EMBARCADOS EXPERIENCE

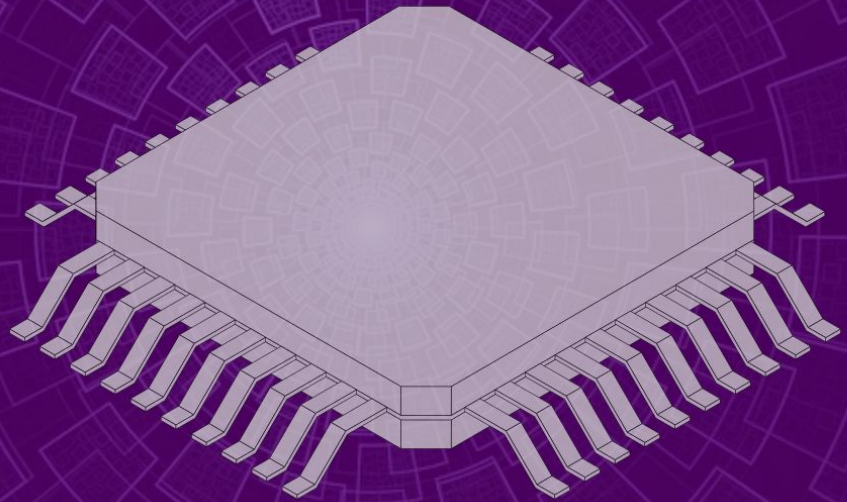
2020

Patrocínio Ouro

MOUSER ELECTRONICS

Realização

EMBARCADOS



28 de Novembro

Evento online com muito networking
e troca de conhecimento.

EMBARCADOS EXPERIENCE 2020

Patrocínio Ouro

MOUSER ELECTRONICS

Realização

EMBARCADOS

TDD e projetos embarcados: vale a pena?

Renata Camilo

SOBRE MIM

Graduação

Engenharia elétrica

EESC · USP
Escola de Engenharia de São Carlos
Universidade de São Paulo

Estágios

NXP

EMBARCADOS

Curadoria IoT

ONOV LAB

Grupo de estudos

IoT SANCA

Atualmente

Desenvolvimento de software embarcado

V2COM
WEG Group

Mestrado na área de 5G

UNICAMP

PROJETO

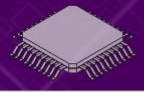
Desenvolvedora

Objetivo: Firmware embarcado do produto X

Tempo: 3 meses

Opções de desenvolvimento:

- **Opção 1** - DLP
- **Opção 2** - TDD

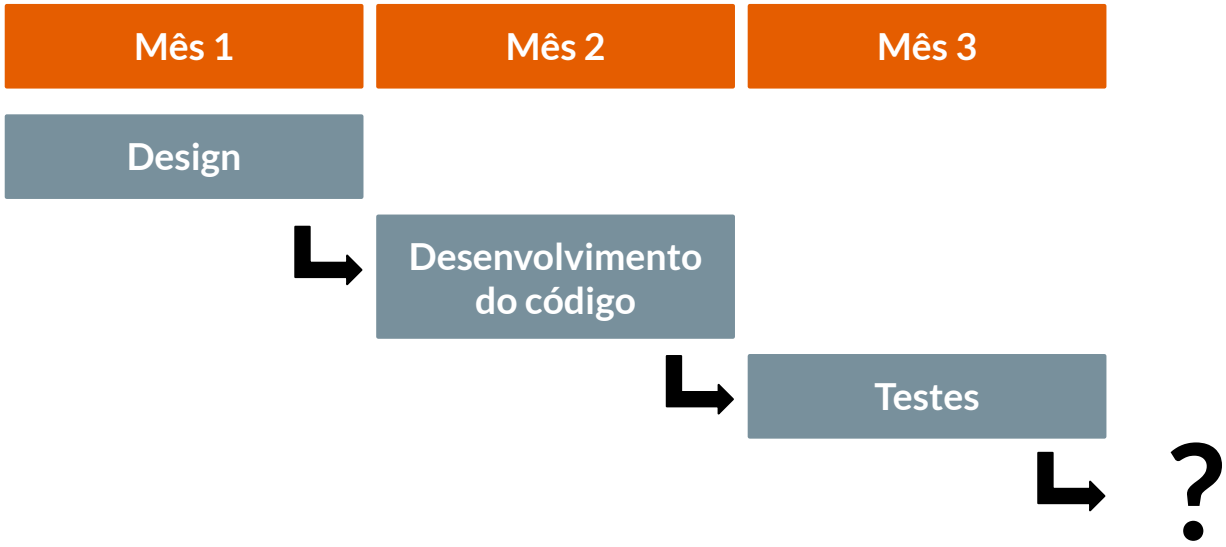


OPÇÃO 1

Debug

Later

Programming



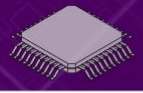
DLP - RESULTADOS

Muitos bugs foram encontrados durante os testes...

