DOI 10.12694/scpe.v14i3.857
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 14, Number 3, pp. 155–167. http://www.scpe.org

ELL-I: AN INEXPENSIVE PLATFORM FOR FIXED THINGS

PEKKA NIKANDER[*], VADDINA KAMESWAR RAO[†], PETRI LIUHA[‡], AND HANNU TENHUNEN[§]

Abstract. The Internet of Things (IoT) vision is enticing; each and every "thing" in the world is expected to be eventually connected to the Internet, thereby becoming a part of the "context" within which the applications live. In most of the IoT research, the focus has been in enabling movable things to communicate, including phones, tablets, RFID tags, watches, and jewellery, to name but a few. In such an approach, the things are expected to have their own batteries or receive temporary power over short distance electro-magnetic field. This approach has also dominated the more fixed side of the IoT research, including a large fraction on the work on stationary sensors and actuators, focusing also there on battery-based operations and wireless communication.

In this paper, we introduce an alternative view to the world of stationary Internet-connected things. We argue that a large majority of the fixed or stationary things would benefit from being permanently powered using wireline connections, and while doing so, it becomes natural to use the same wires also for their communication and contextual needs. Such an approach allows the appliances to become part of the the wider application context. With this in mind, we introduce the ELL-i platform, a new open source initiative for provide a low-cost flexible prototyping and production platform for extensible, Power-over-Ethernet based smart appliances. We describe the first ELL-i prototyping board, a number of application concepts, and discuss its business model.

Key words: Internet of Things, Context Awareness, Direct Current (DC) power transmission, Power over Ethernet

1. Introduction.

Making real-life appliances intelligent and Internet-connected not only enriches the amount of environmental information available to applications but also creates the promise of making the environment itself a situation-aware part of the "applications". As we can easily see from the multitude of sensors that are available in today's smartphones, once we realise the Internet of Things (IoT) vision [5] of making each "light bulb" Internet connected [12], the amount of available sensor data is likely to multiply. Furthermore, through making the appliances not only sensitive but also active, the appliances may themselves become more aware of the nearby humans, reacting and pro-acting in a situation-aware manner [29].

Considering physical space, a key aspect of making a room, building, or any physical space context aware is to have a sufficient number of sensors and providing a framework for pre-processing the data [6]. This is typically expected to take place through installing a *sensor network* to an already existing building or space, and routing the collected data to an aggregation service. In the large majority of the recent works, the sensor network is expected to be a wireless one, to the extent that the most widely referred surveys carry the word "wireless" in their title [1, 2].

At the same time, most of the works have implicitly ignored how the sensors are to be powered or assume that they have a battery that will last for their lifetime. Additionally, in most of the published research, the presented use cases and applications assume the use of some wireless connectivity technology, although the primary ideas do not specifically refer to either wireless or wired networking, or to a any specific information and communications technology. Looking at the situation from greenfield applications point of view, or more widely from a sustainability perspective, assuming that the sensors will be replaced every few years, or even every tens of years, does not look like a very rewarding alternative [45].

While it may well be possible to create sensor nodes that scavenge energy from the environment and use radio or light bursts to communicate with the environment [52], creating wireless powered every day appliances that have some other active function than mere sensing seems improbable. Hence, it seems prudent to assume that a large majority of different appliances will remain wired, including air conditioners, blinds, coffee machines, doorbells, electroliers, and freezers, to mention but a few.

In this paper we argue that it makes economic sense to introduce a public open source platform for wired, communicating, intelligent "things". Once established, such a platform would allow designing and producing inexpensive appliances that are able to monitor their environment, participate in wider situation data collection and perusal, and act in accordance with the users' explicit and sometimes even implicit intentions. In particular,

[*]ELL-i open source co-operative, Helsinki, Finland. (pekka.nikander@ell-i.org). Questions, comments, or corrections to this document may be directed to that email address.
[†]Department of Information Technology, University of Turku, Finland. (vadrao@utu.fi).
[‡]EIT ICT Labs, Helsinki Node, Finland. (petri.liuha@eitictlabs.eu).
[§]Kungliga Tekniska Högskolan, Stockholm, Sweden. (hannu@kth.se).

we introduce the ELL-i platform, an open source initiative that aims to provide a low-cost prototyping and production platform for embedding wired intelligence into buildings, appliances, and other implements.

As its first technical platform, the ELL-i initiative provides open source hardware and software for building inexpensive embedded intelligence into devices, allowing them to communicate and be powered with Power-over-Ethernet (PoE) [25], an established but rapidly evolving standard for providing up to 100 Watts of electric power [21] through standard Ethernet cables. The currently available ELL-i prototyping board is compatible with the popular Arduino [7] prototyping platform.

The ELL-i initiative is organised as a co-operative, providing an egalitarian ownership model that anchors the property rights with the community. The governance model is designed to be meritocratic, assuring that the platform will evolve in a technically sound manner. The incentive models are designed to inspirit participation from a wider open source developer community while still allowing mass-market production by commercial companies.

The rest of this paper is organised as follows. First, in Section 2 we consider related work from a relatively wide perspective. Then, in Section 3 we introduce the ELL-i platform from a technical point of view. In Section 4 we briefly describe some example applications that are immediately viable with ELL-i, followed in Section 5 with a succinct description of the business model. Section 6 enlists our ongoing work. We conclude the paper in Section 7, attempting to summarise the lessons learned so far.

2. Related work. Proliferation of cheap and versatile sensors, actuators, computing, storage and networking solutions, augmented with the desire to connect people with their devices in a meaningful way, have contributed significantly to the IoT research. In this work we focus on an almost banal way for providing both the communication and powering solutions to fixed and immovable devices, which would anyhow be powered by wirelines, using Power over Ethernet (PoE) standard.

In this section, we give a general and broadly accepted definition for Internet-of-things (Sect.2.1), explore IoT applications (Sect.2.2), and delve quickly into the power management and security related aspects in IoT and M2M research (Sect.2.3). We conclude by enumerating some other open source embedded PoE platforms (Sect.2.4).

