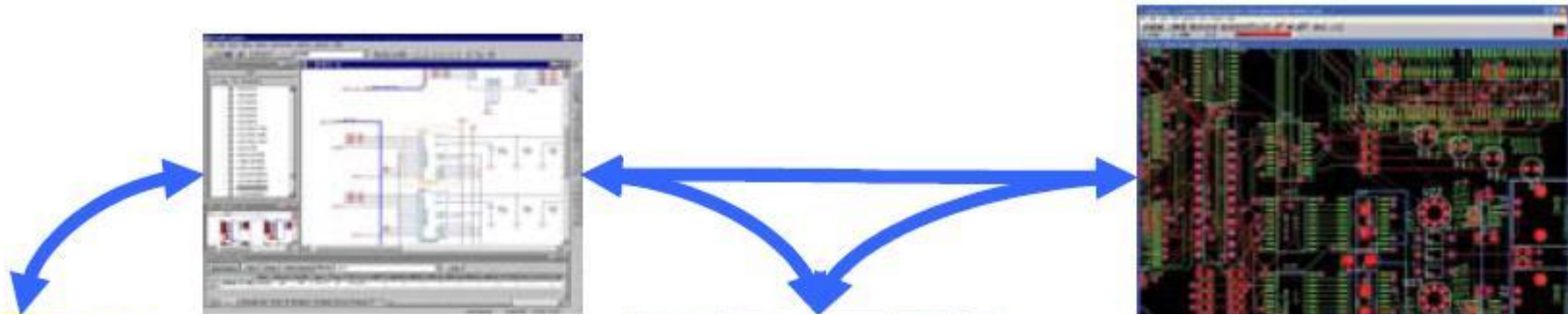


Teste do software embarcado e idéias para projeto do hardware: plataforma de prototipação com processador alvo



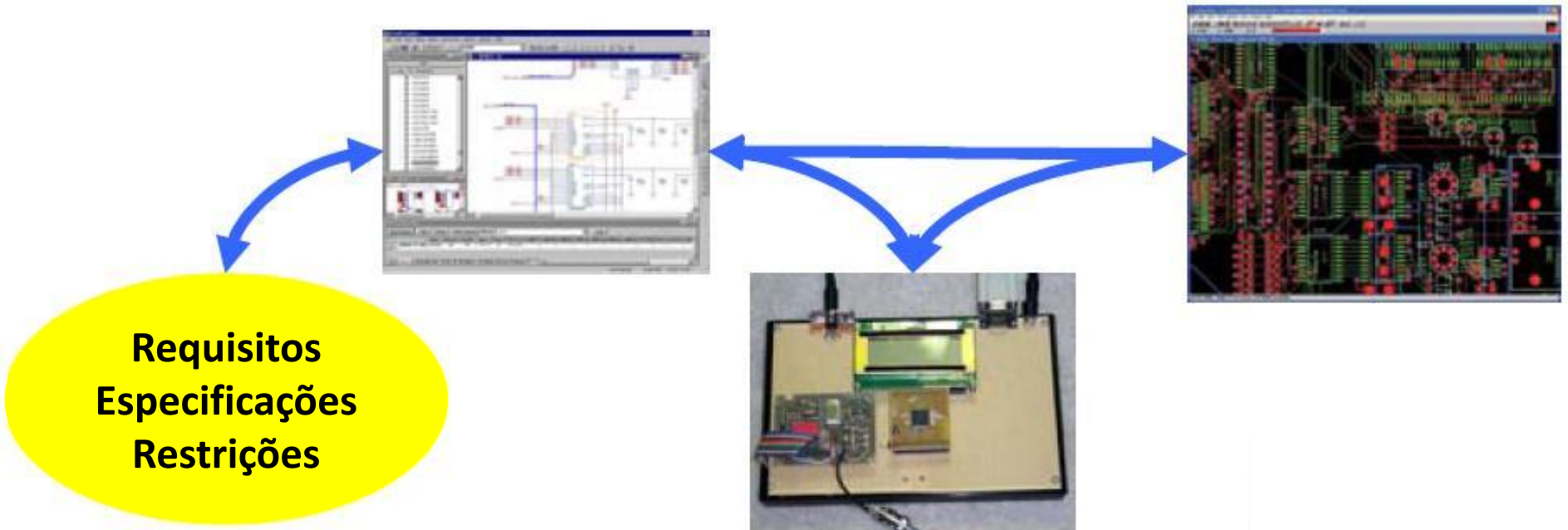
# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)

---



- Reuniões com o cliente para levantamento de requisitos, funcionalidades, restrições, prazos, ...
- Uso de ferramentas para modelagem da solução proposta (ex. FSMs; fluxogramas; diagramas UML; entre outros) – auxilia o entendimento não apenas da equipe de software/hardware, mas também a interface com o cliente
- Se disponível, uso de simulador, cross-compiler e plataforma para desenvolvimento do software e primeiros contatos com o projeto de hardware
- **Busca e compra de componentes (*procurement*)**
- **Uso de ferramentas de CAD (ex. Orcad) para projeto do hardware. Projeto do PCB, roteamento, layout, planta baixa. Uso de simuladores de hardware para validação do circuito (ex. Spice)**

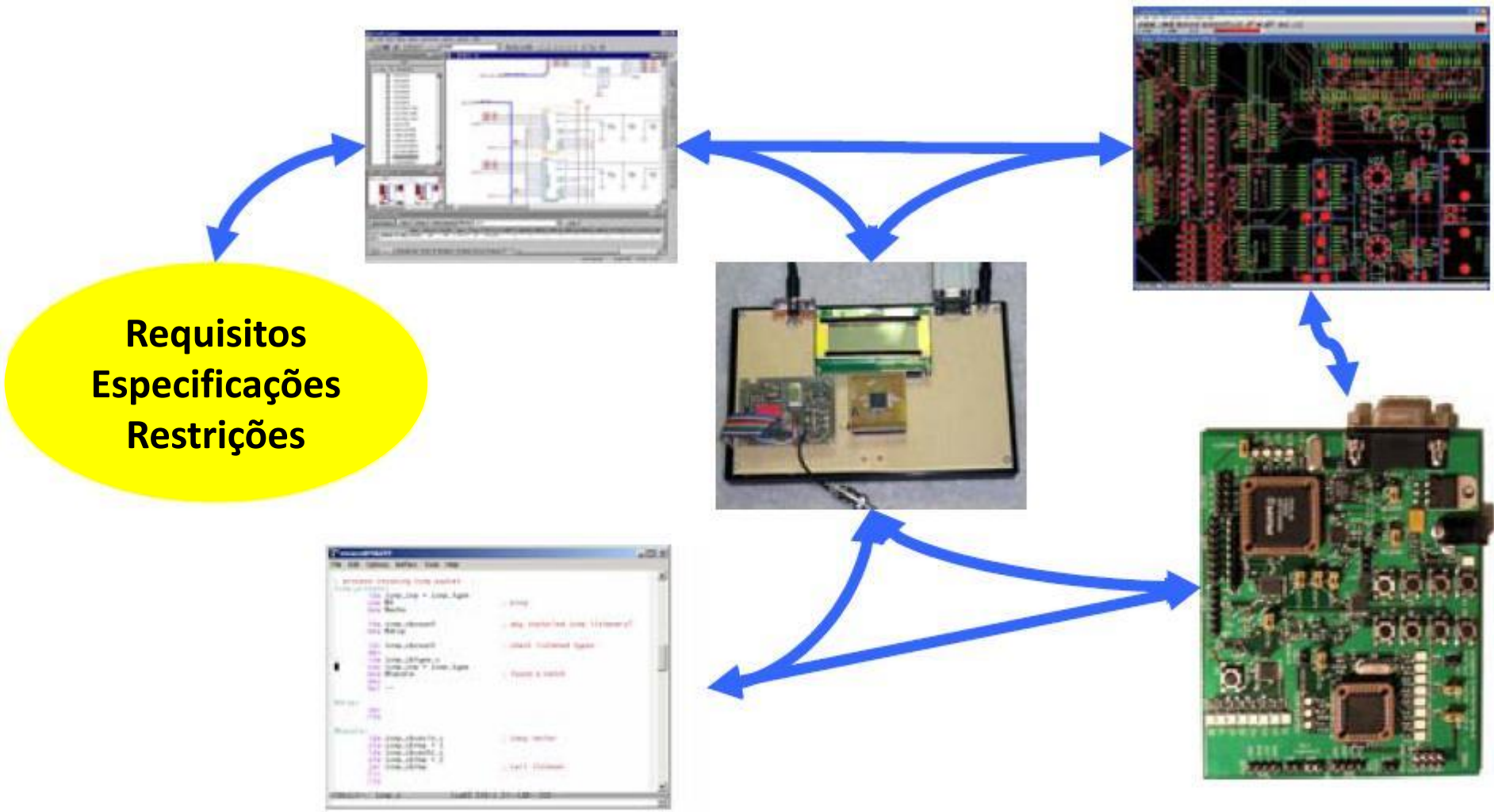
# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)



- Para projetos simples, é interessante uma prototipagem inicial do circuito em um proto-board, de forma a corrigir bugs de SW/HW a partir dos requisitos iniciais. O desenvolvimento das placas finais e soldagem possui um custo mais elevado em relação ao protótipo em proto-board.



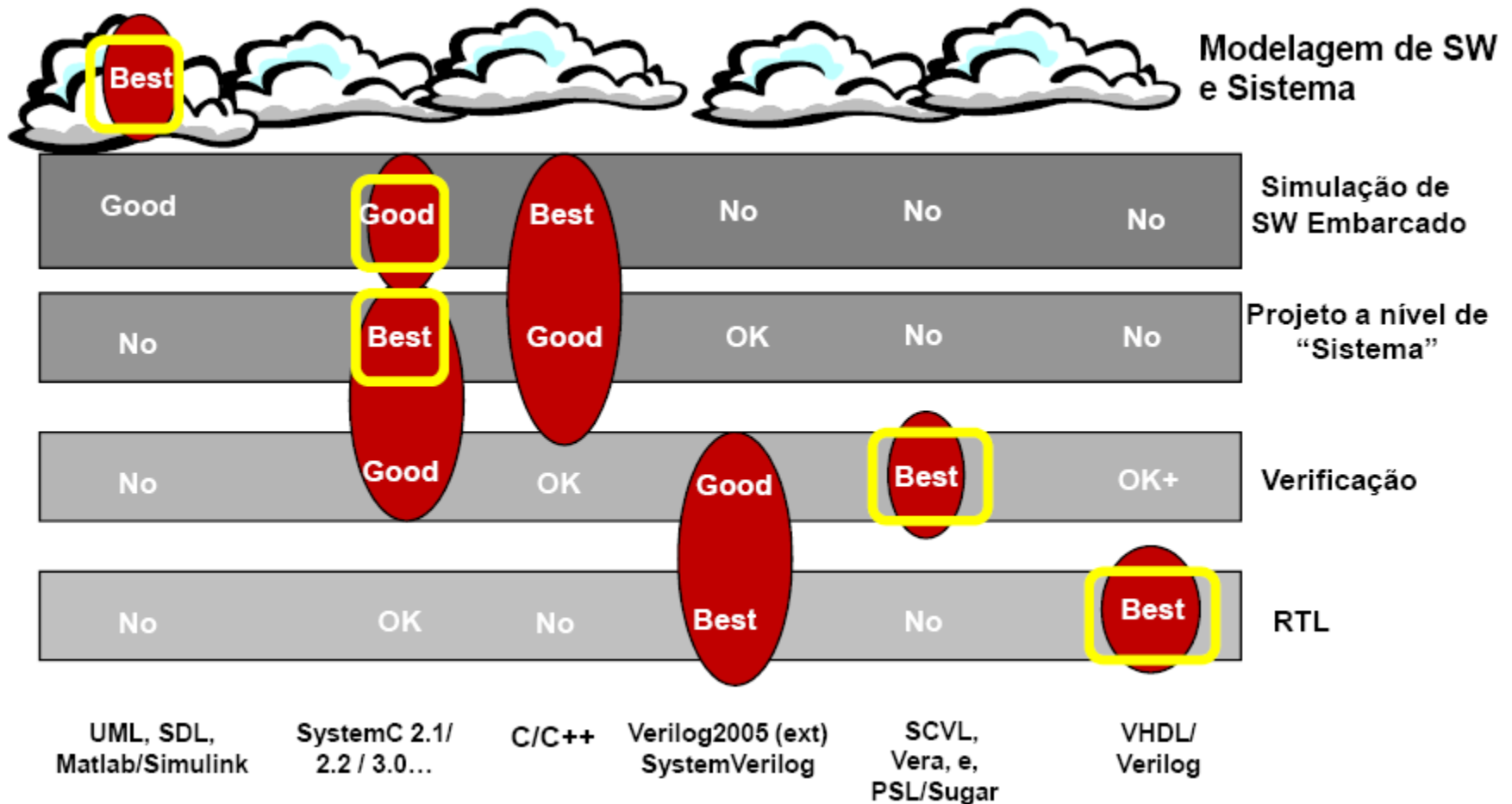
# Fluxo de desenvolvimento para uma aplicação típica (e de baixa complexidade)





# Ferramentas de desenvolvimento

# Ferramentas de desenvolvimento



# Ferramentas de desenvolvimento

---

I. Ferramentas de modelagem (FSM, Fluxograma, UML)

II. Ambiente de desenvolvimento (SDK, IDE, API)

- *Cross-compiler* (compilador cruzado), *linker*, *loader*
- Simulador

III. Linguagem de programação

IV. Emulador

V. Analisador lógico

VI. Analisador de protocolos

VII. Osciloscópio

VIII. Gerador de formas de onda

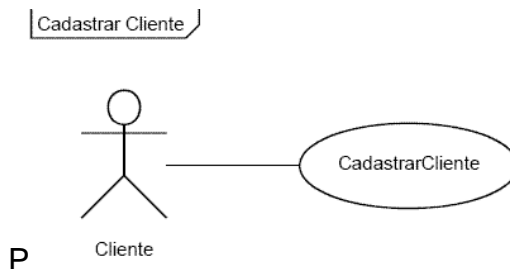
**Software**



**Hardware**

## UML

- Diversas opções de ferramentas para diversas linguagens, ou apenas para modelagem independente de linguagem (ex. Jude, Dia, plug-ins para eclipse)
- Sistema pode ser modelado em diversos níveis de abstração através de diversos tipos de diagramas (ex. diagramas de classes e objetos visando programação orientada a objetos)
- O comportamento do sistema pode ser modelado precisamente utilizando-se diagramas como, por exemplo, diagrama de seqüência
- A utilização de diagramas UML facilita a troca de informações entre componentes das equipes (software, hardware, software/hardware), e também com o contratante do projeto.
- Uso de UML nas etapas de desenvolvimento de sistemas embarcados deverá continuar aumentando devido a crescente complexidade das aplicações

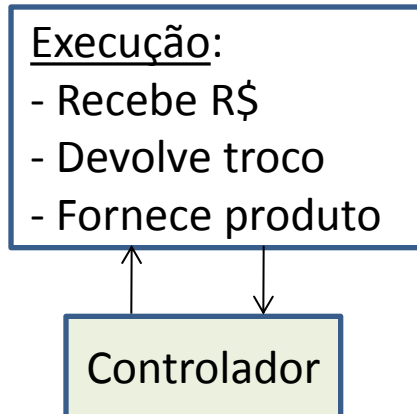


| Name <span>↗</span>                                     | Creator <span>↗</span>              | Platform / OS <span>↗</span>    | First public release <span>↗</span> | Latest stable version <span>↗</span> | Software license <span>↗</span>        | Open source <span>↗</span> | Programming language used <span>↗</span> | Approach <span>↗</span>   | Languages generated <span>↗</span>     |
|---|-------------------------------------|---------------------------------|-------------------------------------|--------------------------------------|--|----------------------------|--|---|--|
| <a href="#">Acceleo</a> <span>↗</span>                  | <a href="#">Obeo</a> <span>↗</span> | Java / Eclipse (cross-platform) | 2006-03                             | 2.5.1                                | EPL                                    | Yes                        | Java                                     | MDA, template   | JEE, C#, Java, PHP, Python.            |
| <a href="#">ArgoUML</a>                                 | <a href="#">Tigris.org</a>          | Java (cross-platform)           | 1998-04                             | 0.28                                 | BSD                                    | Yes                        | Java                                     |   | C++,C#,PHP4,PHP5                       |
| <a href="#">BoUML</a> <span>↗</span>                    | Bruno Pagès                         | C++/Qt (cross-platform)         | 2005-02-26                          | 4.15<br>2009-09-26                   | GPL                                    | Yes                        | C++                                      | MDA, template   | Java, C++, PHP, Python, IDL.           |
| <a href="#">Dia</a> <span>↗</span>                      | Alexander Larsson/GNOME Office      | GTK+ (cross-platform)           | 2004?                               | 0.97                                 | GPL                                    | Yes                        |  | Java, C++, ADA (using dia2code)   |  |
| <a href="#">Eclipse UML2 Tools</a>                      | Eclipse Foundation                  | Java (cross-platform)           | Planning                            | 1.1 Planned                          | GPL?                                   | Yes?                       | Java                                     |   | Java (or Eclipse project supported?)   |
| <a href="#">Jink UML</a> <span>↗</span>                 | Nether                              | Java (cross-platform)           | 2008-12-11                          | .745                                 | MIT                                    | Yes                        |  |   |  |
| <a href="#">Modelio Free Edition (see Objecteering)</a> | Modeliosoft                         | Windows                         | 2009                                | 1.0                                  | ?                                      | No                         | Java, C++                                | full UML2 support; integrated BPMN support. XML import; HTML and MS-Word document generation. | Java, C#, C++, XSD, WSDL               |
| <a href="#">StarUML</a>                                 | Plastic Software                    | Windows                         | 2005-11-01                          | 5                                    | GPL, modified                          | Yes                        | <a href="#">Delphi</a>                   | Plug-in architecture: C++, Delphi, C#, VB,  |  |
| <a href="#">Visual Paradigm for UML</a>                 | Visual Paradigm Int'l Ltd.          | Java (cross-platform)           | 2002-06-20                          | 7                                    | Commercial with Free Community Edition | No                         | <a href="#">Java</a>                     | Full UML, SysML, ERD and BPMN Support   | Java, C#, C++, PHP, Ada, Action Script |
| <a href="#">Umbrello UML Modeller</a>                   | Umbrello Team                       | Linux                           | 2006-09-09                          | 2.0.0                                | GPL                                    | Yes                        | C++, KDE                                 |   | C++, Java, Perl, PHP, Python... 16     |

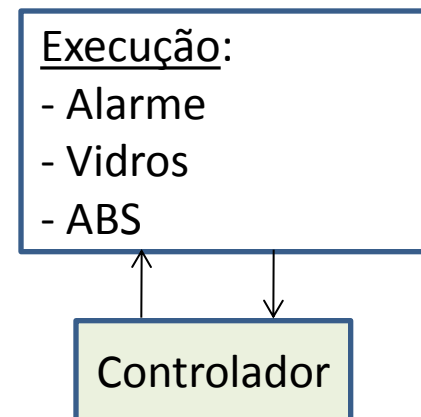
## FSM – *Finite State Machine*

- Sistemas embarcados, normalmente, são compostos por um módulo de “controle” e um módulo para “execução das operações”.

