Kurzfassung

Eingebettete Systeme und Mikrocontroller insbesondere sind heute ein integraler Bestandteil vieler Produkte. Verglichen mit PCs stellt die Softwarebereitstellung Entwickler*innen vor besondere Herausforderungen. Während PCs meist mit dem Internet verbunden sind und Betriebssysteme verwenden, welche das Laden von Software übernehmen, werden eingebettete Systeme häufig ohne Betriebssystem betrieben und verfügen über limitiertere Kommunikationsmittel.

In dieser Arbeit werden Softwarebereitstellungsmethoden mittels Debugging Schnittstellen und Embedded Bootloadern präsentiert. Embedded Bootloader sind selbst auf einem Mikrocontroller laufende Firmware, welche verfügbare Kommunikationsschnittstellen wie den Universal Serial Bus (USB) verwenden, um neue Programme zu empfangen und in den Flash Speicher des Mikrocontrollers zu programmieren. Es wird ein generischer Ansatz zur Bootloaderprogrammierung gewählt und ein Bootloader Programmierframework präsentiert, welches das Schreiben von erweiterbarem und anpassbarem Bootloadercode ermöglicht. Das Framework ist in der Rust Programmiersprache geschrieben, einer aufsteigenden Systemprogrammiersprache, die im Gegensatz zu C und C++ Speichersicherheitsmechanismen bietet und dennoch vergleichbare Geschwindigkeit und Speicherbelegung erreicht.

Stichwörter: USB, Eingebettete Systeme, Firmware Upgrade Bootloader, Debugging Schnittstelle, Rust

Abstract

Embedded systems and microcontrollers in particular have become an integral part of many products. Compared to Personal Computers, deploying software for microcontrollers presents the developer with unique challenges. Unlike Personal Computers which are usually connected to the internet and run operating systems which handle loading of software, embedded systems often run no operating system and are limited to the communication interfaces provided by the microcontroller used.

In this thesis software deployment via debug interfaces and embedded bootloaders will be presented. An embedded bootloader is firmware running on a microcontroller which uses availabe communication interfaces like the Universal Serial Bus (USB) to receive new software and program it into the microcontrollers flash memory. A generic approach to bootloader design will be taken, presenting a bootloader programming framework which allows writing extensible and adaptable bootloader code. The framework is written in the Rust programming language, an upcoming systems programming language, which in contrast to the established C and C++, has memory safety mechanism while achieving similar levels of performance and memory footprint.

Keywords: USB, Embedded Systems, Firmware Upgrade Bootloaders, Debug Interfaces, Rust