2.1. Internet of Things and Machine-to-Machine. Perhaps the first definition of what later became Internet of Things was given by the Internet pioneer Vint Cerf in this ACM conference talk in 1997 [12], where he envisioned each light bulb having their own Internet address. The term Internet-of-Things (IoT) was first used by Kevin Ashton in 1999, when he referred to identifiers placed in objects so that they could be monitored and managed by computers [4]. It became related to the famous and more generic definition of the disappearing computer by Mark Wiser [53], where he described that miniaturization and ubiquity of sensors would eventually lead to disappearing to the computational elements "into the fabric of everyday life". This has been later denoted to as pervasive or ubiquitous computing.

More recently, one of many definitions of IoT is given by the Cluster of European Research Projects (now IERC), who defined IoT as a "dynamic networked global infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual things have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated to the information network" [51].

A second dimension and branch of development of IoT is the so called Machine-to-Machine or M2M concept, which further refers to linking the plethora of digital devices (from simple microcontrolled devices to smartphones and large industrial instrumentation). This refers more to the communications protocols required to allow devices to communicate with standard communications, typically wireless or even more specifically cellular systems. A couple of popular projects developing the technology are the Eclipse M2M Industry Working Group [18] and ITU-T Focus Group M2M [27].

2.2. Internet of Things applications. IoT applications include a variety of applications ranging from tracking and tracing of objects to applications applying wireless sensor networks and Machine-to-Machine (M2M) systems. Typical applications are in military use cases (like intelligence, surveillance), environmental use cases (e.g. monitoring conditions that affect farming and animal husbandry, or monitoring macro-scale phenomena in soil, forest, marine or atmospheric contexts), health applications (remote monitoring of patients,

tracing of care personnel or equipment), home applications, or other commercial applications, such as interactive smart spaces in office or other working environments.

The use cases include applications with large area networks and high mobility but also no-mobility use cases. Wu et. al. discuss the requirements for future applications from the M2M perspective [54]. While many of them assume wireless connectivity, a good part of them are in fact for fixed things, like in home management, industrial automation and metering, sensors, and lighting. More generally, there are IoT use cases with no mobility in smart homes or smart buildings and smart cities; e.g., for the purpose of reducing the consumption of resources associated with buildings (electricity, water) or improving the satisfaction level of humans populating them [39].

2.3. Power management and security in IoT and M2M research. In this work, IoT devices have been classified by the amount of power required for their efficient operation. Some IoT devices use batteries, some scavenge energy from radio waves (RFID and NFC), while others are powered by separate power supply from the mains. Smart and tiny embedded systems have sensors, actuators, CPU, memory and a low-power communication device [34]. Usually, these can be powered by a battery. On the other hand, Wireless Sensor Networks (WSN) use energy efficient battery powered wireless devices for transmitting sensor data [2].

From the security point of view, as the complexity of the encryption algorithm increases, their battery consumption increases as well. It has been shown that among symmetric encryption algorithms used in security protocols, the AES128 algorithm increases the battery consumption and the processing time by 75% and 65% respectively when compared to "no encryption" results [20]. Furthermore, increasing the key size for the encryption also increases the battery consumption and processing time.

2.4. Open source embedded platforms. Today, there are a few tens of slightly different open source embedded platforms. The Raspberry Pi [42] forms a category of its own, offering an unbeatable price point in the embedded Linux category. In the lower category, the Arduino [7, 23] project has gained the largest user community, with half a dozen different official boards and a number of derivatives.

The Arduino project is a somewhat loose collaboration led by a number of people that developed the original Arduino design. They have decided to open source all of the software and hardware but to keep the Arduino and associated graphics as a trademark. At the moment there are three companies that are authorized by the project leaders to produce official Arduino boards [37]. As another example, the Raspberry Pi is produced by the Raspberry Pi Foundation, which is a UK registered charity supported by the University of Cambridge Computer Laboratory and Broadcom [28]. For its first three years of operation, the foundation income was annually less than £10.000. The foundation income and spendings for 2012 were not registered at the UK Charity Commission at this writing.

There are other Arduino and Raspberry Pi based PoE solutions [11, 47, 48, 49] currently available in the market. Some of them are complete PoE solutions, while others are shields that can be plugged into an Arduino. The Arduino Ethernet with PoE [48] is an Arduino development board with the built in Wiznet Ethernet interface. The Arduino Ethernet Shield with the PoE Module [47] allows an Arduino board to connect to the Internet. It is based on Wiznet W5100 Ethernet chip which provides a network IP stack capable of both TCP and UDP. The PoEthernet Shield [49] gives access to the Internet via the Ethernet library and also allows the Arduino based projects to power itself from the Ethernet line. The Raspberry Pi Power Over Ethernet (PoE) Adapter [11] is designed to be plugged directly on to a Raspberry Pi and power it with +5 Volts at up to 1.0 Amps output.

All the above solutions focus on providing power to the MCU and its digital peripherals, powered through the 3.3V and/or 5V systems. The long term vision for ELL-i is significantly different. ELL-i is a multi-voltage domain design making the "raw" 48V from PoE available. ELL-i is aiming to be an inexpensive platform for low-cost PoE devices that come with integrated cloud services.

3. ELL-i platform architecture. In this section, we describe the ELL-i technical architecture in sufficient detail to ground the discussion that follows[1]. First, we describe the overall technical architecture in a bottom-up-fashion, including hardware, firmware, and the forthcoming cloud components. After that, we introduce the current implementation. We conclude the section with a brief discussion of the design choices made so far.

[1]For further details, please refer to the open source design files at `http://www.github.com/ELL-i`.

