Mendel University in Brno
Faculty of Business and Economy

MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947

Bachelor thesis

Supervisor:
Ing. Jan Kolomazník, Ph.D.

Dmitrii Titarenko

Brno 2025

Acknowledgment

I would like to express my gratitude to everyone who contributed to completion of this thesis.
I would like to thank my thesis supervisor, Ing. Jan Kolomazník, Ph.D. , for his guidance and support. Furthermore, I express my gratitude to Zbynek Fedra Ph.D. for guiding me during the creation of this thesis.
I am also very grateful to all the professors in the Department of Informatics at Mendel University for their teaching, openness, and constant support throughout my study.
A big thank you to family members and friends for their support and encouragement.
Finally, I express gratitude to the people who directly or indirectly contributed to the creation of this work. To those who came before me and on whose shoulders I stand.

Declaration

I hereby declare that this thesis entitled **MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947** was written and completed by me. I also declare that all the sources and information used to complete the thesis are included in the list of references. I agree that the thesis could be made public in accordance with Article 47b of Act No. 111/1998 Coll., Higher Education Institutions and on Amendments and Supplements to Some Other Acts (the Higher Education Act), and in accordance with the current Directive on publishing of the final thesis.

I am aware that my thesis is written in accordance to Act No. 121/2000 Coll., on Copyright and therefore Mendel University in Brno has the right to conclude licence agreements on the utilization of the thesis as a school work in accordance with Article 60(1) of the Copyright Act.

Before concluding a licence agreement on utilization of the work by another person, I will request a written statement from the university that the licence agreement is not in contradiction to legitimate interests of the university, and I will also pay a prospective fee to cover the cost incurred in creating the work to the full amount of such costs.

Brno 03.08.2025 .....

signature

Abstrakt

Titarenko, D. MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947. Bachelor thesis. Brno, 2025.

Práce zkoumá podporu MicroPython na Zephyr RTOS s využitím vývojové desky FRDM-MCXN947 od NXP. Samo o sobe Zephyr RTOS poskytuje širokou podporu a snadno použitelné API pro mnoho embedded zařízení a jejích periferii, ale podpora MicroPython je stále limitovaná a nekonzistentní s MicroPython porty vyvíjený pro jíná zařízení. Práce analyzuje současné limity MicroPython na Zephyr RTOS a vybírá funkčnost pro implementaci. Implementace zahrnuje inicializaci vývojovýho prostředí pro Zephyr a MicroPython, které bude upravené pro FRDM-MCXN947, provádění funkčního testování periferii vývojové desky v vývojovém prostředí pro Zephyr a MicroPython a porovnáním kompatibility nativného poru a Zephyr portu MicroPython.

Klíčová slova: MicroPython, závěrečná práce, Zephyr RTOS, FRDM-MCXN947

Abstract

Titarenko, D. MicroPython Utilizing Zephyr Port and NXP FRDM-MCXN947. Bachelor thesis. Brno, 2025.

This thesis explores MicroPython support on Zephyr RTOS using the NXP FRDM-MCXN947 development board. Zephyr RTOS itself provides extended support and easy-to-use APIs to many embedded devices and their peripherals, but the support of MicroPython remains limited and not consistent with MicroPython ports developed for other devices. This work analyzes the current limitations of MicroPython on Zephyr RTOS and selects specific functionality for implementation. The implementation involves initializing Zephyr and MicroPython development environments adjusted to FRDM-MCXN947, conducting functional testing of the development board peripherals with Zephyr and MicroPython environment, and comparing the compatibility of native and Zephyr ports of MicroPython.

Key words: MicroPython, thesis, Zephyr RTOS, FRDM-MCXN947

Contents

List of Figures

1 Introduction

The world of microcontrollers and embedded devices continues to grow, and such devices become more common with every day. Even though they are seldom noticeable we more often might find ourselves surrounded by them. From home appliances, to cars, to factory machines to city-wide networks embedded devices reach wide and deep in our lives.

But with the growing count of embedded devices grows complexity of functions they implement. Hence arises a need for Operating Systems(OSs) to manage sets of complex programs and provide a layer of abstraction to ease development in such constrain but demanding environments.

To cover demand of an Operating System in embedded devices the Zephyr Real Time Operating System (RTOS) was created. Zephyr RTOS is an open-source operating system with build-in security and optimization for resource limited devices. Zephyr kernel supports ARM, Intel x86, ARC, RISC-V, Nios II, Tensilica Xtensa and large number of development boards, among others NXPs FRDM-MCXN947. ALso Zephyr has rich API that allows developers to write high-level code for embedded devices.

