International Journal of Engineering Education Vol. 34, No. 1, pp. 263–272, 2018
Printed in Great Britain

0949-149X/91 $3.00+0.00

IoT Rapid Prototyping Laboratory Setup*

KIMMO KARVINEN
Department of Automation and Systems Technology, School of Electrical Engineering, Aalto University, Maarintie 8, Espoo, Finland. ORCID 0000-0003-3946-4163. E-mail: kimmo.karvinen@iki.fi

TERO KARVINEN
Westminster Business School, the University of Westminster, 35 Marylebone Road, London NW1 5LS, United Kingdom. E-mail: tero.karvinen@iki.fi

Even novice engineers and non-technical students can design and build an Internet of Things (IoT) prototype in four days. We present a setup for rapid IoT prototyping in a classroom, identify necessary skills and combine these to a workshop that allows students to turn their ideas into prototypes. Our approach enables fast prototyping cycle, using a common and well-established development board and a computer. Arduino Uno is used for device prototyping and a Python program running on the same computer handles the needed Internet communications. A web server handles device and web client connectivity, logging and data monitoring. The method makes it possible to learn the needed basic skills in manageable steps, allowing students to focus on the actual prototype instead of struggling with the wireless and Internet communication problems. After the prototyping phase, the device can be ported to inexpensive and small ESP8266 based microcontroller. Compared to developing IoT prototypes directly with ESP8266, the setup presented is considerably faster. The whole process is based on free software tools which provides a possibility to utilize prototypes commercially, without a risk of a third party changing or discontinuing services. We arranged an experimental four-day workshop for university students (n = 19) from diverse backgrounds and varying levels of technical skills. All teams successfully built a working prototype based on their own ideas. Student self-assessment of programming skills was noticeably improved during the course. We later repeated the experiment with another group (n = 27) in a university of applied sciences, getting similar results. Our results indicate that this method is effective for learning IoT prototyping skills in a short time.

Keywords: embedded systems; ESP8266; free software; Internet of Things (IoT); rapid prototyping; novice engineering education

1. Introduction

The Internet of Things (IoT) is expected to change the way devices are connected and embedded into environment. The influence of IoT will cover vast amount of different areas such as education, medical, transportation and countless others [1]. Cloud computing, existing networks and wireless networks already today provide technical foundation for IoT services and devices, but IoT must evolve to everyday objects and become an integral part of our environment to redeem its expectations [2]. This change is not in distant future. For example, in 2015 Gartner estimated that there will be 13.5 billion connected things in use in the consumer sector in 2020 [3].

Prototypes are often created to evaluate and test new ideas and designs. Prototypes can reveal design weaknesses, clarify technical requirements and give a chance to try out innovations to ensure they are fit for the designed purpose. Prototyping is usually an iterative process where the prototype is reviewed and modified repeatedly. Rapid prototyping is a process where a trial version of the design is created in an early stage of the development life cycle. [4] It allows a ''test drive'' of an idea with an acceptable possibility to radically change the approach or even throw away infeasible design. Rapid prototyping is also suitable for teaching and learning the use of new technology in a meaningful context as presented in this paper.

The use of prototypes [5] and related approaches such as problem, design and project oriented approaches [6] have been used in engineering education. Based on student self-evaluation surveys, these approaches have been successful [5, 6]. In the workshop presented in this paper, the results from student self-evaluations are supported by the functional prototypes successfully built in each student group.

In this paper, we define Internet of Things (IoT) as embedded systems that communicate over the Internet. Multiple definitions of IoT can be found in literature, showing interest in this area [7]. Some definitions and visions of IoT emphasize the network aspect, others concentrate on the objects [7].

Our focus is on a setup that allows students to turn their own IoT innovations into working prototypes. This setup can be taught in a four-day-workshop. Learning IoT prototyping is divided into three manageable sections, allowing building IoT devices without need for server programming. Setup utilized enables fast prototyping cycle using wired environment combining desktop computer to a hobby development platform. Fast and beginner friendly workflow is suitable for rapid prototyping

* Accepted 11 October 2017.

and makes novice and interdisciplinary IoT courses possible. The prototype can be easily developed to the next stage by turning it into a wireless, small and battery operated ESP8266 device. Process of prototyping in wired, steady environment without wireless and Internet communication problems provides an effective way for developing programs for ESP8266. Code created will work directly with ESP8266 microprocessor just by replacing serial read/write functions with WLAN functions. The whole process is based on free software and open source tools which makes it suitable for further development and commercial projects.

This paper makes three key contributions:

1. Approach that enables novice students to turn their ideas into working IoT prototypes during a four-day-workshop. For results of case workshop, see chapter 5.
2. Defining necessary basic skills for learning and prototyping IoT. For summary of skills, see Fig. 2.
3. Open source setup for IoT prototyping utilizing a development board and a computer.

