SEMANTIC INTERFACE FOR RESOURCE CONSTRAINED WIRELESS SENSORS

Arto Ylisaukko-oja[1], Pasi Hyttinen[2], Jussi Kiljander[1], Juha-Pekka Soininen[1] and Esa Viljamaa[1]
[1] *VTT Technical Research Centre of Finland, Oulu, Finland*
[2] *VTT Technical Research Centre of Finland, Kuopio, Finland*

Keywords: Ultra-low power wireless sensors, semantic sensor web.

Abstract: In this paper, a functional implementation of a semantic interface for a resource constrained, battery operated wireless sensor is presented. The concept is demonstrated by a home greenhouse application where the semantic interface is applied to moisture sensors which are connected to a database. The solution is based on M3 architecture for Smart Spaces. The paper discusses the enabling technologies that make the semantic messaging more viable for resource constrained devices. Performance figures concerning power consumption, battery duration and memory consumption are presented along with ideas for further development.

1 INTRODUCTION

Providing a semantic interface for accessing information of resource constrained wireless sensors is still a novel concept. The objective of semantic level interoperability is to enable meaningful sharing of information between a variety of devices – including situations and new applications which have been impossible to take into account at design time. It is hard to address such demands if e.g. the data format and their semantics are defined and implemented for each use case separately. Also approaches such as device profiles have their limits in this sense: Bluetooth device profiles, for example, inherently assume a certain application domain, such as audio or health device. Such an approach does not support well any novel ways of utilizing the same information in novel, unexpected applications.

Wireless sensors are often low capacity devices in terms of energy, processing, communication, physical size and cost. A typical wireless sensor is battery operated and uses one of the standard or proprietary radio communication technologies. Standardized radio technologies such as IEEE 802.15.4 or Bluetooth Low Energy have low power consumption and battery based operation as one of the most important design criteria. These protocols also define short message payloads – simple, short messages are sent, preferably infrequently. The target is to conserve battery power and achieve reasonable times of operation before the battery needs to be replaced. It is a challenge to apply semantic interfaces to such devices, since memory, processing time and message length overheads easily increase, potentially leading to more expensive and power-hungry devices.

The scope of the work presented in this paper was to adapt semantic interface to a resource constrained wireless sensor that we call an *Active Tag*. The Active Tag has a radio access to a database, where it can publish its sensor data in a semantic form. In addition, the Active Tag can read data from the database using similar semantic messages. It can use this information to adjust its own behaviour.

The aim was to achieve an implementation that would still enable an ultra-low power implementation from small batteries, also without excessive component cost. This meant that minimizing the resulting overhead was essential, to prevent excess complexity in software and to keep radio messages as compact as possible – only this way can the memory requirements be kept modest and the operation time of the battery reasonable. The aim was to keep the power consumption in a level that would allow several months or years operation time in typical wireless sensor applications.

The framework for our solution was the M3 smart space architecture, for which open source

Ylisaukko-oja A., Hyttinen P., Kiljander J., Soininen J. and Viljamaa E..
SEMANTIC INTERFACE FOR RESOURCE CONSTRAINED WIRELESS SENSORS.
DOI: 10.5220/0003698905050511
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (SSW-2011), pages 505-511
ISBN: 978-989-8425-80-5
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

software has been developed over the last few years (Smart-M3 in SourceForge). Some of this development has been targeting to decreased complexity and improved execution efficiency. Therefore, software libraries for implementation were available. However, they had certain limits – especially about supported communication interfaces – that had to be overcome when adapting to wireless sensor domain.

The overall proof-of-concept demonstration was implemented as a greenhouse smart space application, where the Active Tags work as moisture sensors in plant jars. They use semantic level, RDF (Resource Description Framework) based messaging to communicate via *RIBS* (RDF Information Base Solution). (RDF Vocabulary Description Language 1.0). The RIBS is the central knowledge base and semantic information broker (SIB) in the smart space. Below the semantic communication level there is simple data access communication between Active Tag and RIBS. This communication follows the smart space access protocol (SSAP), with WAX encoding (Suomalainen and Hyttinen, 2011) that is suitable for resource limited devices. WAX (Word Aligned XML) and RIBS are the key technologies in minimizing the overhead caused by the semantic level interface.

