Institut für Angewandte Forschung Hochschule Offenburg

forschung im fokus

Ausgabe Nr. 19 / 2016

Die Welt nachhaltig verändern – NaSiO, das Institut für nachhaltige Silikatforschung Offenburg

Physikalische Charakterisierung von Solarmaterialien

„Künstlerisch forschen in der Musik“ als KlangKomposition und RadioKunst

Institut für verlässliche Embedded Systems und Kommunikationselektronik

Prof. Dr.-Ing. Axel Sikora, Leiter des Instituts ivESK

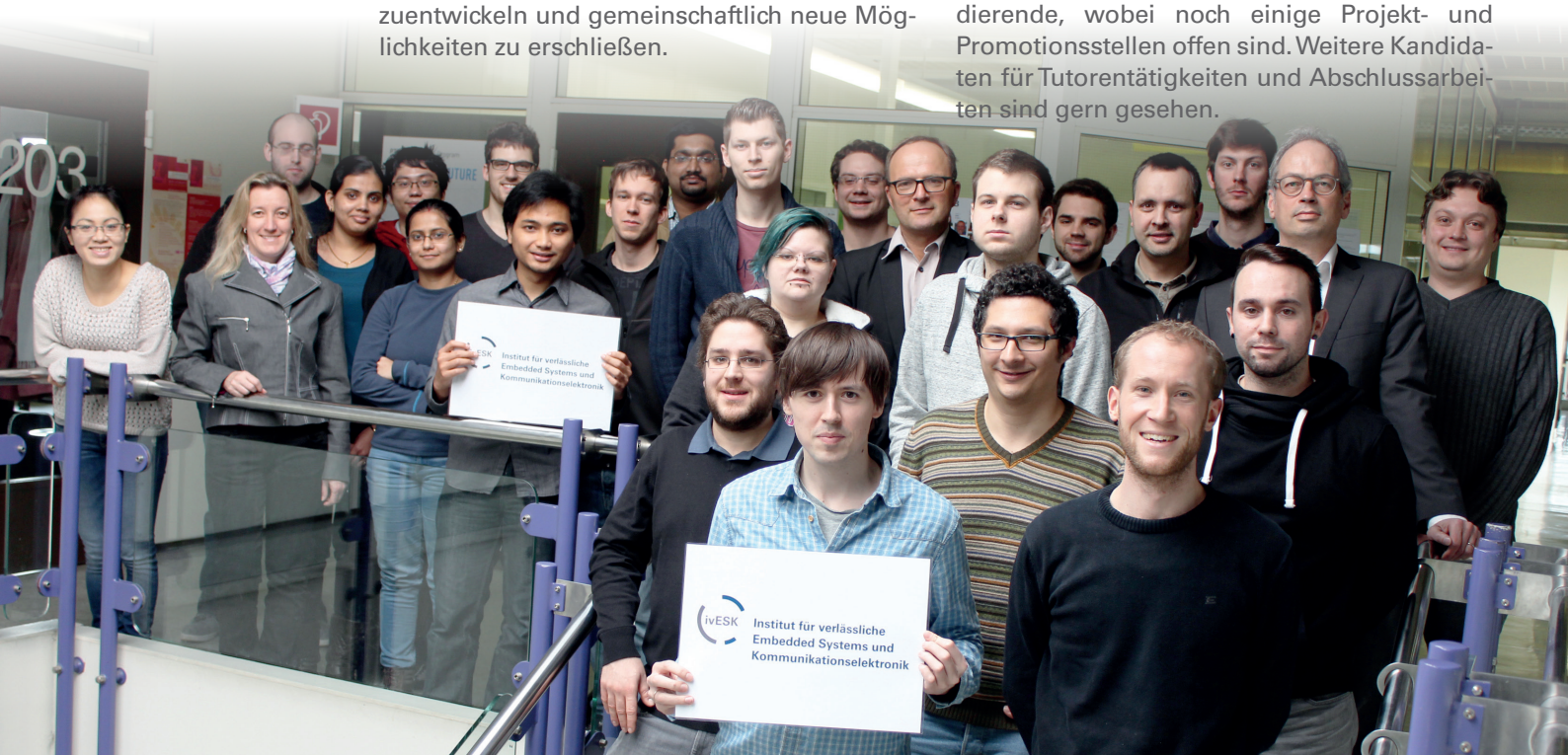
Das „Internet der Dinge“ durchdringt die industriellen und persönlichen Anwendungen zunehmend. Hierzu zählen beispielsweise Smart-Metering und Smart-Grid, Industrie- und Prozessautomation, Car-to-Car- bzw. Car-toX-Kommunikation, Heim- und Gebäudeautomation, Telehealth- und Telecare-Anwendungen. Die drahtgebundene und drahtlose Vernetzung von Embedded Systemen und deren Anbindung als sogenannte cyberphysische Systems (CPS) spielen hierbei eine immer wichtigere Rolle. Da auch immer mehr Systeme funktionskritische Aufgaben autonom übernehmen, gewinnen Zuverlässigkeit und Sicherheit immer mehr an Bedeutung. Entsprechend müssen die Aspekte der Datensicherheit und der Privatsphäre (Privacy) ebenfalls berücksichtigt werden.