Consequência: muito tempo tentando resolvê-los

Todos os bugs foram resolvidos!

Mais bugs aparecem!



DLP - RESULTADOS

- O próprio desenvolvedor pode encontrar bugs muito tempo depois devido a erros cometidos no desenvolvimento que não foram vistos antes.

- Quanto tempo leva para encontrar um bug? Pode levar muito tempo.
  - Atraso na entrega do projeto!

- Se outras partes do código dependiam dessa que continha o bug?
  - Outros bugs podem ser descobertos nessas outras partes.
  - Mais gasto de tempo tentando encontrá-los e consertá-los.

- Adicionar código ou fazer update depois de um tempo: pode quebrar o que já foi feito!

DLP - RESULTADOS

Vamos ver a

Opção 2 - TDD ?

OPÇÃO 2

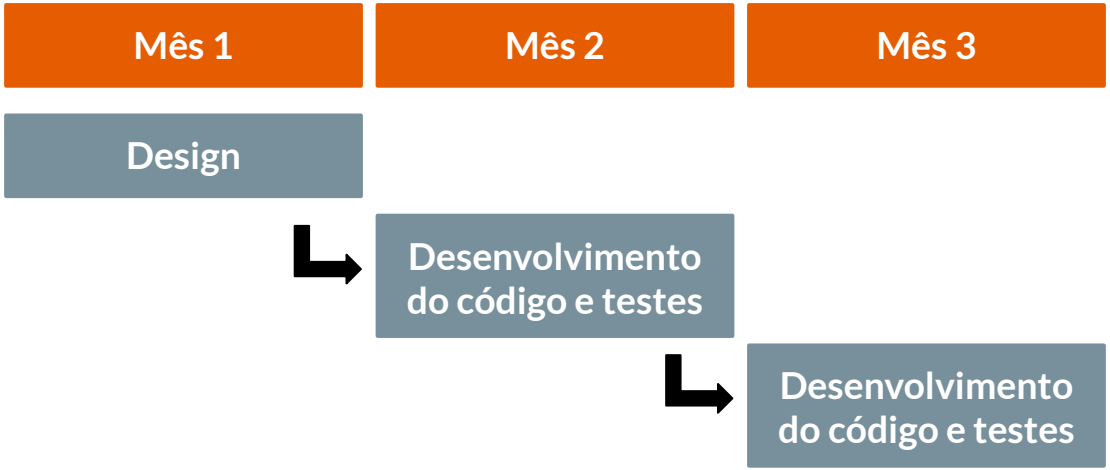
Mas calma, o que é TDD?

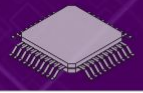
Test

Driven

Development

TDD


Mês 1
Mês 2
Mês 3
Design
Desenvolvimento do código e testes
Desenvolvimento do código e testes



TDD

- Desenvolvedores cometem erros o tempo todo.
- Se os testes forem feitos durante o desenvolvimento, bugs que seriam propagados já são facilmente resolvidos.
- Testes podem ser usados como documentação e um ‘manual de instruções’ de como a implementação deve operar.
- Por isso devemos testar **TUDO** o que desenvolvemos!

TDD

Nossa, mas eu preciso testar tudo?

Vão ser muitos testes!

Não vai dar tempo!