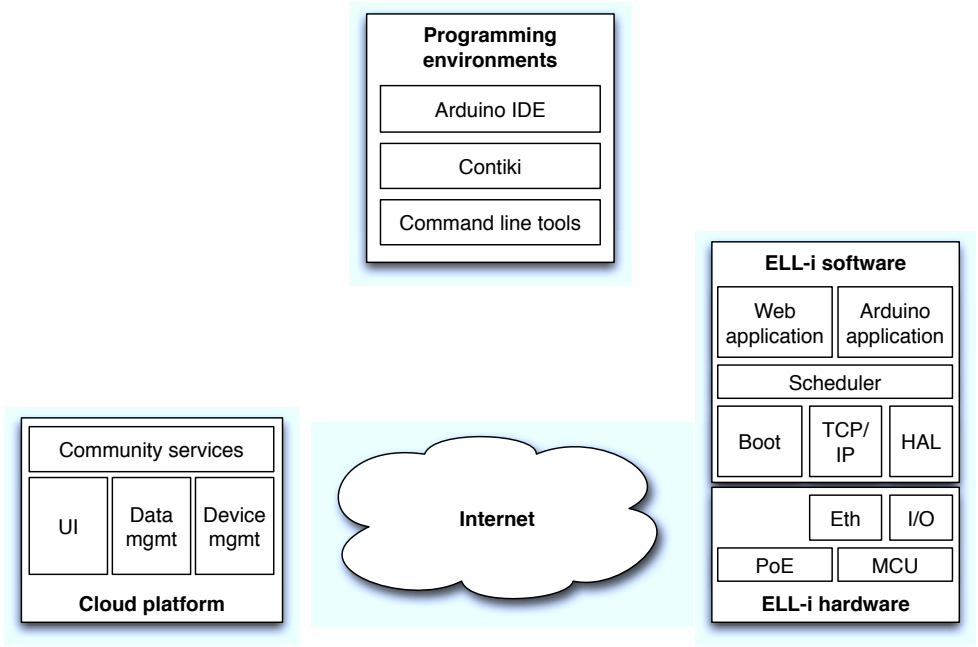

FIG. 3.1. *ELL-i platform architecture*

3.1. Overall technical architecture. The ELL-i platform consists of a hardware architecture, a runtime (firmware), cloud-based components, and support for a set of programming environments. The basic architecture is depicted in Figure 3.1.

The hardware architecture is based on combining power and communications on a single cable, making both of them available to applications. In practise, the current implementation is based on the IEEE 802.3at [25] Power-over-Ethernet standard, which is an economical solution when the power consumption is roughly in the 1–100 Watts range. In the long run, we expect to expand the power range to both smaller and higher power levels; for example, there are no technical reasons why Power-over-Ethernet technology could not be applied to thicker wires and higher currents.

The main goal of the runtime and the cloud components is to make it easy to provide connectivity for the appliances. Hence, the goal of the runtime is to package TCP/IP based protocols, such as HTTP [19] and CoAP [46] in an easy-to-use way. As it is undesirable to perform packet processing in an interrupt context and as many of the protocols depend on timeouts, abstracting the protocol processing away from the runtime API requires some kind of threading.

ELL-i is positioned as an easy-to-start platform. Hence, we have focused on providing an Arduino IDE extension that allows Arduino skills to be readily applied on ELL-i. However, we also realise that the default Arduino runtime is geared for hobbyists. Therefore we will also support XCode and Eclipse.

3.1.1. Hardware. The platform hardware consists of an MCU, an Ethernet communications module, a power supply module, and an I/O subsystem that connects the system to the real world. The MCU provides the smarts, including a computational core, non-volatile (flash) and volatile (RAM) memory, and peripherals for connecting with the communications, power, and I/O subsystems. It executes a program stored into its flash, converting commands received over the Internet into changes into its I/O state and vice versa. The MCU program is primarily responsible for implementing the TCP/IP protocols. In a typical case, it takes also (partial) responsibility for controlling the DC/DC power supplies, thereby reducing the overall cost of the system.

The Ethernet communications module implements the IEEE 802.3 [24] communications standard, sharing the same cable with the PoE-based powering of the device. It consists of three components: the isolation

magnetics, the physical interface (PHY) and the media access control interface (MAC). The firmware takes care of the actual packet processing.

The power supply module consists of a Power-over-Ethernet signalling module and a DC/DC power supply module. The former implements the IEEE 802.3af or 802.3at Power-over-Ethernet standard, including the physical-layer signalling that consists of an initial resistive-load based detection phase, a current-sink based classification phase, an optional second classification mode added in 802.3at, and an operating-voltage ramp-up phase.

The DC/DC power supply is responsible for converting the incoming Power-over-Ethernet 48V voltage into the 5V and/or 3.3V needed by the MCU and other digital electronics. In a specific ELL-i application, there may be an additional power supply producing "bulk" power for the "real world" components, such as a constant-current PSU for a high-power LED light, a circuit generating suitable current pulses for a solenoid, or a precisely controlled PSU generating modulated currents for a direct-current driven electric motor. In most real world applications there is a difference between the power requirements of the digital electronics, which typically require constant voltage but variable current, and that of the real-world actuators, which typically require controlled current but are relatively insensitive to the actual voltage.

Finally, the I/O subsystem adjust the electrical signals generated by the sensors and required by the actuators with the levels created and tolerated by the MCU. In a typical case, the I/O subsystem consists of operational amplifiers acting in amplifier and/or level shifter role. In many cases also opto-isolators or other isolators are needed.

3.1.2. Runtime. The runtime system forms the basis of any firmware running on the ELL-i platform. It consists of a software library that is linked into any specific firmware build by an application developer. The runtime provides a number of APIs and services that make application development easier than building everything from scratch.

In ELL-i, the main goals of the runtime are to provide an easy-to-start but still powerful programming environment for both novices and professionals, and to offer facilities for communicating with the cloud components in the Internet. To achieve that, the ELL-i runtime consists of a boot module, a hardware abstraction layer (HAL), a tiny scheduler providing concurrency, an Internet-communications module, an optional built-in Webserver, and a set of APIs providing programmatic access to all of these.

The set of APIs may be divided into five groups, corresponding to the booting, HAL, scheduler, Internet, and Web-services. In the current implementation, the boot and HAL APIs provide an Arduino-compatible programming environment, allowing programmers to initialise the hardware using the Arduino `setup()` function and to run a main loop within the `loop()` function. Both of these are called by a `main()` function, allowing more advanced programmers to override the Arduino approach. The familiar Arduino HAL APIs are supported, such as accessing GPIOs and serial lines.

While the original Arduino runtime does not provide any concurrency, thereby causing e.g. busy loops to stall the whole MCU, in the ELL-i runtime there is a small scheduler running "under" the Arduino APIs. Using the scheduler, the Internet and Web-server modules are implemented so that they always run concurrently with any Arduino sketch.

The scheduler API provides services for starting and stopping concurrent threads and adjusting the scheduling algorithms. By default threading is fully pre-emptive, allowing the TCP/IP stack to work on a higher priority than any Arduino sketch. Any I/O modules requiring precise timing need to run on their own threads, or in their interrupt context, in order to preempt packet processing. Dynamic memory management is not supported and is strongly discouraged; any threads need to have their stacks to be allocated at compile time.