Diesen Themen widmet sich das Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK) an der Hochschule Offenburg, das im Herbst 2015 von Prof. Dr.-Ing. Axel Sikora und Prof. Dr. rer. nat. Dirk Westhoff gegründet wurde, um die bislang sehr erfolgreichen Forschungs- und Entwicklungsarbeiten in den Laboren der beiden Professoren weiterzuentwickeln und gemeinschaftlich neue Möglichkeiten zu erschließen.

Hierbei stehen folgende Arbeitsgebiete im Zentrum der Aktivitäten:

- Konzeption und Implementierung von effizienten und modularen drahtgebundenen und drahtlosen Kommunikationsprotokollen unter Nutzung von Embedded Systemen – u. a. mit 6LoWPAN, Wireless M-Bus, M-Bus
- Konzeption und Implementierung von durchgängigen Sicherheitsarchitekturen für Kommunikationslösungen unter Nutzung von Embedded Systemen – u. a. embedded TLS 1.2, PKI-Lösungen für verteilte Anwendungen
- Konzeption und Implementierung von effizienten und sicheren embedded Rechnerplattformen – u. a. Embedded Linux (Speed-Boot, Virtualisierung)
- Test- und Verifikation von Kommunikationslösungen – u. a. mit dem automatisierten physischen Testbed (APTB) oder mit Netzwerksimulation und -emulation
- Ende-zu-Ende Sicherheitslösungen zwischen leistungsschwachen Geräten und leistungsstarken Komponenten sowie die Anbindung an Cloud-Ecosysteme

Am Institut arbeiten gegenwärtig 12 Vollzeitmitarbeiter sowie etwa ebenso viele Studierende, wobei noch einige Projekt- und Promotionsstellen offen sind. Weitere Kandidaten für Tutorentätigkeiten und Abschlussarbeiten sind gern gesehen.



emb::6 – Ein 6LoWPAN OpenSource-Projekt

Prof. Dr.-Ing. Dipl.-Ing. Dipl. Wirt.-Ing. Axel Sikora, Dipl.-Inf. (FH) Manuel Schappacher, Dipl.-Ing. (FH) Edgar Schmitt, Artem Yushev M. Eng., Fesseha Tsegaye Mamo M. Sc.

Das Internet der Dinge ist auf dem Vormarsch. Immer mehr mikroelektronische Systeme werden vernetzt. IPv6 über IEEE802.15.4, bekannt als 6LoWPAN, ist hierbei ein populäres Vernetzungsprotokoll. In diesem Artikel wird das Open Source Projekt emb::6 vorgestellt, das mit einer Reihe von Alleinstellungsmerkmalen eine effiziente, sichere, stabile und einfach zu nutzende Implementierung für übergreifende und medienunabhängige Vernetzungen im Internet der Dinge bereitstellt.