DEPENDE

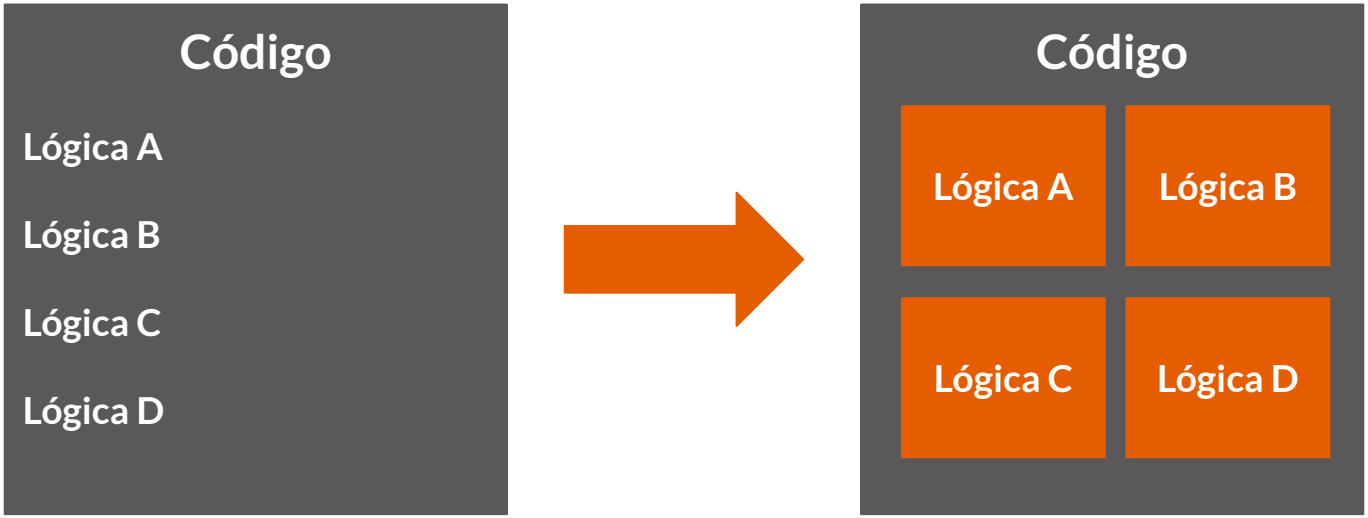
Você tem tempo de testar tudo?

Precisa testar tudo?

Posso testar só o que for mais importante?

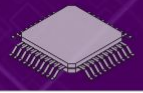
A quantidade de testes vai depender do seu design. Se ele for **modularizado** a quantidade tende a ser menor.

SISTEMA MODULAR



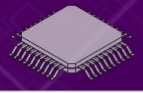
DICA

https://www.youtube.com/watch?v=7O57OZGWdVA



TDD + SISTEMA MODULAR

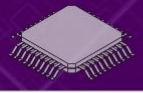
- Cada módulo pode ser testado individualmente, utilizando **testes unitários**.
- E depois são feitos testes de integração entre os módulos.
- Menos testes e mais agilidade em comparação a um sistema não modularizado.
- Facilidade de integração.



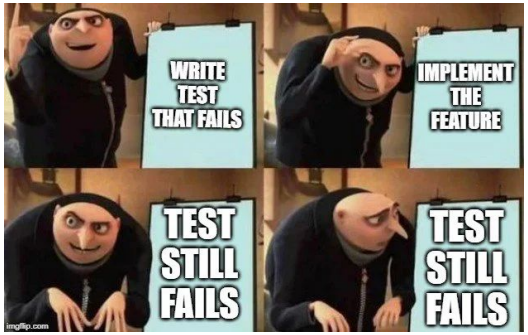
TDD - TO DO LIST

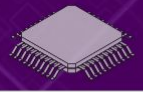
- Primeiro passo: fazer uma lista de testes
  - listar tudo o que necessita ser testado;
  - diversas situações e cenários;
  - incluir casos especiais.

- Depois de tudo implementado, você pode ser a necessário adicionar mais testes a lista. Não tem problema! Pois seguindo os passos a seguir vamos evitar problemas.



TDD - OPERAÇÃO





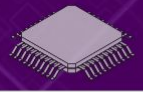
TDD

- A implementação estará incompleta até que todos os testes estejam passando. Testes contínuos e automatizados.

- Refatore sem medo, os testes estão lá para evitar e reduzir bugs.

- E, caso eles apareçam, vai ser bem mais fácil de resolvê-los.

- Facilidade na atualização do código, principalmente se ela for feita por outro desenvolvedor.



EXEMPLO

- Desenvolvimento TDD de uma função que retorna a soma de dois inteiros que são passados como argumento.

