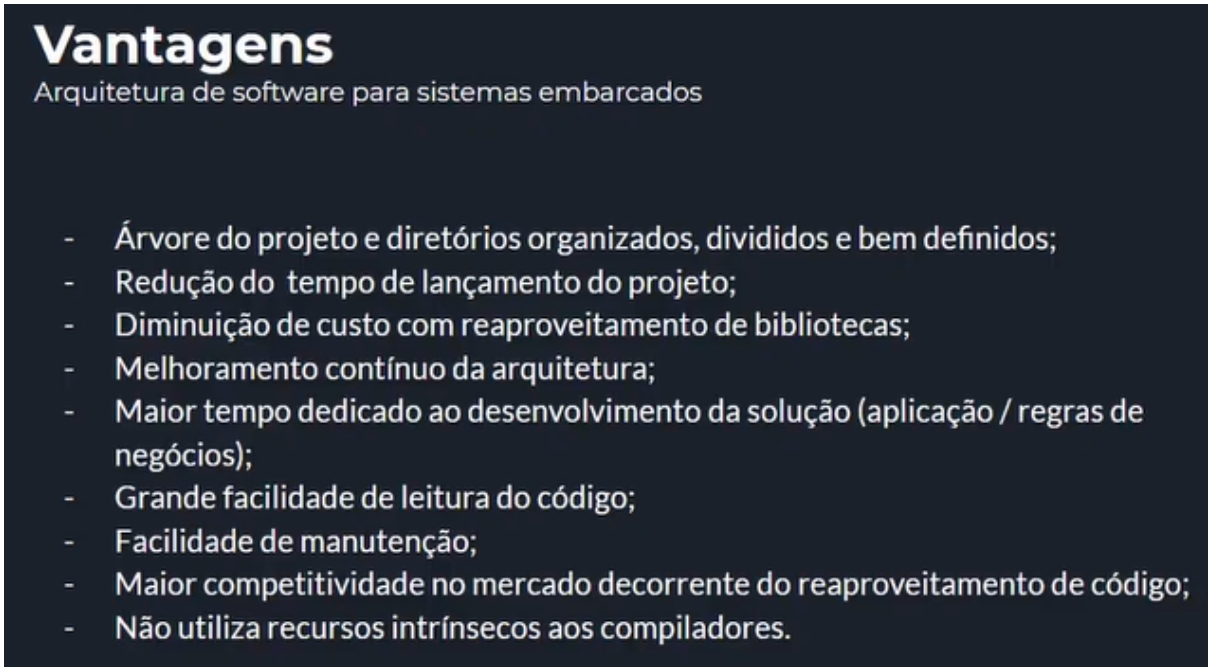
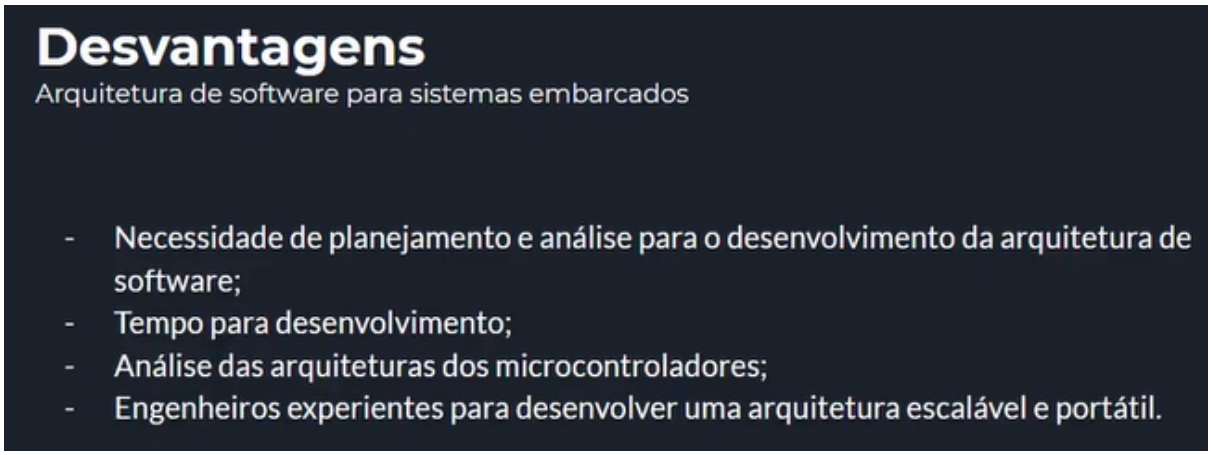
Arquitetura de Software para Sistemas Embarcados