The Internet API provides services for operating TCP, UDP and CoAP [46]. The TCP/IP stack is based on uIP [15]. We support Contiki [16] style protothreads [17], thereby making it easier to support existing uIP-based protocol stacks and applications. The optional built-in Web-server allows the ELL-i platform to provide AJAX-style Webservices towards the local network. For security reasons, the TCP packet lifetime is limited to one hop, thereby making the webservice unreachable from the Internet even in the case the node has a public IP address and full Internet connectivity.

The runtime library has been carefully built in a way that allows the linker to leave out any modules that are not used by a particular application. However, whenever the default `main()` function is used, the runtime

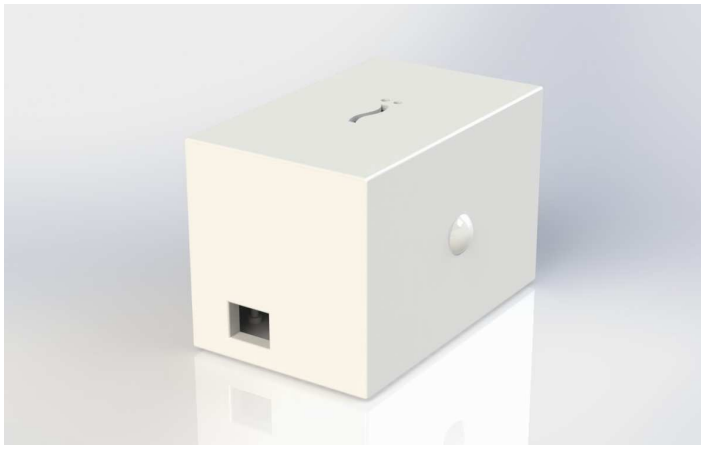

FIG. 3.2. *ELL-i case design*

includes the UDP/IP stack, the CoAP protocol, and a basic service module that registers the node with the cloud components and by default sends sensor data to the cloud.

3.1.3. Cloud components. In addition to the software components running on the ELL-i hardware, the ELL-i platform contains also cloud components, running on a compatible cloud runtime, such as Amazon Web Services (AWS). The cloud components provide interaction, update and community services to the ELL-i boards, and they are enabled by default in each ELL-i development board.

In the default configuration, a newly installed ELL-i development board attempts to contact its counterpart component at the cloud side, making the cloud-side component aware that the board is running and has Internet connectivity. At the same time, the user may direct their browser to a board-specific URL and view the board status. The board may also periodically send sensor information to the cloud, such as a reading from the MCU-internal temperature sensor. This data is then visualised to the user.

Another cloud-based service is the ability to remotely upgrade the firmware on ELL-i boards. If the remote upgrade is enabled, the firmware contains a tiny CoAP-based communications module that is able to download new firmware page-by-page to the RAM, and flash it. By default, each page must be protected with a symmetric cryptographic key, shared between the board-specific cloud component and the board itself. If the remote upgrade fails and the board no longer boots, it still remains possible to completely re-flash the board using a serial line and a separate programming board.

3.1.4. Programming environments. The ELL-i platform is designed to support multiple programming environments, including the Arduino, Eclipse, XCode, and for the old timers, plain `emacs` and `make`. At this writing only the Arduino IDE is formally supported, even though some of the early adopters do use command-line tools. The aim is to make it *easy* to move from the Arduino IDE to the more advanced programming environments, with clear instructions on how to convert Arduino sketches into proper C++ source files. This also requires clear explanations of what is happening under the hood in the runtime.

On the compiler side, at the moment we are still using the very GCC version provided by the default Arduino IDE. However, our plan is to move to LLVM [32] as soon as possible, and to enable global link-time optimisation. This together with well-written static initialisation of global C++ objects, such as the Arduino `Serial`, allows the compiler in many cases to optimise away not only virtual function calls but use constant propagation down to the level where the Arduino syntactic sugar may be optimised completely way, providing the same level of efficiency as would normally be achievable through bare-metal access while still working with high-level C++ abstractions. We are also considering whether it would be possible to pre-run the Arduino `setup()` function and some of the preceding setup machinery during the compilation time [43], providing boot-time optimisations.

3.2. Current implementation. At this writing in September 2013, the currently available implementation consists of the first prototyping board, an initial runtime, and support for the Arduino and Contiki APIs.

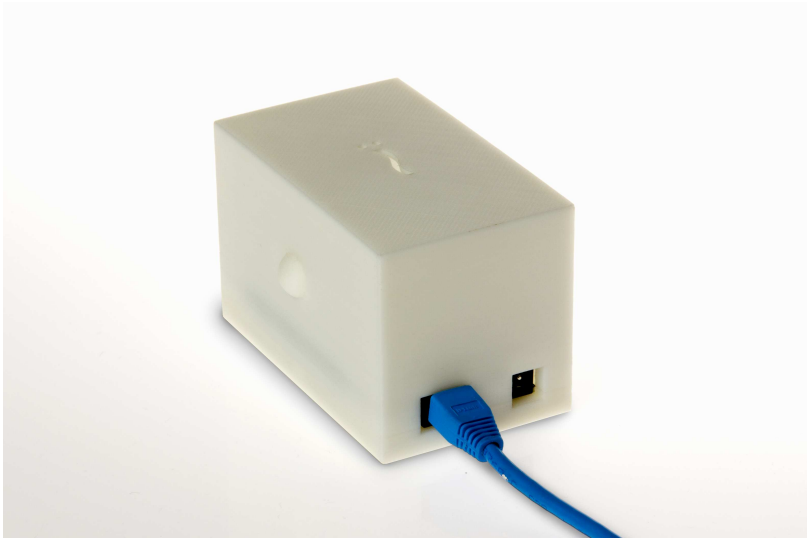

FIG. 3.3. *ELL-i 3D-printed prototype case*

Work on the cloud-side components is to be started. The goal of the first prototyping board was to create Arduino-compatible hardware and software that would support a more powerful MCU than the present basic Arduino's do and to allow custom DC/DC power supply design. This goal has been reached. With the first prototyping board, prospecting developers can utilise the existing Arduino examples, learning material, and shields, to a large extent.

The first ELL-i prototyping board was designed during spring 2013. These initial boards have an STM32F051 [50] ARM Cortex-M0 microcontroller, Microchip ENC28J60 [38] Ethernet chip, Linear LTC4267 PoE controller with build-in flyback SMPS controller, and Arduino-compatible headers. We also have a 3D printed case for the prototyping board, with enough of space for a few Arduino shields. The current computer generated case design is depicted in Figure 3.2 and a corresponding 3D-printed prototype in Figure 3.3.