The overall scenario includes also a Gardener Terminal device. NFC and optical tag technologies are used in combination with uCode technology (Koshizuka and Sakamura, 2010) to configure the smart space appropriately.

2 BACKGROUND AND RELATED WORK

2.1 Semantic Web

The Semantic Web is a vision of a next generation World Wide Web (WWW) in which the semantics of the information is explicit and openly shared in the Internet. Explicity and the availability of the ontology definitions allow run-time interpretation and new intelligent Web applications and services.

The core technologies comprising the Semantic Web stack include Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL) and SPARQL (SPARQL, SPARQL 1.1). Information interoperability in the Semantic Web is based on defining common ontologies. RDFS and OWL provide vocabularies for describing the concepts and relationships between these concepts, i.e., ontologies. The RDF is used to present the ontologies in the form of a subject, predicate and object triples, so it is a very natural way to make statements about information. SPARQL query language provides SQL-like query mechanisms for RDF data. The SPARQL 1.1 expands the 1.0 version by defining also mechanisms for path queries and for modifying the data in RDF database. (T. Berners-Lee et al., 2001).

2.2 Semantic Sensor Networks

There are also activities focusing on utilizing Semantic Web technologies to sensor networks. The Semantic Sensor Web (SSW) approach targets to improve the interoperability of sensor networks by adding temporal, spatial, and thematic metadata to the measurement data. The SSW aims to achieve this by extending the OGC and SWE specifications with Semantic Web technologies (Sheth et al., 2008). Sense2Web is a platform for publishing and linking sensor data to the Semantic Web (Barnaghi and Presser, 2010). Sense2Web Linked-sensor-data platform enables users to publish RDF serializes information about their sensors, associate this data with existing RDF sensor data, link their sensor data to other resources and make the information publicly accessible for other semantic web applications via SPARQL endpoints. In (Patni et al., 2010) a framework for publishing sensor data to Linked Open Data Cloud is presented. This is achieved by converting the sensor descriptions from SWE's XML based Observations and Measurements (O&M) standard to RDF format. It is also noteworthy that W3C's Semantic Sensor Networks Incubator Group (SS-XG) has started to define ontologies for describing sensor data (W3C's Semantic Sensor Networks Incubator Group).

The aforementioned approaches provide necessary technologies and valuable knowledge for enabling semantic sensors for the IoT. However, these approaches do not concentrate on how the real-life constrained sensors with limited power, memory and processing capabilities are able to present their information in, usually very sparse, semantic format. In addition these approaches do not present how the sensors could utilize the available machine interpretable data to improve their own functionality. The main contribution presented in this paper is a novel approach for constrained sensors to publish and access information in semantic form. In our approach we utilize the Semantic Web based M3 concept. M3 is an infrastructure for providing semantic interoperability in physical environments.

We present how constrained sensors can not only publish their information for global use in semantic form, but also how the sensors are able to improve their quality of service (QoS) by utilizing the semantic information produced by other parties.

3 SYSTEM MODEL

M3 is a concept for utilizing the Semantic Web ideas and technologies to provide semantic level interoperability between devices in physical environments. By utilizing the ontology based information model the M3 based software agents can more autonomously interpret the meaning of information and therefore obtain greater degree of smartness and flexibility than could be achieved with traditional use case specific data models. M3 utilizes RDF, RDFS and OWL for presenting the semantics of information in a computer-interpretable manner. In the core of M3 is a functional architecture that specifies how the semantic information can be accessed in a physical space. The M3 functional architecture consists of Knowledge Processors (KP) and Semantic Information Brokers (SIB). For SIB, we use a specific implementation called RIBS (RDF Information Base Solution). SIBs or RIBSs are basically shared RDF databases of semantic information that provide publish/subscribe based interface for KPs. The role of KPs is to provide applications for end-users by interacting with each other via the RIBS. Smart Space Access Protocol (SSAP) defines the rules for KP-RIBS interaction. M3 utilizes existing solutions for the communication and service level meaning that it is possible to implement the SSAP protocol with different service and communication level technologies.