Máquina de venda  
de refrigerantes



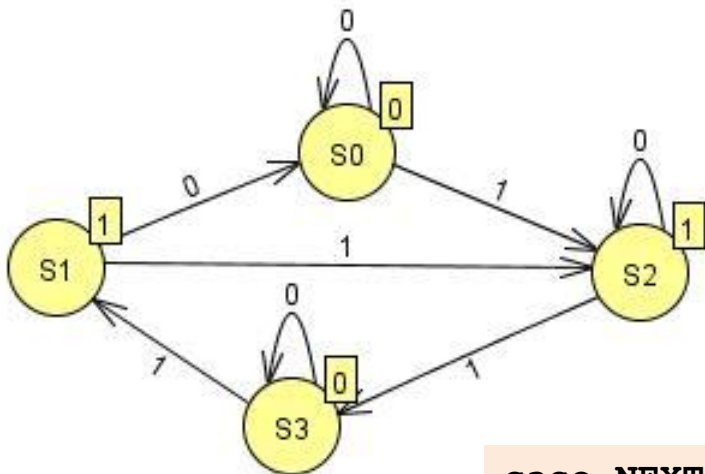
Automóvel



# Modelagem

O comportamento da sequência de atividades em aplicações embarcadas pode ser modelado por FSMs de diversas formas:

## Diagramas de estados (grafos)



## Tabelas de transição de estados

| Estado Atual | Próximo Estado |     | Saída Atual (z) |
|--------------|----------------|-----|-----------------|
|              | X=0            | X=1 |                 |
| S0           | S0             | S2  | 0               |
| S1           | S0             | S2  | 1               |
| S2           | S2             | S3  | 1               |
| S3           | S3             | S1  | 0               |

```

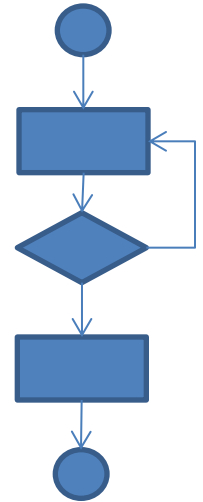
case NEXT_STATE is
  when S0_C =>
    if (x = 0)
      z = 0;
      NEXT_STATE <= S0_C;
    else
      z = 0;
      NEXT_STATE <= S2_C;
    end if;
  when S1_C =>

```

## Linguagens de programação

## Fluxograma

- Bastante útil para auxiliar na organização do fluxo de dados e controle de programas em geral para sistemas embarcados
- Indispensável no desenvolvimento de programas em assembly
- Diversas ferramentas disponíveis (MS-Visio, Dia, ...)



## Linguagem natural, algoritmos, diagramas de blocos

- O tipo de modelagem a ser utilizada depende da aplicação alvo, e do nível de abstração
- Descrições em linguagem natural ou em algoritmos são bastante úteis em complemento a outras abordagens de modelagens, ou até mesmo como única forma de modelagem no caso de sistemas com menor complexidade



# Ambiente de desenvolvimento

---

- **SDK** (*Software Development Kit*) – conjunto de ferramentas de desenvolvimento
- SDKs podem ser compostas por apenas uma simples **API** (*Application Programming Interface*) para utilização em uma determinada linguagem de programação, ou podem possuir hardware sofisticados para interface com sistemas embarcados
- Ferramentas de SDKs, normalmente, são disponibilizadas em **IDEs** (*Integrated Development Environment*).
- IDEs incluem sistemas de ajuda, documentação, e facilidades para depuração.
- SDKs também podem incluir código exemplo, templates, documentação auxiliar, entre outros.

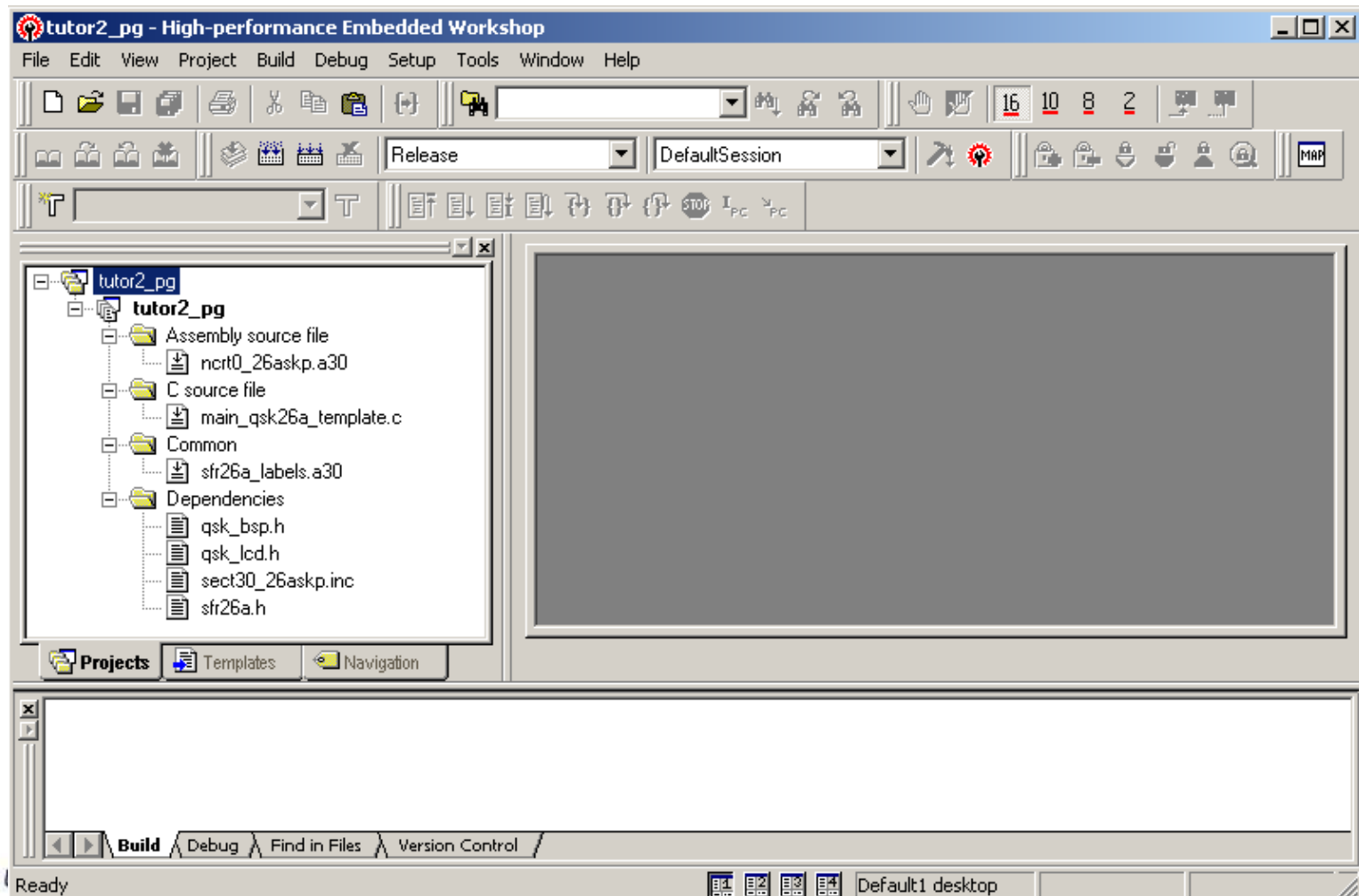
## IDE típica

The screenshot displays the Topview IDE interface for an Atmel AT89C51 microcontroller. The interface includes several key windows:

- Program:** A table of assembly code with columns for Address, BP, Code, and Mnemonic. A callout box labeled "Programa" points to this window.
- Internal Data Memory:** A grid showing memory addresses and their corresponding hexadecimal values. A callout box labeled "Memória interna" points to this window.
- Registers:** A list of Special Function Registers (SFRs) and their current values, such as the Accumulator (ACC) at 12 and Program Counter (PC) at 0000. A callout box labeled "Registradores" points to this window.
- Memory Bit Status:** A table showing the status of individual memory bits (Bit00 to Bit15). A callout box labeled "Bits de status" points to this window.
- SFR Bit Status:** A table showing the bit status for specific SFRs, including the Accumulator (ACC) and B Register (B). A callout box labeled "Bits dos SFR (8051)" points to this window.

# Ambiente de desenvolvimento

[IDE da Renesas](#) – SDK, ambiente de projeto, APIs, templates, simulador, facilidades para depuração de hardware, programação de microcontroladores, entre outros.





# Ambiente de desenvolvimento

**Keil** – Empresa do grupo ARM. IDE para diversas arquiteturas (ARM, 8051, ...)

The screenshot displays the Keil IDE interface with several key components:

- Register Window:** Shows the current state of registers (R0-R15, SP, CPSR, SPSR).
- Source Code Window:** Displays C code for a serial interface, including functions like `void init_serial`, `int sendchar`, and `int getkey`.
- Disassembly Window:** Shows the corresponding assembly instructions for the source code, such as `BC R2, #0x00000000` and `STR R1, [R2, #0x1C]`.
- Logic Analyzer Window:** Displays a waveform graph showing digital signals over time, with a time scale from 30.25000s to 35.25000s.
- Command Window:** Shows the execution of commands like `start`, `save_record`, and `end`.
- Watch Window:** Displays the values of variables being monitored during execution, such as `start` and `save_record`.
- Memory Window:** Shows the contents of memory addresses, including the `save_record` variable.

The disassembly window shows trace intermixed with source code.

Code coverage and profiling information display in the source window.

The toolbox contains user definable buttons to run commands or debug

The logic analyzer shows changes to variables and signals over time.

Symbol names may be dragged and dropped to other debugger windows.

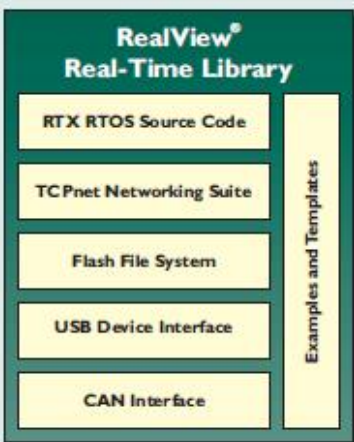
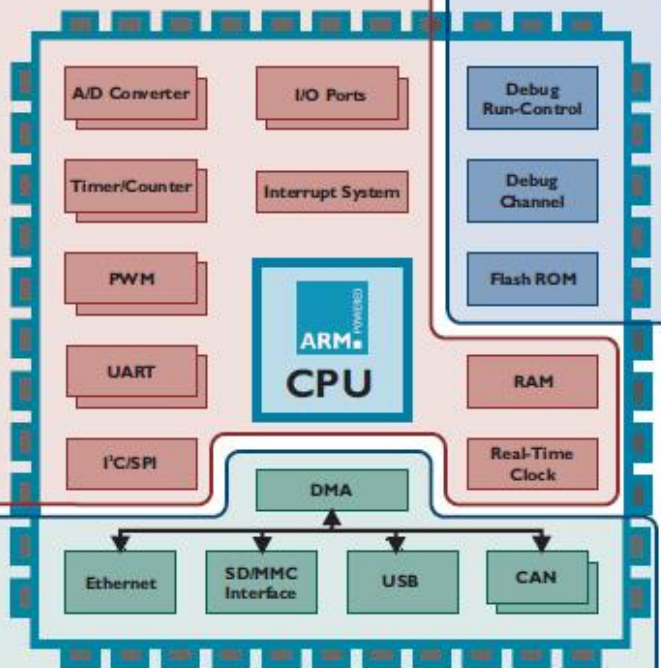
Memory and watch windows display important program variables.

Enter debug commands into the command tab of the output window.



## Microcontroller Development Kit (MDK)

- Best-in-class ARM RealView<sup>®</sup> C/C++ Compiler.
- Genuine Keil  $\mu$ Vision<sup>®</sup> IDE/Debugger/Simulator.
- Royalty-free RTX Real-Time Operating System.
- Easy device configuration with Device Database support for more than 260 ARM Powered devices.



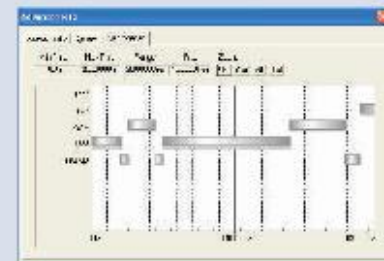
## Real-Time Library (RL-ARM)

- RTX Real-Time OS with Source Code.
- TCP/IP Suite with Server Applications.
- File System for ROM and Memory Cards.
- Supports USB Standard Device Drivers.
- CAN Driver with RTOS Interface.



## ULINK2<sup>®</sup> Adapter

- JTAG & Serial Wire Interface.
- Target Debugging.
- Flash Programming.
- Real-Time Trace and on-the-fly debugging.



## Evaluation Boards



Keil provides a wide range of evaluation boards for devices based on ARM7, ARM9 and Cortex-M3.

# Ambiente de desenvolvimento

---

**Ambientes de desenvolvimento, normalmente, disponibilizam um compilador cruzado (ex. gcc, sdcc, keil, ...) e facilidades para simulação.**

- **Compilador cruzado (*cross-compiler*)**

- SDCC: Compilador para plataformas Linux x86, Windows e Mac OS.

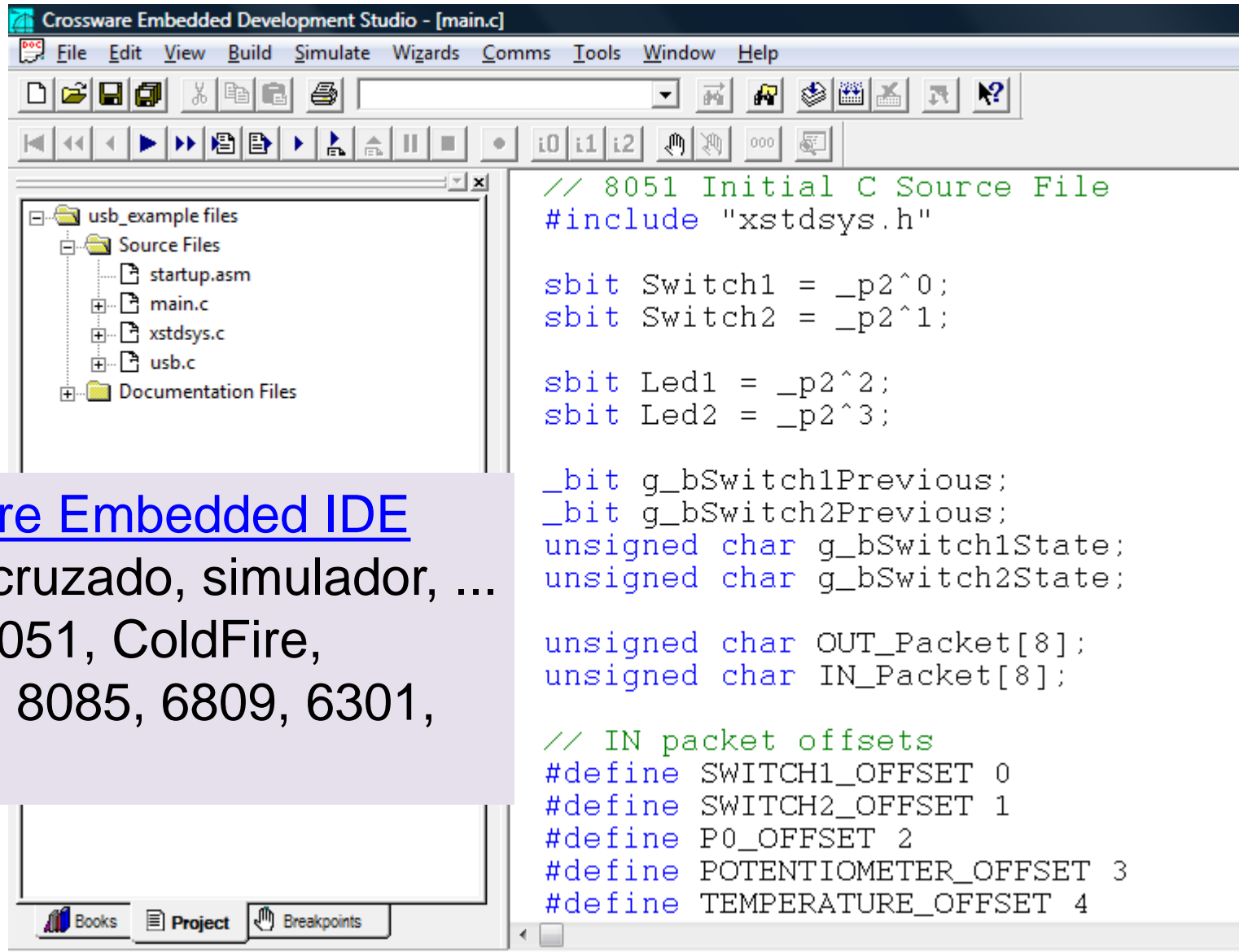
- Gera binários para 8051, DS390, Z80, HC08 e PIC

<http://sdcc.sourceforge.net>.

