

MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

Virtualisierung eines Echtzeit-Betriebssystems zur Steuerung eines Roboters mit Schwerpunkt auf die Einhaltung der Echtzeit

By: Halil Pamuk, BSc

Student Number: 51842568

Supervisor: Sebastian Rauh, MSc. BEng

Wien, March 22, 2024

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Wien, March 22, 2024

Signature

Kurzfassung

Erstellung einer Echtzeit-Robotersteuerungsplattform unter Verwendung von Salamander OS, Xenomai, QEMU und PCV-521 in der Yocto-Umgebung. Die Plattform basiert auf Salamander OS und nutzt Xenomai für Echtzeit- Funktionen. Dazu muss im ersten Schritt die Virtualisierungsplattform evaluiert werden. (QEMU, Hyper-V, Virtual Box, etc.) Als weiterer Schritt folgt die Anbindung eines Roboters über eine VARAN-Bus Schnittstelle. Das gesamte System wird in der Yocto-Umgebung erstellt und konfiguriert. Das Hauptziel der Arbeit ist es, herauszufinden, wie die Integration von Echtzeit-Funktionen und effizienten Kommunikationssystemen in eine Robotersteuerungsplattform die Reaktionszeit und Zuverlässigkeit von Roboteranwendungen verbessern kann

Schlagworte: Schlagwort1, Schlagwort2, Schlagwort3, Schlagwort4

Abstract

Abstract

Keywords: Echtzeit, Virtualisierung, Xenomai, VARAN

Contents

1	Introduction	1
1.1	State of the art	2
1.2	Problem and task definition	3
1.3	Objective	4
2	Methodology	5
3	Salamander Operating System	6
3.1	Task priorities	7
3.2	Memory Management	7
4	Results	9
5	Discussion	10
6	Summary and Outlook	11
	Bibliography	12
	List of Figures	13
	List of Tables	14
	List of Code	15
	List of Abbreviations	16
A	Anhang A	17
B	Anhang B	18

1 Introduction

Diese Masterarbeit wurde in der Firma SIGMATEK GmbH & Co KG erstellt.

Firma SIGMATEK

Yocto

Salamander 4 - harte Echtzeit

Xenomai

QEMU

Trace-cmd

Kernelshark

VARAN-Bus PCV-521

Windows Ubuntu WSL

1.1 State of the art

1.2 Problem and task definition

In robotics, accurate and timely control is crucial to ensure precise movements and reliable interaction with the environment. However, existing robot control systems often have limitations in terms of real-time capabilities and communication efficiency, which can have a detrimental effect on response time and reliability. The challenge is to develop a powerful real-time robot control platform that overcomes these limitations and improves the performance of robotic applications.

1.3 Objective

The main objective of this work is to create a real-time robot control platform that integrates Salamander OS, Xenomai, QEMU and PCV-521 in the Yocto environment.

This integration is expected to lead to a significant improvement in the response time and reliability of robot applications.

2 Methodology

In diesem Abschnitt werden sämtliche theoretischen Konzepte und Randbedingungen sowie praktische Methoden, die zur Erreichung der Zielsetzung dieser Masterarbeit beigetragen haben, detailliert beschrieben.

Zu Beginn der Arbeit wurde eine ausführliche Analyse der einzelnen Virtualisierungsmöglichkeiten von Salamander 4 durchgeführt. Im Besonderen wurde hier die Virtualisierungsperformance von Ubuntu 22.04, Windows 10 und WSL unter QEMU verglichen.

First, a comprehensive literature review is conducted to understand the current trends and challenges in real-time robot control. Based on the literature study, a suitable virtualisation platform is selected.

After the selection of the virtualisation platform, the robot control platform is implemented. This step includes the installation and configuration of Salamander OS, Xenomai, QEMU and PCV-521 in the Yocto environment. Once the platform has been implemented, the robot is connected via a VARAN bus interface. Finally, the platform is evaluated to determine how the integration of real-time functions and efficient communication systems improves the response time and reliability of robot applications.

- Evaluation der Virtualisierungsplattform: Ich werde verschiedene Virtualisierungsplattformen wie QEMU, Hyper-V, Virtual Box usw. evaluieren. Dies ist ein wichtiger Schritt, um die beste Plattform für meine Anforderungen zu finden.

- Erstellung und Konfiguration des Systems in der Yocto-Umgebung: Ich werde das Yocto-Framework verwenden, um mein Embedded Linux System zu erstellen und zu konfigurieren. Yocto bietet viele Tools und Funktionen, die mir bei der Erstellung und Konfiguration meines Systems helfen können.

- Verbesserung der Reaktionszeit und Zuverlässigkeit von Roboteranwendungen: Mein Hauptziel ist es, herauszufinden, wie die Integration von Echtzeitfunktionen und effizienten Kommunikationssystemen die Reaktionszeit und Zuverlässigkeit von Roboteranwendungen verbessern kann. Ich strebe an, die Leistung und Zuverlässigkeit meiner Roboteranwendungen zu verbessern, indem ich ihre Fähigkeit verbessere, in Echtzeit auf Ereignisse zu reagieren.

- Anbindung eines Roboters über eine VARAN-Bus Schnittstelle: Ich plane, einen Roboter in mein System zu integrieren. Ich werde eine VARAN-Bus Schnittstelle verwenden, um eine schnelle und zuverlässige Kommunikation zwischen dem Roboter und dem Steuerungssystem zu gewährleisten.