Vou utilizar:

- Ceedling (framework de teste unitário)

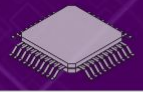
TDD + PROJETOS EMBARCADOS

Mas onde eu vou rodar esses testes?

O hardware que estou usando tem limitações, não tenho **espaço** o suficiente para o código e os testes!

E ele pode nem estar **pronto** ainda, ou com **problemas**!

No seu **PC**!

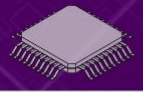


VANTAGENS

- Desenvolvimento mais rápido e evita bugs;
- Sem problemas de espaço e RAM;
- Não preciso estar com o hardware para estar desenvolvendo (↓ custo).

DÚVIDA

- Como eu vou fazer com partes que dependem exclusivamente do hardware, como o RTOS, I/Os e chamadas de callbacks?

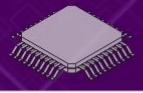


MOCKS

Objetos utilizados para ‘simular’ iterações de saída com dependências externas aos nossos testes.

E para que os **MOCKS** são usados?

- Para testar se uma ou mais rotinas de uma dependência externa foi chamada corretamente;
- Testar quantas vezes essas rotinas foram chamadas;
- Testar se essas rotinas foram chamadas com os parâmetros corretos.

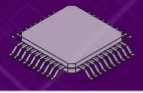


STUBS

Objetos utilizados para ‘simular’ iterações de chegada com dependências externas a implementação.

E para que os **STUBS** são usados?

- Para testar retornos de uma dependência externa;
- Testar o comportamento da implementação frente aos diferentes retornos (ex: sucesso, falha, ..).

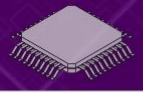


FAKES

Objetos utilizados para ‘simular’ iterações de chegada de dados com dependências externas a implementação.

E para que os **FAKES** são usados?

- Para testar retornos com dados de uma dependência externa;
- Testar o comportamento da implementação frente aos dados recebidos (ex: leitura de um valor).



RESULTADOS

Implementação finalizada ✓

Todos os testes estão passando ✓

Refatoração foi feita e o código está organizado ✓

Finalmente!

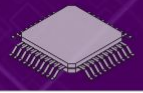
Hora de juntar **FW** + **HW**

...

...

...

NÃO FUNCIONOU



PROBLEMAS

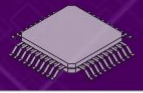
Existem problemas que são exclusivos de hardware além do seu código nem chegar a compilar.

Ok, as vantagens do **TDD** são bem atraentes. **MAS**,a parte de bugs no final do desenvolvimento estão me lembrando **DLP**!

CALMA!
AINDA NÃO ACABOU!
TEMOS UM BÔNUS

Vamos ver a

Opção 3 - TDDVP ?



TDDVP

Test

Driven

Development na

Vida real de

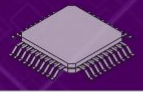
Projetos embarcados

COMBINAÇÃO

Testes no PC

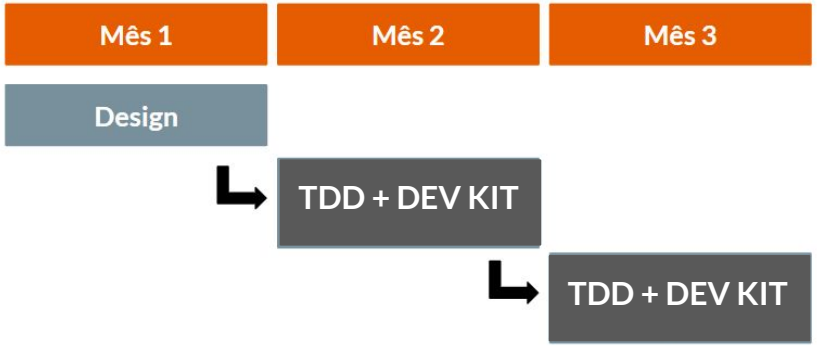
+

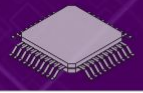
Testes de compilação usando dev kit



TDDVP

- Bugs aparecem mais cedo e são menores!
- Evita problemas de RAM, tempística, wdog ... Problemas exclusivos do HW.
- União do melhor dos dois mundos: menos bugs e um produto mais robusto!