Exemplo de utilização (linha de comando Windows):

```
C:\PIC\source\>sdcc --debug -mpic14 -p16f627 toggle_led.c
```

# Ambiente de desenvolvimento

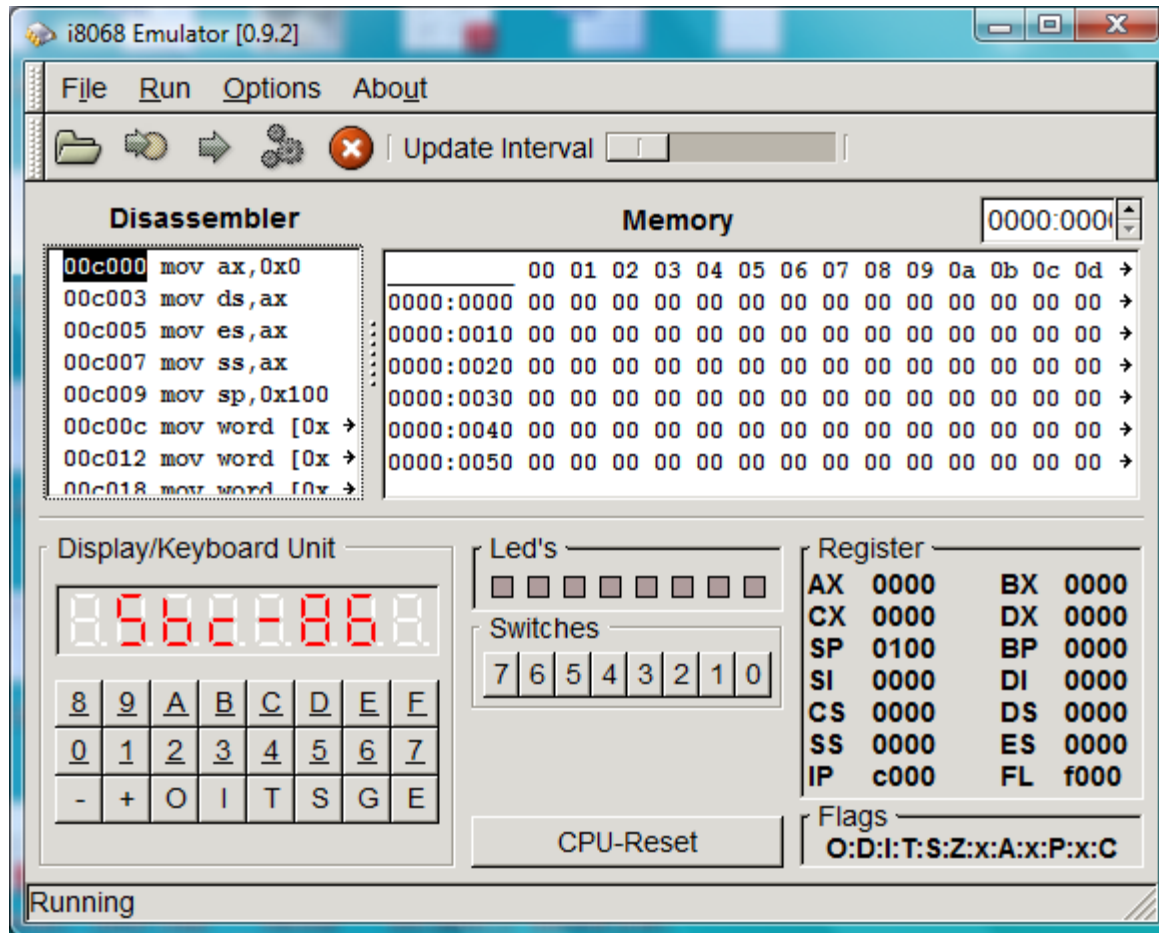


## Crossware Embedded IDE

Compilador cruzado, simulador, ...  
para ARM, 8051, ColdFire,  
68XXX, Z80, 8085, 6809, 6301,  
68HC11

# Ambiente de desenvolvimento

## Simulador para o 8086





## GPsim - Simulador para o PIC

```
gpsim -pp16f627 -s toggle_led.cod toggle_led.asm
```

The screenshot displays the GPsim development environment. On the left, the 'Source Browser' window shows the assembly code for 'toggle\_led.asm'. The code includes initialization routines, a main function, and a 'toggle\_led' function. The current execution point is at the 'GOTO \_00106\_DS\_' instruction. On the right, the '0.21.2' window shows the simulation controls, including buttons for 'step', 'over', 'finish', 'run', 'stop', and 'reset', and a 'Simulation mode' dropdown. Below this, the 'Breadboard [Currently in development]' window shows a schematic of a PIC16F627 microcontroller with its pins labeled (porta2 to porta7, perth0 to perth7). The breadboard is currently empty, and the PIC is highlighted with a black box.

# Ambiente de desenvolvimento

## EdSim51 - Simulador para o 8051

The screenshot displays the EdSim51DI software interface for simulating an 8051 microcontroller. The window title is "EdSim51DI - Version 2.1.1 | adcToDac.asm".

- Registers:** A table on the left shows registers R0-R7, TH0-TL0, TH1-TL1, and PC. The PC register is highlighted with the value 0x0034 and the label "8051".
- Assembly Code:** The central pane shows assembly code for "main:" starting with "SETB IT0" and "SETB EX0".
- Data Memory:** A table at the bottom left shows memory addresses 00-70 with hex values.
- I/O Devices:** A right-hand pane lists various hardware components like "Display-select Decoder", "Keypad", "LEDs", "ADC", and "Motor Sensor".

The screenshot shows the hardware simulation interface with various components:

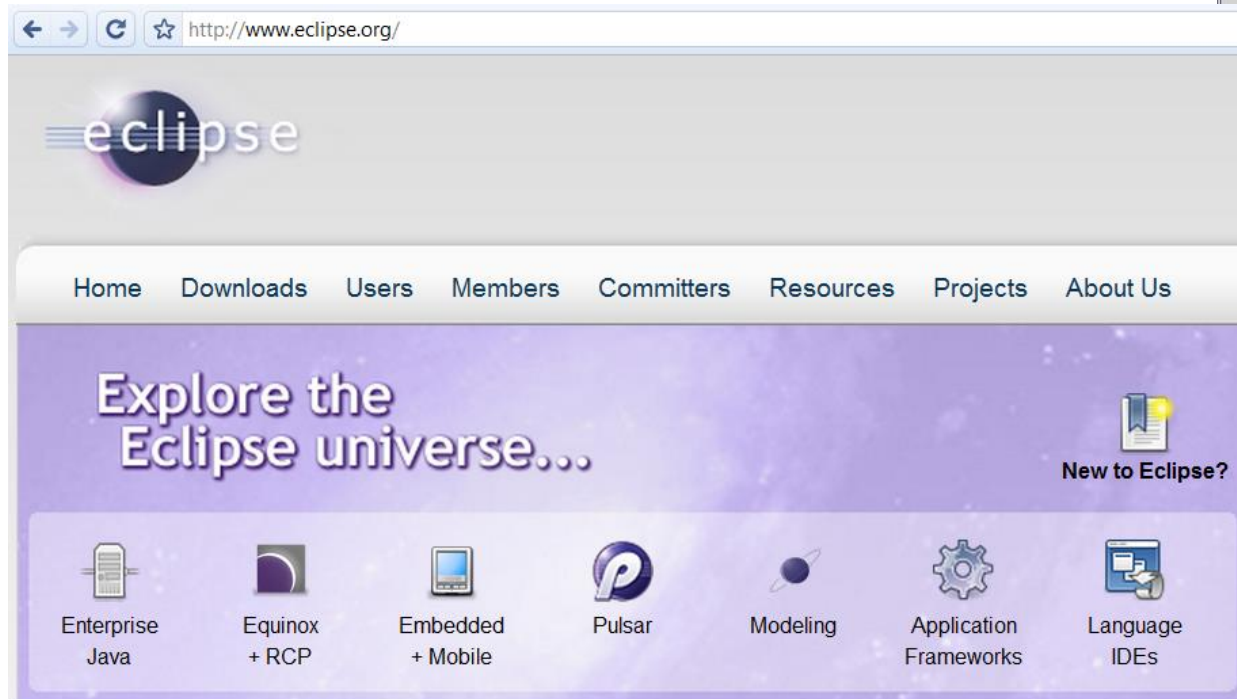
- Keypad:** A 4x3 grid of buttons labeled 1-9, \*, 0, #.
- DAC:** A section with a "Scope" button and a "2.96 V output" display.
- ADC:** A section with a "3.61 V input" display and an "ADC" output showing "11111111".
- Motor:** A section with a "MAX" and "MIN" indicator and a "Motor Enabled" button.
- UART:** A section with "Odd Parity 8-bit UART @ 2400 Baud" and "Rx Reset" / "Tx Send" buttons.
- Other:** A "DI / LD" indicator and a "Scope" button for the DAC.

# Ambiente de desenvolvimento

SDK Android do Google é composto por:

- Emulador para teste do software desenvolvido
- Plugin para IDE do Eclipse
- APIs para Java
- <http://developer.android.com/sdk/>

Conceitualmente, seria um “simulador”





# Celulares com Android



ANDROPHONES.com



# Linguagens de programação

## Assembly

- Baixo nível
- Controle a nível de quantidade de ciclos por instrução/rotina/programa
- Controle total
- Difícil manutenção

## BASIC, Forth

- Interpretadas
- Fáceis de usar
- Lentas

## Linguagem C

- Alto nível
- Abstrai detalhes da arquitetura
- Permite acesso baixo nível (a nível de bit)
- *Assembly in-line*
- Acesso direto a portas de I/O
- Possibilidade de definição de tamanho de palavra (int)
- Grande disponibilidade de ferramentas (compiladores, ...)

## Linguagem C++

- Vantagens do C com orientação a objetos
- Carência de ferramentas

## Programa Exemplo: Loop

```
/* pulses pin PORTB<3>
eight times */

pulse:
    movlw    0x08
    movwf   counter

pulse_lp0:
    bsf     PORTB, 3
    bcf     PORTB, 3
    decfsz  counter, F
    goto    pulse_lp0
    return
```

Código Assembly

```
/* pulses pin PORTB<3>
eight times */

void pulse()
{
    int i;
    for (i=0; i<8; i++){
        output_high(PIN_B3);
        output_low(PIN_B3);
    }
    return;
}
```

Código C



## Ineficiência dos compiladores

```
/* pulses pin PORTB<3> eight
times */
```

```
0000: movlw  0x8
0001: movwf  0x20
0002: bsf    0x6,0x3
0003: bcf    0x6,0x3
0004: decfsz 0x20
0005: goto   002
```

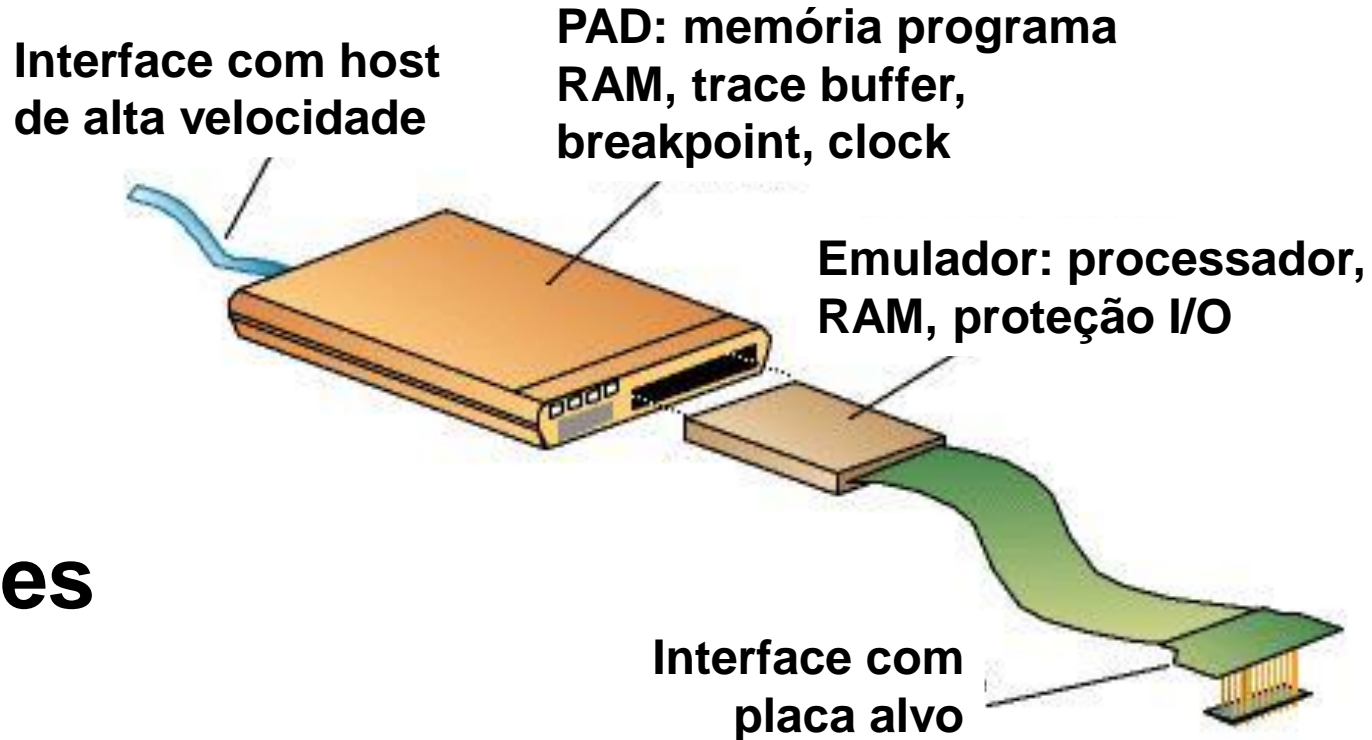
Código assembly  
do desenvolvedor

```
/* pulses pin PORTB<3> eight times
*/
```

```
0005: CLRF   21
0006: MOVF   21,W
0007: SUBLW  07
0008: BTFSS  03,0
0009: GOTO   014
000A: BSF    03,5
000B: BCF    06,3
000C: BCF    03,5
000D: BSF    06,3
000E: BSF    03,5
000F: BCF    06,3
0010: BCF    03,5
0011: BCF    06,3
0012: INCF   21,F
0013: GOTO   006
```

Código assembly gerado  
pelo compilador C

# Emulador – *In-Circuit Emulator*



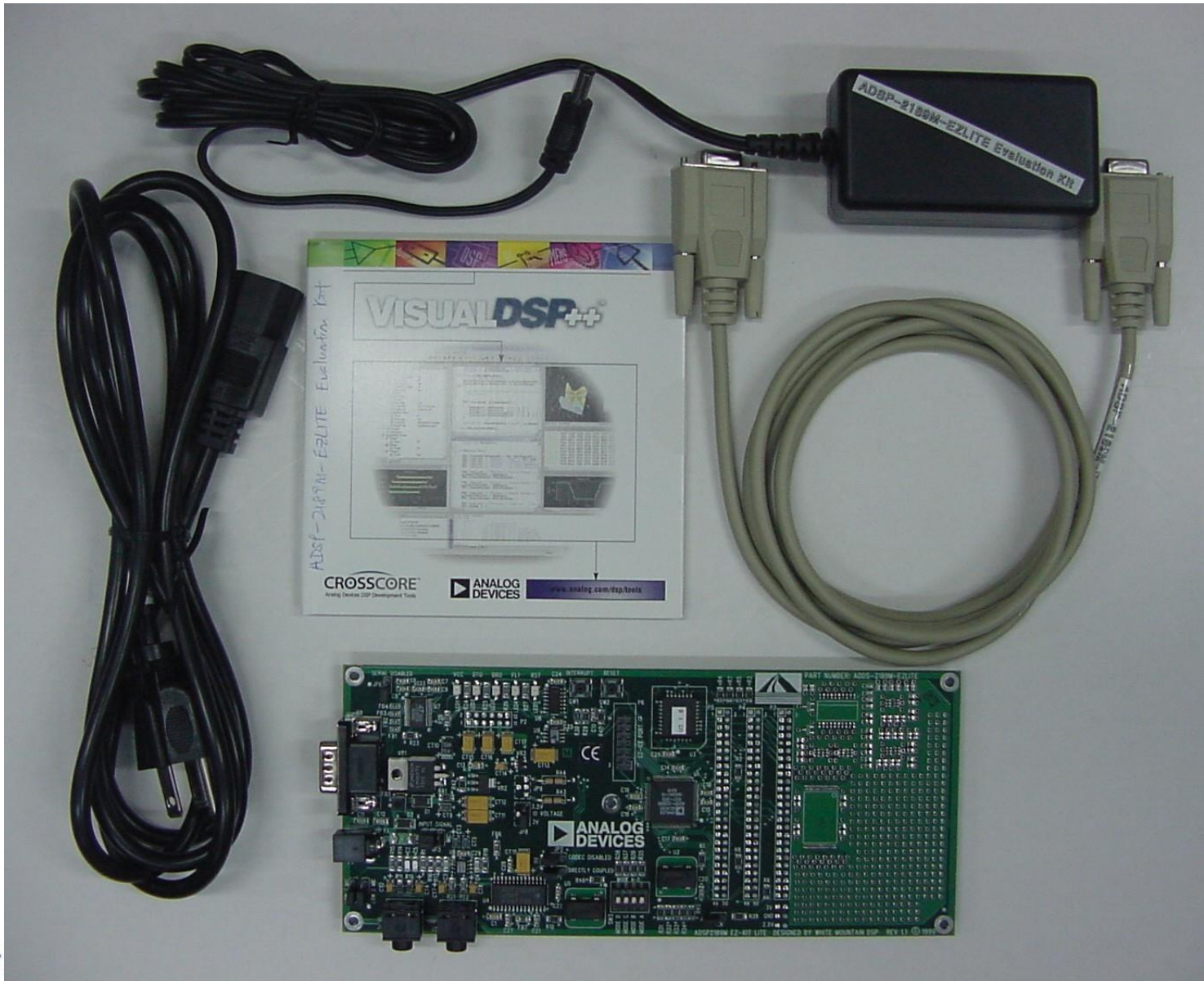
## Emuladores

- Hardware contendo processador alvo
- Possibilita teste “real” do software, antes de concluído o projeto da placa
- Placa alvo é testada, sem o processador, utilizando pads do emulador no socket destino
- Ferramenta com alto grau de controlabilidade e observabilidade
- Depuração em tempo real