Writing software for embedded devices is still a complex task regardless of what underlying technologies are used. Writing it in a language such as C adding an additional complexity due to need of managing program memory allocation and deallocation by hand. Leaving unhandled memory sector could lead to memory leaks or worse opens an opportunity for an attacker to execute malicious code on an embedded device. The consequences of such problems become much grater when occurring in the embedded world. MicroPython is an optimized subset of Python 3 programming language for embedded devices. It aims to ease writing software by managing memory using its garbage collector system, using easy to read Python-like syntax and providing various modules to enable work with different peripherals. Additionally, MicroPython allows for code portability, meaning that code written for FRDM-MCXN947 could be ported and ran on ESP32 with minimal updates to code.

But MicroPython does not support every single device straightaway – ported versions of MicroPython are submitted to the MicroPython repository, by a manufacturer or an enthusiasts. Later submitted port will be reviewed and tested by MicroPython maintainers, which is a lengthy process, for example MicroPython port for Zephyr was under review for 2 years.

By combining Zephyr RTOS and MicroPython in one technological stack the best of both technologies could be utilized. Potential exists for developers to write highly readable, easy-to-understand and efficient code with MicroPython that make use of various hardware support introduced by Zephyr RTOS. Yet the state of this development environment is not yet firm and have plenty of rough edges and unapparent problems that could arise during the process of software development.

The aim of this thesis is to construct a method for configuring the development

environment for MicroPython, Zephyr RTOS and FRDM-MCXN947 development board, provide insight of compatibility challenges of both platforms and propose potential solutions for extending MicroPython port to Zephyr RTOS. The finding of this thesis will contribute to understanding of the state of both platforms and their integration, informing future developers and increasing usability of MicroPython and Zephyr.

1.1 Goal of this thesis

Goal of the thesis is to establish development environment and workflow for developing applications for embedded devices with the MicroPython Zephyr port. This work could be used in future to ease start of application development and as a reference.

2 Background

This chapter introduces the reader to an information about Zephyr RTOS, MicroPython and FRDM- MCXN947 board that is needed for understanding this thesis.

2.1 Zephyr RTOS

Zephyr is an Operation System designed for resource-constrained and embedded system from simple sensors to smart industrial embedded solutions with emphasis on safety. It supports a broad list of embedded devices, development boards and peripherals. Zephyr offers extensive number of features and services including multithreading, inter-thread data passing, inter-thread synchronization, dynamic memory allocation, interrupt service, power management, networking, file system. Zephyr project is open-source, distributed under Apache 2.0 license and was created under Linux Foundation organization.(Zephyr Project, Introduction, 2024)