In our case, the Active Tags are essentially KPs in the system. In the demonstration, the Active Tags work as intelligent moisture sensors in plant jars; the basic idea is to indicate if the moisture of the soil is correct. This is done by two different ways:

1. By inserting the moisture data from Active Tags to RIBS. The gardener can read the moisture value by his mobile terminal. The mobile terminal is also a KP and it also has a connection to RIBS.
2. By blinking bright LED indication in each of the sensors in jars, giving visual location indication for the gardener about the plants in need of water.

The gardener presence is also inserted to the SIB: when the gardener touches an NFC *location tag*, his presence is inserted to the SIB. The Active Tags regularly query the presence information and use this to avoid blinking the LEDs in vain, therefore reducing power consumption.

4 APPROACH

In this Chapter, we describe the main technical solutions applied in the implementation of our home greenhouse demonstration for Active Tags using semantic interfaces.

4.1 WAX Encoding of SSAP

Word Aligned XML (WAX) is data encoding scheme where each XML tag, attributes and user data is packed into memory locations that are multiples of the word length of the processor. For 32 bit processors the word length is four bytes and thus WAX tags, attributes and data are placed into memory using four byte alignment. Four byte alignment allows data to be treated as byte array or as word array. The word processing is beneficial from the processor point of view since the arithmetic logic units (ALU), system bus and memories are often designed for word operations.

Word alignment means also that lengths of tags, attributes and data are multiples of words. The minimum length is one word and intuitively this is the favourable length. The lengths of the tags and attributes of the SSAP fields are four bytes each. Since the XML end tag has three reserved characters "<", "/" and ">" the actual name of the tag may contain only one character. As an example, the WAX encoding of the SSAP message tag use letter M, and thus start tag is "<M> " and end tag is "</M>". Note that the extra space character after ">" in the begin tag is important and required for the word alignment.

SSAP message tag without WAX encoding use tags "<SSAP_message>" (14 bytes) and "</SSAP_message>" (15 bytes). As byte counts indicate the message overhead in WAX encoding of SSAP is 3-4 times lower than in the SSAP without encoding. The message overhead can be eliminated totally by removing all metadata of the SSAP message and using for example some fixed binary structure for the user data. While this could be optimal for transmission and memory usage, it will cause difficulties in interoperability since interpretation of the messages requires a priori knowledge of the structure of the message. In this respect the WAX encoding is a good compromise between semantic interoperability and size of the message. The metadata is present, but its length is minimized.

WAX parser may use word alignment for speeding up the parsing process. Instead of byte comparison, the WAX parser can use word comparison for matching tags and attibutes. Roughly speaking, the integer comparison is four times faster and than byte comparison for 32-bit processor, which is also a power consumption advantage. Similar advantage is present also when composing WAX message.

A huge benefit is that WAX parser is much more compact than a generic XML parser – Expat XML parser used in our earlier work was 206 kB of code size – too large to even fit to the wireless sensor platform we are next describing.

4.2 Hardware and Software Platforms

Fig. 1 shows the architecture of the KP – RIBS network we needed to implement for our study. The hardware and software platforms are described in the following.

Many wireless sensor platforms that are intended for research purposes contain 16- or 8-bit microcontrollers such as Atmel AVR series or Texas Instruments MSP430 series devices. However, we ended up using a 32-bit computing platform. The main reason for this decision is code efficiency: if same functionality is performed with 32-bit ARM based architecture, the resulting amount of instructions is considerably lower than in case of 16-bit devices, let alone 8-bit computing platforms. This is a considerable benefit, keeping the memory overhead caused by the program code of semantic interface reasonably low. Although 32-bit architectures are more power hungry per instruction than lower word length architectures, this is very much compensated by the shorter execution time (assuming equal clock frequencies).

Even if more power efficient and modern CortexM3/M0 would have been basically preferred, we ended up choosing an ARM7 based platform by Freescale, namely MC13224V system-on-chip that includes ARM7TDMI-S processing core and an IEEE 802.15.4 compatible 2.45 GHz short-range radio transceiver. While there was a comparable, Cortex-M3 based system-on-chip available from ST Microelectronics (STM32W), we found that the MC13224V with its total 96 kB divided between program code and RAM memory gave greater flexibility. Thus larger RAM buffers would be possible to implement if needed, as opposed to the fixed, 8 kilobytes RAM of STM32W. In addition, MC13224V has the IEEE 802.15.4 MAC layer readily implemented in ROM.