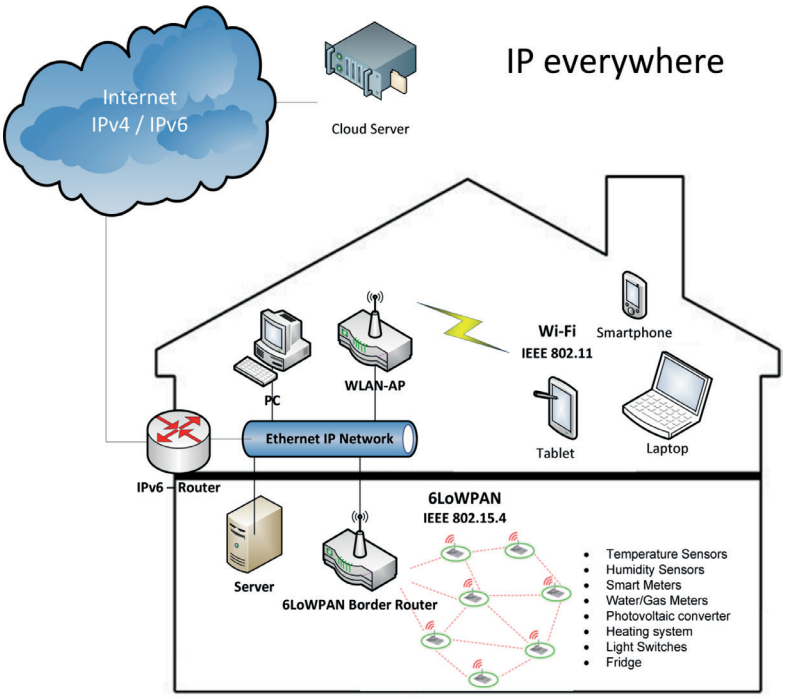
The Internet of Things is on the rise. More and more microelectronic systems are getting connected with each other. One of the most popular networking protocol for IoT is IPv6 over IEEE802.15.4 commonly known as 6LoWPAN. In this article, the open source project called emb::6 is presented providing an efficient, safe, stable and easy to use implementation for connecting devices to the IoT including lots of exclusive features.

1. IPv6 als integrierende Netzwerkschicht

Die Grundidee von IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN, sprich six-louhpähn) besteht darin, das IPv6-Protokoll auch für lokale Funknetze zu nutzen, um hierbei homogene und durchgängige Ende-zu-Ende-adressierte Netzwerke zu ermöglichen. Auf diese Weise kann das Internet der Dinge eine Verschaltung mehrerer IP-basierter Netze unter Nutzung individueller IPv6-Adressen werden, sodass zur Kopplung nur Layer-3-Router, aber keine Kopplungselemente höherer Schichten, insbesondere keine Gateways, benötigt werden. Darüber hinaus wird die Standardisierung in den öffentlicheren Raum der Internet Engineering Task Force (IETF) gehoben.

Auf der Ebene der Funknetze kommen neben dem IEEE802.15.4-Standard auch andere Funkprotokolle zum Einsatz. Spezifikationen liegen u. a. für Bluetooth, WiSun, LoRaWAN und ULE vor. Auf diese Weise besteht die Möglichkeit, dass sich IPv6 als medienunabhängige Netzwerkschicht auch für die Vielzahl von Funksystemen durchsetzt und das Internet der Dinge letztendlich auch auf dem Internetprotokoll basiert („IP everywhere“). Eine mögliche zukünftige Installation ist in Abb. 1 gezeigt.