# Emulador – *In-Circuit Emulator*

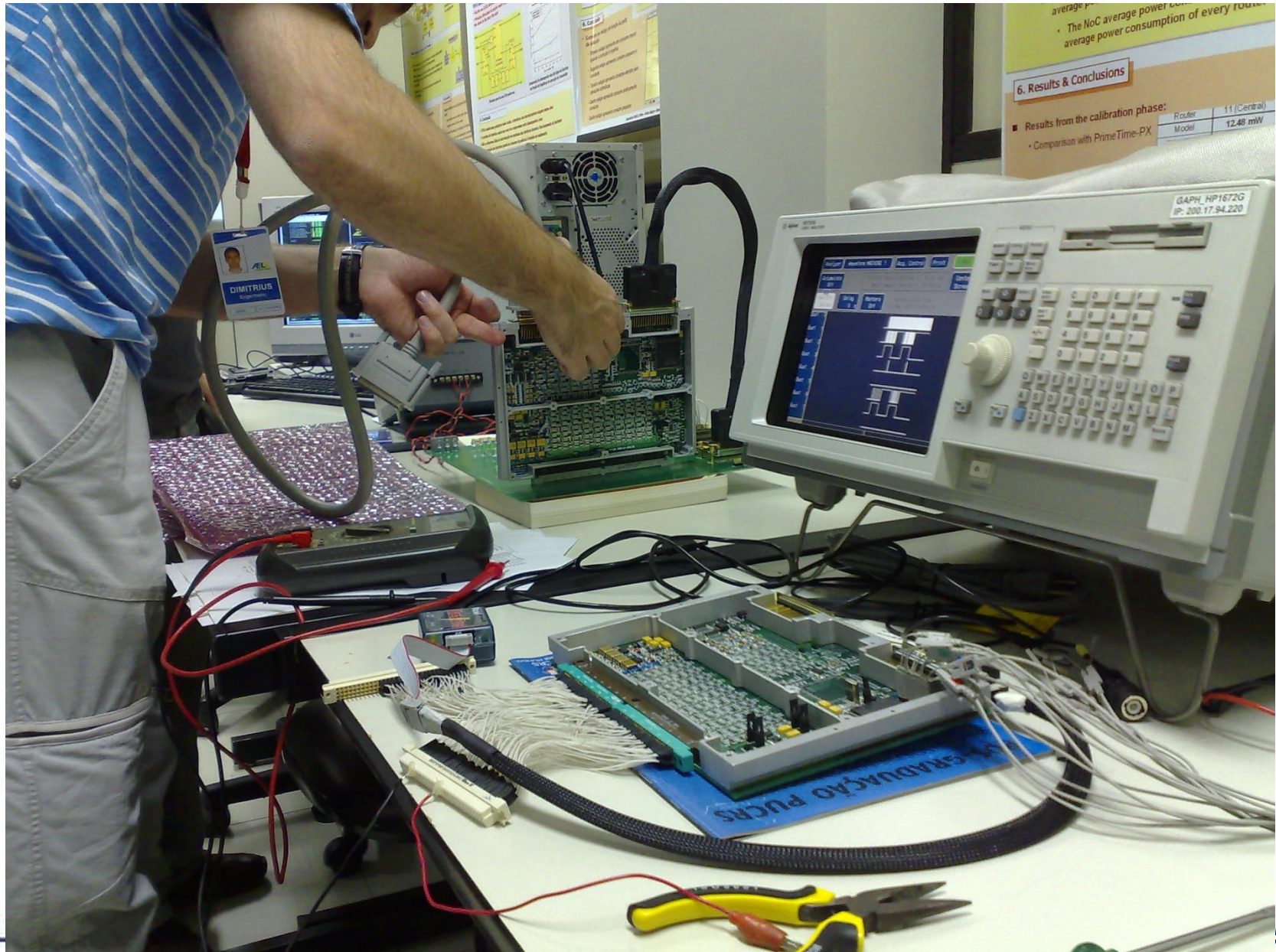


# Emulador – *In-Circuit Emulator*



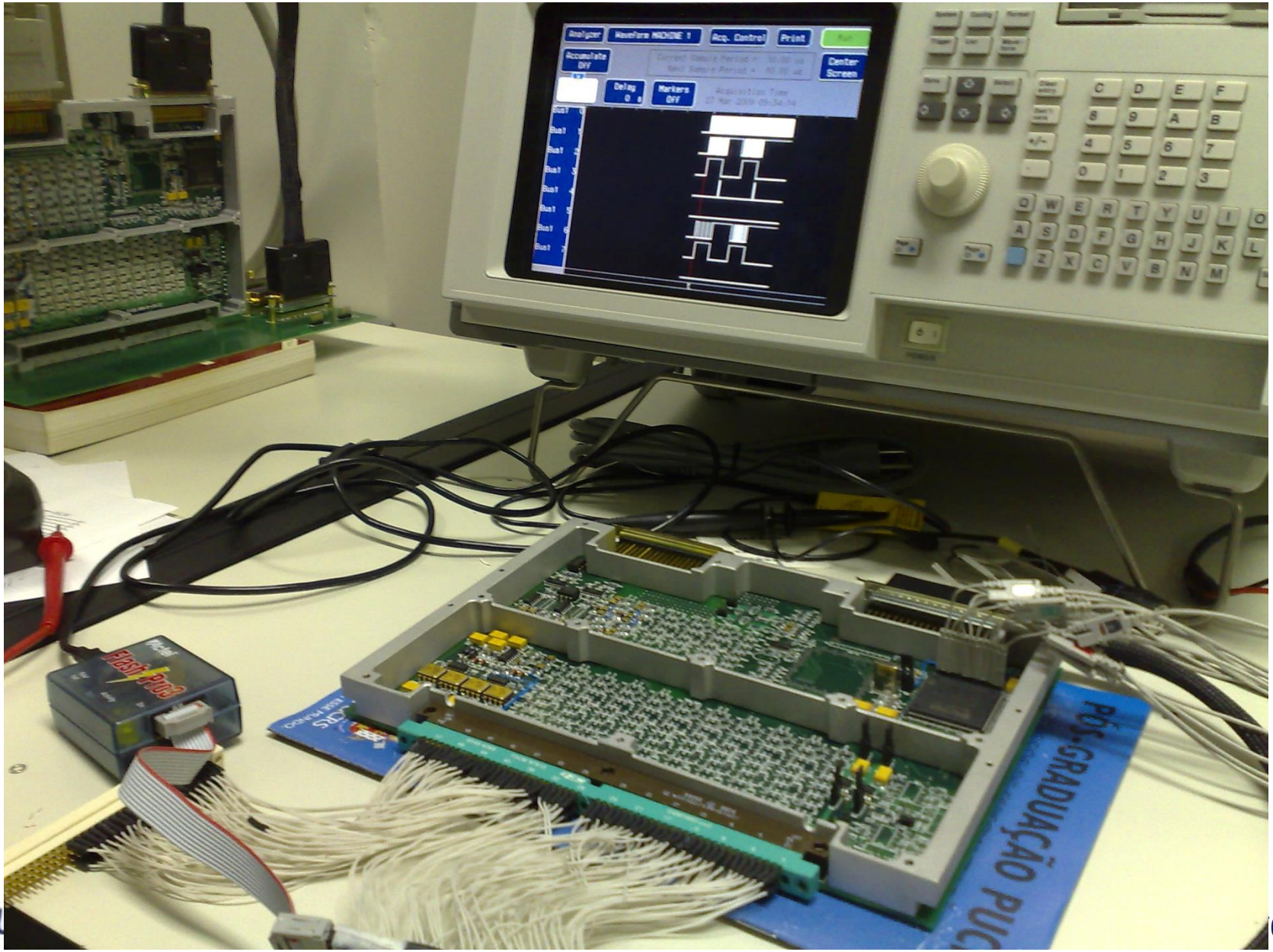


# Analizador Lógico





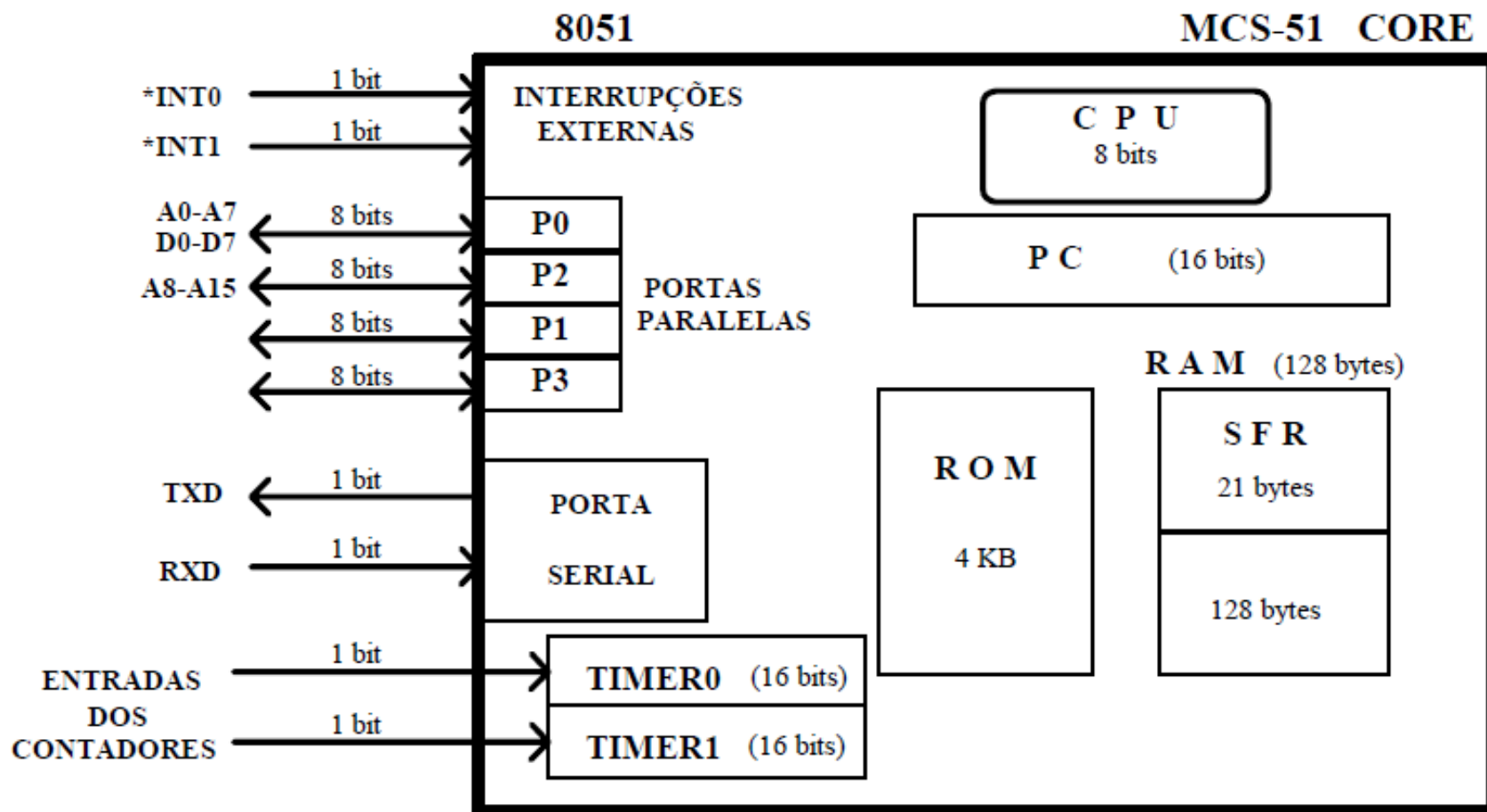
# Analizador Lógico



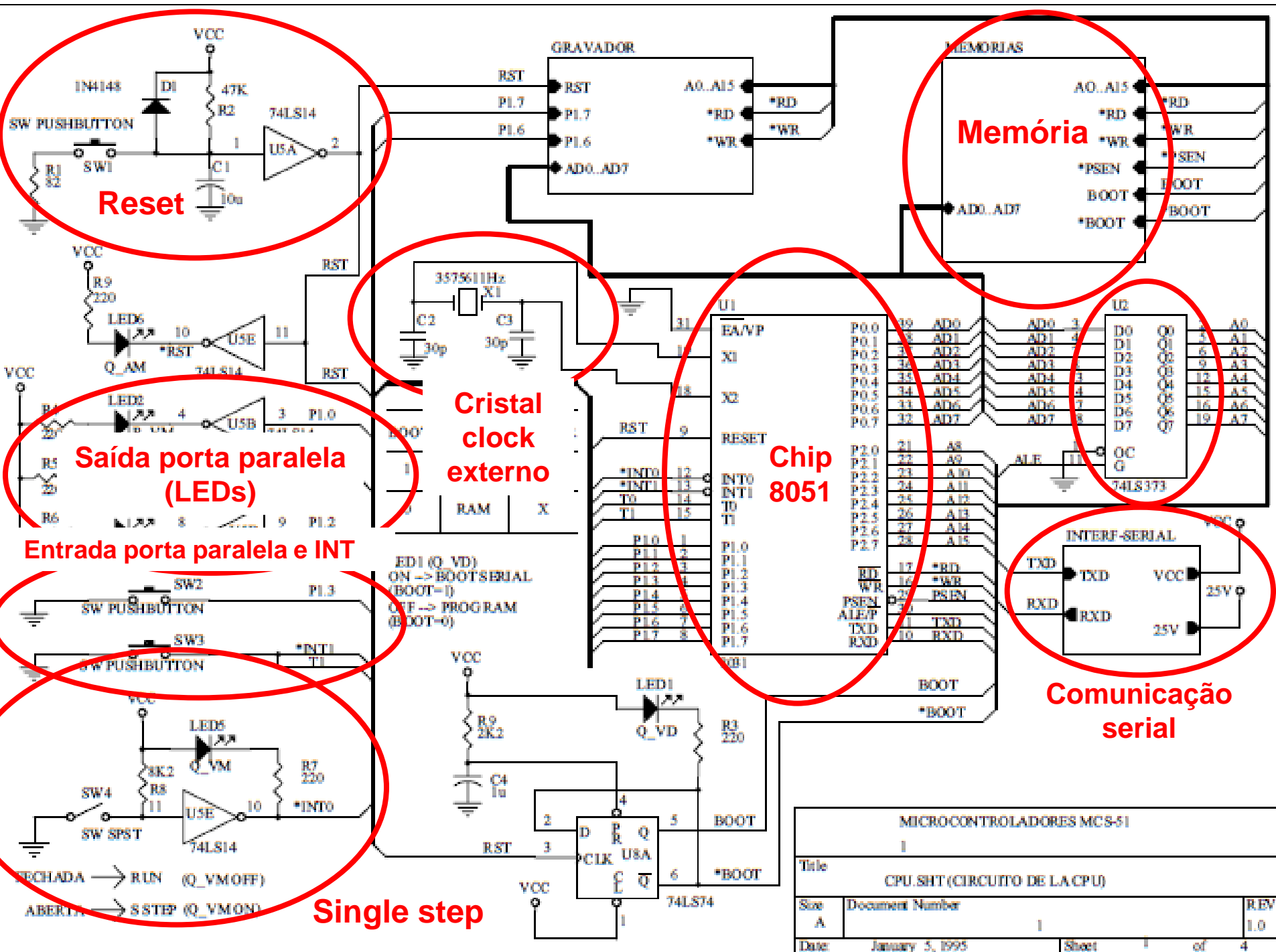
# Plataforma 8051



# 8051



- 5 Interrupções (2 externas, 2 dos timers/counters e 1 da porta serial)



**Reset**

**Memória**

**Cristal clock externo**

**Chip 8051**

**Saída porta paralela (LEDs)**

**Entrada porta paralela e INT**

**Comunicação serial**

**Single step**

|                              |                 |       |        |
|------------------------------|-----------------|-------|--------|
| MICROCONTROLADORES MCS-51    |                 |       |        |
| 1                            |                 |       |        |
| Title                        |                 |       |        |
| CPU_SHT (CIRCUITO DE LA CPU) |                 |       |        |
| See                          | Document Number | REV   |        |
| A                            | 1               | 1.0   |        |
| Date                         | January 5, 1995 | Sheet | 1 of 4 |

# Ferramentas para 8051

---

- IDE Keil
- Compilador cruzado sdcc
- Simuladores
- Emuladores

Bascom



[www.mcselec.com](http://www.mcselec.com)

System Clock (MHz)  Update Freq.