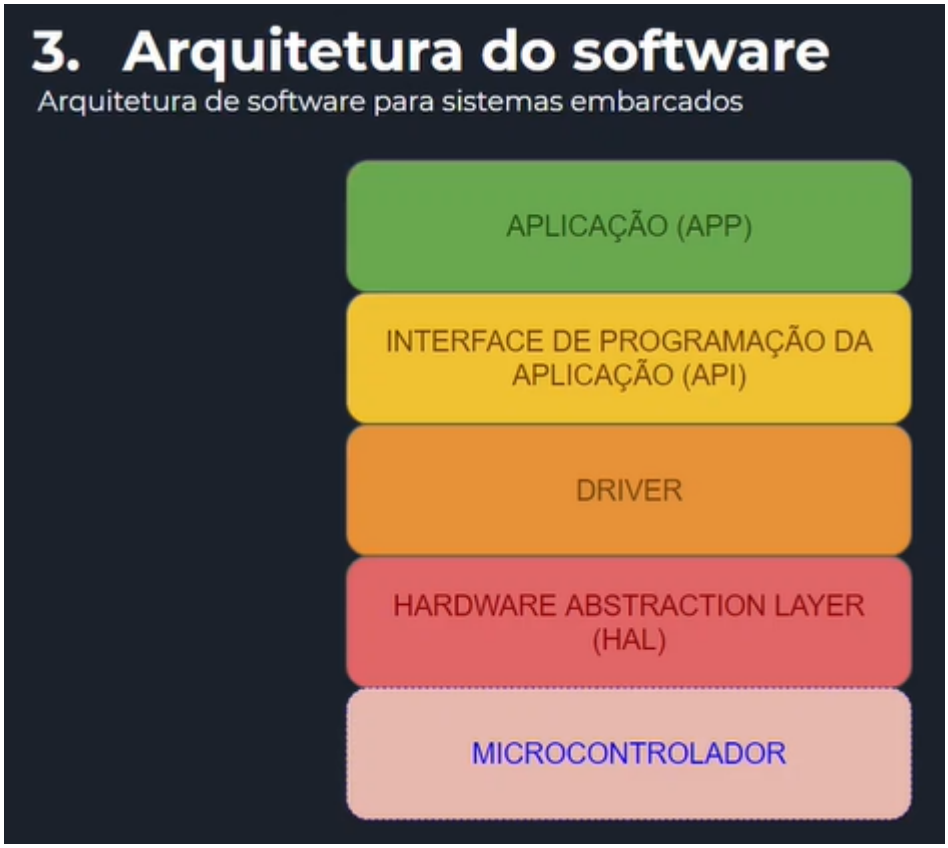
Qualidades de um software portátil

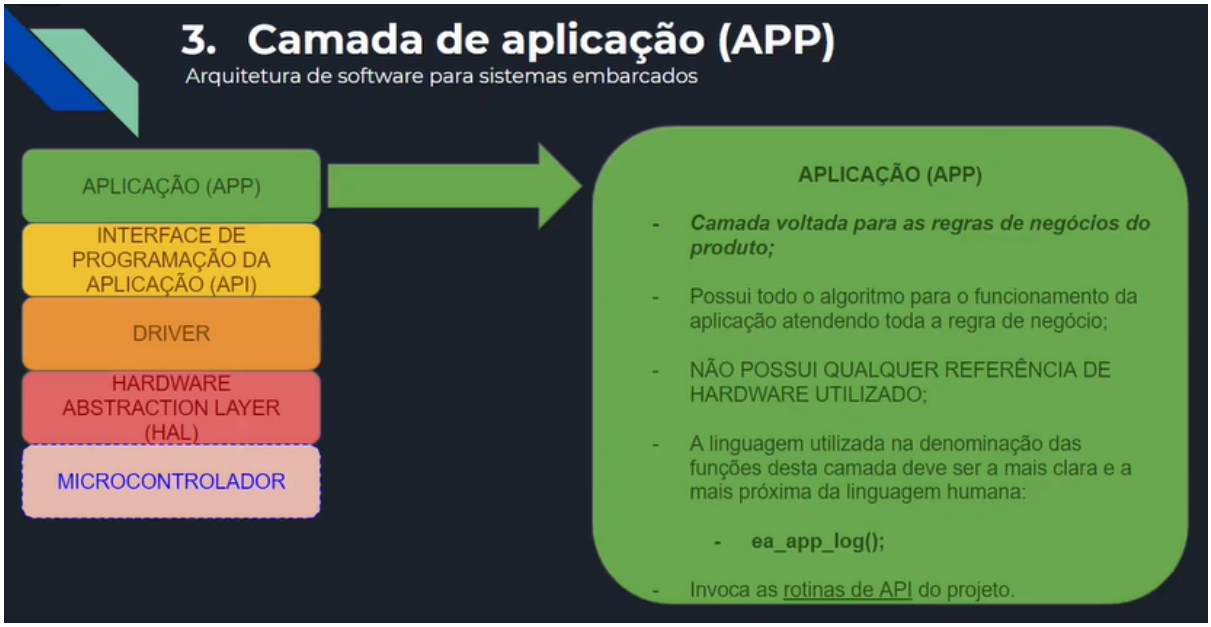
Arquitetura de software para sistemas embarcados

