Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Salvador

30 de abril 2020

Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Universidade Federal da Bahia - UFBA

Escola Politécnica

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Wagner Oliveira

Coorientador: Paulo César

Salvador

30 de abril 2020

Nestor Dias Pereira Neto
Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS/ Nestor Dias Pereira Neto. – Salvador, 30 de abril 2020-
37p. : il. (algumas color.) ; 30 cm.

Orientador: Wagner Oliveira

Dissertação (Mestrado) – Universidade Federal da Bahia - UFBA
Escola Politécnica
Programa de Pós-Graduação em Engenharia Elétrica, 30 de abril 2020.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Salvador, 24 de novembro de 2012:

Wagner Oliveira
Orientador

Professor
Convidado 1

Professor
Convidado 2

Salvador
30 de abril 2020

Resumo

Segundo a **??**, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: latex. abntex. editoração de texto.

Abstract

oihbqptipbõq4tnpot4photnj4yojnj4ynojp

Keywords: latex. abntex. text editoration.

Sumário

1 Introdução

1 - Falar do crescimento de pesquisas sobre robótica,

Nos últimos anos novas técnicas para construção de robôs tem sido bastante estudadas, em especial áreas como robótica móvel e robótica colaborativa tem chamado bastante atenção dos pesquisadores. As principais características dessas áreas são o alto grau de percepção do ambiente que rodeia o robô, juntamente com a busca de níveis cada vez mais maiores de autonomia, tanto do ponto de vista da capacidade energética quanto do ponto de vista das tomadas de decisão, além de uma execução cada vez mais precisa em seus movimentos.

As consequências desses trabalhos podem ser percebida em robôs com

2 - a busca por sistemas com um grau maior de autonomia A robótica tem se caracterizádo pelo grande nível de percepção do ambiente e pelos sistema complexos de controle do movimentos

O ROS como um framework que esta se tornando o padrão no desenvolvimento de robótica

1 - Fazer um link com a complexidade dos sistemas robóticos modernos e o ROS

Agrupar inúmeros “blocos"de softwares usados em robótica, fornecer drivers para hardwares específicos (sensores e atuadores), gerenciar troca de mensagens entre os nós que fazem parte do sistema, são as função do ROS. Essas características fezem com que o ROS seja reconhecido com um pseudo sistema operacional (PYO et al., 2017). Dessa maneira o ROS se tornou muito ágil no desenvolvimento de novas aplicações para robótica. Usando nós já desenvolvidos e testados por outros desenvolvedores podemos criar novos sistemas completos apenas gerenciando esses nós na rede interna do ROS. Essa abordagem fez com que o número de pacotes para o ROS cresça em uma taxa muito rápida, desde o ano de seu lançamento, 2007, até 2012 o ROS aumentou de 1 para 3699 pacotes (YAMASHINA et al., 2005).

Com essa distribuição de tarefas através de vários nós podemos criar sistemas cada vez mais complexos, apenas inserindo novos nós na rede ROS, essa rede é gerenciada pelo ROS Master, que é apensas mais um nó do sistema, mas com a função de ser um servidor de nome e serviços para o restante dos nós. Ele identifica os nós na rede, assim todos os nós podem se comunicar com os outros através de conexões peer-to-peer, Figura 1. Para desenvolver novas aplicações para integrarem o crescente grupo de pacotes ROS, o desenvolvedor deve respeitar os protocolos de comunicação da rede, as bibliotecas do ROS facilitam a vida do desenvolvedor, por já fornecer funções prontas para o desenvolvimento

de novos códigos compatíveis e que possam se registrar na rede. Detalhes dos protocolos e interno podem ser visto em (ROS, 2011a), (ROS, 2018) e (ROS, 2011b).

Desenvolvimento com fpga, SoC. dificuldade e maior tempo de desenvolvimento

1 - explicar a ideia do FPGA e a dificuldade do desenvolvimento

O potencial que os FPGAs possuem para melhorar o desempenho de sistemas que utilizam processamento digitais de sinal é conhecido já algum tempo, as possibilidades de paralelismo, criação de estruturas de DSP dedicadas à aplicação são recursos muito interessantes que a possibilidade do hardware configurado oferecem, mas em contra partida as facilidade de desenvolvimento encontradas em aplicações que fazem uso de softwares não são encontradas nas mesmas proporções no mundo do hardware, sendo assim:

Juntar fpga com robótica através do ros dando foco na facilidade de desenvolvimento

2 - Fala sobre o FPGA escolhido

explicar a parte da comunicação entre o computador/ros e o SoC

melhorar a comunicação, comunicação eficiente, pacote pronto e de fácil integração com qualquer sistema ros,

O mais genérico possível para se enquadrar a qualquer projeto é que o desenvolvedor tenha interesse em incluir um FPGA ao sistema

Definir o problema (a pergunta)

- **Como estabelecer a comunicação entre o ROS e um sistema de processamento auxiliar embarcado em um FPGA?**

1.1 Justificativa

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver uma solução para estabelecer comunicação entre *Field Programmable Gate Array - FPGA*, configurado como um co-processador de vídeo e o *Robot Operating System - ROS* avaliando o impacto desta aplicação ao sistema.