SBUF

|              |           |     |      |         |
|--------------|-----------|-----|------|---------|
| R/W          | TH0       | TL0 | R7   | B       |
| 0x00 0x00    | 0x00 0x00 |     | 0x00 | 0x00    |
| RXD TXD      | TMOD      |     | R6   | ACC     |
| 1 1          | 0x00      |     | 0x00 | 0x00    |
| SCON         | TCON      |     | R5   | PSW     |
| 0x00         | 0x01      |     | 0x00 | 0x00    |
| pins bits    | TH1 TL1   |     | R4   | IP      |
| 0xFF 0xFF P3 | 0x00 0x00 |     | 0x00 | 0x01    |
| 0xFF 0xFF P2 |           |     | R3   | IE      |
| 0xFF 0xFF P1 |           |     | R2   | PCON    |
| 0xFF 0xFF P0 |           |     | R1   | DPH     |
|              | PC        |     | R0   | DPL     |
|              | 0x0034    |     |      | 0x00    |
|              |           |     |      | SP 0x07 |

8051

Data Memory

| addr | 0x00 | 0x00 | value |
|------|------|------|-------|
| 00   | 00   | 00   | 00    |
| 10   | 00   | 00   | 00    |
| 20   | 00   | 00   | 00    |
| 30   | 00   | 00   | 00    |
| 40   | 00   | 00   | 00    |
| 50   | 00   | 00   | 00    |
| 60   | 00   | 00   | 00    |
| 70   | 00   | 00   | 00    |

Remove All Breakpoints

```

RST Step Run New Load Save Copy Paste
Executed 0x0032: SETB 0A8H
0000| JMP main ; jump to the
ORG 3 ; external 0
0003| JMP ext0ISR ; jump to the
ORG 0BH ; timer 0 into
000B*| JMP timer0ISR ; jump to time
ORG 30H ; main program
main:
0030| SETB IT0 ; set externa
0032| SETB EX0 ; enable exte
0034| CLR P0.7 ; enable DAC V
0036| MOV TMOD, #2 ; set timer 0
0039| MOV TH0, #-200 ; | put -200
; | with syste
003C| MOV TL0, #-200 ; | put the s
; | 236 (256
003F| SETB TR0 ; start timer
0041| SETB ET0 ; enable time
0043| SETB EA ; set the global
    
```

- P0.7 1 Display-select Decoder CS|DAC WR
- P0.6 1 Keypad Column 2
- P0.5 1 Keypad Column 1
- P0.4 1 Keypad Column 0
- P0.3 1 Keypad Row 3
- P0.2 1 Keypad Row 2
- P0.1 1 Keypad Row 1
- P0.0 1 Keypad Row 0
- P1.7 1 LED 7|Seg. dp|DAC DB7|LCD DB7
- P1.6 1 LED 6|Seg. g|DAC DB6|LCD DB6
- P1.5 1 LED 5|Seg. f|DAC DB5|LCD DB5
- P1.4 1 LED 4|Seg. e|DAC DB4|LCD DB4
- P1.3 1 LED 3|... d|..DB3|..DB3|.. RS
- P1.2 1 LED 2|... c|..DB2|..DB2|LCD E
- P1.1 1 LED 1|Seg. b|DAC DB1|LCD DB1
- P1.0 1 LED 0|Seg. a|DAC DB0|LCD DB0
- P2.7 1 SW 7|ADC DB7
- P2.6 1 SW 6|ADC DB6
- P2.5 1 SW 5|ADC DB5
- P2.4 1 SW 4|ADC DB4
- P2.3 1 SW 3|ADC DB3
- P2.2 1 SW 2|ADC DB2
- P2.1 1 SW 1|ADC DB1
- P2.0 1 SW 0|ADC DB0
- P3.7 1 ADC RD|Comparator Output
- P3.6 1 ADC WR
- P3.5 1 Motor Sensor
- P3.4 1 Display-select Input 1
- P3.3 1 AND Gate Output|Display-se..t 0
- P3.2 1 ADC INTR
- P3.1 1 Motor Control Bit 1|Ext. UART Rx
- P3.0 1 Motor Control Bit 0|Ext. UART Tx

DI / LD

7 6 5 4 3 2 1 0

2.96 V output

Scope DAC

1 2 3 AND Gate Enabled

4 5 6 Key Bounce Disabled

7 8 9 Pulse

\* 0 #

U Odd Parity 8-bit UART @ 2400 Baud

Rx Rx Reset

Tx Tx Send


3.61 V input

11111111

ADC

MAX MIN Motor Enabled

BF 0 AC 0x00 IR 0x00 DR 0x00



# Ferramentas para 8051

## Emulador para 8051

**Facilidades:** 8051 da Atmel com memória flash embarcada. Operações de tempo real até 24MHz. Clock built in ou outro clock conectado a placa em emulação.

**Frequências:** 4 MHz, 4,608 MHz, 8 MHz, 9,216 MHz, 10 MHz, 12 MHz, 16 MHz, 18,432MHz, 20 MHz e 24MHz.

Até 60 KB de memória de programa.

Até 256 Bytes de memória de dados interna. Dispositivos DIP de 20 e 40 pinos.

Conexão ao host via USB. IDE para Windows XP.

### Dispositivos:

AT89C1051, AT89C1051U, AT89C2051, AT89C4051, AT89C51, AT89C51RC, AT89S51, AT89C52, AT89S52, AT89C55, AT89C55WD, AT89S53, AT89S8252, AT89S8253

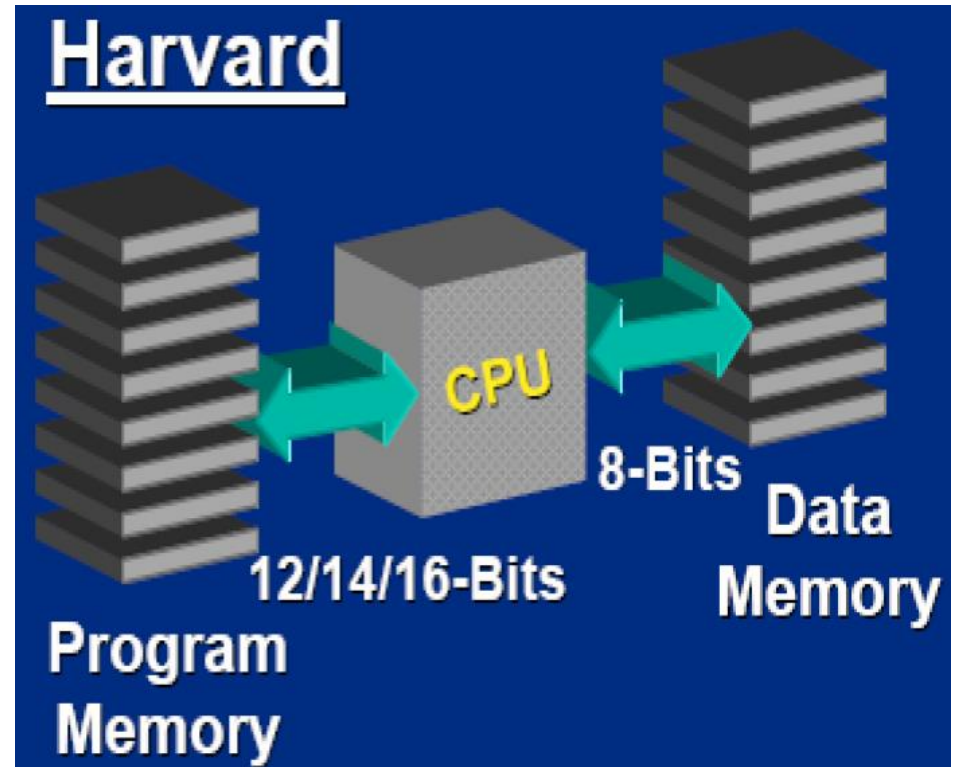


# Plataforma PIC



# PIC

- Fabricante Microchip
- RISC
- Série 16 possui 35 instruções
- Arquitetura Harvard
- Barramentos separados para memória de dados e memória de programa.



## PIC16F88

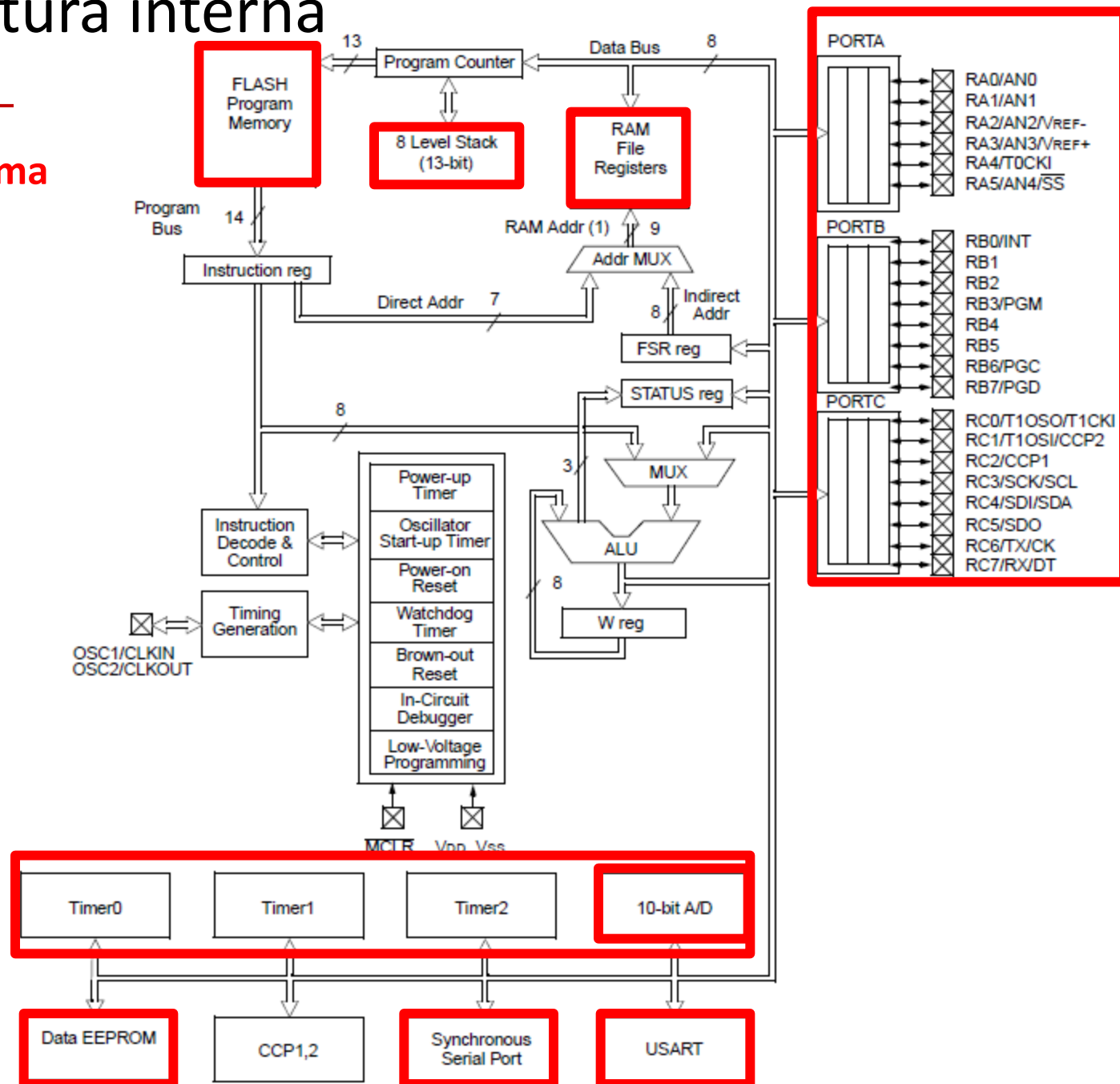
- 8 bit
- Memória: 68 bytes RAM/68 bytes EEPROM
- 18 Pinos: 13 pinos de I/O





# PIC – arquitetura interna

- Memória de programa
- Pilha para chamada de sub-rotinas
- Portas (I/O)
- Memória de dados
- Timers
- Serial síncrona
- Serial assíncrona
- Conversor A/D



# PIC

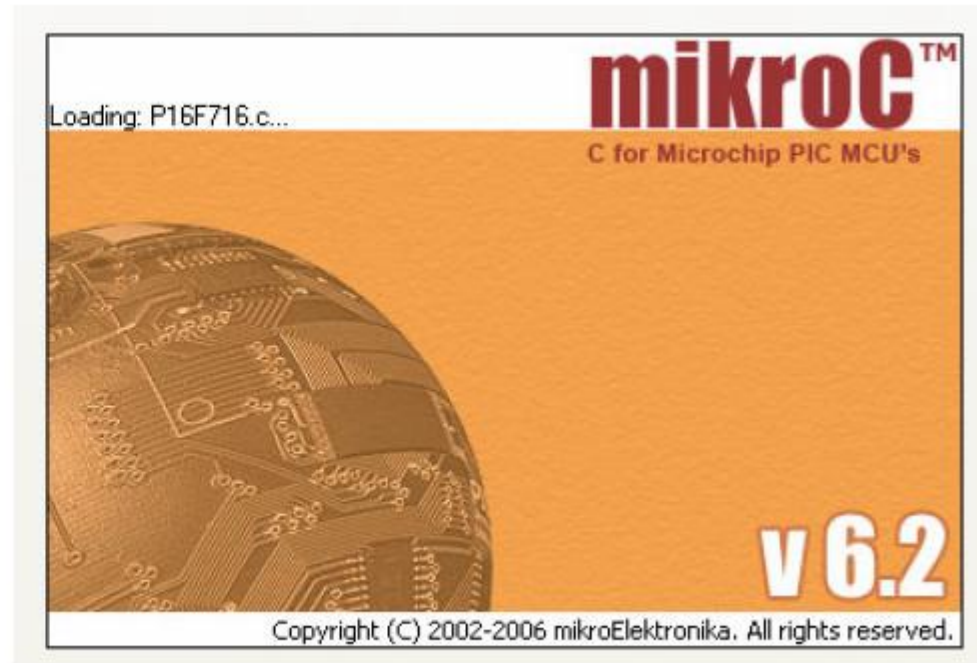
|                                  |   | PIC12CXXX | PIC14000 | PIC16C5X | PIC16C6X | PIC16CXXX | PIC16F62X | PIC16C7X | PIC16C7XX | PIC16C8X | PIC16F8XX |
|----------------------------------|---|-----------|----------|----------|----------|-----------|-----------|----------|-----------|----------|-----------|
| <b>Software Tools</b>            | MPLAB® Integrated Development Environment       | ✓         | ✓        | ✓        | ✓        | ✓         | ✓         | ✓        | ✓         | ✓        | ✓         |
|                                  | MPLAB® C17 Compiler                             |           |          |          |          |           |           |          |           |          |           |
|                                  | MPLAB® C18 Compiler                             |           |          |          |          |           |           |          |           |          |           |
|                                  | MPASM/MPLINK                                    | ✓         | ✓        | ✓        | ✓        | ✓         | ✓         | ✓        | ✓         | ✓        | ✓         |
| <b>Emulators</b>                 | MPLAB®-ICE                                      | ✓         | ✓        | ✓        | ✓        | ✓         | ✓**       | ✓        | ✓         | ✓        | ✓         |
|                                  | PICMASTER/PICMASTER-CE                          | ✓         | ✓        | ✓        | ✓        | ✓         |           | ✓        | ✓         | ✓        |           |
|                                  | ICEPIC™ Low-Cost In-Circuit Emulator            | ✓         |          | ✓        | ✓        | ✓         |           | ✓        | ✓         | ✓        |           |
| <b>Debugger</b>                  | MPLAB®-ICD In-Circuit Debugger                  |           |          |          | ✓*       |           |           | ✓*       |           |          | ✓         |
| <b>Programmers</b>               | PICSTART®Plus Low-Cost Universal Dev. Kit       | ✓         | ✓        | ✓        | ✓        | ✓         | ✓**       | ✓        | ✓         | ✓        | ✓         |
|                                  | PRO MATE® II Universal Programmer               | ✓         | ✓        | ✓        | ✓        | ✓         | ✓**       | ✓        | ✓         | ✓        | ✓         |
| <b>Demo Boards and Eval Kits</b> | SIMICE  | ✓         |          | ✓        |          |           |           |          |           |          |           |
|                                  | PICDEM-1  |           |          | ✓        |          | ✓         |           | ✓†       |           | ✓        |           |
|                                  | PICDEM-2  |           |          |          | ✓†       |           |           | ✓†       |           |          |           |
|                                  | PICDEM-3  |           |          |          |          |           |           |          |           |          |           |
|                                  | PICDEM-14A                                      |           | ✓        |          |          |           |           |          |           |          |           |
|                                  | PICDEM-17                                       |           |          |          |          |           |           |          |           |          |           |
|                                  | KEELOQ® Evaluation Kit                          |           |          |          |          |           |           |          |           |          |           |
|                                  | KEELOQ Transponder Kit                          |           |          |          |          |           |           |          |           |          |           |
|                                  | microID™ Programmer's Kit                       |           |          |          |          |           |           |          |           |          |           |
|                                  | 125 kHz microID Developer's Kit                 |           |          |          |          |           |           |          |           |          |           |
|                                  | 125 kHz Anticollision microID Developer's Kit   |           |          |          |          |           |           |          |           |          |           |
|                                  | 13.56 MHz Anticollision microID Developer's Kit |           |          |          |          |           |           |          |           |          |           |
|                                  | MCP2510 CAN Developer's Kit                     |           |          |          |          |           |           |          |           |          |           |

# PIC – Ferramentas

---

mikro C

[www.mikroe.com](http://www.mikroe.com)