3 Salamander Operating System

Salamander 4 ist das proprietäre Betriebssystem. Es basiert auf der Linux-Version 5.15.94 und integriert zusätzlich Xenomai 3.2, eine Echtzeit-Entwicklungsumgebung [2]. Es handelt sich um ein 64-Bit-System, was auf die x86_64-Architektur hinweist. Das Echtzeitverhalten wird durch die Verwendung von Symmetric Multi-Processing (SMP) und Preemptive Scheduling (PREEMPT) erreicht. Darüber hinaus nutzt es IRQPIPE, um Interrupts in einer Weise zu verarbeiten, die die Echtzeitanforderungen des Systems erfüllt. Xenomai besteht aus 3 Teilen. Diese sind der Tabelle 1 zu entnehmen

Table 1: Lang

Teil	Beschreibung
i-pipe	Kernelerweiterung für das Domain-Konzept
Xenomai Kernel	Benutzt die i-pipe, und hängt sich als root-Domain ein
Xenomai User	Programme (LRT) verwenden diese Bibliothek, um Xenomai Funktionen verwenden zu können.

Xenomai beruht auf einem Domain-Konzept, das bedeutet, dass alle IRQ an die erste Domain gesendet werden. (root – Domain / Xenomai) Nur wenn diese nichts mehr zu tun hat, dann darf die 2. Domain arbeiten. Das bedeutet, erst wenn alle Xenomai Task in einem Wartezustand sind, arbeiten die Linux-Tasks.

In der Regel unterbricht der Prozessor beim IRQ-Handling seine aktuellen Aktivitäten, um einen Interrupt zu bearbeiten, während die IRQ-Behandlung von Xenomai einen Interrupt-Pipeline-Mechanismus verwendet, der das gleichzeitige Abrufen und Vorbereiten eines anderen Interrupts ermöglicht, während ein Interrupt bearbeitet wird, was die Leistung verbessert und die Latenzzeit verringert.

Was Xenomai4 von seinem Vorgänger Xenomai3 unterscheidet, ist die vollständige Neugestaltung der Ausführungsphase mit hoher Priorität. Dies geschah aus Gründen der Portabilität und Wartungsfreundlichkeit: I-pipe - die zweite Iteration der ursprünglichen Adeos-Interrupt-Pipeline - wurde vollständig durch Dovetail ersetzt.

Table 2: Lang

Xenomai spezifische Funktionen	Linux spezifische Funktionen
Tasks	Dateizugriffe
Mutexes, Semaphoren, Events	Netzwerk

Ein Aufruf dieser Funktionen erfordert die entsprechende Domain. Wenn der Task in der falschen Domain läuft, dann wird ein Domain-Wechsel forciert. Ein Domainwechsel von Xenomai nach Linux geht relativ einfach. Aber der Wechsel von Linux nach Xenomai braucht Unterstützung, und dafür ist die Hilfe des Gatekeepers notwendig. Das bedeutet, der Gatekeeper hilft einem Task von Linux nach Xenomai zu wechseln.

3.1 Task priorities

Es gibt grundsätzlich 4 Gruppen

Table 3: Übersicht der Prioritätsgruppen und ihrer Beziehungen

Prioritätsgruppe	Bereich
Xenomai Priorität	0 bis 99
Linux RT Priorität	1 bis 99
Linux (Nice Level) Priorität	-20 bis 19
RTK Priorität	0 bis 14

3.2 Memory Management

Es gibt verschiedene Speicherbereiche

Linux/System/Programm Speicher Der Speicher, den Linux und Programme belegt haben. Dieser Speicher ist intern in viele Teile aufgeteilt. (DMA, ...)

LRT-Heap Speicher Speicher den der LRT verwendet, oder welcher über ein CIL Funktionen angefordert wird.

App Heap, App Code, ...

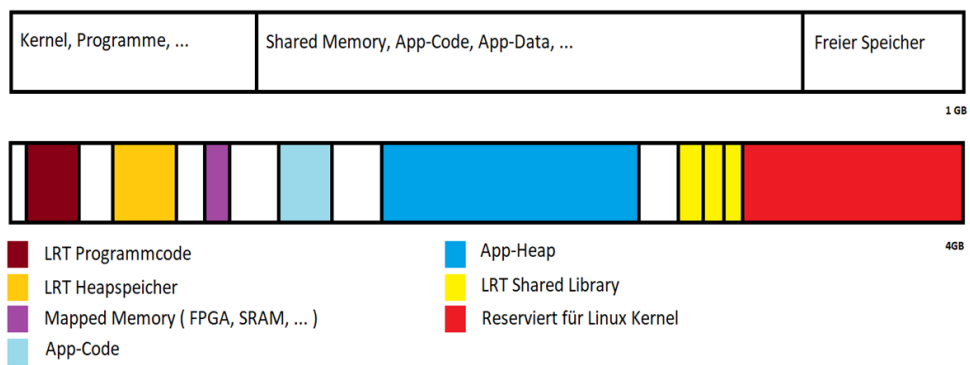


Figure 1: Lang

4 Results

5 Discussion

6 Summary and Outlook

I am using [1] for citing.

Bibliography

- [1] Clarke Brian Blackadar. “Historical Review of the Causes of Cancer”. In: *World Journal of Clinical Oncology* 7.1 (Feb. 10, 2016), pp. 54–86. ISSN: 2218-4333. DOI: [10.5306/wjco.v7.i1.54](https://doi.org/10.5306/wjco.v7.i1.54). pmid: [26862491](https://pubmed.ncbi.nlm.nih.gov/26862491/).
- [2] *Xenomai Real-time Linux*. URL: <https://xenomai.org/> (visited on 03/21/2024).

List of Figures

Figure 1 Kurz	8
-------------------------	---

List of Tables

Table 1 Kurz	6
Table 2 Kurz	7
Table 3 Übersicht der Prioritätsgruppen und ihrer Beziehungen	7

List of Code

List of Abbreviations

ABC Alphabet

WWW world wide web

ROFL Rolling on floor laughing

A Anhang A

B Anhang B