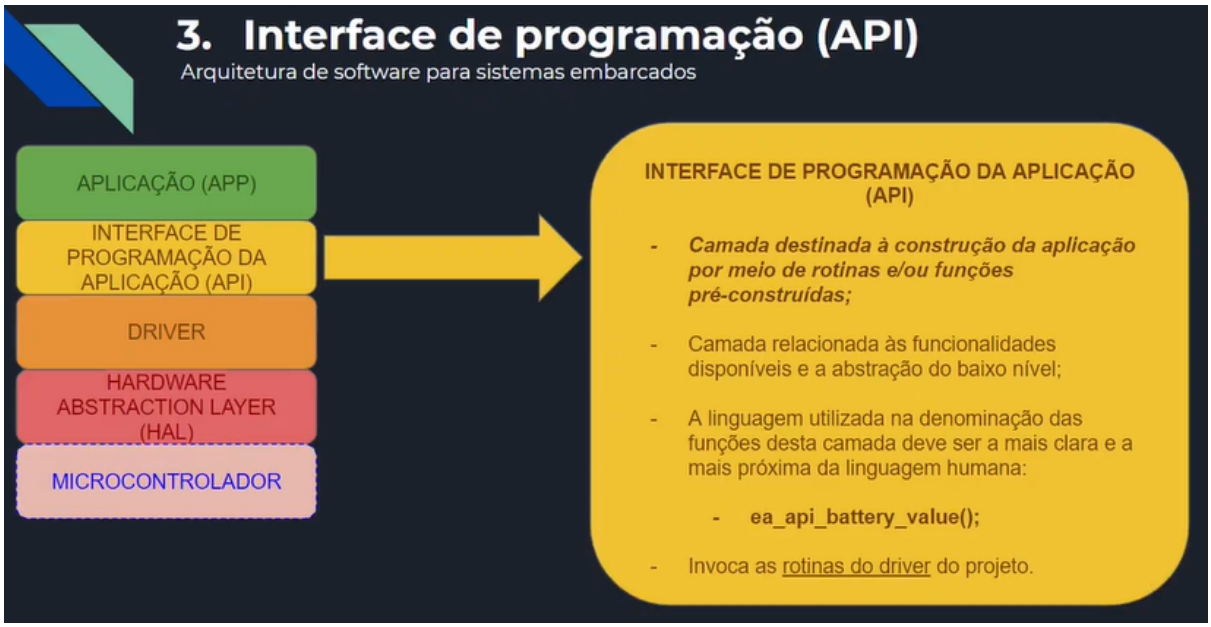
- Modular;
- Camada de abstração do hardware é simples e escalável;
- Camadas fracamente acopladas;
- Facilidade em identificar a integração entre camadas;
- Alta coesão, legível e simples;
- Padronização de código;
- Complacente ao ANSI-C e MISRA-C;
- Uso de encapsulamento e abstração de tipos de dados
- Devidamente documentado.

