

# Python em sistemas embarcados: Sim ou Não?

PythonBrasil[7]  
São Paulo

Gustavo Sverzut Barbieri

# o que é um sistema embarcado?

- sistema embutido em um aparelho
- "tudo que não é PC ou servidor"
- costuma ser específico, poucas funções
- não necessariamente "low end specs"
- ... porém é comum ser "low end"

# exemplos de sistemas embarcados



# por que python em embarcados?

- melhor linguagem do mundo :-)
- gerência de memória
- tipagem dinâmica e forte
- orientada a objetos e mais...
- vasta biblioteca padrão e módulos extras
- bem conhecida, fácil achar programadores

# por que não python em embarcados?

- consumo de memória RAM
- consumo de disco/flash
- é lento (CPU)
- demora pra iniciar (import)
- código fonte visível
- falta de acesso baixo nível
- falta de capacidades RealTime

MITO x VERDADE? depende...

técnicos

políticos

psicológicos ou imaginários

# resolvendo problemas técnicos

---

- melhorar arquitetura do software
- melhorar algoritmos
- usar bons design patterns
- evitar retrabalho (caches)
- uso de bindings (C, C++)

- acessos de baixo nível
- otimização de caminhos críticos
- gerência mais eficiente de memória
- acesso a frameworks otimizados:
  - gstreamer, xine, mplayer, vlc
  - gtk, qt, efl, x11, sdl, opengl

# resolvendo demais problemas

---

- análise se realmente faz sentido
- apresente evidências de sucesso
- crie protótipo ou prova de conceito
- apresente ganhos esperados:
  - fica pronto mais rápido (TTM)?
  - mais fácil de manter?
  - programador mais barato?
  - SEMPRE DIZER QUANTO MAIS!

- apresente problemas possíveis:
  - gastar mais CPU ou memória?
  - demorar para iniciar?
  - código fonte visível x licença...
  - DIZER VALORES DOS IMPACTOS!

# caso real: canola2

- centro multimídia para Nokia
- 64Mb de RAM, ARM de 400Mhz
- tela enorme de 800x480
- Linux, X11, multi-tarefas
- fotos, música, vídeos e muito mais!



CANOLA2

# história do canola

- canola1: SDL + C/GObject
- nasceu protótipo e "evoluiu"
- 4-8 meses de desenvolvimento
- antes do iPhone, causou "WOW"



- novos requisitos
- pouco tempo (originalmente 3 meses)
- "VAI EXPLODIR!"

# história do canola

---

- vamos refazer o software
- ... mas não dá tempo!
- ... dá se for em Python!
- ... mas é lento!
- ... não é! PROTÓTIPO PROVANDO ;-)
- chefia aprova. #TODOSFELIZ
- hora extra sem fim. #TODOSTRISTE

# provando com protótipo

- PyGame com `sprite.RenderUpdates`
- animações tão rápidas como C
- ... afinal 99% era feito em C/SDL ;-)
- base para um canvas-2D

# problemas pré-conhecidos

---

- canvas-2D
- media scanner
- arquitetura e plugins

# resolvendo canvas-2D

---

- pesquisa por sistemas existentes
- GooCanvas, Cairo, Qt, EFL
- bindings inexistentes
- performance não atingia SDL (em ARM!)
- criar tudo com SDL seria proibitivo
- EFL era a melhor opção:
  - otimizado para ARM
  - bindings criados

# resolvendo media scanner

---

- pesquisa por sistemas existentes
- Maemo Media Scanner, Tracker, ...
- bindings inexistentes
- performance MUITO ruim
- criado LightMediaScanner, com bindings

# resolvendo arquitetura e plugins

---

- python tornou tudo muito fácil
- plugins via zipimport
- arquitetura MVC inspirada no Zope
- economizou MUITO tempo de projeto

# resolvendo problemas inesperados

---

- ficou lento! :-)
- procura de decorators de `__init__` para classe
- atrasar leitura de módulos secundários
- atrasar início de tarefas
- ficou rápido! :-)

- grande maioria de nossos projetos em C
- maiores problemas:
  - espaço em disco/flash
  - pouco tempo para o projeto
  - necessidade de bindings
  - baixa complexidade dos projetos
  - código fonte visível
  - existência de python na plataforma

- use a ferramenta certa para o caso certo
- pondere: python nem sempre serve para tudo!
- problemas técnicos são fáceis de resolver
- demais problemas podem ser resolvidos

Obrigado!

Gustavo Sverzut Barbieri

<barbieri@profusion.mobi>