Die Vorgaben des IETF sind seit nunmehr etwa drei Jahren stabil. Sie sind auch die Grundlage für mehrere quelloffene und kommerzielle Implementierungen, deren Interoperabilität in einem ersten Plug-Test verifiziert wurde.



2. Marktübersicht

Bevor die neue Implementierung vorgestellt wird, soll kurz erläutert werden, warum die Autoren der Meinung sind, dass eine weitere quelloffene Implementierung sinnvoll ist. Denn schließlich gibt es bereits verschiedene kommerzielle und auch quelloffene Implementierungen für 6LoWPAN, wobei im Rahmen dieses Beitrags vor allem die quelloffenen Varianten betrachtet werden sollen. Diese Implementierungen sind allerdings aus Sicht der Autoren alle mit gewissen Nachteilen behaftet.

Verbreitete Implementierungen wie µIPv6 (Contiki), BLIP, RIOT sind sehr stark an das zugrunde liegende Betriebssystem bzw. an betriebssystemähnliche Strukturen gebunden und damit nur beschränkt in andere Systemumgebungen portierbar. Hierbei erfüllen diese Betriebssysteme oft nicht die Anforderungen an professionelle Systeme und finden außerhalb der Kommunikationsanwendungen auch praktisch keine Verbreitung. Hinzu kommt gerade bei µIPv6 die Verwendung von sehr betriebssystemspezifischen Konstrukten (Protothreads), die eine Portierung erschweren. BLIP 2.0 ist gar in dem C-Derivat NesC programmiert und somit sogar unmittelbar nicht verwendbar.

Darüber hinaus versuchen viele der Aktivitäten mit viel Aufwand komplette Systemumgebungen mit dezidierten Simulatoren und weiteren Werkzeugen aufzusetzen. Aus Sicht der Autoren ist es viel effizienter, hier auf bestehende Werkzeuge zurückzugreifen und lediglich die Einbindung der eigenen Implementierung stabil und einfach vorzubereiten. Auf diese Weise ist dann auch die Integration in Gesamtsysteme bedeutend einfacher möglich.

Aus diesen Gründen fiel die Entscheidung, eine eigene Implementierung vorzulegen, die sich durch eine reine, modulare, portierbare Implementierung in nativem C auszeichnet und einfach anpassbar und parametrierbar ist. Die Integration eines sicheren Transportlayers, vor allem für den ebenfalls im Institut der Autoren entwickelten (D)TLS-Stack soll vorbereitet werden.

In einem Projekt, gefördert durch das ZIM-Programm vom Bundesministerium für Wirtschaft und Energie (BMWi), wurden gemeinsam mit dem Dresdner Unternehmen AN Solutions GmbH der sogenannten emb::6-Stack implementiert und verifiziert. Hierbei diente die Implementierung des µIPv6-Stacks aus dem Contiki-Projekt als Grundlage und wurde so weit umgebaut, dass die oben erwähnten Anforderungen erfüllt sind. Zusätzlich wird mit einem teilautomatisierten Verfahren sichergestellt, dass weitere Updates und Bugfixes der Community für die Ursprungssoftware in den neuen Stack übernommen werden. In Absprache mit den Autoren des Contiki-Projekts wurde die nunmehr erstellte Implementierung als quelloffenes Projekt mit einer identischen, sehr einfachen Lizenzregelung bereitgestellt.

Erste Portierungen auf einige verbreitete Hardware-Plattformen liegen mittlerweile vor. Hierzu zählen gegenwärtig Mikrocontroller wie Atmel AVR8, TI MSP430 sowie ARM-basierte MCUs mit Cortex-M0+- und M3-Architekturen. Als Transceiver werden gegenwärtig at86rf2128(b) und at86rf230 von Atmel und TI's CC1120 und CC1200 unterstützt. Auch eine Portierung auf den neuen cc13xx-SOC von TI erfolgte letztens in einem Industrieprojekt. Tab. 1 zeigt den Speicherbedarf unterschiedlicher Konfigurationen bei typischen Parametrierungen.