For Arduino-compatibility, we designed the LTC4267-based flyback DC/DC converter to produce 5V, using a standard linear regulator to produce 3.3V out of the regulated 5V supply. That allows us to replicate Arduino Due and Arduino Mega I/O pins with some accuracy. The I/O pins have been assigned so that there is a separate timer/counter channel available for all of the digital I/O pins, an ADC channel for all of the analog pins, and the DAC on the STM32F051 is connected to the Due DAC0 line.

As the first programming environment, we have been focusing on supporting the Arduino IDE. Starting with the Arduino IDE 1.5 extension mechanism [35], we have developed our own Arduino-compatible runtime that compiles about over 90% of the sketches provided with the Arduino IDE. However, while most of the APIs are supported at the syntax level, there is still quite some underlying code missing.The existing functionality is enough for most of the basic sketches, though, thereby helping people to get started with ELL-i.

3.3. Design choices. With the ELL-i platform, our goal has been to produce an easy-to-use, flexible, and inexpensive platform for fixed appliances. These goals have guided our design choices. Indeed, the primary choice of using Power-over-Ethernet stems from this goal, as it allows us to use a single cable for both power and communications, thereby potentially reducing the overall cost while still being flexible and easy-to-use.

Considering the component choices in the first prototyping board, we have tried to select the lowest-cost components that are easy-to-use and flexible enough. Consequently, we opted for the lowest end ARM series, Cortex-M0, as they are easy-to-use as a familiar 32-bit platform and most likely to hit the lowest price point in the area within a few years[2]. The chosen Ethernet chip is the lowest-cost denominator of those widely used

[2]At the moment, there are still older Cortex-M3 based MCUs on the market that are sold cheaper than the newer Cortex-M0 MCUs in small lots. However, the lowest price point is most likely to be replaced with the Cortex-M0 MCUs within the next couple

in the market. At the PoE and DC/DC PSU side there is still room for optimisation, but we are currently planning to leave that for later.

For hardware-level expandability, we are still working to determine the right approach. At the moment, we are using the Arduino-compatible I/O connector design for development-time extensions. However, it is clear that they are not a good choice for production use, for many reasons[3].

Considering the connectors and mechanics, our goals are again the same: easy-to-use, flexible, and inexpensive. For Ethernet, the goal is to support both the familiar RJ45 connectors and the punch-down connectors for fixed design, the punch-down connectors being clearly cheaper. In the new design, we are looking at various options for the connectors between the production and development boards. At the moment, we are planning to allow different approaches for tying the boards together, including using screws and having suitable notches in the packages.

At the firmware side, it looks likely that we will end up implementing our own runtime. Looking at the various open source RTOS options available, they typically include more than what we need and with a license that is not fully compatible with our planned business model; see below. The cloud-side components will be based on the lowest common platform-as-a-service (PAAS) denominator, which at the moment is AWS APIs, also supported by e.g. the AppScale [13] open source framework.

4. Application examples. As a platform, the ELL-i design is open ended. In this section, we briefly describe a few designs that we presume the ELL-i platform making easier or less costly to implement than before. Furthermore, wherever applicable, we attempt to illustrate how the open source nature of ELL-i will benefit the individual applications.

4.1. Illumination. The very first demo that we implemented using ELL-i was a driver for a commercial high-power RGB LED projector. Since then, we have repeated the LED driver exercise a few times.

From the electronics point of view, an adjustable high-power LED driver is a relatively simple device, as LEDs in general are current-driven components. That is, the voltage drop over a LED remains relatively constant independent of how much current is flowing through the LED, and the amount of current determines how brightly the LED shines. The typical voltage drop over a high-power LED is around 3V. Hence, for driving a LED from a voltage source that is higher than the voltage drop of the LED it is enough to limit the amount of current. For indicator LEDs the current is usually limited with a suitable series resistor; however, such an arrangement would be very inefficient for driving a high-power LED out of the PoE nominal 48V. Fortunately, a buck converter, the simplest of the so-called switching mode power supply (SMPS) [36] topologies, is quite easy to regulate to produce a constant, adjustable current. All that is needed is a simple current-measurement circuit and a simple algorithm to drive the switching FET in within the driver. While achieving high efficiency requires somewhat more work, even an electronics amateur, like us, is able to design a simple MCU-controlled buck converter in a couple of weeks time.

So, from the hardware point of view, driving LED-based lamps would be an ideal application for the ELL-i platform. In addition to the components already in ELL-i, a LED driver at its simplest form requires only few inexpensive components, including a switching FET, a choke coil, and a diode. Consequently, ELL-i may be an excellent platform for implementing future, more intelligent show lighting systems, flexible retail lighting, and other illumination applications where it is beneficial to control each and every lamp separately. In such an environment, ELL-i implements Vint Cerf's vision of each light bulb having its own IP address [12].

To communicate with the external world, and ELL-i node may implement the Ethernet or TCP/IP variants of the commonly used lighting control protocols, including DALI [14] and DMX [9].

4.2. Automation. Another area where the ELL-i platform may bring benefits is building and industrial automation. In the building automation area, there are already a number of Power-over-Ethernet products on the market, including locks and other security and safety equipment. However, to a large extent the existing

of years.

[3]The Arduino-compatible connectors are physically quite large, requiring more space than what is available within many devices and e.g. within a pattress box. Furthermore, the Arduino-compatible connectors are basically unbuffered, making it somewhat tricky to get good ADC readings or driving higher loads through them.

automation systems are still based on standards, such as BACnet [10], KNX [30], LonTalk [26], or ModBus [40], which generally utilise twisted pair control cables and separate power supply.

From that point of view, ELL-i provides an open source platform to build upon. As such, ELL-i provides little benefit directly to automation, as it is lacking direct connectivity with sensors and actuators, and it does not as such implement any of the existing protocols. Therefore, as a standalone component it does not integrate with the existing automation systems.

Being a small, connected, intelligent, power-convoying low-cost component, ELL-i could potentially be co-located with the sensors and actuators, providing power to them and integrating both intelligence and connectivity directly to them. Consequently, it would allow replacing the hierarchy of separate field devices (sensors and actuators) and the programmable logic controllers (PLC) with a flat logical hierarchy, allowing the physical connectivity to be arranged in any topology.

