Tim Sagaster, BSc

# RavenOS

A Real Time Operating System based on SmartOS

**Bachelor's Thesis**

to achieve the university degree of

Bachelor of Science

Bachelor's degree programme: Computer Science

submitted to

**Graz University of Technology**

Supervisor

Univ.-Prof. Dipl.-Inf. Univ. Dr.rer.nat. Marcel Carsten Baunach

Institute of Technical Informatics
Head: Univ.-Prof. Dipl.-Inform. Dr.sc.ETH Kay Uwe Römer

Graz, June 2021

## Abstract

The founding of the Aero Space Team Graz (ASTG) at the end of 2019, with the goal to develop and manufacture rockets, necessitated the development of a Real Time Operating System (RTOS). This thesis follows the steps needed to develop a functioning RTOS called RavenOS. RavenOS is based on SmartOS, which is developed by the Embedded Automotive Systems (EAS) group at TU Graz, and aims to provide the same Application Programming Interface (API). The implementation of the kernel differs in many parts due to the different architectures. RavenOS uses events and resources to provide synchronization for multi tasked applications. The RTOS will be used as the foundation for further development to fulfill the needs of amateur rocketry.

## Acknowledgements

This thesis was written in the academic year 2020/2021 at the Embedded Automotive Systems (EAS) group at the Institute of Technical Informatics (ITI) at TU Graz.

I would first like to thank my supervisor Professor Marcel Baunach, whose expertise was invaluable and who made this thesis possible by allowing me to use resources from lectures and laboratories held by his group. I would further thank all of his colleagues at the EAS group, who helped me with any problems I encountered.

I would like to express my gratitude towards all of the members of the Aero Space Team Graz (ASTG), who work tirelessly on the projects that give this thesis practical use.

Lastly, I want to thank my girlfriend, family and friends, who had to endure endless conversations about this thesis and problems I encountered while developing RavenOS, for their patience and helping me to rest my mind when I was not writing.

Graz, June 2021 Tim Sagaster

# Contents

# Chapter 1

# Introduction

This thesis describes the process of writing an RTOS for the ARMv7-M architecture. The RTOS created has the name RavenOS and is based on SmartOS, more precisely on the SmartOS version that is recreated by Students in the Real-Time Operating Systems Laboratory (448.026) held by the EAS Group at the ITI at TU Graz (RTOS Lab).

## 1.1 Motivation

At the end of 2019, the Aero Space Team Graz (ASTG)[1] was founded. The goal of this interdisciplinary team of university students in Graz is to develop, manufacture and test rockets, intending to compete in the European Rocketry Challenge (EuRoc)[2] in Portugal and the Spaceport America Cup (SAC)[3] in New Mexico.

This team's purpose is not only to take part in competitions but also to pursue innovation and research, therefore it was decided to develop an in-house RTOS which can be used in the varying projects, instead of using a third party RTOS. To eliminate the problem of porting the RTOS to multiple architectures, a common Micro Controller Unit (MCU) was selected. The decision was made to use the dual core MCU STM32H745ZI from STMicroelectronics[4] with a Cortex-M7 and a Cortex-M4 core. This MCU might be a bit excessive for most tasks, but it should provide most functionality the team could ever need. For now, only the more powerful Cortex-M7 core of the MCU is used for simplicity.

Taking advantage of this situation, I contacted Professor Marcel Baunach for a potential bachelor thesis. Being the head of the EAS Group at the ITI at TU Graz, the very same group that holds the RTOS Lab course, he was the ideal supervisor for this project. He agreed to supervise this thesis, generously allowing me to use resources of the RTOS Lab. The RTOS Lab uses the MSP430 MCU from Texas Instruments which is based on a different processor architecture.

[1] `www.astg.at`
[2] `www.euroc.pt`
[3] `www.spaceportamericacup.com`
[4] `www.st.com`

## 1.2 Objective

The aim of this thesis is to implement the basic functionality of an RTOS. Therefore, the lab guide[2] from the RTOS Lab will be followed and the functionalities described there will be implemented on this architecture. In addition, a documentation for RavenOS will be made to aid further development of the RTOS by future members of the ASTG.

## 1.3 Structure

The basic structure of this thesis will follow the structure of the lab guide[2].

First, in **Chapter 2**, a general overview of the design philosophy of RavenOS is given.

In **Chapter 3**, the implementation of the system timer and the general configuration of the MCU is explained.

In **Chapter 4**, the layout of the relevant memory sections of the MCU is shown. The data structures needed for tasks are outlined, this includes the TCB, task queues and the individual task stacks.

In **Chapter 5**, the way the ARM architecture handles exceptions and how the RTOS uses this, to enter and exit the kernel is described.

In **Chapter 6**, the different exception types that can be used to enter kernel mode are explained.

In **Chapter 7**, events and their Control Block (CB) and SVCalls are described.

In **Chapter 8**, the CB and SVCalls of resources are explained. In addition the scheduler protocol used in RavenOS is explained in detail.

Lastly, in **Chapter 9**, a short summary is given. Furthermore a future outlook on how the RTOS will be used and improved upon is outlined.

## 1.4 Resources

As mentioned before, the lab guide[2] from the RTOS Lab was used as foundation for the development process. Additionally, some parts of the framework of the RTOS Lab were used and adapted. This includes mainly the data structures used within the kernel and some macros used to place these structures in the correct memory sections.

For information on how the Cortex-M7 works, the programming manual[4] was used.

To correctly configure the STM32H745ZI, the corresponding data sheet[3] was used.

For information on the subroutine call standard of the ARM architecture, the documentation [1] from ARM was used.

## 1.5 Tools

The STM32CubeIDE 1.4.2[5] from STMicroelectronics was used for the development process. This provided powerful debugging capabilities as well as the possibility to automatically generate basic code for initializing and setting up the MCU correctly.

The STM32CubeIDE comes with an integrated toolchain to compile the project. The version used for this thesis was "GNU Tools for STM32 7-2018-q2-update", which is a

[5] `www.st.com/en/development-tools/stm32cubeide.html`

patched version by STMicroelectronics of the "GNU ARM Embedded 7-2018-q2-update" toolchain.

For the development process, the development board Nucleo-H745ZI-Q[6] from STMicroelectronics was used.

For generating the documentation in Appendix C, Doxygen version 1.9.1 from `www.doxygen.nl` was used.

For debugging and generating Figure 8.2 in Chapter 8, the Logic Analyzer from AZ-Delivery[7] was used together with the Logic software from Saleae [8].

[6] `www.st.com/en/evaluation-tools/nucleo-h745zi-q.html`
[7] `www.az-delivery.de/en/products/saleae-logic-analyzer`
[8] `www.saleae.com/downloads/`