Stack Configuration	cs_xpro_212b Cortex M0+ Flash/RAM	cs_stk3600 Cortex M3 Flash/RAM	cs_atany900 AVR8 Flash/RAM	ua_atany900 AVR8 Flash/RAM
COAP:	10.4 / 1.4 kB	10.0 / 1.4 kB	10.8 / 1.4 kB	
RPL:	16.0 /0.4 kB	14.7 / 0.6 kB	12.7 / 0.5 kB	12.6 / 0.5 kB
IPV6:	18.7 / 2.6 kB	17.5 / 2.6 kB	15.8 / 2.1 kB	15.8 / 2.1 kB
6LOWPAN:	7.4 / 0.3 kB	6.6 / 0.3 kB	5.4 / 0.4 kB	5.4 / 0.4 kB
Others: (Demo, REST, MAC, Util, BSP)	74.9 / 5.6 kB	64.2 / 7.1 kB	24.3 / 4.6 kB	19.1 / 2.3 kB
Total:	127.4 / 10.3 kB	113.0 / 10.9 kB	69.0 / 9.0 kB	52.9 / 5.4 kB

3. Architektur

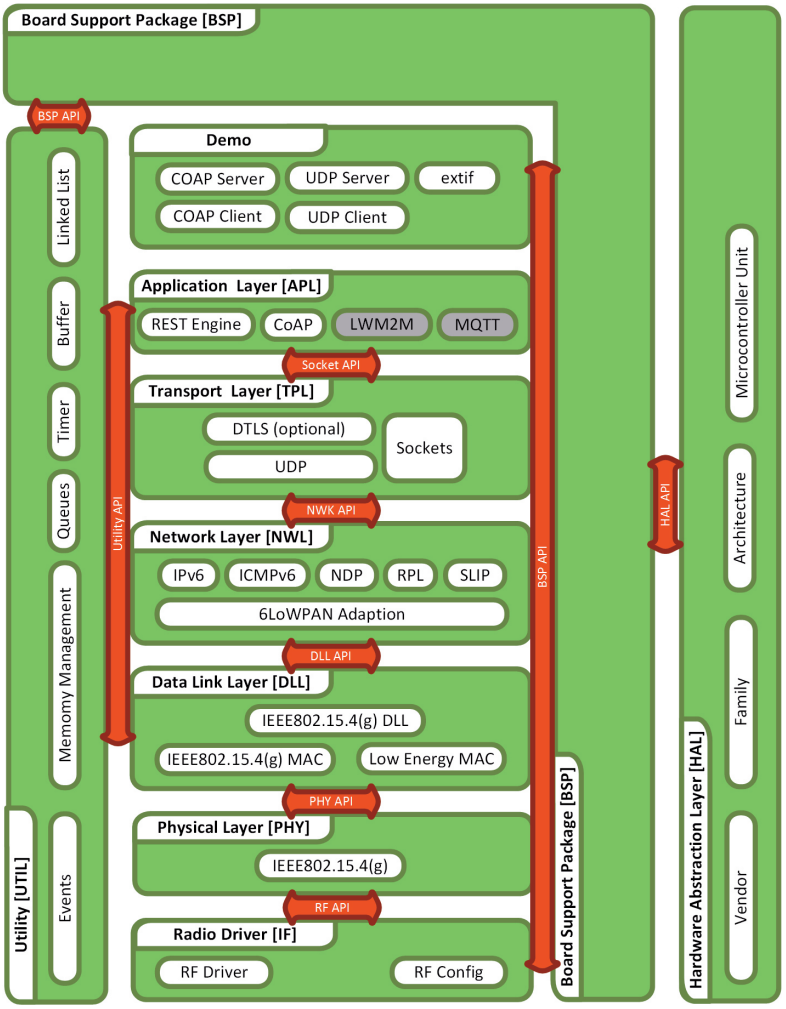


Abb. 2 zeigt den Aufbau der emb6-Protokollimplementierung. Im Zentrum der Entwicklung steht natürlich die Netzwerkschicht, die nach oben für die Anwendungsschichten die Schnittstellen für die Kommunikation anbietet und nach unten die Dienste der Sicherungs- und der physischen Schicht nutzt. Die einzelnen Schichten werden in den folgenden