Beyond the basic functionality challenges, applying ELL-i in automation systems would require a number of non-functional challenges to be addressed. First and foremost, it should be demonstrated that the ELL-i system is reliable as the known and proven old solutions. As it is unlikely to happen as such any soon, one way to address the reliability concern is to build reliability through redundancy. That is, a system consisting of the ELL-i platform based nodes can be engineered to be inherently redundant, placing two ELL-i nodes at each sensor or actuator side, using standard Ethernet network redundancy approaches[4], and addressing software-level redundancy through differing runtime systems.

4.3. Education. A third area where ELL-i may turn out to be useful is education. As an open system, ELL-i itself can be studied and modified by the students, and it could as such be extended to be a platform for teaching basic programming. Secondly, as an easy-to-use embedded system ELL-i may be used in various student projects where real-life interaction is needed.

4.3.1. ELL-i platform itself as course material. The ELL-i platform has been designed to be not only easy to use but also easy to understand. The hardware represents a complete design with an MCU, Ethernet-based communications, and a DC/DC PSU. The chosen MCU is a relatively simple but still a powerful one, from a family that is commonly used in open source projects. The CPU core within the MCU is a modern 32-bit RISC architecture with simple 3-stage pipelining and no instruction reordering, being easy to teach and to understand. The MCU-related part of schematic is very similar to the relevant part of the MCU manufacturer's evaluation board. The Ethernet communications module consists of a single chip and the magnetics and has a wide support in the open source community.

At the software side, the firmware has been written in a modular manner, as was explained above in Section 3.1.2. Hence, studying simple firmware designs that do little but initialise the MCU and blink some LEDs is relatively easy. The modularity then allows more complex features, such as communications and scheduling, to be added in a piecewise manner to the design. This allows the teacher to demonstrate the full runtime in a gradual manner, starting from MCU initialisation and simple I/O, continuing through the SPI and I2C protocols up to simple Web services.

Hence, the ELL-i platform may be used to introduce and demonstrate all the elements of a modern Web-connected computer, including both the hardware and software, without the complexities.

Given that the Arduino APIs and sketches are C++ programs in practise, it would be possible to write a set of C++ proxy classes and templates that compile an Arduino sketch so that it runs in an emulator, allowing the functioning of such a sketch to be simulated and illustrated on a PC. Furthermore, with the Ethernet interface, another set of proxy classes could be used to instrument a sketch, allowing it to be run on the ELL-i platform while simultaneously inspecting the variables and I/O ports. The instrumentation classes would communicate the application state over the Ethernet to a PC, which would then use the simulator user interface to illustrate the application state.

4.3.2. ELL-i as a student project platform. Due to the simplicity and the modular design of the ELL-i platform, it is a relatively easy platform for students to build their platforms upon. Arduino compatibility allows adoption of almost any of the some 300 existing Arduino shields, typically with some minor software-side modifications. In many cases, the more powerful MCU on the ELL-i platform allows the sensors or other

[4]Dual cabling with redundant switches, spanning tree protocols to avoid bridging loops, etc.

components on the shield to be used more intelligently than with the basic Arduino boards. Ethernet-based powering allows the devices to be placed relatively freely anywhere, thereby allowing the students to build smart spaces or place smart fixed devices at desired locations.

The simplicity and flexibility of the platform also allows the students to replace many parts of the platform with their own components. For example, at this writing, a student group at University of Turku have completed a preliminary port of the FreeRTOS [8].

EIT Smart Spaces summer school. In June 2013, a couple of weeks after the first ELL-i prototyping boards had arrived, two of us held a one-day ELL-i tutorial at the European Institute of Innovation and Technology (EIT) Smart Spaces summer school in Grenoble, France. The tutorial lasted for six hours, within which the computer science students built a simple buck converter based LED driver, using the ELL-i platform, discrete components, and the Arduino IDE.

During the tutorial, the students learned how to handle basic physical electronic components, including resistors, capacitors, coils, FET transistors, diodes, and LEDs. Starting from a very simple design, they built in a stepwise manner an adjustable constant-current DC/DC converter that generated some 300 mA of current from the nominally 48V unregulated DC power available from the PoE header on the ELL-i prototyping board. The background of the students varied from ones that had only taken theoretical basic courses in electronics, as a part of their basic studies, to a few that had built simple Arduino-based prototypes before. None of the students had previous knowledge of how the DC/DC power supplies actually work.

5. Business model. In this section, we briefly look at the ELL-i business model, which somewhat deviates from most typical business models, being closer to the Linux kernel community [22, 33].

As stated, the goal of the ELL-i project include providing an inexpensive and easy-to-use platform for all kinds of fixed appliances. As always, in electronics production achieving a low price requires high volumes. Hence, the ELL-i business model aims to make the platform widely available at a low cost, thereby encouraging it to be widely used, leading to increasing volumes and thereby even lower prices. In other words, the project does not aim to create profits as such but to distribute the added value in the form of lower prices, thereby contributing to the economy as a whole.

The ELL-i project itself has been incorporated as a co-operative, with the intention that the hardware and software copyrights are co-owned by the developers. That is, the aim is that the co-operative owns the copyright for all of the hardware and software components within the core ELL-i platform and the developers who contribute to the project may join the co-operative as members if they wish to do so.

We envision that the project and the co-op will create interest among commercial for-profit companies. This is important both for creating volumes and for distributing the created value to the economy. Hence, the ownership, governance and licensing models have been planned and will be adjusted to make the platform lucrative for commercial licensees.

In order to get the platform there in the first place, we first need to entice a group of enthusiastic developers that want to contribute to the project. For that, we presume that we first need to sell a few thousand boards. We expect that 1–4% of the early adopters will turn into active developers [3]; see Section 5.1 below.

The ELL-i business model needs to be balanced between the early market need of creating a sizeable developer community and the somewhat later need of attracting commercial companies in order to create volumes and benefiting the larger society.

The resulting structure has probably a relatively low monetary value compared to its overall utility value, as the ownership will be more anchored than in most alternative models. However, we consider this as a virtual from the macroeconomic perspective [41], somewhat similar to what state-owned enterprises provide but detached from the populist political system.