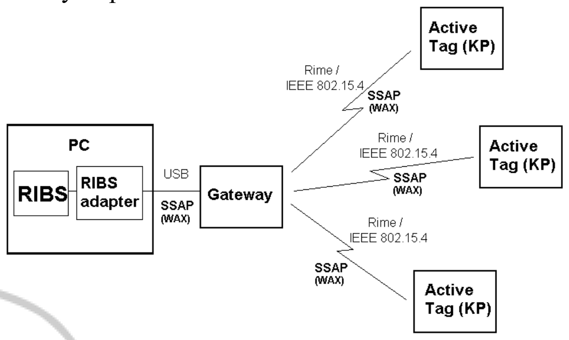

Figure 1: The star network of Active Tags connected to RIBS.

The core of our research was the semantic interface in a resource constrained sensor. Thus we wanted a platform that would provide existing support for the underlying wireless communication mechanism. We decided to run Contiki operating system on the MC13224V SoC: Contiki includes a simple IEEE 802.15.4 compatible protocol called Rime, and Contiki had been readily ported to this platform (The MC1322x Open Source Project); (Contiki Operating System. ANSI C was used to write programs on Contiki OS. To avoid building new hardware, we decided to build the Active Tag on the MC13224V based, commercially available platform by Redwire LLC, called *Econotag*.

Since IEEE 802.15.4 radio is not standard equipment in PCs, we decided to use one Econotag as a gateway in each network. That is, the device containing the RIBS database needs to have an Econotag connected to its USB port. In addition, software had to be implemented to adapt to the currently supported RIBS interfaces (basically only socket connections are supported at the moment). For this purpose, we implemented a simple PC software program called *RIBS adapter* that runs in the RIBS device, communicating with the gateway device in the USB port and using a socket connection to communicate the data to the RIBS.

The RIBS can be run in any PC. In our demonstration, we had it running on a Via Artigo A1100 compact PC. As moisture sensors in Active Tags, we selected the VG400 sensor probes from Vegetronix. These sensors can be powered directly from a general-purpose IO pin of the microcontroller. The sensor can measure water level or moisture in soil of a plant jar.

Fig. 2 shows the Active Tag related portion of the demonstration system. On the left, two battery

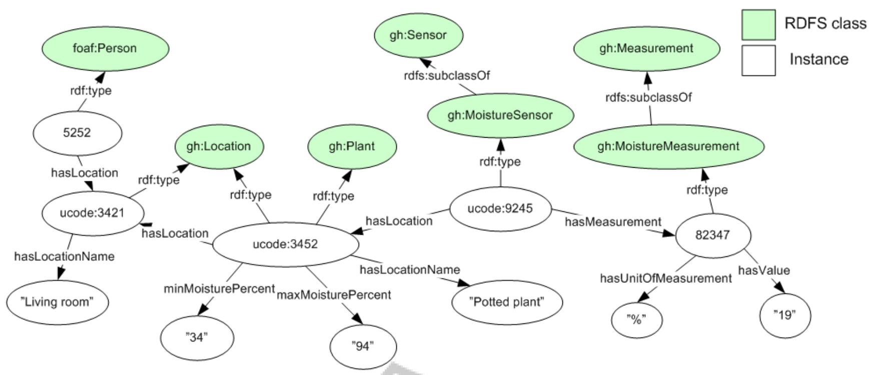

Figure 3: Ontology for Home Greenhouse.

operated Active Tags with moisture sensors are shown. Here, the moisture sensors are measuring water level in the coffee mugs. On the right, RIBS PC is shown with attached Gateway to support IEEE 802.15.4 communications of the Active Tags.

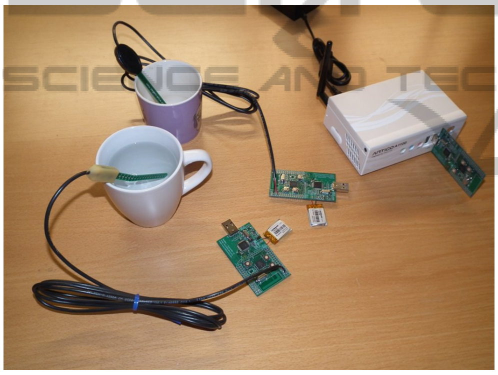

Figure 2: Two Active Tags with moisture sensors and the RIBS PC with Gateway.