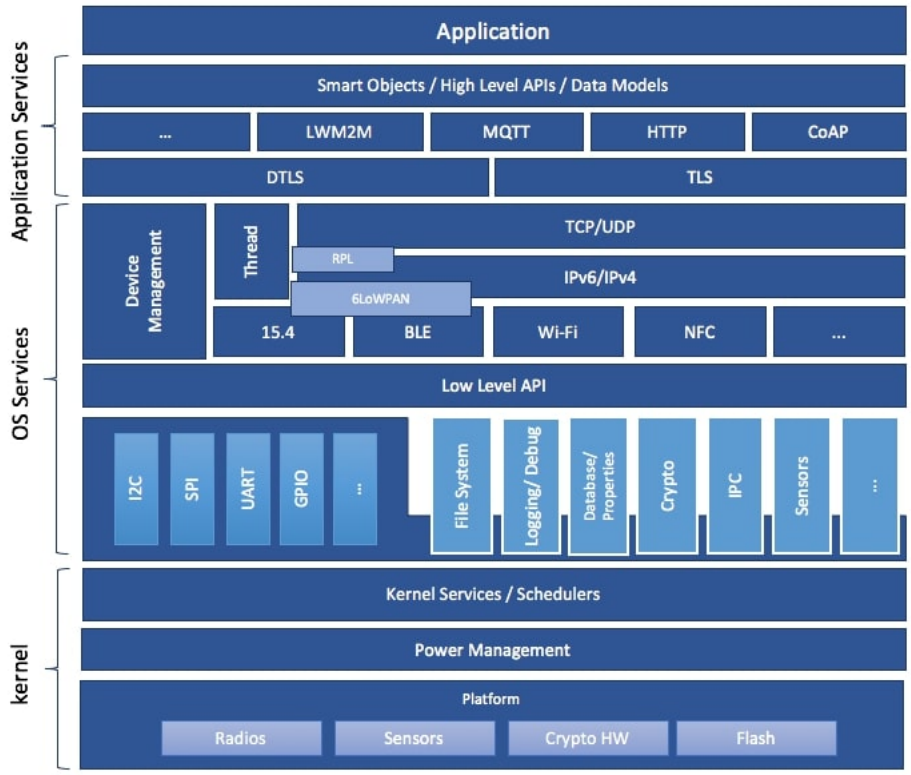

Figure 1: Zephyr System Architecture
Source: (Zephyr Project, Zephyr Security Overview, 2024)

2.1.1 West

West is a part of Zephyr's tool-chain used for building and configuring. West can initiate Zephyr workspace from official upstream repository, update or change version of a local Zephyr workspace to any version in official repository, build Zephyr application from source, flash built application to a board.(Zephyr Project, West (Zephyr's meta-tool), 2024)

2.1.2 Kconfig

Kconfig is Zephyr's kernel, peripheral drivers and subsystems configuration system that allow to configure Zephyr at a build time. Kconfig goal is to enable configuration without introducing changes to the source code.

The initial board configuration can be found in **<board>_defconfig** files. For example configuration file for FRDM-MCXN947 is located at **boards/nxp/frdm_mcxn947/ frdm_mcxn947_mcxn947_cpu0_defconfig**. The board configuration for NXPs' FRDM-MCXN947 is as follows:

Listing 1: FRDM-MCXN947 Kconfig configuration

```
1 CONFIG_CONSOLE=y
2 CONFIG_UART_CONSOLE=y
3 CONFIG_SERIAL=y
4 CONFIG_UART_INTERRUPT_DRIVEN=y
5 CONFIG_GPIO=y
6 CONFIG_PINCTRL=y
7 CONFIG_ARM_MPU=y
8 CONFIG_HW_STACK_PROTECTION=y
9 CONFIG_TRUSTED_EXECUTION_SECURE=y
```

Kconfig values can be set to a **<board>_defconfig** files, temporarily with terminal graphical interfaces or with a **prj.conf** file at application level which overrides the initial configuration during application build.(Zephyr Project, Configuration System (Kconfig), 2022)

2.1.3 Devicetree

Devicetree is a data structure to describe hardware. It is a community driven standard that is heavily used in Zephyr project. In Zephyr devicetrees are usually build inherently meaning that for example FRDM-MCXN947 has a devicetree configuration **board/nxp/frdm_mcxn947/frdm_mcxn947_mcxn947_cpu0.dts** which mainly enables peripheral devices, but includes FRDM-MCXN947 specific configuration from **frdm_mcxn947.dtsi** (include file), which in turn includes **frdm_mcxn947-pinctrl.dtsi** file that mostly defines pinmux groups. Additionally the **frdm_mcxn947_mcxn947_cpu0.dts** includes **nxp_mcxn94x.dtsi** file that defines memory ranges for SRAM, FLEXSPI and peripherals and includes

nxp_mcxn94x_common.dtsi include file where most of devices including CPU, GPIO, CTIMER and others are defined and assigned memory ranges.(devicetree.org, Devicetree Specification Release v0.4, 2023)

Same as Kconfig Devicetrees can be overwritten or have some specific devices configured differently with *overlay* files, which as well needs to be placed in build directory, from there *west* tool will use it to edit the Devicetree configuration.

2.2 MicroPython

MicroPython is an open-source project founded by Damien George. MicroPython is an implementation of the Python programming language that is optimized to be run on embedded and resource constraint devices. It implements the entire Python 3.4 syntax with some selected features from the later versions such as *async/await* from Python 3.5 , additionally on par with Python it uses garbage collection system for memory management. MicroPython final build include a compiler that compiles MicroPython code to bytecode and an runtime interpreter of the compiled bytecode. Programs could be written directly to the MicroPython REPL(Read–eval–print loop) or be loaded onto MicroPython host device with use of serial connection and utility programs like *ampy*.