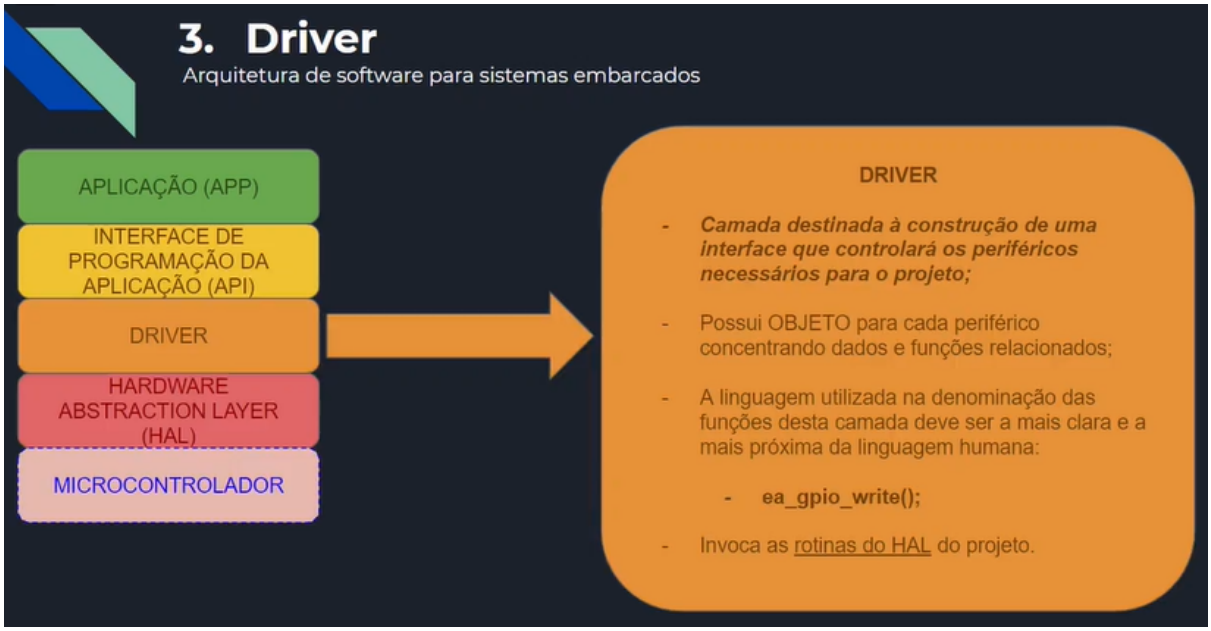
3. Arquitetura do software

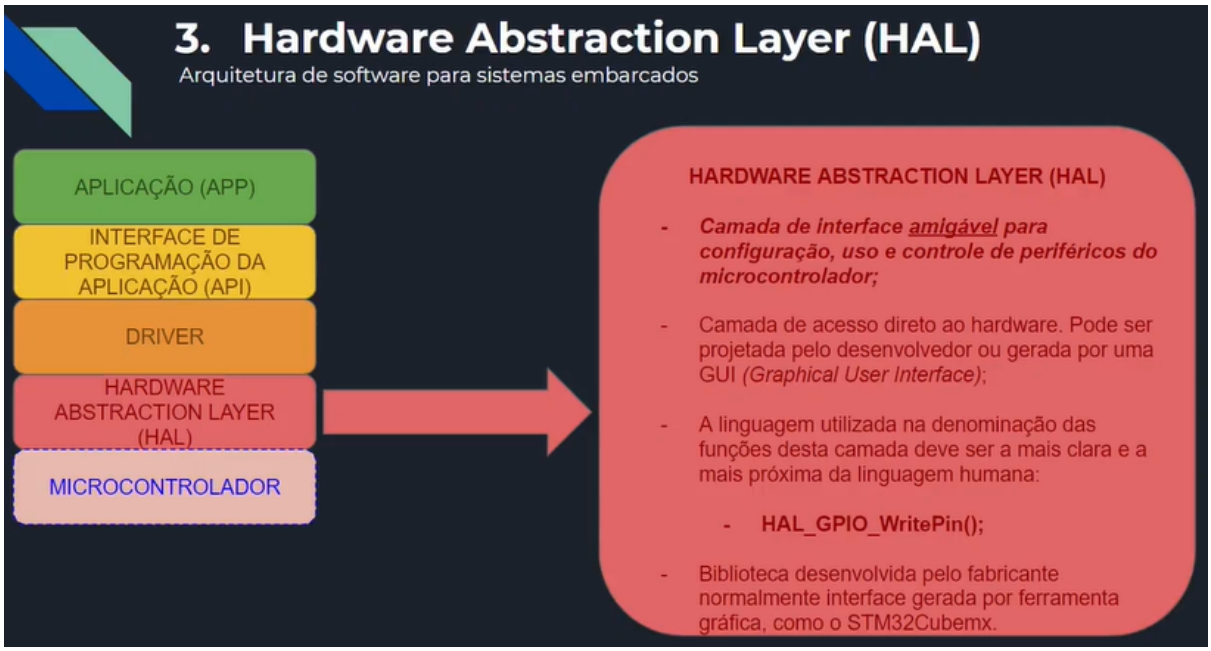
Arquitetura de software para sistemas embarcados