4.3 Home Greenhouse Ontology

The ontology developed for the case study contains four logically distinct physical entities: Person, Location, Plant and Sensor. The Person class is imported from the Friend of a Friend (FOAF) ontology and the "foaf" namespace is thus used. For Location, Plant and Sensor classes namespace "gh" (form greenhouse) is used. These entities are modelled as RDFS classes. The RDFS is used as the ontology language because the RIBS provides RDFS level reasoning.

The Location class presents a physical location such as a town, house, room or pot, for example. In this demonstration it is used to both present when some person is in the same room with an Active Tag (the Gardener Terminal is used for gardener presence indication) and with which plant the Active Tag is located in a same jar. The Plant class presents necessary information about the plant. In this demonstration the minimum and maximum preferences for soil moisture are used. The actual potted plant is modelled as an instance of both Location and Plant classes.

The Sensor class models a physical object capable of measuring its surroundings. Instances of the Sensor class can be associated with a Location class instance by using the "gh:hasLocation" property. To allow sensor to have multiple different measurements with different unit types the measurements of a sensor is modelled as a separate class. The Measurement class has properties for presenting the value and unit of the measurement. Subclasses for Sensor and Measurement class are used to model the exact type of the sensor and measurement respectfully. In this demonstration only the MoistureSensor and MoistureMeasurement classes are used.

Fig. 3 illustrates an example instant of the home greenhouse ontology in the RIBS. Note that part of the ontology description is related to the Gardener Terminal.

4.4 Software Operation in Active Tags

To implement an ultra-low power battery operated wireless sensor, it is essential to minimize the time the radio is switched on. Furthermore, all the parts should be in the deepest possible sleep mode for a major part of time. This has effect on what publish-subscribe methods are preferred.

The Active Tag uses *insert* and *update* operations to publish its data along the common ontology model (presented in the section C) into the RIBS. Due to the nature of the greenhouse application, this can take place quite infrequently, such as once per minute (at least when gardener is not present – if he is, the responsiveness of the Active Tags can be automatically improved by shortening the communication interval). To read data from RIBS, we avoid *subscribe*, since it would basically require the Active Tag to be ready for communication at an arbitrary moment. Instead, the Active Tag use *query* on a regular basis (following the moisture data *insert* or *update*) to ask for gardener presence data from RIBS. In the beginning of operation, Active Tag also uses query for getting the maximum and minimum moisture values from RIBS.

Between the moisture measurement and above-mentioned communication operations, the Active Tag is in deep sleep mode: the microcontroller in *hibernate* sleep mode and the radio and moisture sensor being powered off. The internal RC oscillator of the microcontroller is used as a wake-up circuit. If a more precise wake-up circuit is needed, a low power oscillator using a 32 kHz crystal is also available.

5 VALIDATION

5.1 Power Consumption and Battery Duration

The battery life of a wireless sensor is essentially determined by the average power consumption. This is a combination of sleep and active mode power consumptions. To determine this, the current consumption of the sleep mode was measured with a digital multimeter and the active mode current was measured with a current probe and an oscilloscope. During the measurements, the Active Tag was running a cycle where it wakes up, measures moisture, inserts moisture data to RIBS, queries gardener presence and goes back to sleep. No LED is blinked in the test sequence.

During measurements, the Active Tag was operated from a 3V supply without voltage regulation. This is equivalent of using two 1.5V AA size batteries in series without regulation. The current consumption during sleep was measured to be 10.5 µA (all RAM pages and microcontroller state retained). This is equal to 32 µW. It is possible also not to retain the microcontroller state, in which case the sleep current was measured to be 5.1 µA.

The sleep mode consumption defines the absolute maximum for the battery duration. For two 1.5V, 2700 mAh alkaline batteries in series the 10.5 µA consumption yields a theoretical duration of 29 years (self-discharge not taken into account). The practical duration is of course further determined by the wake-up interval and the power consumption during the active cycle.

