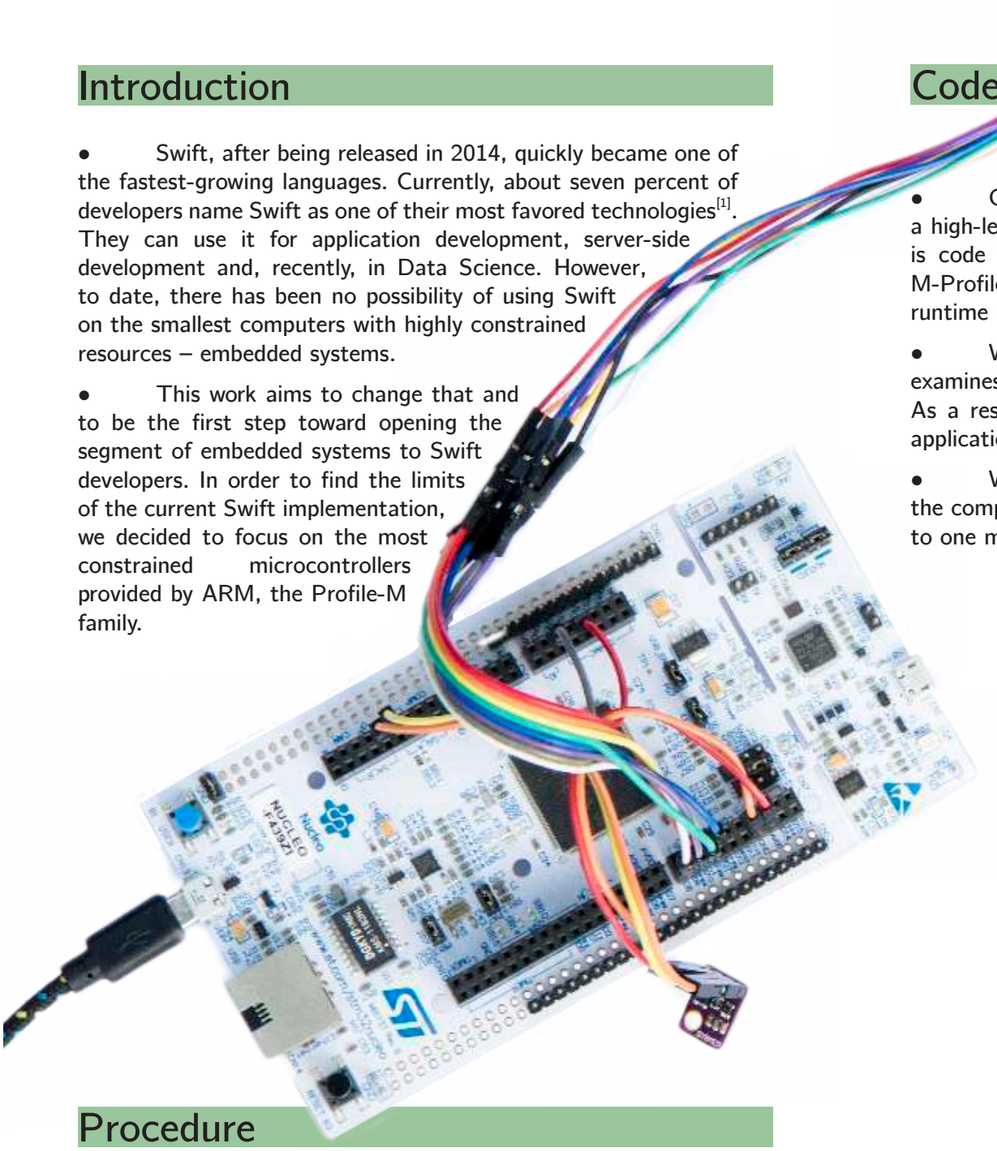
SWIFT FOR EMBEDDED SYSTEMS

Alan Dragomirecký – Petr Máj (Supervisor) – Czech Technical University – Faculty of Information Technology

Introduction

- Swift, after being released in 2014, quickly became one of the fastest-growing languages. Currently, about seven percent of developers name Swift as one of their most favored technologies[1]. They can use it for application development, server-side development and, recently, in Data Science. However, to date, there has been no possibility of using Swift on the smallest computers with highly constrained resources – embedded systems.
- This work aims to change that and to be the first step toward opening the segment of embedded systems to Swift developers. In order to find the limits of the current Swift implementation, we decided to focus on the most constrained microcontrollers provided by ARM, the Profile-M family.