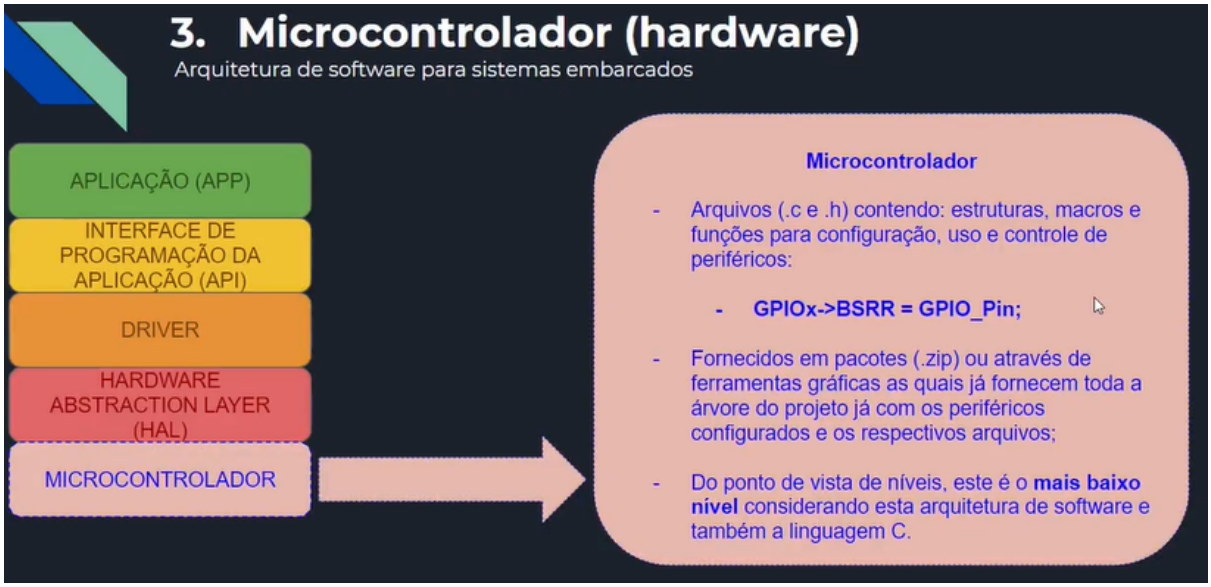


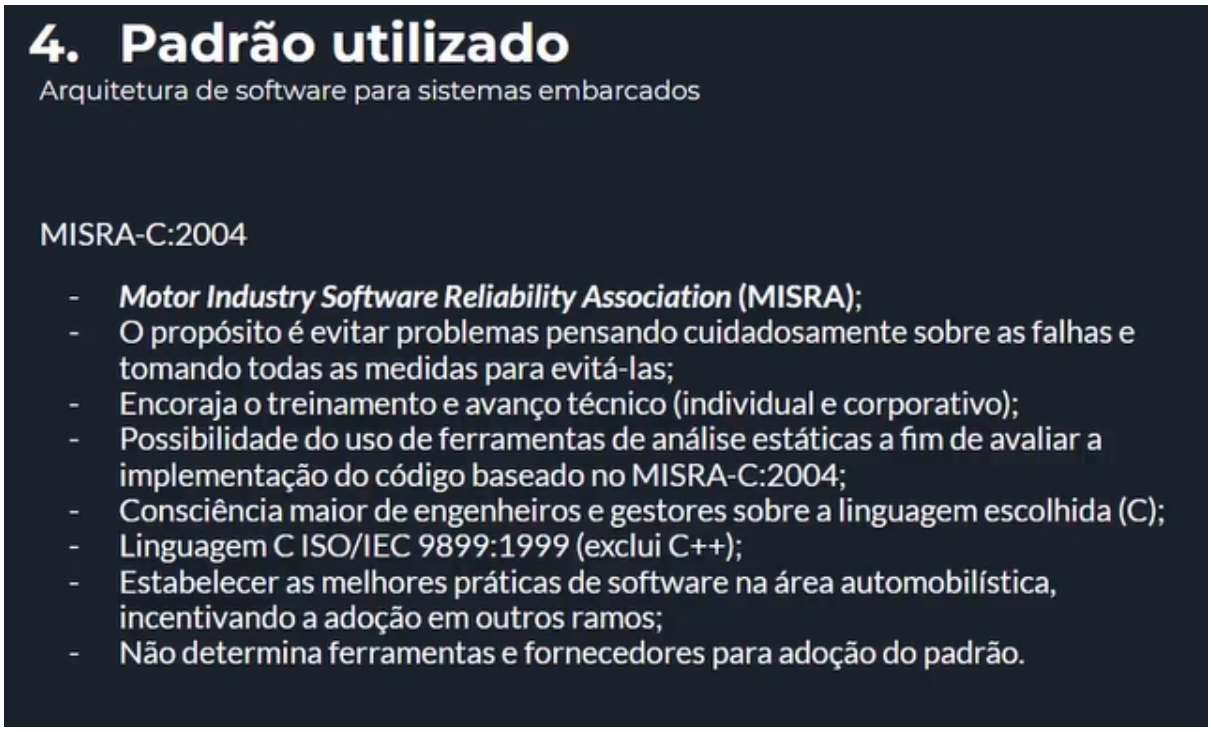

3. Camada de aplicação (APP)
Arquitetura de software para sistemas embarcados
APLICAÇÃO (APP)
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
DRIVER
HARDWARE ABSTRACTION LAYER (HAL)
MICROCONTROLADOR
APLICAÇÃO (APP)
- Camada voltada para as regras de negócios do produto;
- Possui todo o algoritmo para o funcionamento da aplicação atendendo toda a regra de negócio;
- NÃO POSSUI QUALQUER REFERÊNCIA DE HARDWARE UTILIZADO;
- A linguagem utilizada na denominação das funções desta camada deve ser a mais clara e a mais próxima da linguagem humana:
- ea_app_log();
- Invoca as rotinas de API do projeto.