Abschnitten detaillierter erläutert. Neben der Netzwerkfunktionalität sind alle wichtigen Basisfunktionen wie Timer, Buffer-Management und Speicherverwaltung herausgelöst und in generische Utility-Module zusammengefasst.

Zur flexiblen Unterstützung unterschiedlicher Mikrocontroller und Transceiver auf unterschiedlichen Boards wurden alle hardwarespezifischen Bestandteile in einem Board Support Package (BSP) zusammengefasst, das den Hardware Abstraction Layer (HAL) bedient.

- Anwendungsschicht: Die Anwendungsschicht (Application Layer, APL) ist ein optionaler Bestandteil des emb6-Stacks. Hierbei kann in Abhängigkeit von der angestrebten Funktionalität zwischen verschiedenen Implementierungen ausgewählt werden, wobei bislang nur CoAP im Open-Source-Bereich bereitgestellt wird. Parallel sind Implementierung LWM2M und MQTT in Bearbeitung.
- Transportschicht. Die Transportschicht (transport layer) stellt eine an den klassischen BSD-Socket angelehnte Socket-Schnittstelle zur Verfügung. Nach Standard wird – wie in vielen Anwendungen üblich – das verbindungslose User Datagram Protocol (UDP) unterstützt. Eine Integration von TCP ist möglich.
- Netzwerkschicht: Die Netzwerkschicht (network layer) ist der bisherige Kern der Aktivitäten. Er enthält folgende Teilmodule:
  - Die oberen Teilschichten realisieren das standardisierte IPv6-Protokoll und die zugehörigen Management Protokolle wie das Internet Control Message Protocol (ICMPv6) und das Neighbor Discovery Protocol (NDP).
  - Diese wird dann mithilfe der unteren 6LoWPAN-Teilschicht an die Anforderungen des IEEE802.15.4 adaptiert. Die geringere Rahmengröße des IEEE802.15.4-Standards erfordert eine effiziente Einbettung, weil der fixe Teil des Standard-IPv6-Headers bereits 40 Byte beträgt. Würden diese in jedem Rahmen zusammen mit den bis zu 25 Bytes MAC-Overhead des IEEE802.15.4 Standards übertragen, wären mit bis zu 65 Bytes schon mehr als die Hälfte des Rahmens mit Steuerinformationen gefüllt. Entsprechend erscheint sowohl eine Kompression des Headers als auch eine Fragmentierung sinnvoll und notwendig. Diese Anpassungen werden zusammen mit einigen anderen Funktionen als 6LoWPAN in den RFCs 4944, 6282 und 6775 beschrieben. Hierbei werden dann IPv6 und UDP Header komprimiert. Die IP-Pakete mit einer maximalen Rahmengröße (Maximum Transmission Unit, MTU) von 1280 Bytes werden fragmentiert und in die maximal 127 Byte großen IEEE 802.15.4-Rahmen verpackt.
  - Das IPv6-Modul wird ergänzt durch das Routing-Protokoll, das die Verkehrsweiterleitung in vermaschten Topologien übernimmt. Zum Einsatz kommt hierbei das „Routing Protocol for Low power and Lossy Networks“ (RPL, sprich: rippl). Weitere wichtige Bestandteile sind der Trickle Algorithmus (RFC 6206) für das Timing der Informationspakete, die Definition der Routingmetriken in RFC 6551 sowie die Objective Function Zero (RFC 6552), die für die Wahl des bevorzugten Elternknotens benötigt wird.
  - Zusätzlich ist ein Serial Line Internet Protocol (SLIP) Modul vorhanden, das die Verwendung des Stacks über eine serielle Verbindung von einem anderen Gerät aus erlaubt.
- Sicherungsschicht: Hier wird im Wesentlichen die Funktion des IEEE 802.15.4-MAC realisiert. Für weiterleitende Knoten (Routing Nodes) wird die Funktion des Full Function Devices (FFD) benötigt, während für Endknoten, sogenannte Leaf Nodes, bzw. Non-Routing Nodes die einfacheren Reduced Function Devices (RFD) ausreichen. Die Kopplung mit der physischen Schicht ist recht eng, weil viele Transceiver bereits Funktionen für den Kanalzugriff, wie z. B. CSMA oder Auto Retransmission, in Hardware anbieten.