# PIC: etapas de desenvolvimento

---

## 1. Escrever o programa

- MPLAB
- C ou Assembly

## 2. Compilar o programa

- CCS PCM

## 3. Programa para gravação

- Arquivo .HEX
- Usar PICSTART e MPLAB

## 4. Colocar PIC no gravador de EPROM

- Observar pinagem

## 5. Alimentação da placa

- Fonte/bateria de 9V

## 6. Depurar o programa

- Raramente funciona na primeira tentativa

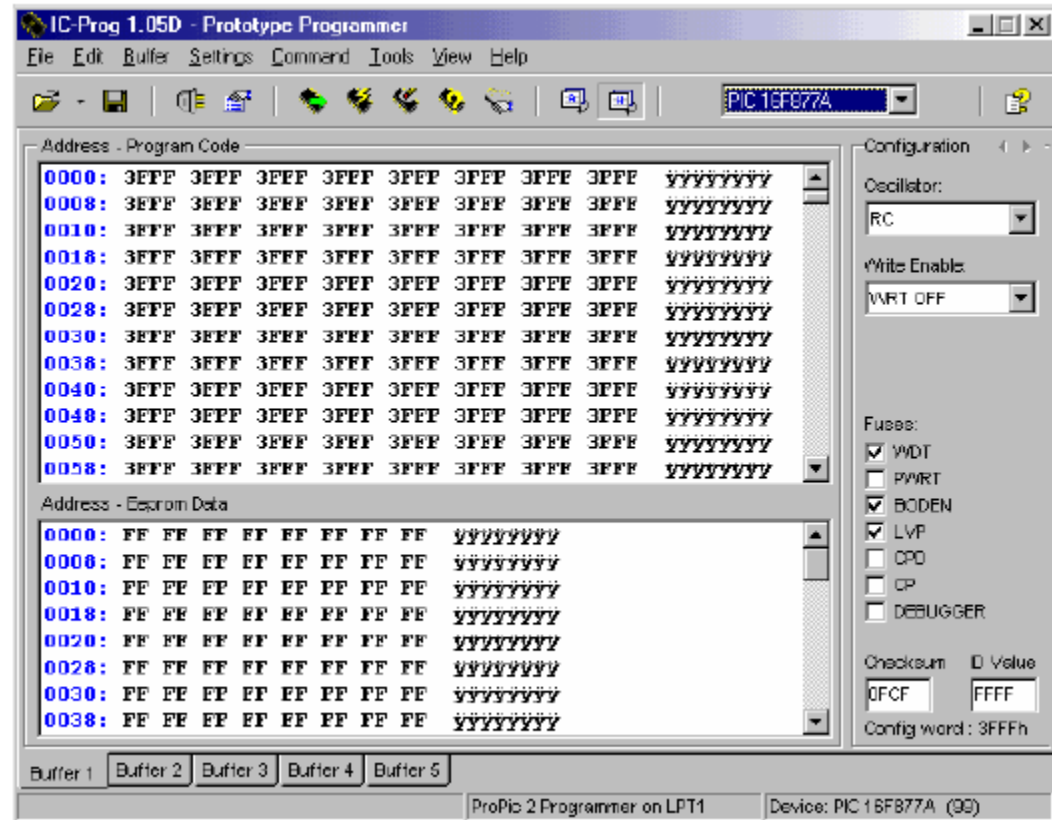
## 7. Repetir a partir do passo 1



# Gravação de PIC e 8051

IcProg

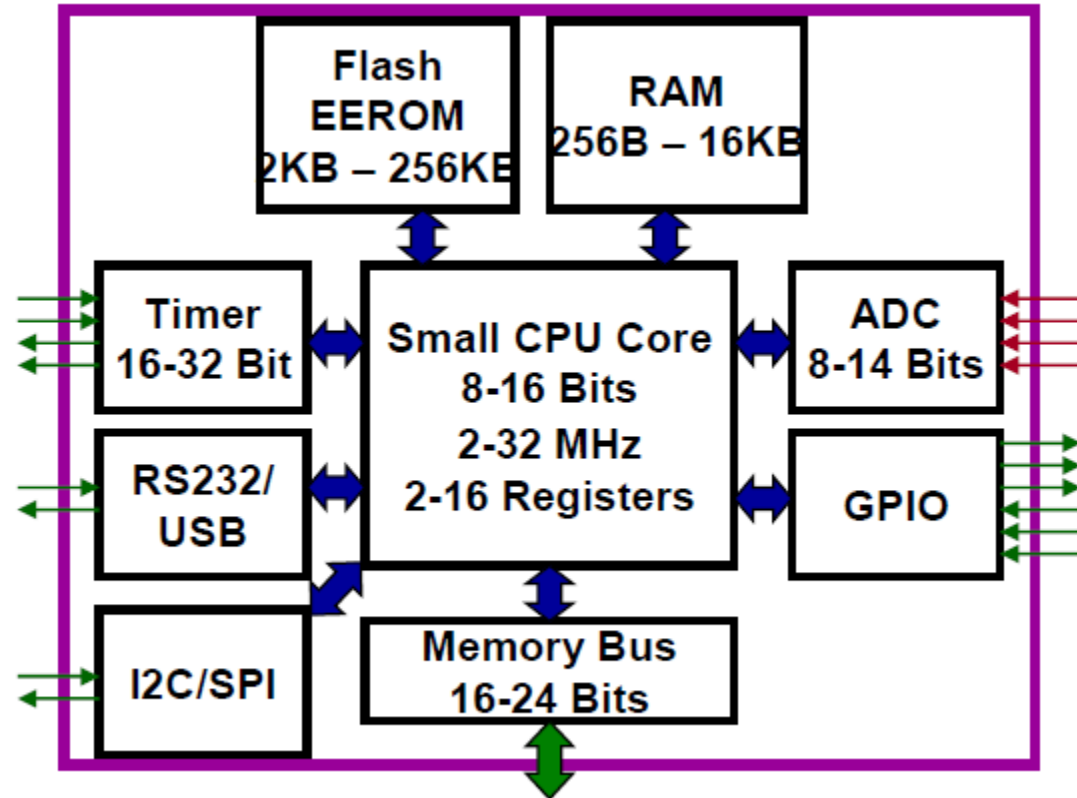
[www.icprog.com](http://www.icprog.com)



# Plataforma AVR

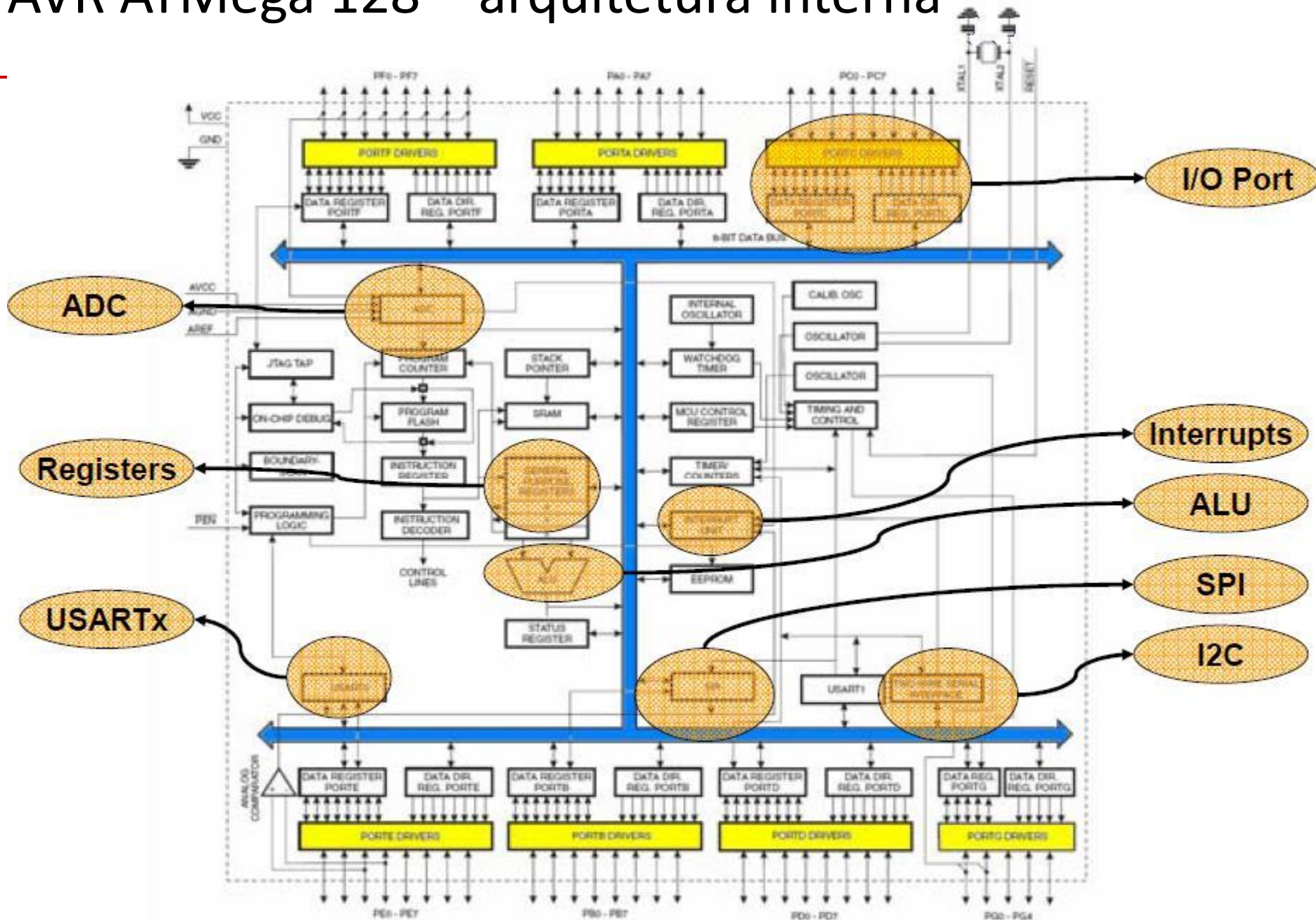


# AVR – arquitetura interna



- Core RISC com ~100 instruções
- Velocidades de clock modestas (4-16 MHz)
- Barramento de 8 bits e 32 registradores de uso geral de 8 bits
- Flash programável *in-circuit* (~1000 ciclos)
- Pequena quantidade de EEPROM e SRAM
- Diversos periféricos embarcados (*UART, SPI, ADC, PWM, WDT*)

# AVR ATMega 128 – arquitetura interna



# AVR – IDE

---



[www. hpinfotech.com](http://www.hpinfotech.com)



HP InfoTech

**CodeVisionAVR**

C Compiler, Integrated Development Environment,  
Automatic Program Generator and In-System Programmer  
for the Atmel AVR Family of Microcontrollers

Version 1.25.6 Evaluation

© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.

<http://www.hpinfotech.com>

Freeware, for evaluation and non-commercial use only

# AVR vs. PIC

---

## PIC

- Disponibilidade em encapsulamento DIP para uso direto em placas de prototipação
- Valores na ordem de US\$1 a US\$9
- Desvantagem: Custo das ferramentas – Compilador ~US\$200; Debug ~US\$150.

## AVR

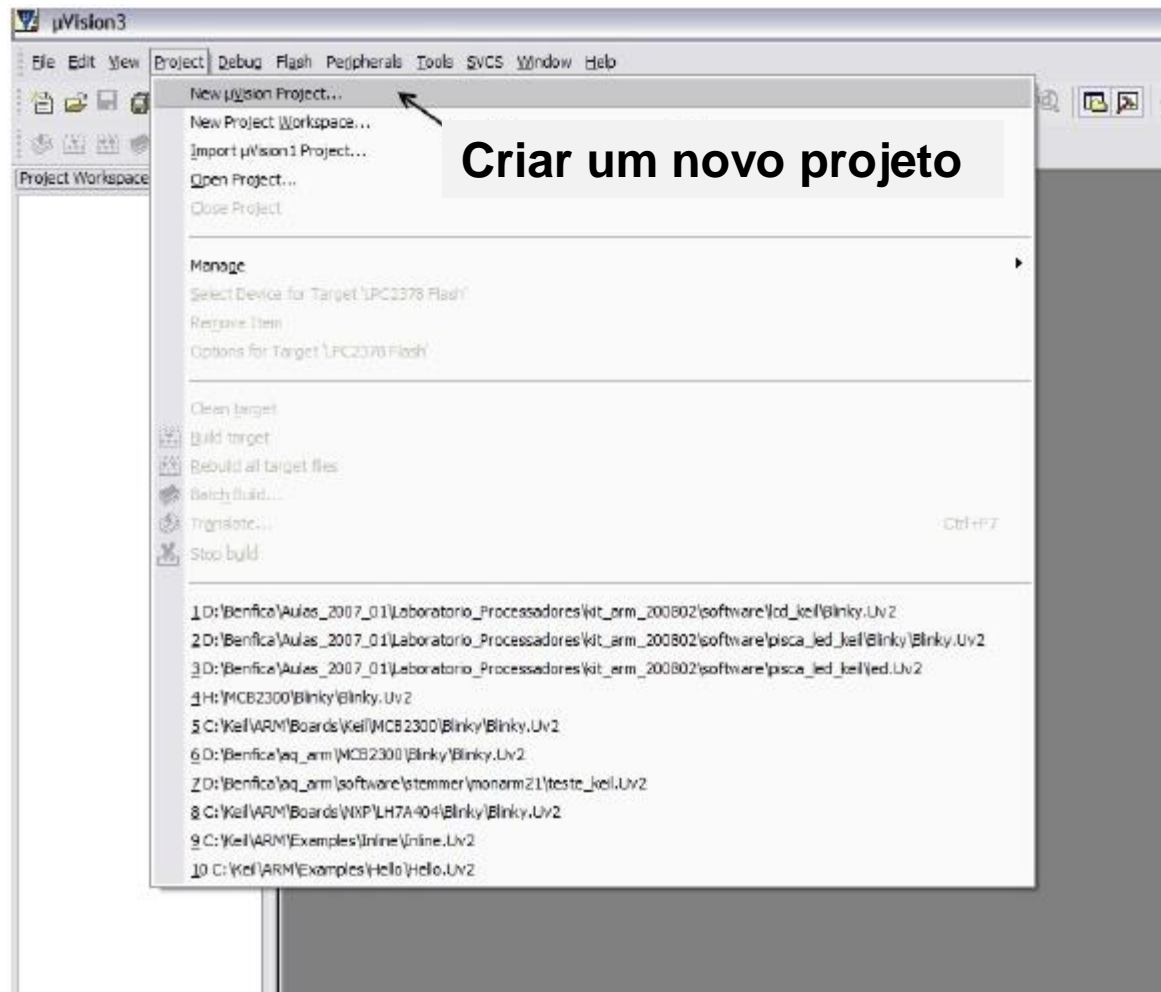
- Ferramentas gratuitas (gcc)
- IDE disponível para Windows, Mac e Linux, incluindo debug
- AVR-Dragon da Atmel custa em torno de US\$50 e pode ser utilizado para programação e depuração
- Desvantagem: poucas famílias de dispositivos disponíveis (pouca variedade) ao se comparar com o PIC

# Tutorial Keil

# Tutorial: criação de projeto e geração de binário para ARM

## “Criação de novo projeto”

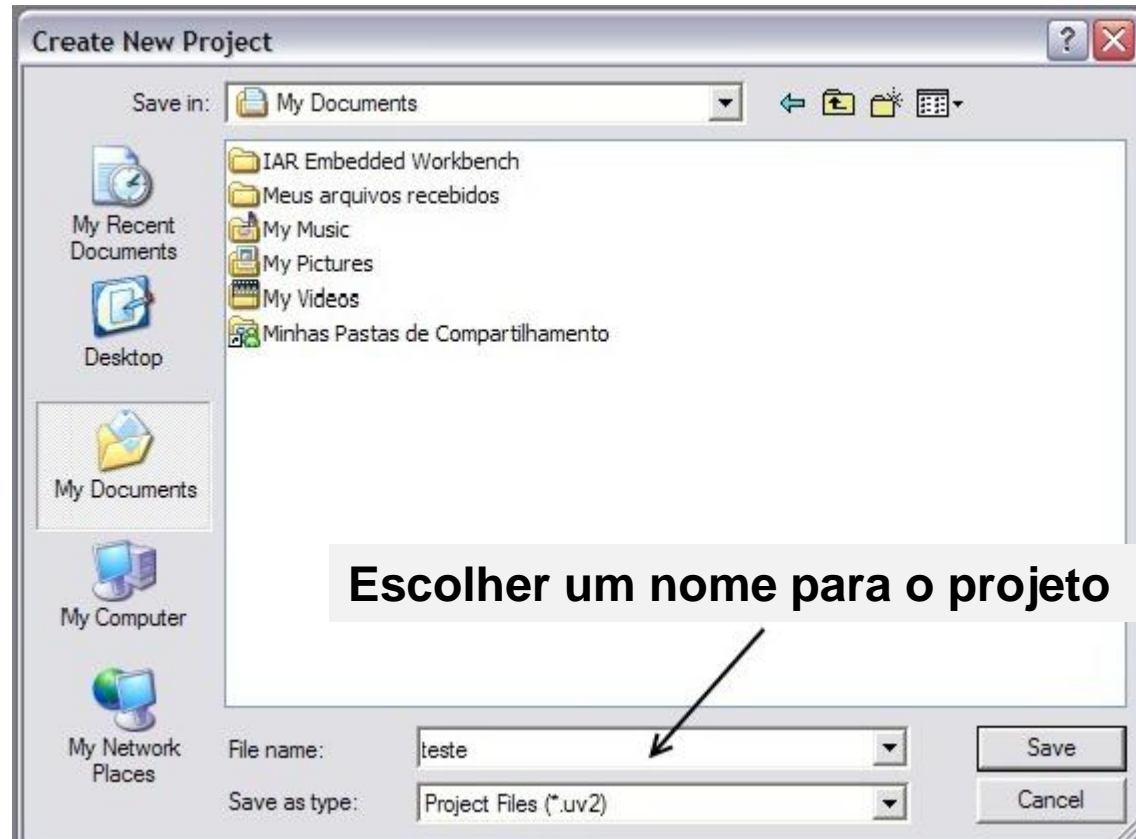
[Keil](#) – IDE para diversas arquiteturas (ARM, 8051, PIC, ...)