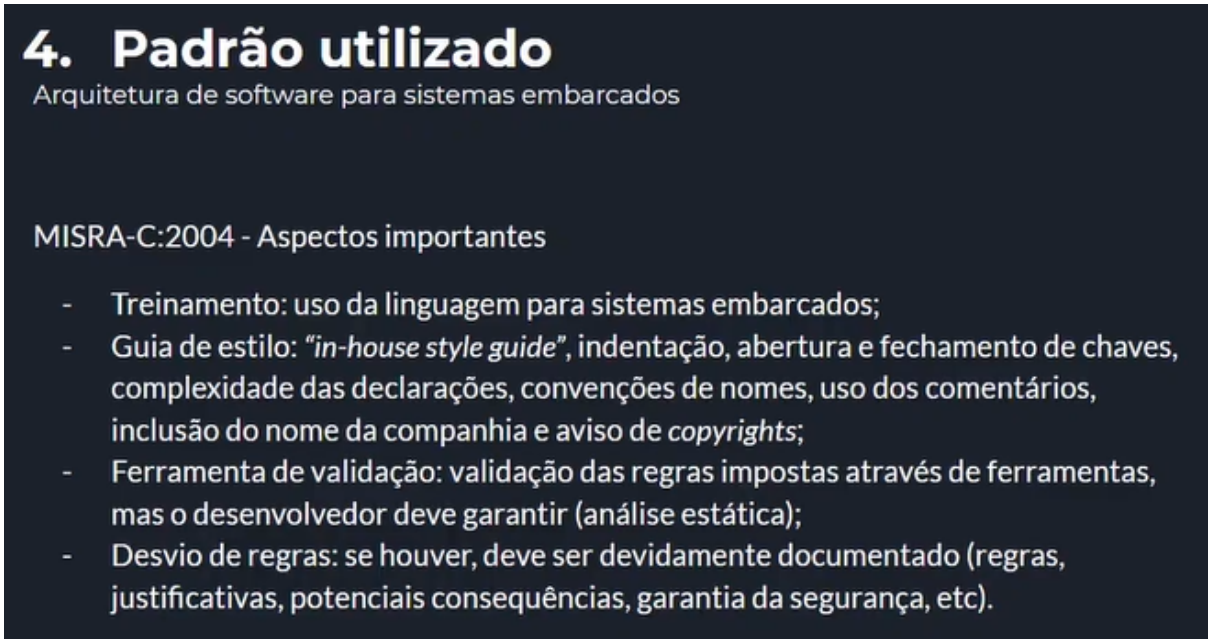

3. Interface de programação (API)
Arquitetura de software para sistemas embarcados
APLICAÇÃO (APP)
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
DRIVER
HARDWARE ABSTRACTION LAYER (HAL)
MICROCONTROLADOR
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
- Camada destinada à construção da aplicação por meio de rotinas e/ou funções pré-construídas;
- Camada relacionada às funcionalidades disponíveis e a abstração do baixo nível;
- A linguagem utilizada na denominação das funções desta camada deve ser a mais clara e a mais próxima da linguagem humana:
- ea_api_battery_value();
- Invoca as rotinas do driver do projeto.


3. Driver
Arquitetura de software para sistemas embarcados
APLICAÇÃO (APP)
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
DRIVER
HARDWARE ABSTRACTION LAYER (HAL)
MICROCONTROLADOR
DRIVER
- Camada destinada à construção de uma interface que controlará os periféricos necessários para o projeto;
- Possui OBJETO para cada periférico concentrando dados e funções relacionados;
- A linguagem utilizada na denominação das funções desta camada deve ser a mais clara e a mais próxima da linguagem humana:
- ea_gpio_write();
- Invoca as rotinas do HAL do projeto.