Hervorzuheben sei in diesem Zusammenhang, dass die IEEE802.15.4-Funktionalität des emb6-Stacks um sogenannte Wake-On-Radio-(WOR)-Funktionen erweitert wurden, die ein Aufwecken der Transceiver über die Funkschnittstelle erlaubt. Allerdings wird diese Funktion zum gegenwärtigen Zeitpunkt vor allem von Transceiver effizient unterstützt, die nicht kompatibel zu den Vorgaben des IEEE802.15.4 sind.

- Physische Schicht: Ein Großteil der physischen Schicht, wie z.B. Modulation wird in der Regel durch den Transceiver gehandhabt. Andere Aufgaben, wie z. B. CRC-Prüfungen und Berechnungen für empfangene oder zu sendende Pakete, werden oft in Software erledigt.
- Radio-Treiber-Schicht: Aufgrund der Hardwareunterstützung vieler Transceiver für

Literatur:
[1] P. Nguyen, M. Schappacher, A. Sikora, V.F. Groza, „Extensions of the IEEE802.15.4 Protocol for Ultra-Low Energy Real-Time Communication“, Proc. I²MTC, 2016 IEEE International Instrumentation and Measurement Technology Conference, 23.-26. May 2016, Taipei (Taiwan)
[2] A.K.Nsiah, A. Sikora, A. Walz, A. Yushev, „Embedded TLS1.2 Implementation for Smart Metering & Smart Grid Applications“, Journal of Electronic Science and Technology, Vol. 13, No. 4, Dec. 2015, pp.373-378
[3] M. Schappacher, E. Schmitt, A. Sikora, P. Weber, A. Yushev, „A Flexible, Modular, Open-Source Implementation of 6LoWPAN“, 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS2015), 24-26 September 2015, Warsaw, Poland, pp. 838-844
[4] A. Sikora, „Funknetzwerke für das Internet der Dinge: 6LoWPAN OpenSource-Projekt: emb6“, Elektronik Wireless 2016

MAC-Funktionen besteht die unterste Schicht des Protokollstapels im Wesentlichen aus der Ansteuerung der Transceiver-Funktionen. Diese werden im sogenannten Radio Driver zusammengefasst.

5. Inbetriebnahme, Betrieb und Test

Wie gesagt, wurde bei der Implementierung auf eine einfache Benutzerbarkeit der Software geachtet.

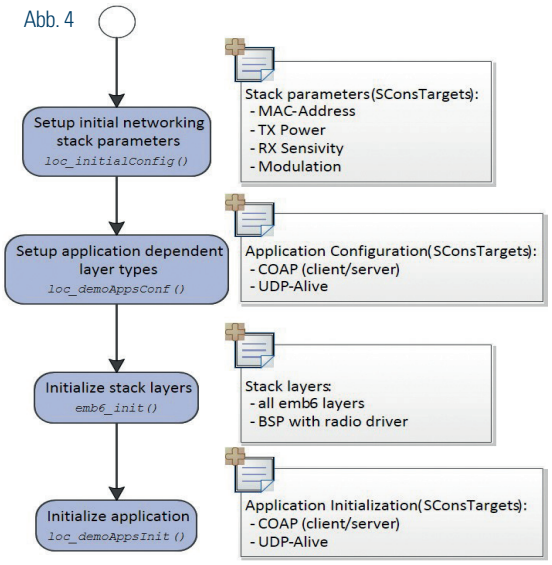
```
typedef struct netstack {
      const struct netstack_headerCompression* hc;
      const struct netstack_highMac*           hmac;
      const struct netstack_lowMac*            lmac;
      const struct netstack_framer*            frame;
      const struct netstack_interface*         inif;
}s_ns_t;
```