1.2.2 Objetivos Específicos

- Estudar os assuntos relevantes ao projeto: Verilog HDL, RTOS, Nios II, TCO/IP Stack, ROS;
- Conhecer com detalhes os protocolos da rede TCP/IP usada para comunicação interna dos nós e serviços ROS;
- Desenvolver plataforma com Nios II como base para o andamento do projeto;
- Implementa um sistema operacional de tempo real - RTOS na plataforma base;
- Estabelecer comunicação entre o ROS e o sistema Nios II (embarcador no FPGA) através da tecnologia Gigabit Ethernet;
- Testar aplicações de processamento de vídeo em hardware em conjunto com ROS;
- Avaliar a performance com a inclusão do FPGA ao sistema.

1.3 Organização

No primeiro caítulo...

Parte I

Referenciais teóricos

2 Field Programmable Gate Array - FPGA

2.1 Cyclone IV - Intel

2.2 Soft processor NiosII

2.3 Kit de desenvolvimento DE2-115

3 Robot Operating System - ROS

3.1 Sistema multiagentes

4 FreeRTOS

O FreeRTOS é um sistema de tempo real de grande sucesso desde de seu início, é suportado por mais de 35 architectures e “foi baixado uma vez a cada 175 secundos no ano de 2018” (FREERTOS, 2019). O FreeRTOS é um projeto de *free* e de código aberto disponível sobe licença MIT. O FreeRTOS foi a escolha para esse trabalho por não possuir restrições ao seu uso e também por sua popularidade, o que facilita a busca por informações de seu uso uma tarefa menos árdua.

4.1 Entendendo o FreeRTOS

FreeRTOS é um kernel de um sistema operacional de tempo real para sistemas embarcados, permite que aplicações sejam organizadas como coleções de independentes threads mantendo os requisitos de tempo real do sistema. Foi desesenvolvido em 2003, inicialmente por Richard Barry e posteriormente mantida pela *Real Time Engineers Ltd.*, em 2017 o projeto FreeRTOS passou aser adminitrado *Amazon Web Services*.

O kernel básico é formado por

4.2 Gerenciamento de Memória

4.3 Tasks

4.4 Interrupções

hgdhfhg dgadfadg.

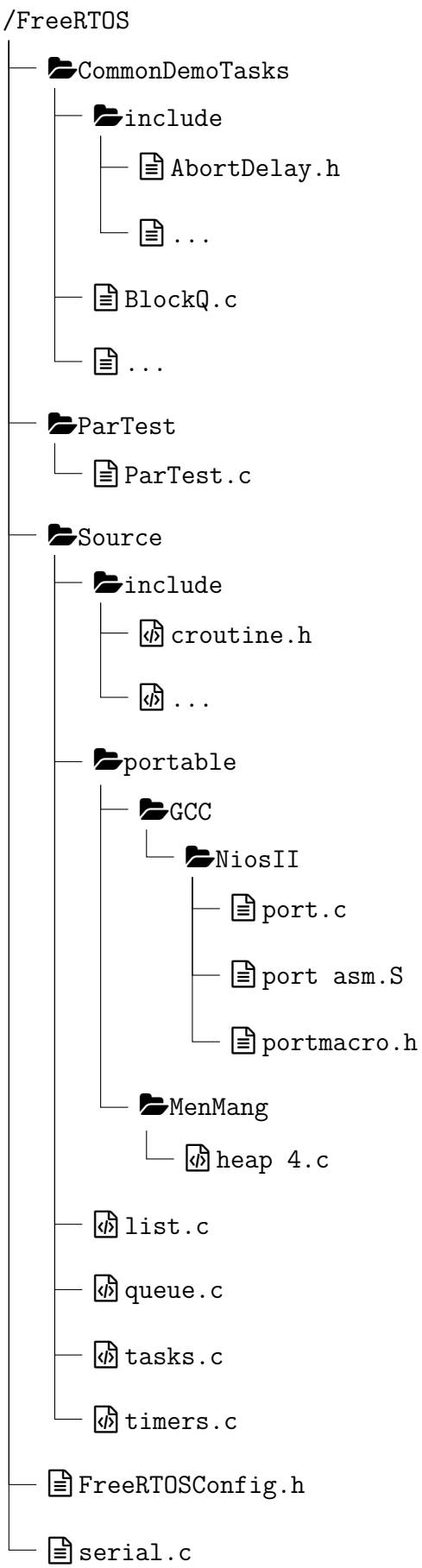
4.5 FreeRTOS+TCP

4.5.1 Orcanização do código fonte

4.5.2 Adicionar os arquivos TCP/IP ao projeto FreeRTOS

======= FreeRTOS+TCP é um TCP/IP *stack* de código aberto para o FreeRTOS, ou seja, é um extenção do do FreeRTOS que fornece uma interface progamação socket para sistemas embarcados.

«««< HEAD jfyhjfjf


/FreeRTOS
CommonDemoTasks
include
AbortDelay.h
...
BlockQ.c
...
ParTest
ParTest.c
Source
include
croutine.h
...
portable
GCC
NiosII
port.c
port asm.S
portmacro.h
MenMang
heap 4.c
list.c
queue.c
tasks.c
timers.c
FreeRTOSConfig.h
serial.c

======= »»»> 2f33d0a9c82916e49400f2b78807da4390f7d32a

Parte II

Desenvolvimento

5 Hardware

6 Software

Parte III

Resultados

7 estudos funturos

7.1 Pellentesque sit amet pede ac sem eleifend consectetuer

8 Conclusão

Capítulo conclusão

Referências

FREERTOS. *History*. 2019. Amazon Web Services, Inc. Disponível em: <https://www.freertos.org/RTOS.html>. Acesso em: 25 outubro 2019. Citado na página 21.