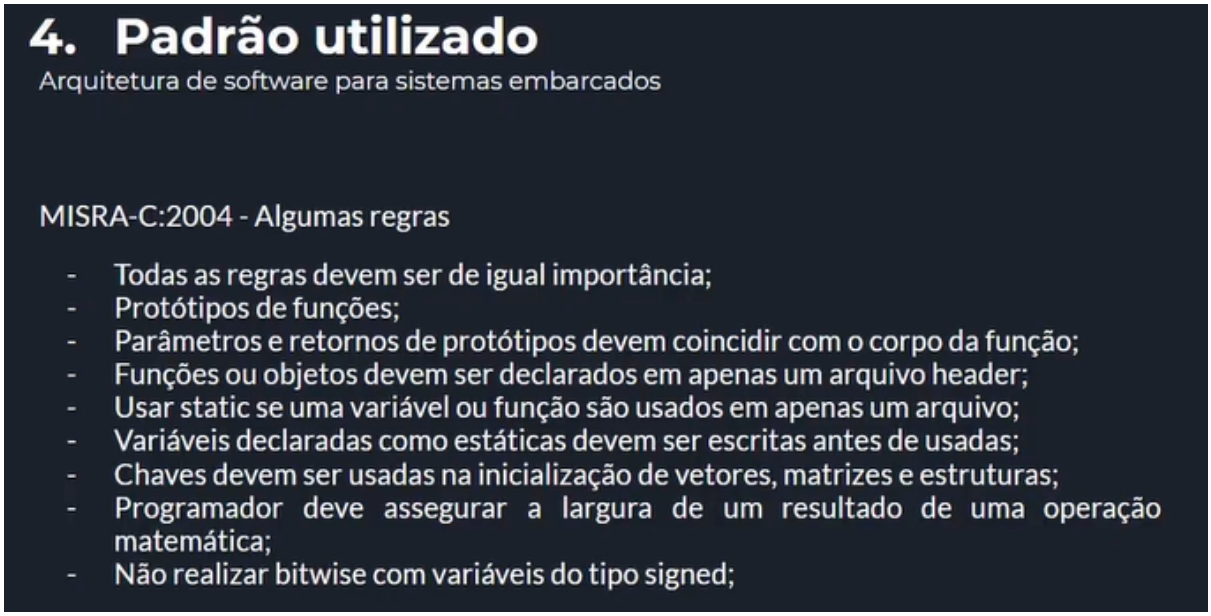

3. Hardware Abstraction Layer (HAL)
Arquitetura de software para sistemas embarcados
APLICAÇÃO (APP)
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
DRIVER
HARDWARE ABSTRACTION LAYER (HAL)
MICROCONTROLADOR
HARDWARE ABSTRACTION LAYER (HAL)
- Camada de interface amigável para configuração, uso e controle de periféricos do microcontrolador;
- Camada de acesso direto ao hardware. Pode ser projetada pelo desenvolvedor ou gerada por uma GUI (Graphical User Interface);
- A linguagem utilizada na denominação das funções desta camada deve ser a mais clara e a mais próxima da linguagem humana:
- HAL_GPIO_WritePin();
- Biblioteca desenvolvida pelo fabricante normalmente interface gerada por ferramenta gráfica, como o STM32Cubemx.


3. Microcontrolador (hardware)
Arquitetura de software para sistemas embarcados
APLICAÇÃO (APP)
INTERFACE DE PROGRAMAÇÃO DA APLICAÇÃO (API)
DRIVER
HARDWARE ABSTRACTION LAYER (HAL)
MICROCONTROLADOR
Microcontrolador
- Arquivos (.c e .h) contendo: estruturas, macros e funções para configuração, uso e controle de periféricos:
- GPIOx->BSRR = GPIO_Pin;
- Fornecidos em pacotes (.zip) ou através de ferramentas gráficas as quais já fornecem toda a árvore do projeto já com os periféricos configurados e os respectivos arquivos;
- Do ponto de vista de níveis, este é o mais baixo nível considerando esta arquitetura de software e também a linguagem C.


4. Padrão utilizado
Arquitetura de software para sistemas embarcados
MISRA-C:2004
- Motor Industry Software Reliability Association (MISRA);
- O propósito é evitar problemas pensando cuidadosamente sobre as falhas e tomando todas as medidas para evitá-las;
- Encoraja o treinamento e avanço técnico (individual e corporativo);
- Possibilidade do uso de ferramentas de análise estáticas a fim de avaliar a implementação do código baseado no MISRA-C:2004;
- Consciência maior de engenheiros e gestores sobre a linguagem escolhida (C);
- Linguagem C ISO/IEC 9899:1999 (exclui C++);
- Estabelecer as melhores práticas de software na área automobilística, incentivando a adoção em outros ramos;
- Não determina ferramentas e fornecedores para adoção do padrão.



https://erinwrightwriting.com/in-house-style-guides-for-small-businesses-benefits-preparation/#:~:text=A house style guide outlines,issues%2C and product name formatting.



4. Padrão utilizado

Arquitetura de software para sistemas embarcados

MISRA-C:2004 - Algumas regras

- A quantidade de bits que devem ser deslocados em operações com shift, deve obedecer a largura do tipo base menos um: char i = 1, x = 0; x = i << 7;
- Não aplicar operações de bitwise em variáveis do tipo signed;
- Aplicação do operador unário de menos "--" em tipos unsigned pode gerar valores negativos devido a possibilidade da promoção de tipo para int;
- Operações de incremento e decremento não devem ser utilizados na mesma expressão, devem ser usados separadamente;
- Não utilizar o operador de atribuição "=" em expressões de controle (if). Se isto for necessário, realizar separadamente.

5. Organização dos diretórios

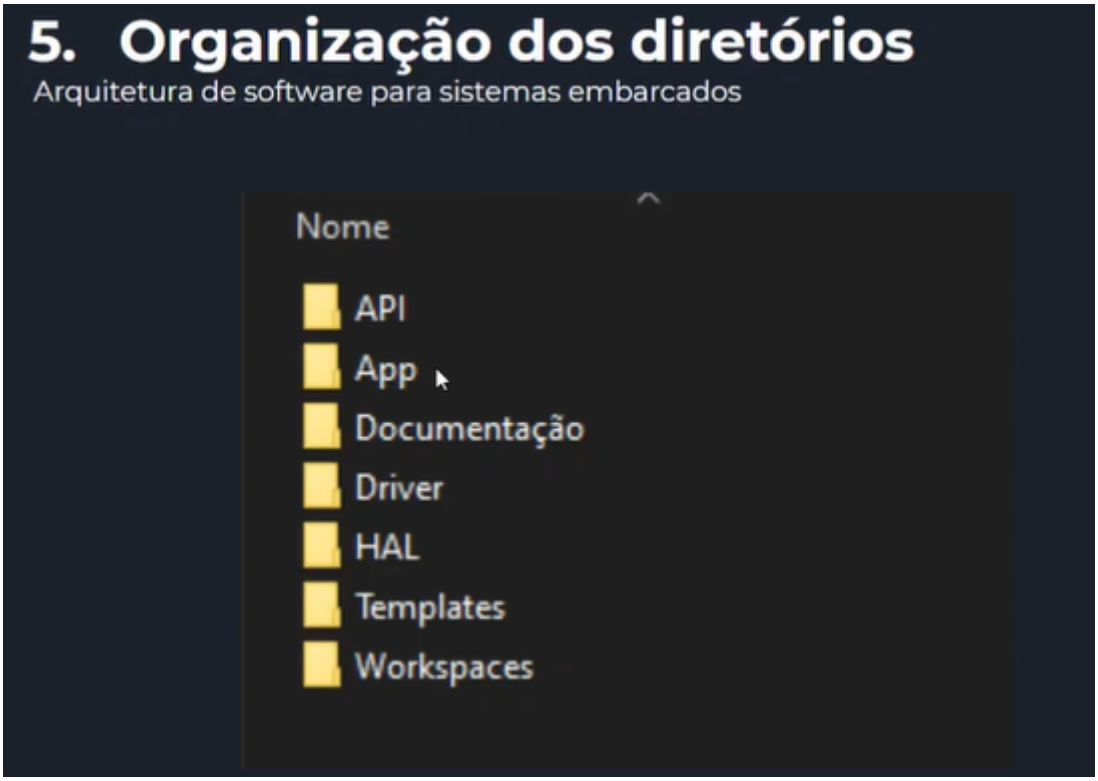
Arquitetura de software para sistemas embarcados