According to the measured current profile, the active cycle consumes roughly 11.3 mA for the duration of 236 ms and 28.8 mA for the duration of 388 ms. The current consumptions are at expected levels, but the active cycle length is remarkably long: the amount of data communicated during one active cycle is approximately 1 kB with Rime and IEEE 802.15.4 headers included, which takes 32 ms of transmission/reception time. However, the time to get a response from RIBS is now dependent on the operation of the PC the RIBS is located in. This is bound to increase the time the radio is kept on. Nevertheless, the current implementation seems to include extensive delays in operation that need to be carefully sorted out in further development.

Despite the obvious need for optimization, we get an average current consumption of 241 µA from 3 volts @ 60 second wake-up interval. For two 1.5V, 2700 mAh alkaline batteries in series this yields a theoretical battery duration of 1.3 years (battery self-discharge and LED blinking consumption not taken into account). Even with 10 s wake-up interval we would still get 2.7 months battery duration.

5.2 Microcontroller Resources

The resulting code size, with *join, insert* and *query* operations enabled, was 39.7 kB. This is 26% increase compared to a reference software that includes only the underlying Contiki OS and the Rime protocol – the reference software also does not include the application software. Increase in RAM memory use was more dramatic: from 14.3 kB in reference software to 25.3 KB in our Active Tag software (77% increase). RAM requirements need further analysis in our further work.

6 CONCLUSIONS

A functional semantic interface was implemented for a battery operated, resource constrained wireless sensor. The sensor publishes data in a semantic database, but also queries semantic data, adjusting its operation accordingly.

Despite obvious need for improvement in active cycle duration, the sensor achieves theoretical battery duration of 1.3 years from two 1.5V AA batteries at 60 s wakeup/communication interval. Code size was increased by 26% compared to a reference implementation with only Contiki OS and radio protocol implemented (excluding the semantic interface and the application software). RAM requirement increased by 77%, respectively.

One possibility to minimize radio on time could be to use the Gateway as a message buffer in RIBS communications, avoiding the need to wait for PC program execution – in this case, *subscribe* could be a viable alternative to *query*. Possibilities to minimize message length further should also be studied – an average WAX message length to be sent is now 160 bytes. Binary XML is a potential solution. Shortening the messages would make semantic interface more viable with technologies such as Bluetooth Low Energy that use even shorter maximum message payloads than IEEE 802.15.4 radio.

Applying query languages such as SPARQL in a resource constrained wireless sensor is another possible research topic in the future.

REFERENCES

Smart-M3 in SourceForge: http://sourceforge.net/projects/smart-m3/

RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004. URL: http://www.w3.org/TR/rdf-schema/ [Accessed 14 October 2010].

Suomalainen, J., Hyttinen, P.: Security Solutions for Smart Spaces. The Second International Workshop on Semantic Interoperability for Smart Spaces (SISS 2011). July 18-21, 2011, Munich, Germany.

Koshizuka, N., Sakamura, K.: Ubiquitous ID: Standards for Ubiquitous Computing and the Internet of Things. Pervasive Computing, IEEE, 2010, pp. 98-101

T. Berners-Lee, J. Hendler, O. Lassila, “The Semantic Web,” Scientific American Magazine, May 17, 2001.

SPARQL Query Language for RDF, W3C Recommendation 15 January 2008. URL: http://www.w3.org/TR/rdf-sparql-query/

SPARQL 1.1 Query Language W3C Working Draft, 12 May, 2011, URL: http://www.w3.org/TR/sparql11-query/

A. Sheth, C. Henson, and S. Sahoo, “Semantic sensor web,” Internet Computing, IEEE, vol. 12, no. 4, pp. 78–83, July-Aug. 2008.

Barnaghi, P. and Presser, M.: Publishing Linked Sensor Data. In Kerry Taylor, Arun Ayyagari, David De Roure (Eds.): The 3rd International workshop on Semantic Sensor Networks 2010 (SSN10) in conjunction with the 9th International Semantic Web Conference (ISWC 2010), 7-11 November 2010, Shanghai, China.

Patni, H.; Henson, C.; Sheth, A.: "Linked sensor data", International Symposium on Collaborative Technologies and Systems (CTS), 2010, vol., no., pp.362-370, 17-21 May 2010.

W3C’s Semantic Sensor Networks Incubator Group home page. URL: http://www.w3.org/2005/Incubator/ssn/

The MC1322x Open Source Project: http://mc1322x.devl.org/

Contiki Operating System: http://www.sics.se/contiki/