MicroPython's core development is focused on implementing and maintaining core features of the MicroPython like Python language features, libraries, memory management and MicroPython interpreter. The responsibility for adapting and porting MicroPython to different platforms lies on the community around it. Every MicroPython port introduces required adaptations and addresses hardware features and limitations of its platform. Consequently, MicroPython support is not linear on all platforms, because some might lack the necessary configuration for enabling a part of functionality or even lack reimplementation of a number of core libraries. Additionally, the slow pace of adding to source code features for various platforms created by the community means that even fully functional and tested ports or features might wait for months before being reviewed. Despite all of this, there are already many supported devices and architectures that MicroPython can run on. MicroPyton has additional support to be run on operating system Zephyr RTOS and on OSes from UNIX family, as well as experimental Windows port.

MicroPython remains in beta-stage, hence it is a subject to possible API and code-base changes in the future.(Nicholas H. Tollervey, 2017)

2.3 FRDM-MCXN947

The FRDM-MCXN947 is a low-cost development board designed by NXP semiconductors. FRDM-MCXN947 integrates Dual Arm Cortex-M33 microcontroller, a neural processing unit, P3T1755DP I3C temperature sensor, TJA1057GTK/3Z CAN PHY, Ethernet PHY, SDHC circuit, RGB LED, touch pad, high-speed USB, MCU-Link debuger, push buttons and has an option to be extended with external

devices. (NXP semiconductors, UM12018 FRDM-MCXN947 Board User Manual, 2024)

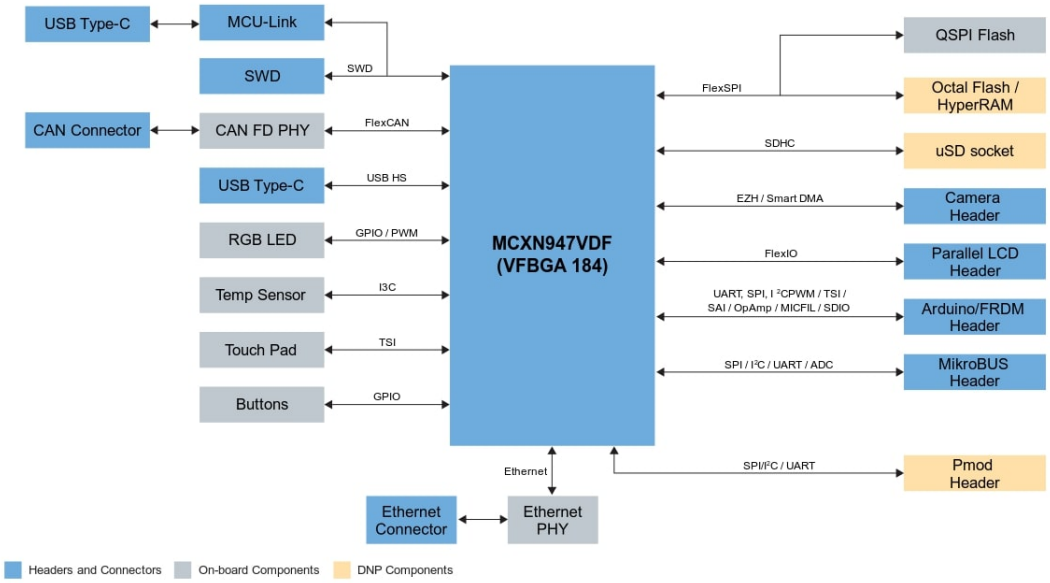

Figure 2: FRDM-MCXN947 Block diagram
Source: (NXP semiconductors, FRDM Development Board for MCX N94/N54 MCUs , 2025)

2.3.1 Signal Multiplexing

FRDM-MXCN947 enables use of several functions for different pins by utilizing Signal Multiplexing. For example pin **P0_10** which is an red RGB pin can use **GPIO** functionality directly, **FLEXCOMM** by utilizing **FC0_P6** FLEXCOMM device, **CTIMER** by utilizing **CT0_MAT0** CTIMER device, and **FLEXIO** functionality by utilizing **FLEXIO0_D2** device.

Only one function can be used at a time on a pin and only one pin can be assigned to a peripheral device. (NXP semiconductors, MCX Nx4x Reference Manual, 2025)

2.3.2 LinkServer

LinkServer is an NXP command-line utility that provides target flashing capabilities and firmware updates for FRDM-MCXN947. This is typically used as a backend for flashing FRDM-MCXN947 in *west* utility. (NXP semiconductors – LinkServer for Microcontrollers, 2025)

2.4 MicroPython port to Zephyr

While Zephyr RTOS provides a feature-packed and expandable development and system to be used in embedded world standard development in C can be time consuming. MicroPython's Zephyr port brings advantages of both MicroPython and Zephyr to single environment, allowing to write hight-level Python like code and rapidly prototype and debug, while also leveraging hardware agnostic Zephyr APIs and wide support of different embedded devices and their peripherals.