Os benefícios

- Projeto bem organizado;
- Melhora a portabilidade e manutenção;
- Facilidade em reconhecer a arquitetura do software e camadas;
- Sistema de diretórios com os respectivos arquivos .c e .h;
- Grande facilidade na substituição de diretórios e arquivos, se necessário.

5. Organização dos diretórios

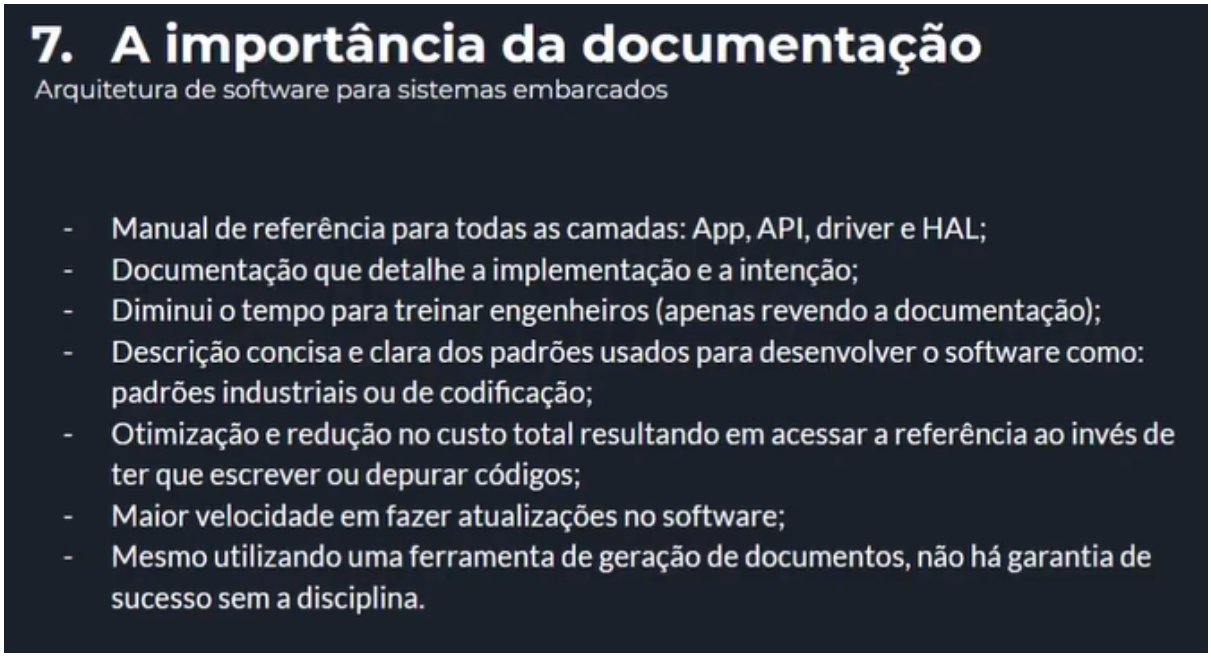
Arquitetura de software para sistemas embarcados



6. *House coding* e convenções

Arquitetura de software para sistemas embarcados

- O idioma utilizado no desenvolvimento do software e documentação;
- Convenções de nomes de variáveis, funções e arquivos;
- Determina como variáveis e funções devem ser declaradas;
- Determina como cabeçalhos e arquivos *.c* e *.h* devem ser padronizados;
- Determina como devem ser os resumos das rotinas/funções;
- Determina como deve ser a indentação.



Colocar o README dentro da pasta documentação

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/aaf6915e-2c22-47ab-8c09-5cbf19ea3c6a/README.rar