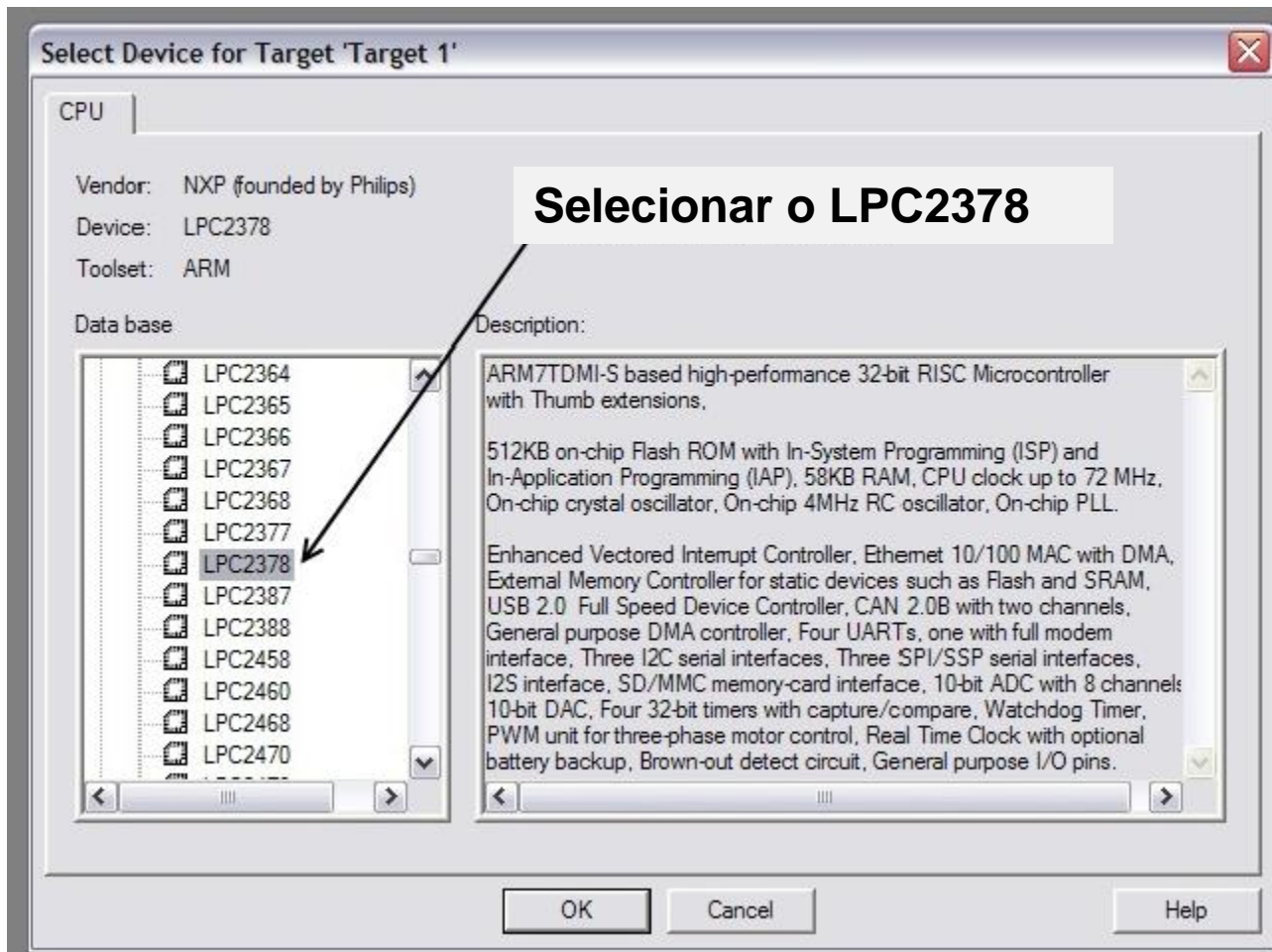
# Tutorial: criação de projeto e geração de binário para ARM

## “Escolha do diretório”



# Tutorial: criação de projeto e geração de binário para ARM

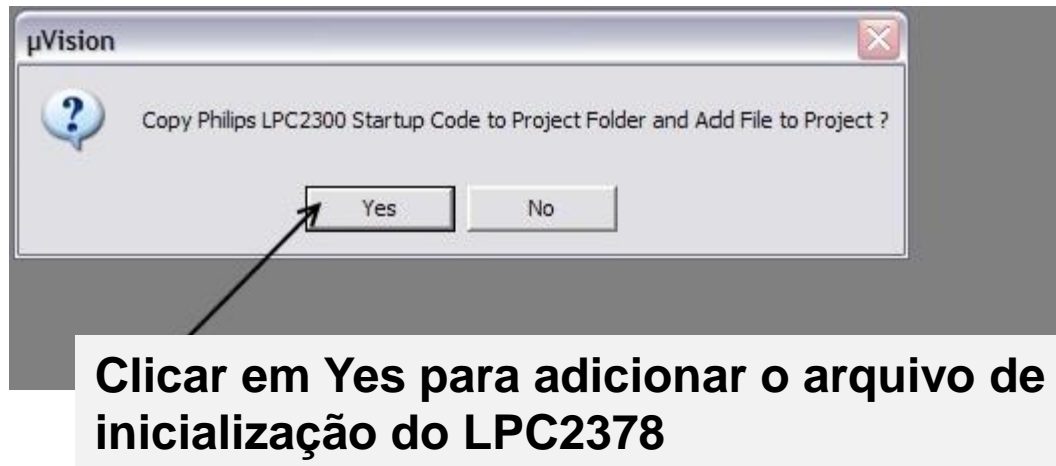
## “Escolha do device ARM7”



# Tutorial: criação de projeto e geração de binário para ARM

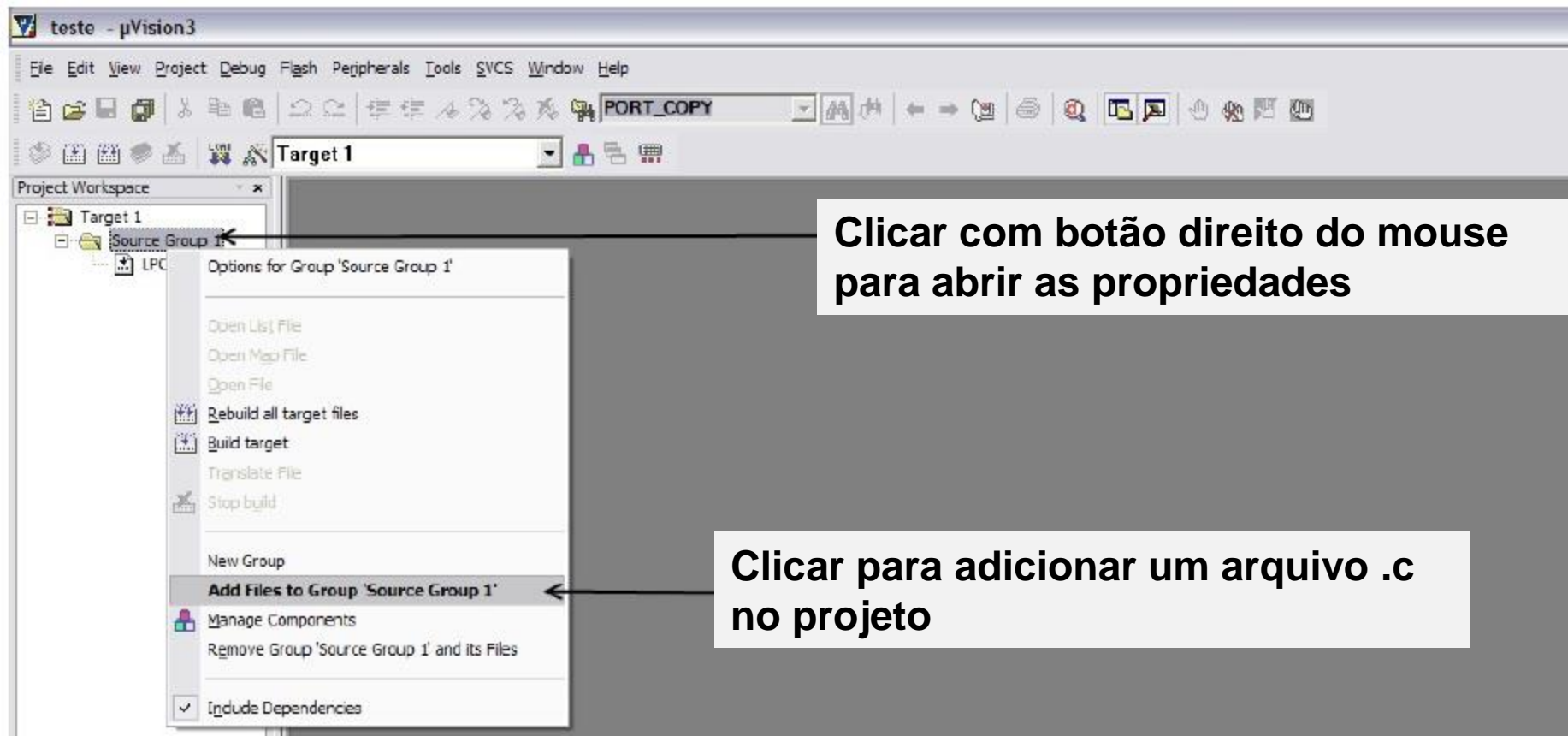
## “Adicionar inicialização”

---



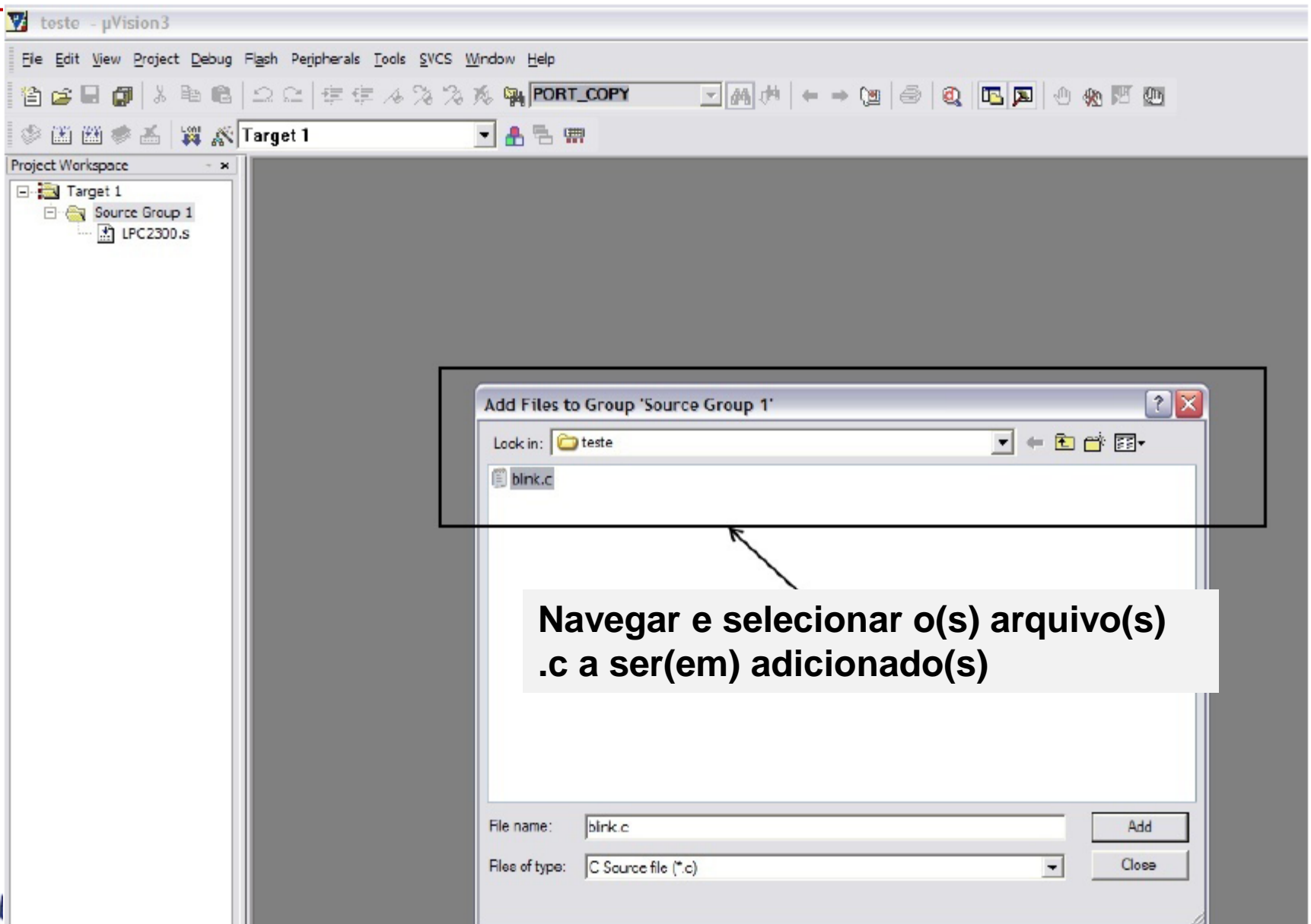
# Tutorial: criação de projeto e geração de binário para ARM

## “Criar um novo arquivo”



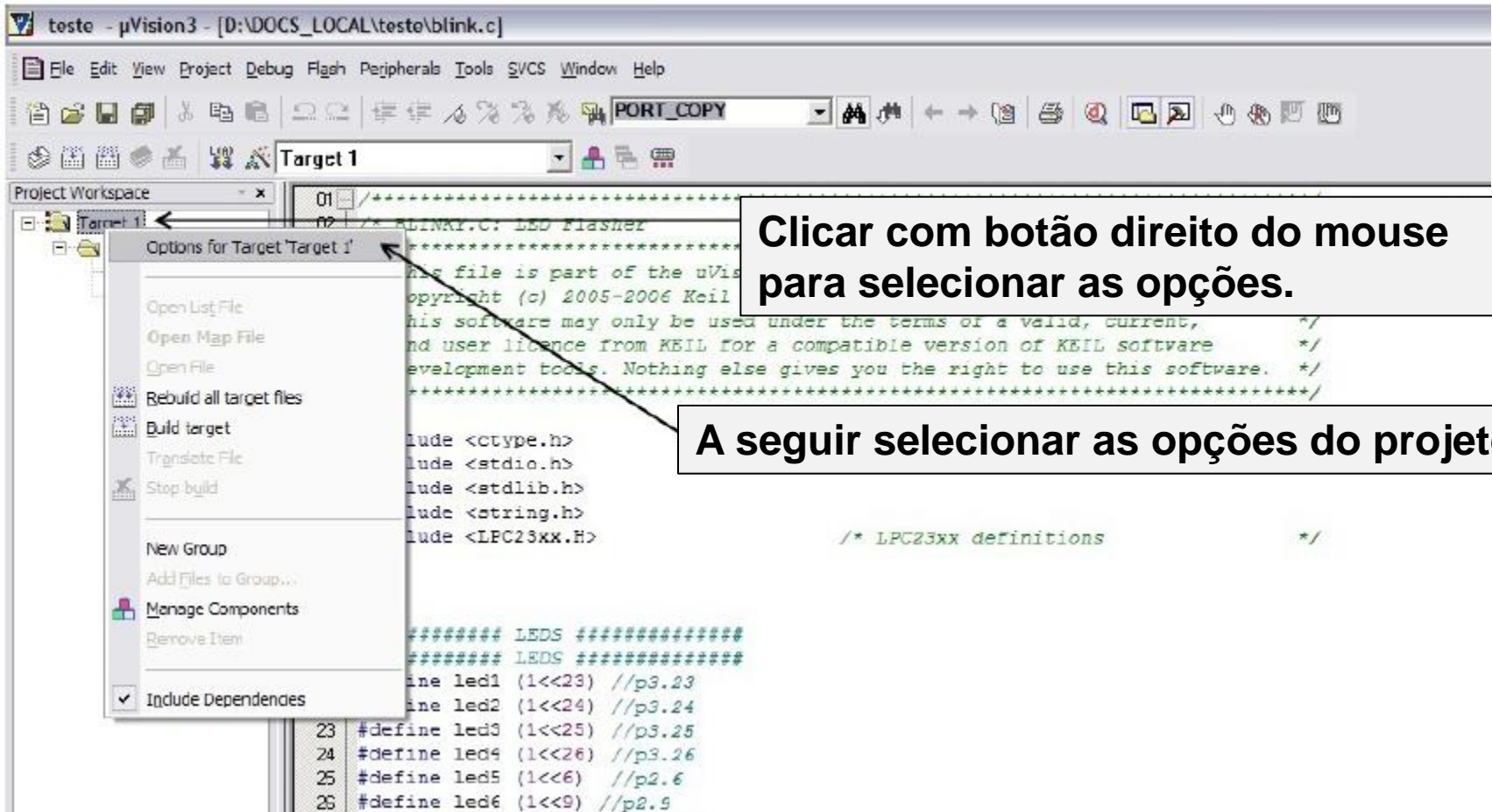
# Tutorial: criação de projeto e geração de binário para ARM

## “Selecionar arquivo”



# Tutorial: criação de projeto e geração de binário para ARM

## “Propriedades do projeto”





# Tutorial: criação de projeto e geração de binário para ARM

## “Propriedades do projeto”

The screenshot shows the Keil uVision3 IDE interface. The main window displays a C source file named 'blink.c' with the following content:

```
01 //*****  
02 /* BLINKY.C: LED Flasher */  
03 //*****  
04 /* This file is part of the uVision/ARM development tools. */  
05 /* Copyright (c) 2005-2006 Keil Software. All rights reserved. */  
06 /* This software may only be used under the terms of a valid, current,  
07 /* end user licence from KEIL for a compatible version of KEIL software  
08 /* development tools. Nothing else gives you the right to use this software. */  
09 //*****  
10  
11 #include <...>  
12 #include <...>  
13 #include <...>  
14 #include <...>  
15 #include <...>  
16  
17  
18  
19 //*****  
20 //*****  
21 #define led1  
22 #define led2  
23 #define led3  
24 #define led4  
25 #define led5  
26 #define led6  
27 #define led7  
28 #define led8  
29  
30  
31  
32  
33 void espera(  
34 {  
35     long int  
36     for (x=60  
37     {  
38         //va  
39     }
```

The 'Options for Target 'Target 1'' dialog is open, showing the 'Target' tab. The device is set to 'NXP (founded by Philips) LPC2378'. The 'Code Generation' section has 'ARM-Mode' selected, and the 'Use MicroLIB' checkbox is checked. A callout box with the text 'Selecionar essa opção' points to the 'Use MicroLIB' checkbox.