But despite growth in support between Zephyr and MicroPython there are still features that lack in the port and issues with coupling of both technologies. For a long time the MicroPythons' Zephyr port have been using an older versions of Zephyr and MicroPython itself. It used MicroPython 1.19.1 and Zephyr 3.1.0 versions which both came out in period between May and June 2022 until September 2024. In September 2022 began work by Maureen Helm to introduce a CI pipeline into MicroPython repository to ease porting MicroPython to latest Zephyr release. From this work emerged last MicroPython Zephyr port version based on MicroPython 1.24.0 and Zephyr 3.7.0.

And though MicroPython Zephyr port already supports the MicroPython modules like socket, time, math, machine and other are implemented and usable in the final MicroPython build they may lack support of some sub-modules like machine's PWM sub-module or functionality of modules and sub-modules like not yet implemented features of machine's I2C sub-module that do not have ability to set clock and data lines.

A MicroPython's Zephyr port is built in a same way any Zephyr application is built. A *west* utility is used and MicroPython port to Zephyr source code as a build target. The final build could be configured and some features or peripherals could be activated or deactivated with *Kconfig* and a final devicetree be overwritten with devicetree *overlays*. Then the result binary file is fleshed to a target board with *west* utility.

2.5 Summary

For utilizing MciroPython Zephyr port on FRDM-MCXN947 board it is needed to build MicroPython port as a Zephyr application, and then flash this application onto the development board, both operations are made with use of Zephyr's *west* utility. Program flashing is made using *LinkServer*.

Pre-build configuration is possible using *Kconfig*, for setting what peripherals and sub-system are to be enabled or disable, and *Devicetree*, for creating or updating a structured description of the underling hardware.

FRDM-MCXN947 is a programmable and extendable development board with many devices, peripherals and systems available.

3 Methodology

This chapter introduces the reader to the set of tools used in this thesis and outlines a methodological approach.

3.1 Zephyr development environment setup and testing

This section will detail the process of setting up a development environment for working with Zephyr RTOS. The setup will be broken into several steps to provide clear and easy-to-follow instructions to the reader.

Firstly, the prerequisites and dependencies would be introduced along with the installation process. A note of warning of potential issues or of an alternatives for some dependencies would be provided. The reader will be briefed on purpose of the key dependencies. After the necessary preparation the Zephyr source code will be obtained and two methods for obtaining it will be presented and discussed. Then the *west* tool will be described in detail and it's abilities for managing the project will be discussed. Next, the installation and use of the *LinkServer* utility and its alternatives will be discussed. Finally, the development setup will be verified with a functional test and test prerequisites discussed.

4 References

Zephyr Project, Introduction [onlne], 2024 Available from: https://docs.zephyrproject.org/latest/introduction/index.html.

Zephyr Project, West (Zephyr's meta-tool) [online], 2024 Available from: https://docs.zephyrproject.org/latest/develop/west/index.html.

Zephyr Project, Configuration System (Kconfig) [onlne], 2022 Available from: https://docs.zephyrproject.org/latest/build/kconfig/index.html.

NXP semiconductors, MCX Nx4x Reference Manual, 2025 MCXNX4XRM.

devicetree.org, Devicetree Specification Release v0.4, 2023.

Nicholas H. Tollervey *Programming with MicroPython: embedded programming with microcontrollers and Python*. O'Reilly Media, Inc., 2017. ISBN 978-1-491-97273-1.

NXP semiconductors, UM12018 FRDM-MCXN947 Board User Manual, 2024.

NXP semiconductors, LinkServer for Microcontrollers [online], 2025 Available from : https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/linkserver-for-microcontrollers:LINKERSERVER.

NXP semiconductors, FRDM Development Board for MCX N94/N54 MCUs [online], 2025 Available from: https://www.nxp.com/design/design-center/development-boards-and-designs/FRDM-MCXN947.

Zephyr Project, Zephyr Security Overview [onlne], 2024 Available from: https://docs.zephyrproject.org/latest/security/security-overview.html.