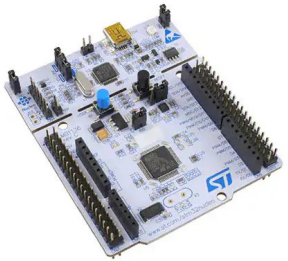


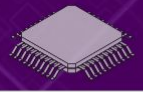
DEMO

Vamos ver na prática como funciona o que foi falado até agora?

Vou utilizar:

- Plataforma Conera™
- Ceedling (framework de teste unitário)
- STM32L010RB





CONERA™

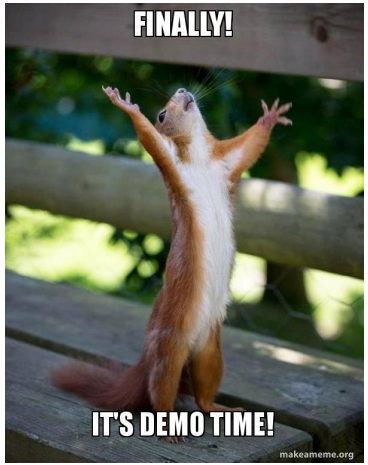
Conera™ é um pacote de software com um poderoso kernel de middleware embarcado. E para as atividades a seguir iremos utilizar a camada Conera™ Core.

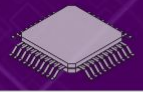
https://v2com.com/conera-v2com-pt/

Não é necessário se preocupar com a implementação específica do timer, GPIOs, ..., drivers específicos do hardware. Isso também vale para o RTOS!

Apenas precisa da interface dos módulos e drivers que serão utilizados na demo.

DEMO





CONCLUSÃO

- A abordagem para um projeto vai depender de diversos fatores, como tempo e orçamento.

- Não existe um padrão a ser seguido.

- Vimos inúmeros pontos positivos, mas as técnicas tem que ser adaptáveis para que o seu projeto necessita.

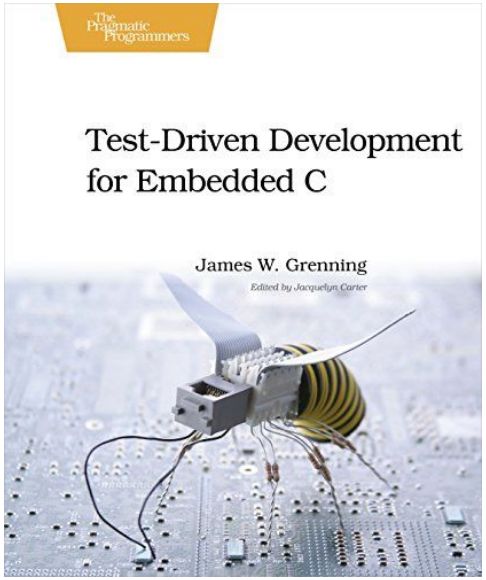
CONCLUSÃO

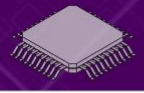
- Ferramentas que deixam o seu produto mais robusto, com menos bugs e menos surpresas para ele ser entregue no prazo estipulado.

- Vale a pena considerar também que o projeto pode ter evolução, e assim fica mais fácil de adicionar features.

- E não se esqueça: você não necessariamente você será o único a mexer no código, poderão haver outros desenvolvedores.

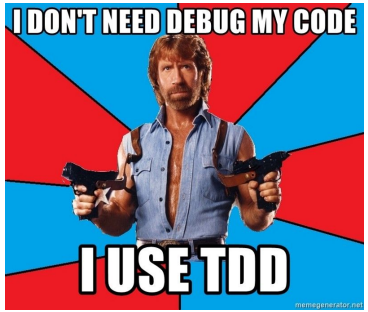
DICA

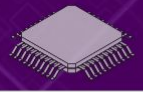

The Pragmatic Programmers
Test-Driven Development for Embedded C
James W. Grenning
Edited by Jacquelyn Carter



E NO FINAL, QUAL A SUA OPINIÃO?

Você acha que usar TDD em projetos embarcados vale a pena?





OBRIGADA!

Dúvidas? Comentários?

CONTATO

renatadecamillo@gmail.com

https://br.linkedin.com/in/renatadecamillo

EMBARCADOS EXPERIENCE 2020

Patrocínio Ouro

MOUSER ELECTRONICS

Realização

EMBARCADOS

28 de Novembro

OBRIGADO!

Evento online com muito networking
e troca de conhecimento.