Zürcher Hochschule
für Angewandte Wissenschaften

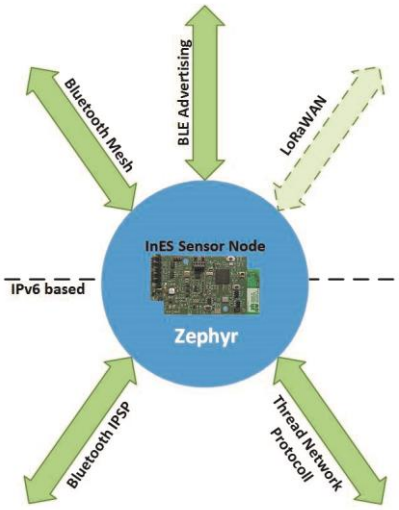
School of Engineering
InES Institute of Embedded Systems

Consolidating IoT Firmware Architectures with Zephyr

Diplomand/in
Benjamin Häring

Dozent/in
Andreas Rüst

Im Team IoT am Institute of Embedded Systems (InES) wird gegenwärtig Firmware meist projekt- oder microcontrollerbezogen implementiert. Dadurch kann Source Code häufig nur einmal verwendet werden, da eine Portierung auf einen anderen Microcontroller zu aufwändig wäre. Hinzukommt, dass die wiederholte Implementierung von grundlegenden Treibern wie I2C, SPI oder UART zeitintensiv ist und dadurch wertvolle Zeit verloren geht. Diese Problematik soll durch die Verwendung eines hardwareunabhängigen Real Time Operating Systems (RTOS) angegangen werden. Eines der vielversprechendsten RTOS ist «Zephyr Project» der Linux Foundation. Zephyr ist als Open Source-Projekt organisiert und erfährt grosse Unterstützung von namhaften Microcontroller-Herstellern wie Intel, NXP, STMicroelectronics und Nordic Semiconductor. Gerade Produkte der drei Letztgenannten werden im Team IoT oft verwendet.
Zu Beginn dieser Arbeit war das Zephyr RTOS am InES noch völlig unbekannt. Das nötige Wissen ist selbständig und in gezielten Arbeitspaketen während der Masterthesis erarbeitet worden. Den Start bildete die Portierung von Zephyr auf den InES-Sensorknoten, welcher vom Team IoT entwickelt wurde. Dazu wurden die notwendigen Board-Dateien implementiert und dem RTOS hinzugefügt. Daraus resultierend können viele der in Zephyr vorhandenen Beispielprojekte mit dem Sensorknoten verwendet werden. Eindrücklich ist hier die starke Hardwareabstraktion, welche das RTOS zur Verfügung stellt. Die fehlenden Sensortreiber konnten, basierend auf den generischen APIs von Zephyr, implementiert werden. Durch diesen Schritt können diese Treiber, ohne Anpassung einer einzelnen Codezeile, in Folgeprojekten eingesetzt werden, unabhängig vom gewählten Microcontroller.
Zusätzliche Energiemessungen zeigen auf, dass Zephyr über einen kompakten Bootprozess sowie ein effektives Power Management-System verfügt. Bei der Aktivierung dieses Systems verwaltet der Kernel selbständig den Energieverbrauch des RTOS. Der Idle-Thread überprüft, wann der nächste Task ansteht. Bei genügend hohem Zeitabstand wechselt der Kernel selbstständig in einen der Schlafzustände des Microcontrollers. Die Peripherie wird ebenfalls über dieses System verwaltet. Nicht benutzte Peripherie wird automatisch deaktiviert. Zusätzlich kann der Entwickler bei Bedarf Peripherie über dieses System suspendieren. Das System funktioniert einwandfrei und Messungen zeigen, dass es mit anderen Low-Power-Anwendungen des Instituts konkurrieren kann.



Systemübersicht mit allen getesteten Kommunikationsprotokollen für den InES-Sensorknoten in Kombination mit Zephyr

Zu weiteren Versuchen wurden Projekte mit LoRaWAN, OpenThread und Bluetooth Low Energy erfolgreich portiert, getestet und ausgemessen. Alle Resultate sind vorwiegend positiv ausgefallen und es konnten keine gravierenden Nachteile festgestellt werden. Zephyr deckt somit einen grossen Teil der Anwendungsgebiete im Team IoT ab, da die gewählten Kommunikationsprotokolle zu den meistgenutzten im Team gehören. Auffallend ist zudem das nützliche und umfangreiche Funktionenset des Kernels. Es beinhaltet Workqueues, Timer und Delays. Dieses Set ermöglicht eine rasche und einfache Portierung.

Das in der Masterthesis erarbeitete Wissen zu Zephyr kann vielfältig eingesetzt werden. Bereits jetzt wurde es als Entscheidungsgrundlage verwendet, um andere Projekte wie Bluetooth Mesh oder einen Security Workshop mit Zephyr zu realisieren. Dabei konnten die Vor- und Nachteile von Zephyr klar aufgezeigt werden. Das neu aufgesetzte zentrale Github Repository des Teams vereint zudem eigene Implementationen wie Boards oder Treiber, Codebeispiele sowie ein Wiki. Das gesammelte Wissen kann von jedem Zephyr-Entwickler im Team verwendet werden, unabhängig davon, welcher der unterstützen Microcontroller verwendet wird. Hinzukommt, dass mit Zephyr auf Linux, Windows sowie auch MacOS entwickelt werden kann. Entwickler können sich somit ohne grosse Hindernisse oder Abhängigkeiten untereinander unterstützen und diskutieren. Insgesamt zeigt die Arbeit, dass bestehende Anwendungen schnell und mit wenig Aufwand auf Zephyr portiert werden können. Die auf Zephyr portierten Anwendungen sind dabei sehr performant und energieeffizient.