5.1. Early market challenges. At the moment, the main challenge ELL-i faces is a marketing one. The first version of the hardware platform exists both as open source schematics and layout and as actual, functioning hardware boards. The software platform is being continuously enhanced, providing the basic facilities that are needed to get started. However, the group of active developers is small, just a handful of people. In order to get the ball rolling, more active developers are needed.

We believe that the ELL-i approach of providing both power and communications over a single cable is beneficial to a fair number of people, especially when the incoming power can be easily converted into a DC

voltage suitable for various kinds of loads. Hence, from the ELL-i point of view, a major challenge is to find the people that are already struggling with the appliance powering problem and demonstrate the viability of the ELL-i solution. That in turn requires much more visibility than what ELL-i enjoys today and a number of additional power supply designs, making it easier to have suitable power tailored for each load.

An initial market presence may be achieved through crowd-funding [31]. That may help with gaining initial production volume and inducing a few new developers; however, such a campaign is also likely to saturate the early market, making it considerably harder to sell new hardware for a while. Hence, at the time a crowd-funding campaign is commenced, there must be sufficient structure in place so that the new developers have a clearly laid approach and incentive to start contributing to the project.

What comes to the powering of appliances, we basically need to learn ourselves – within the core group – to build a more diverse set of power supplies. We also need to make the resulting knowledge available in a form that is easier for relative newcomers to apply than the currently available information, initially as recipes and later on as ready-applicable DC/DC converter units. The basic ELL-i recipe for providing adjustable constant-current power is already there, thereby enabling adjustable LED drivers, and we expect soon to complete our first design for inductive loads, such as solenoids and motors.

5.2. Business model considerations. As a first approximation from the conventional economics point of view, the ELL-i platform seems to form a fairly classical two-sided market [44]. ELL-i offers the platform essentially free-of-charge to the developer community, allowing individual developers to innovate and fulfil their personal needs. At the same time, it offers the platform for commercial companies for a fee. At the same time it attempts to form a joint community of both individual developers and product companies, creating a larger two-sided market where the sides are the hardware component manufacturers and the product users. From that point of view, ELL-i benefits the component suppliers through larger volumes, the product companies through increased bargaining power, and the product users through increased variety and lower average price of the products.

From a societal point of view, if ELL-i succeeds as a platform it may produce essentially a virtual non-profit monopoly, benefiting the society through the increased efficiency of scale of the used components and overall lower user prices of the technology. At the same time, its egalitarian ownership structure anchors the forming intellectual capital within the community while the meritocratic governance structure allows any further development to be primarily technology driven, aiming for long-term engineering excellence instead of short-term profits.

6. Ongoing work. As a platform and a business model, ELL-i is still at its infancy. As a development platform, we are still busy laying out the building blocks: the next version of the hardware is being worked on, the software is receiving constant improvements, and at the tools side we are smoothing out the developer path from the easy-to-start-with but simplistic Arduino IDE to the more powerful full-fledged tools. At the same time we are looking at how to expand the technology to convert a wider variety of environments; for example, it would be very desirable to use the ELL-i platform without requiring any new cabling when renovating existing installations.

From the community building point of view, ELL-i has but started. We are still considering how to best launch a campaign to expand the developer base beyond the founders of the co-operative. It seems clear that we need some sort of a crowd-funding campaign to attract developers and to gain initial volumes.

Finally, the governance and licensing models clearly need much more work. Our understanding of how to structure the business model in a way that allows both fast organic growth and sustained technical development is still inadequate, requiring both better theoretical understanding of the various stakeholder incentives and real-life experimentation in the form of trying out a few alternatives.

7. Conclusions. Realising the Internet-of-Things (IoT) and Machine-to-Machine (M2M) visions will make our environment immensely contextual in the sense that the digitally interconnected artefacts in our environment may jointly form a rich model of what is taking place in any given space, allowing the environment to react in a smart manner. In this paper, we have discussed one potential step towards that vision, the ELL-i initiative and technical platform.

The ELL-i initiative is organised as a co-operative, with the intention of producing an inexpensive open

source hardware and software platform for wired appliances and implements, such as aquaria, bowling alleys, cooktops, dishwashers, all the way to toasters, underlayments, vacuum cleaners, washers, xerox machines, yachts, and kitchen zinks. The currently available ELL-i prototyping board allows researchers and other developers to explore the ELL-i approach, creating prototypes of intelligent, communicating things that can be readily located anywhere at the distance of an Ethernet cable. With its Arduino compatibility, the ELL-i platform provides a very low entry barrier, while still outlining a clear path for more powerful approaches.

From the technical point of view, the main promise of the ELL-i approach is in its ability to provide reasonable amounts of electrical power, combined with the communication and data processing capability, at a very reasonable price point. If ELL-i ever reaches mass-production quantities, it will provide an electronics module that allows almost any wired electrical device to be converted into its "smart" equivalent at the additional production cost of just a couple of euro. Consequently, as even a typical light switch or toaster is priced at a level close to or exceeding ten euro, that may gradually enable complete "smartification" of our urban environment.

The ownership, governance, and incentive models built into the ELL-i co-operative have been designed with sustainability in mind. They aim to create an openly governed, jointly owned platform that is firmly anchored within the society and developed as an essentially public good.

Acknowledgements. The authors would like to thank Antti-Juhani Filpus, Dmitri Fursov, Teemu Hakala, Jukka Korpihete, and Matti Viita for their constructive comments on various versions of this paper, as well as the founding members of the ELL-i open source co-operative for their initiative and courage.