Procedure

- We added a new *BareMetal* platform to the Swift compiler, ported the Swift Runtime, Swift Standard Library, and other required libraries.
- To ensure that everything works after the extensive changes to the whole toolchain, we ported part of the test suite, performing the tests within the QEMU emulator.
- In order to make our solution familiar to Swift developers and to leverage existing tools, we also added support to the Swift Package Manager.

Code-Size Reduction

- One of the biggest challenges when porting a high-level language like Swift to embedded systems is code size. The memory sizes of commonly available M-Profile microcontrollers end at two megabytes. The compiled runtime and standard library itself comprised over five megabytes.
- We created a new tool for program's code-size analysis. It examines object files and performs simulation of the linking process. As a result, it creates a dependency graph of different parts of the application for further analysis.
- With the aid of this tool, new features were implemented to the compiler, enabling code-size reduction of the standard library close to one megabyte.

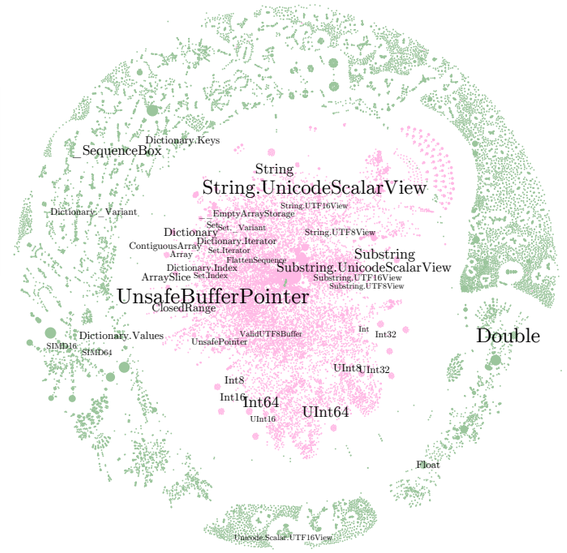


Figure 1: Dependency graph visualization of a simple Swift application. Green areas are those discarded, due to newly implemented compiler features.

Evaluation

- While code size is still higher than on other platforms used for embedded development, it is not a major issue. On the other hand, our results show that the use of Swift for Embedded Systems can result in comparable performance to that of a C-based platform, while making use of all the high-level features Swift offers (See Figure 2).
- From a developer perspective, the use of our solution should feel very familiar and be comparable to the development of a Swift application for Linux - mostly thanks to the support for the Swift Package Manager.

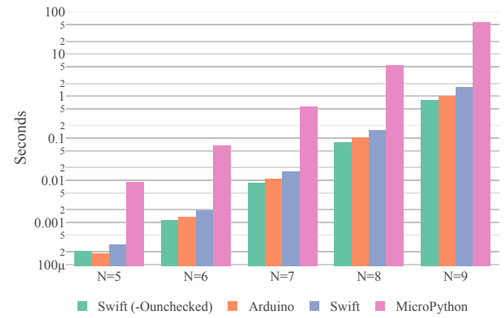


Figure 2: Performance comparison of different embedded platforms running the Fannkuch benchmark.

Conclusion

As a result of this work, it is now possible to use Swift on embedded systems. We believe that Swift can represent a viable alternative to IoT platforms such as Arduino or MicroPython.

References

[1] StackOverflow – Developer Survey Results 2019. Available from https://insights.stackoverflow.com/survey/2019#technology.