Abb. 3 zeigt die Schritte, die zur Konfiguration des emb6-Protokollstapels notwendig sind. Dabei werden sowohl die statischen Parameter der Utilities und des Stacks als auch die Initialwerte der dynamischen, d.h. zur Laufzeit setzbaren Parameter über Präprozessordirektiven gesetzt. Beispielhaft ist in Abb. 4 die Struktur der Schnittstelle zur Parametrierung des emb6-Protokollstapels gezeigt.

Um die Vielzahl an Konfigurationen zu verwalten, wird das SCons-Buildsystem eingesetzt. Dieses erlaubt es, mithilfe von Konfigurationsdateien, relativ schnell neue Buildkonfigurationen zu erstellen oder existierende anzupassen. Zudem kann durch den modularen Aufbau der Dateien auf gemeinsame Bestandteile zurückgegriffen werden.

Die Implementierung wurde insbesondere in Bezug auf die Routing-Funktionalität umfangreich getestet. Eine besondere Rolle spielt hierbei das im ivESK entwickelte automatisierte Testbed.

Abb. 4



6. Ausblick & offene Themen

Das Internet der Dinge ist unaufhaltsam auf dem Vormarsch. Die Community hat hierbei zu großen Teilen verstanden, dass ein wirklicher, nachhaltiger Markterfolg sich nur durch offene, übergreifende und interoperable Lösungen und durch die Bereitstellung von vorentwickelten, getesteten und einfach zu nutzenden Plattformen erreichen lässt.

Der vorgestellte emb::6-Protokollstack ist eine solche Plattform. Wir laden alle Interessierten ein, an diesem Projekt mitzuarbeiten, um eine herstellerunabhängige, leistungsfähige, aber portabel und flexibel einsatzbare Lösung nutzen zu können.

Weitere Entwicklungsthemen sind sicherlich die Erstellung weiterer Portierungen sowie weiterer Applikation-Layer. Interessant ist aber auch die Integration des emb6-Protokollstapels in einen Netzwerksimulator wie NS-3 oder OMNET++. Hierzu sind im ivESK bereits Vorarbeiten geleistet worden. Interessiert sind die Autoren natürlich auch an Erfahrungen beim Einsatz des emb6-Protokollstapels.

Dieser Beitrag ist eine aktualisierte Fassung von [4].

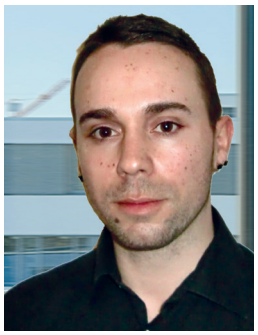
AUTOREN

Artem Yushev M. Eng.
Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK)
artem.yushev@hs-offenburg.de

Dipl.-Ing. (FH) Edgar Schmitt
Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK)
edgar.schmitt@hs-offenburg.de

Dr.-Ing. Dipl.-Ing. Dipl.-Wirt.-Ing. Axel Sikora
Wissenschaftl. Leiter Institut für verlässliche Embedded Systems u. Kommunikationselektronik (ivESK)
axel.sikora@hs-offenburg.de

Dipl.-Inform. (FH) Manuel Schappacher
Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK)
manuel.schappacher@hs-offenburg.de



Fesseha Tsegaye Mamo M. Sc.
Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK)
fesseha.mamo@hs-offenburg.de