REFERENCES

[1] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.
[2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.
[3] Chris Anderson. *Makers: the new industrial revolution*. Random House, 2012.
[4] Kevin Ashton. That 'Internet of Things' thing. *RFiD Journal*, 22:97–114, 2009.
[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.
[6] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.
[7] Massimo Banzi, David A Mellis, David Cuartielles, Clay Shirky, Paul Badger, Tom Igoe, Federico Fissore, Roberto Guido, and Michael Shiloh. Arduino. The official Arduino web page at http://arduino.cc.
[8] Richard Barry. FreeRTOS. *Internet, Oct*, 2008.
[9] Adam Bennette. *Recommended Practice for DMX512: A Guide for Users and Installers*. United States Institute for Theatre Technology, 1994.
[10] Steven T Bushby. Bacnet: a standard communication infrastructure for intelligent buildings. *Automation in Construction*, 6(5):529–540, 1997.
[11] James Carter. *Raspberry Pi Power over Ethernet (PoE) Adapter by Xtronix Ltd*. Whitchurch Hill, Reading, Berks, United Kingdom, June 2013.
[12] Vint Cerf. The next 50 years of networking. In *the ACM97 Conference Proceedings*, 1997.
[13] Navraj Chohan, Chris Bunch, Sydney Pang, Chandra Krintz, Nagy Mostafa, Sunil Soman, and Rich Wolski. AppScale: Scalable and open appengine application development and deployment. In *Cloud Computing*, pages 57–70. Springer, 2010.
[14] Digital addressable lighting interface activity group. DALI Manual, Dali AG, ZVEI-Division Luminaires, 2001.
[15] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98. ACM, 2003.
[16] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki — a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.
[17] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 29–42. Acm, 2006.
[18] Eclipse foundation Machine-to-Machine industry working group – charter. Eclipse Foundation, Inc., Centrepointe Drive, Ottawa, Ontario, Canada.
[19] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1, 1999.
[20] Fadi Hamad, Leonid Smalov, Anne James, et al. Energy-aware security in m-commerce and the internet of things. *IETE Technical review*, 26(5):357, 2009.
[21] Introduction to Power over HDBaseT. HDBaseT Alliance, Beaverton, OR, USA, September 2011.

[22] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research policy*, 32(7):1159–1177, 2003.
[23] Karl A Hribernik, Zied Ghrairi, Carl Hans, and K-D Thoben. Co-creating the Internet of Things–first experiences in the participatory design of intelligent products with Arduino. In *Concurrent Enterprising (ICE), 2011 17th International Conference on*, pages 1–9. IEEE, 2011.
[24] IEEE 802.3 standard for Ethernet. IEEE Computer Society, New York, NY, USA, December 2012.
[25] IEEE 802.3at standard for Ethernet. Amendment 3: Data terminal equipment (DTE) power via the media dependent interface (MDI) enhancements. IEEE Computer Society, New York, NY, USA, September 2009.
[26] IEC 14908 Open data communication in building automation, controls and building management - control network protocol. International Electrotechnical Commission, rue de Varembe, Geneva, Switzerland.
[27] ITU-T focus group on M2M service layer. International Telecommunication Union, Telecommunication Standardization Bureau, Place des Nations, Geneva, Switzerland.
[28] Trevor Johnson, Steve Meirowsky, Johnathon Weare, Ryan Walmsley, and et.al. Raspberry Pi foundation. Wikimedia Foundation, San Francisco, California, USA.
[29] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, et al. People, places, things: Web presence for the real world. *Mobile Networks and Applications*, 7(5):365–376, 2002.
[30] KNX standard. KNX Association, De Kleetlaan, Brussels/Diegem, Belgium, 2011.
[31] Venkat Kuppuswamy and Barry L Bayus. Crowdfunding creative ideas: the dynamics of projects backers in Kickstarter. Technical report, SSRN Working Paper, http://papers. ssrn. com/sol3/papers. cfm, 2013.
[32] Chris Lattner and Vikram Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *Code Generation and Optimization, 2004. CGO 2004. International Symposium on*, pages 75–86. IEEE, 2004.
[33] Gwendolyn K Lee and Robert E Cole. From a firm-based to a community-based model of knowledge creation: The case of the Linux kernel development. *Organization Science*, 14(6):633–649, 2003.
[34] Tomás Sánchez López, Damith C Ranasinghe, Mark Harrison, and Duncan McFarlane. Adding sense to the internet of things. *Personal and Ubiquitous Computing*, 16(3):291–308, 2012.
[35] Cristian Maglie. Arduino IDE 1.5 3rd party hardware specification.
[36] Sanjaya Maniktala. *Switching power supplies A to Z*. Elsevier, 2012.
[37] David A Mellis, David Cuartielles, Federico Fissore, and Alberto Cicchi. So you want to make an Arduino.
[38] *ENC28J60 Data Sheet – Stand-Alone Ethernet Controller with SPI Interface*. Microchip Technology Inc., West Chandler Blvd., Chandler, Arizona, USA, 2008.
[39] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.
[40] Modbus application protocol specification v1.1b. Modbus Organization, Hopkinton, MA, USA, December 2006.
[41] Deborah Groban Olson and John Logue. Fix globalization: Make it more inclusive, democratic, accountable and sustainable. Capital Ownership Group, Ford Foundation, October 2002.
[42] Raspberry Pi. Rasberry Pi Foundation, Caldecote, Cambridgeshire, UK.
[43] Teemu Rinta-aho, Mika Karlstedt, and Madhav P Desai. The Click2NetFPGA toolchain. In *USENIX Annual Technical Conference*, 2012.
[44] Jean-Charles Rochet and Jean Tirole. Platform competition in two-sided markets. *Journal of the European Economic Association*, 1(4):990–1029, 2003.
[45] Diane Rowland, Carolyn DiGuiseppi, Ian Roberts, Katherine Curtis, Helen Roberts, Laura Ginnelly, Mark Sculpher, and Angela Wade. Prevalence of working smoke alarms in local authority inner city housing: randomised controlled trial. *Bmj*, 325(7371):998, 2002.
[46] Zach Shelby, Klaus Hartke, and Carsten Bormann. Constrained application protocol (CoAP). Technical report, Internet Engineering Task Force, 2013.
[47] *Arduino Ethernet Shield with PoE*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[48] *Arduino Ethernet with PoE*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[49] *POEthernet Shield*. Spark Fun Electronics Inc., Longbow Drive, Boulder, Colorado, USA, 2013.
[50] *STM32F05xxx/06xxx advanced ARM-based 32-bit MCUs, Reference Manual*, May 2013.
[51] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, M Eisenhauer, et al. *Cluster SRA 2011*, chapter Internet of Things strategic research roadmap, pages 9–52. European Research Cluster on the Internet of Things, 2011.
[52] Zhong Lin Wang. Toward self-powered sensor networks. *Nano Today*, 5(6):512–514, 2010.
[53] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.
[54] Geng Wu, Shilpa Talwar, Kerstin Johnsson, Nageen Himayat, and Kevin D Johnson. M2m: From mobile to embedded internet. *Communications Magazine, IEEE*, 49(4):36–43, 2011.

Edited by: Ethiopia Nigussie, Andreas Riener and Liang Guang
Received: Sep 1, 2013
Accepted: Sep 30, 2013