| Read/Only Memory Areas              |          |       |         |                                  | Read/Write Memory Areas             |          |            |        |                          |
|-------------------------------------|----------|-------|---------|----------------------------------|-------------------------------------|----------|------------|--------|--------------------------|
| default                             | off-chip | Start | Size    | Startup                          | default                             | off-chip | Start      | Size   |                          |
| <input type="checkbox"/>            | ROM1:    |       |         | <input type="radio"/>            | <input type="checkbox"/>            | RAM1:    |            |        | <input type="checkbox"/> |
| <input type="checkbox"/>            | ROM2:    |       |         | <input type="radio"/>            | <input type="checkbox"/>            | RAM2:    |            |        | <input type="checkbox"/> |
| <input type="checkbox"/>            | ROM3:    |       |         | <input type="radio"/>            | <input type="checkbox"/>            | RAM3:    |            |        | <input type="checkbox"/> |
|                                     | on-chip  |       |         |                                  |                                     | on-chip  |            |        |                          |
| <input checked="" type="checkbox"/> | IROM1:   | 0x0   | 0x30000 | <input checked="" type="radio"/> | <input checked="" type="checkbox"/> | IRAM1:   | 0x40000000 | 0x8000 | <input type="checkbox"/> |
| <input type="checkbox"/>            | IROM2:   |       |         | <input type="radio"/>            | <input type="checkbox"/>            | IRAM2:   | 0x7FE00000 | 0x4000 | <input type="checkbox"/> |

# Tutorial: criação de projeto e geração de binário para ARM

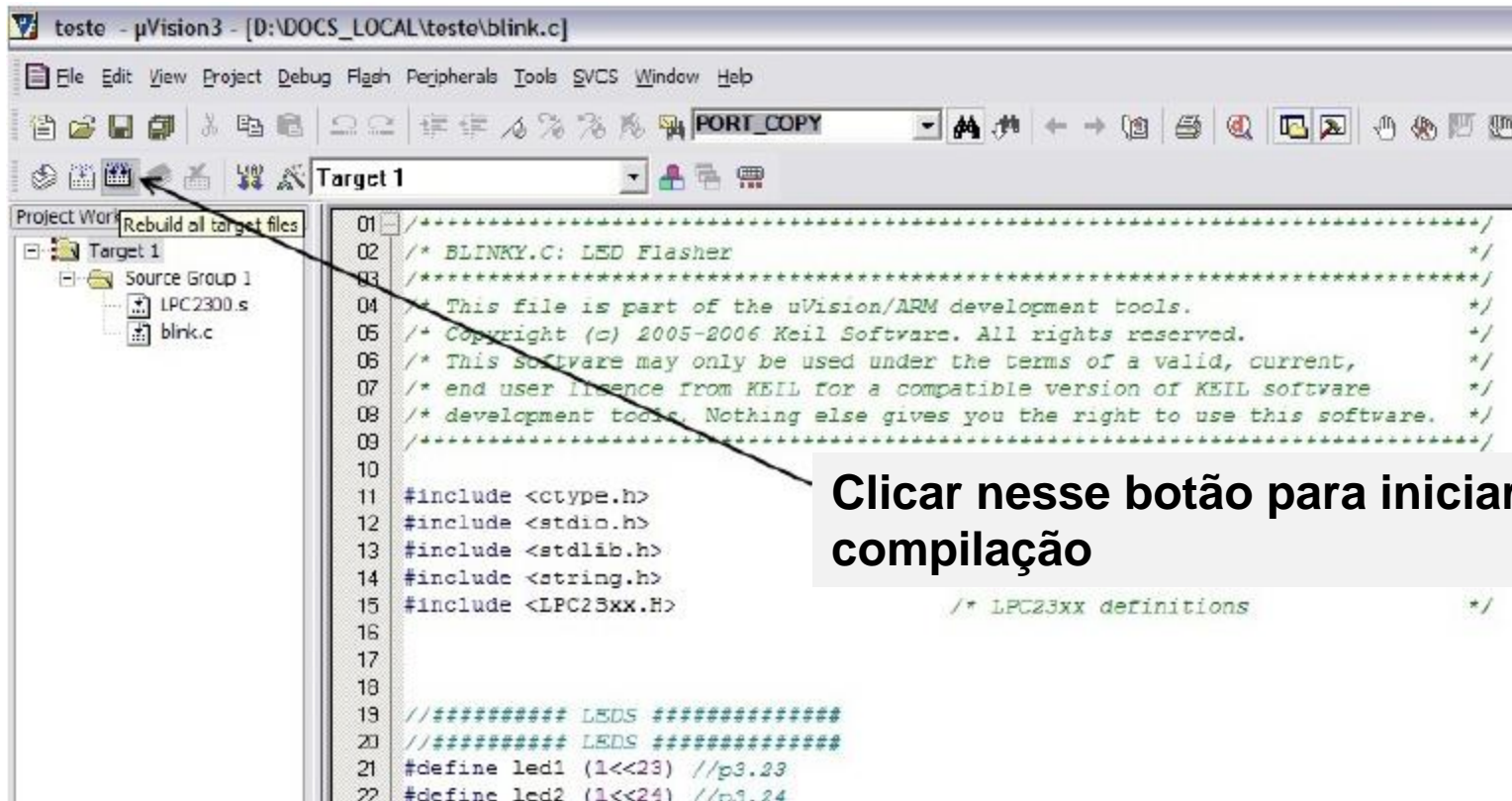
## “Propriedades do projeto”

The screenshot shows the Keil uVision3 IDE interface. The main window displays the source code for a project named 'teste'. The code is a C program for an LED flasher on an ARM microcontroller. The 'Options for Target 'Target 1'' dialog box is open, and the 'Output' tab is selected. The 'Name of Executable' is set to 'teste'. The 'Create Executable' radio button is selected. The 'Create HEX File' checkbox is checked, and the 'Browse Information' checkbox is unchecked. A callout box with arrows pointing to these two checkboxes contains the text: 'Marcar essa opção' (pointing to 'Create HEX File') and 'Desmarcar essa opção' (pointing to 'Browse Information').

```
01 /******  
02 /* BLINKY.C: LED Flasher */  
03 /******  
04 /* This file is part of the uVision/ARM development tools. */  
05 /* Copyright (c) 2005-2006 Keil Software. All rights reserved. */  
06 /* This software may only be used under the terms of a valid, current,  
07 /* end user licence from KEIL for a compatible version of KEIL software  
08 /* development tools. Nothing else gives you the right to use this software. */  
09 /******  
10  
11 #include <ct  
12 #include <st  
13 #include <st  
14 #include <st  
15 #include <LE  
16  
17  
18  
19 //*****  
20 //*****  
21 #define led1  
22 #define led2  
23 #define led3  
24 #define led4  
25 #define led5  
26 #define led6  
27 #define led7  
28 #define led8  
29  
30  
31  
32  
33 void espera  
34 {  
35     long int  
36     for(x=60  
37     {  
38         //va  
39     }
```

# Tutorial: criação de projeto e geração de binário para ARM

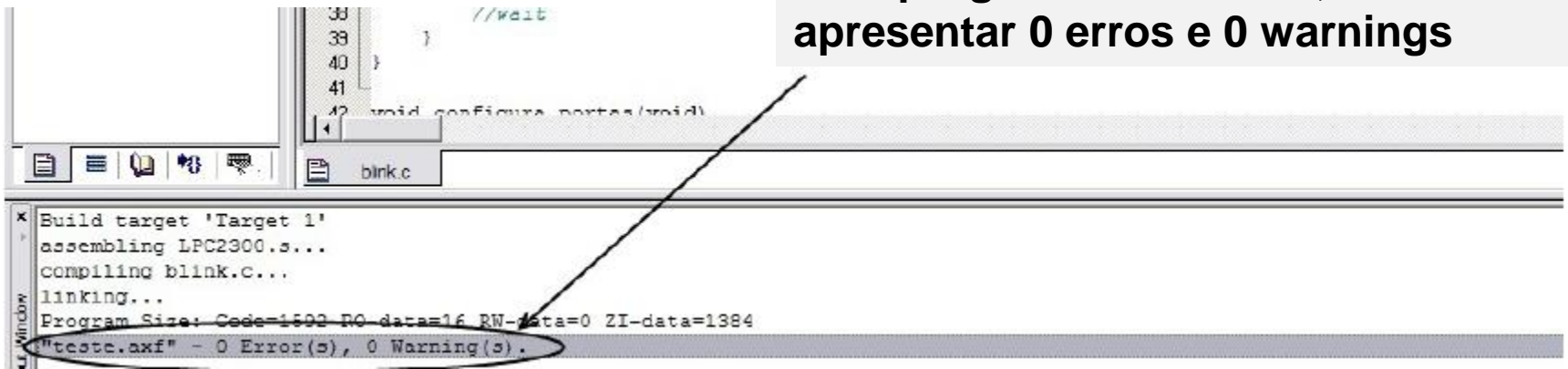
## “Compilação – geração do código objeto”



# Tutorial: criação de projeto e geração de binário para ARM

## “Resultado da compilação”

Se o programa estiver ok, deverá apresentar 0 erros e 0 warnings



The screenshot shows an IDE window with a code editor at the top and a console window at the bottom. The code editor displays the following code:

```
38 //wait
39 }
40 }
41
42 void configura_portas(void)
```

The console window shows the following output:

```
Build target 'Target 1'
assembling LPC2300.s...
compiling blink.c...
linking...
Program Size: Code=1592 RO-data=16 RW-data=0 ZI-data=1384
"teste.axf" - 0 Error(s), 0 Warning(s).
```

A red circle highlights the final line of the console output, and a black arrow points from the text box above to this line.

# Estudo de caso



# Estudo de caso: Controlador de uma máquina de venda de refrigerantes – Prof. Ney Calazans

---

Projetar o circuito de controle para gerência das operações de uma máquina de venda de refrigerantes.

## Especificação:

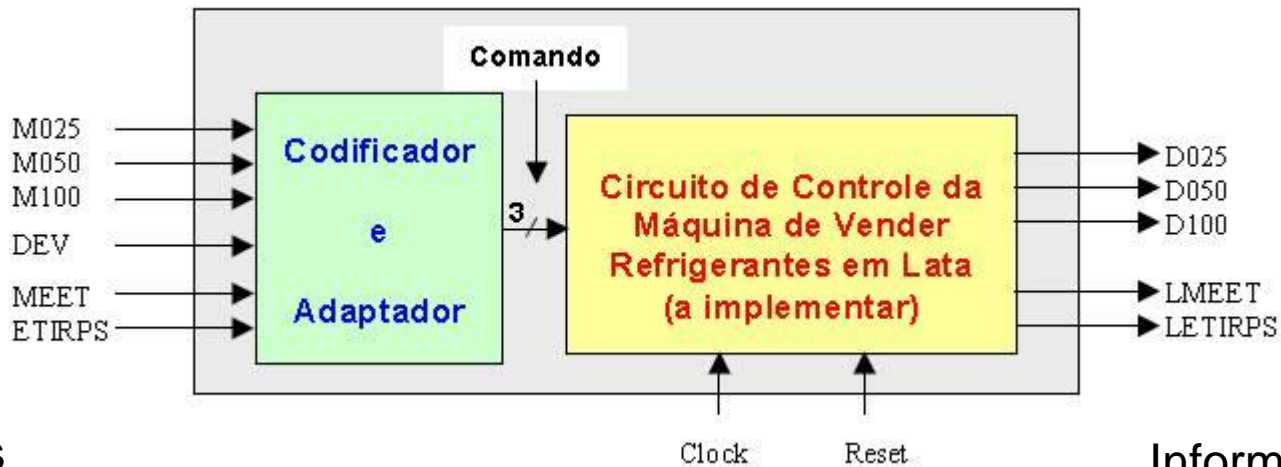
A máquina fornece dois tipos de refrigerantes, denominados MEET e ETIRPS. Estes estão disponíveis para escolha pelo usuário a partir de duas teclas no painel com o nome dos refrigerantes. Ambos refrigerantes custam R\$1,50 e existe na máquina uma fenda para inserir moedas com um sistema eletromecânico capaz de reconhecer moedas de R\$1,00, R\$0,50 e R\$0,25, e capaz de devolver automaticamente qualquer outro tipo de moeda ou objeto não reconhecido. Além disso, durante a compra, o usuário pode desistir da transação e apertar a tecla DEV que devolve as moedas inseridas até o momento. Somente após acumular um crédito mínimo de R\$1,50 o usuário pode obter um refrigerante. A devolução de excesso de moedas é automática sempre que o valor inserido antes de retirar um refrigerante ultrapassar R\$1,50. Uma terceira simplificador consiste em ignorar a composição exata das moedas inseridas na máquina, atendo-se apenas ao montante total inserido.

[Link para a especificação completa.](#)



# Estudo de caso: Controlador de uma máquina de venda de refrigerantes – Prof. Ney Calazans

**Solução:** Diagrama de blocos



Informações  
fornecidas pelos  
sensores

Informações  
enviadas para os  
atuadores (eletro-  
mecânicos)

# Estudo de caso: Controlador de uma máquina de venda de refrigerantes – Prof. Ney Calazans

---

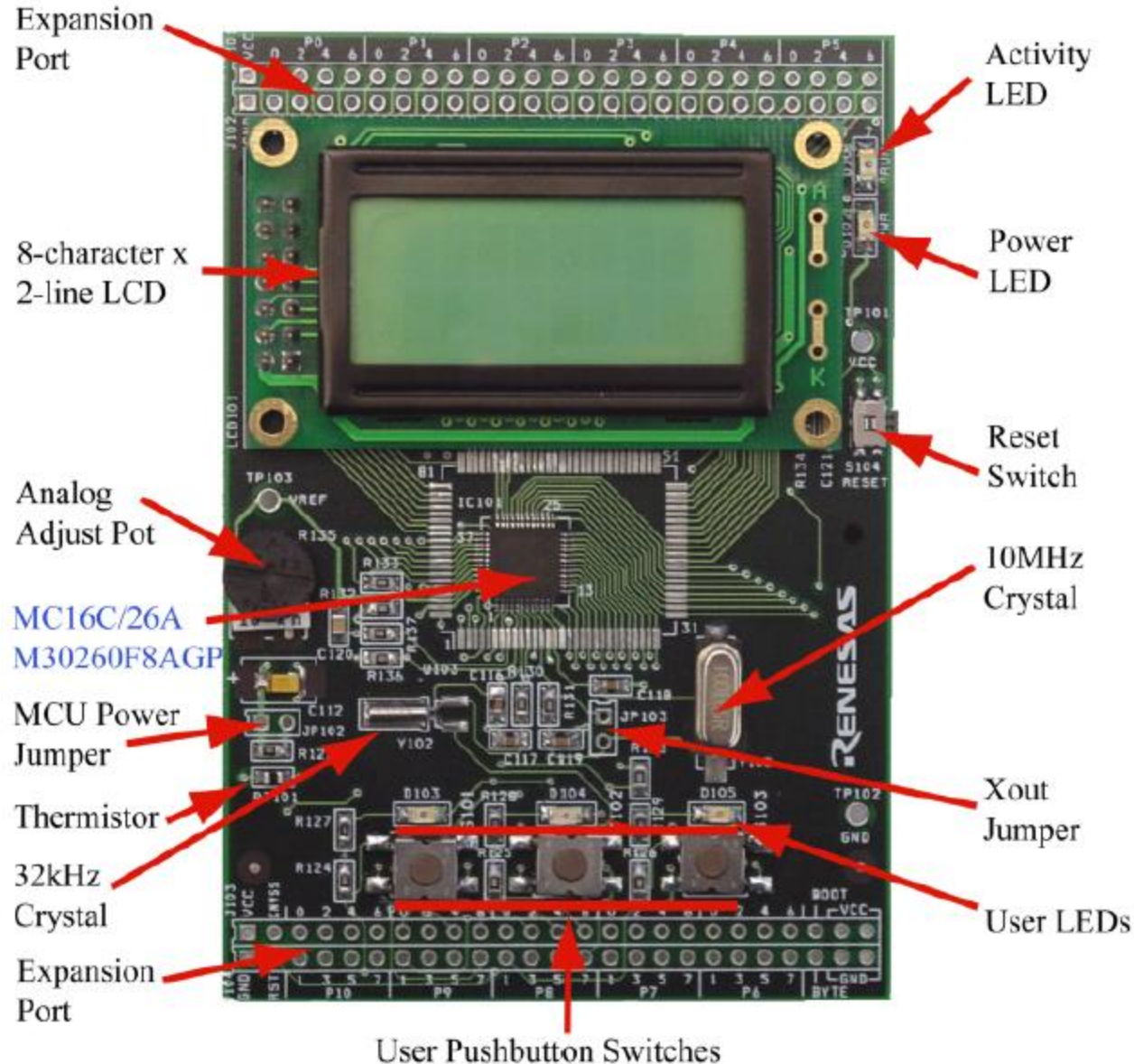
**Solução:** Tabela de estados

| Estado Atual | Comando de Entrada |      |      |               |               |      |        |
|--------------|--------------------|------|------|---------------|---------------|------|--------|
|              | Nada               | M025 | M050 | M100          | DEV           | MEET | ETIRPS |
| S000         |                    | S025 |      |               | S000          | S000 |        |
| S025         |                    | S050 |      |               | S000,<br>D025 |      |        |
| S050         |                    |      |      |               |               | S050 |        |
| S075         |                    |      |      |               |               |      |        |
| S100         | S100               |      |      | S150,<br>D050 |               |      |        |
| S125         |                    |      |      |               |               |      |        |
| S150         |                    |      |      |               |               |      |        |

# Estudo de caso: Controlador de uma máquina de venda de refrigerantes

## Programa prototipado na plataforma da Renesas

- Renesas foi criada por divisões da Mitsubishi e Hitachi
- Utilizado microcontrolador da família M16C/26
- M16C/26 – MCU de 16 bits com CPU da série M16C/60
- Kit QSK26A conectado via USB (usado também como fonte)
- Manual de hardware M16C\_Hardware\_Manual\_Rev0.9.pdf





# Aplicação com smart-card I<sup>2</sup>C e código de barras

