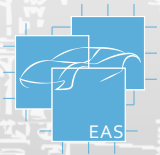
A Study on the Portability of IoT Operating Systems

Renata Martins Gomes and Marcel Baunach



Renata Martins Gomes
renata.gomes@tugraz.at

Institute of Technical Informatics
Embedded Automotive Systems Group
Graz University of Technology

March 2021

Agenda

1. Introduction
2. Portability Aspects and Assessment
3. Porting Experiences in the Literature
4. Conclusion and Outlook

Introduction
Motivation

Internet of Things (IoT):

- New super-infrastructure
  - Health monitoring
  - Cars and traffic
  - City management
- Billions of devices
- Growing variety of
  - Hardware architectures
  - Communication stacks
  - Programming paradigms

IoT Operating System:

- Run on several platforms
- Guarantee dependability
  - Safety
  - Security
  - Real-time
  - Maintainability
  - ...

Introduction
Definitions

Port: realization of software (OS) for a specific target environment:

- *Implementing* specification (from scratch)
- *Porting* (adapting) existing port

Software is *portable* if

- effort of porting $<$ effort of implementing
- lower porting effort $\Rightarrow$ more portable software

Low-level software

- Environment is hardware platform
- Recompiling code is not enough
- Rewriting unavoidable

Introduction

How portable really **are** the existing IoT OSs?

Factors that affect porting effort beyond the software

- Expertise on the software
- Expertise on the target platform
- New target vs. supported targets

Port must guarantee

- Functional behavior
- Non-functional properties

We consider

- Design
- Development process
- Testing process
- Quality of available ports

Agenda

1. Introduction
2. Portability Aspects and Assessment
3. Porting Experiences in the Literature
4. Conclusion and Outlook

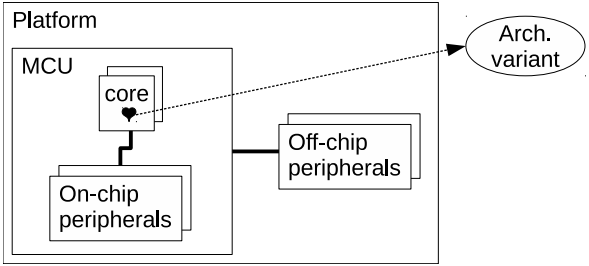
Portability Aspects

Five open-source OSs commonly used in IoT and related domains.

- How is the HAL specified?
- Available ports
  - Incomplete?
  - Unimplemented features?
  - Non-functional differences?
- How easily can low-level code be reused?
  - Source code organization
  - Common code vs. code replication
- How are ports tested?
  - Checked for completeness and correctness
  - Specifications or requirements provided

Portability Aspects

high number
≠
easily portable



OS	Arch.	Arch. variants	MCUs	Platforms
FreeRTOS	23	49	106	117
RIOT	7	15	39	167
seL4	3	10	20	23
Contiki-NG	5	6	8	28
ERIKA3	7	10	18	22

low number
≠
hard to port

Portability Assessment
FreeRTOS

Supports 23 architectures, but...

- Code separated by compilers
- Little code sharing
- Outsources vital OS functionality to application
- Lacks specification

“Porting to a completely different and as yet unsupported MCU is not a trivial task.”

Portability Assessment
RIOT

Highest number of supported platforms...

- Very well structured
- Porting specification
- API for driver communication
- Test system

Still, ports have many open issues.

“this implementation needs major rework”

“there are some inconsistencies throughout the family...”

Portability Assessment
seL4

Only supports 3 architectures...

- Formal specification
- Well structured code
- Verification, testing, benchmarking

Ports have high quality, there is a trade-off.

Portability Assessment
Contiki-NG

Simplicity might make things easy

- Ports have high quality
- Sound processes of testing and benchmarking
- Porting well documented

... and yet...

- Few supported MCUs (8)
- Inconsistencies in code organization

“(application) developer must make sure that processes do not keep control for too much time and that long operations are split into multiple process schedulings”

Portability Assessment
ERIKA3

Automotive OSEK, multicore support

- Source code organization inconsistent
- Automatic testing
- Unimplemented features
- Potential critical bugs

Agenda

1. Introduction
2. Portability Aspects and Assessment
3. Porting Experiences in the Literature
4. Conclusion and Outlook

Porting Experiences in the Literature

Confirm that porting is challenging

- Complex target architectures
- Specialized devices
- Timing dependencies
- Low-level code
- Lack of expertise on OS or hardware

Porting as source of bugs

- High error rate in drivers
- Copy + paste
- Maintenance of replicated code

Potential solutions

- Better documentation and specification
- Verification and testing
- Automate porting

Agenda

1. Introduction
2. Portability Aspects and Assessment
3. Porting Experiences in the Literature
4. Conclusion and Outlook

Conclusion and Outlook

Porting is **not** trivial and is a source of **bugs**!

Good practices

- Good code structure
- Well-structured testing/benchmarking
- Large developer community
- Specification and documentation

For a **dependable** IoT, we must go beyond
→ Formal methods and code generation

- Scalability (code and model size)
- Shifts the effort to formalization and proofs

A Formal Modeling Approach for Portable Low-Level OS Functionality
RM Gomes, B Aichernig, M Baunach
SEFM 2020

Thank you!