Findings can be applied with students ranging from novices to advanced students. With novices, the focus should be on learning embedded systems and building a simple IoT device, while advanced students can benefit from effective workflow and short feedback loop.

Presented method was piloted in four-day-workshop in the University of Lapland with 19 art students. In addition to the case workshop, the experiment was repeated with another group in Haaga-Helia University of Applied Sciences.

2. Teaching IoT embedded system prototyping

IoT devices are embedded systems with input, data processing and output. In this regard, IoT device is like any other embedded system [8]. Unlike traditional embedded systems, IoT devices are not limited to operate according to input from their sensors and interacting with their outputs. By using existing Internet standards, IoT devices can communicate with each other around the world and use cloud services to collect and analyze data as shown in Fig. 1 [2]. Without sensors for input and output components, such as actuators, traditional embedded system would be of limited use. However, IoT devices can be useful if either input or output is connected to cloud. This does not exclude local sensors and outputs, but, by definition, an embedded system is not an IoT device without the ability to input or output to the cloud, over the Internet.

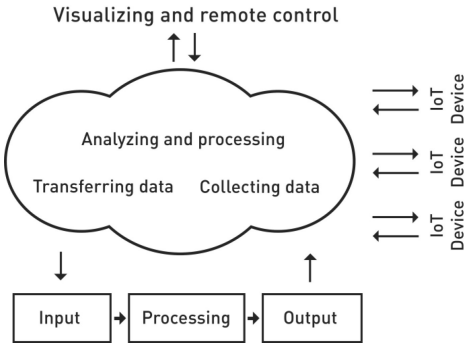

Fig. 1. Structure of an IoT device.

We present that teaching an IoT device structure can be divided into embedded system, embedded system and cloud communication and cloud deployment and development. This structure is presented in Fig. 2. As most students in our case example workshop had limited knowledge about embedded systems, the main part of the first day was dedicated to learning the basics. Using a hobby development platform and focusing only on crucial skills, the very basic embedded system skills can be taught in a one-day-workshop, even for students with limited technical knowledge [9]. The agenda of the workshop is presented in Table 1.

The most central basic skills needed to build a basic embedded system and to create a foundation for understanding IoT devices are shown in Fig. 2.

Table 1. Workshop agenda

Day 1 Technology, theory and personal innovation	**Day 2** Utilizing and publishing results	**Day 3** Debugging and Cloud services	**Day 4** Demoday
• Preliminary knowledge review • Introduction to IoT • Designing embedded system • Designing IoT device • Successful prototyping process • Group project plan • Presenting project plan	• Serious results with easy accessibility tools • Documenting projects • Publishing • Building the group project	• Common problems and solutions • For advanced students: cloud service deployment and operation • Building the group project	• Finalizing the prototype • Building the group project • Group project presentations • Feedback

Students need to be able to use an IDE (integrated development environment) and compile programs for a development platform. Arduino Uno was selected for a development platform for its popularity and broad user base. The low cost and open hardware have made it a frequent choice for numerous research and scientific projects [10]. Arduino is also used for engineering education because of its accessibility, adaptability and compatibility [10–13]. Arduino is widely available from different producers and resellers. In 2013, Arduino had registered over 700,000 official boards [14]. As both software and hardware of Arduino are open-source, there is also an unknown number of non-official boards by various manufacturers in sale.

Input, processing and output can be clarified by combining sensors and output components with code examples [9]. Deeper understanding of programming is not a necessity at this point but in order to utilize code examples and Internet resources, students need to be able to understand the basic program structure and syntax. Utilizing Arduino IDE's built-in serial monitor is necessary for debugging as well as for understanding serial communication in the next phase.

When the concept of input, processing and output is clear, moving from an embedded system to an IoT device with cloud communication is simple. Writing to cloud is similar to using any output, but instead of outputting data to Arduino's pins, a value is written to the serial port. A proxy program then transfers data to the server. Getting input from the cloud works the same way. Instead of using measurement from a sensor, data is received from cloud by writing request to the serial port. The technical details of the process are explained in the next section.

The economic principle of prototyping proposed by Lim et al states that: ''The best prototype is the one that, in the simplest and the most efficient way, makes the possibilities and limitations of a design idea visible and measurable'' [15]. In the same spirit, we aim to leave everything that is not absolutely necessary out of the learning and prototyping process. Server programming is not compulsory for successful IoT prototyping. However, understanding its operating principles is necessary in order to understand how the communication works between the device and the cloud. By limiting the internet communication to passing floating point numbers, we were able to develop a simple and generic prototype for the server backend. The open source server solution makes it possible to deploy cloud software to any server or to develop its features further.

The prerequisites for using the prototyping environment in desktop computers are quite moderate.

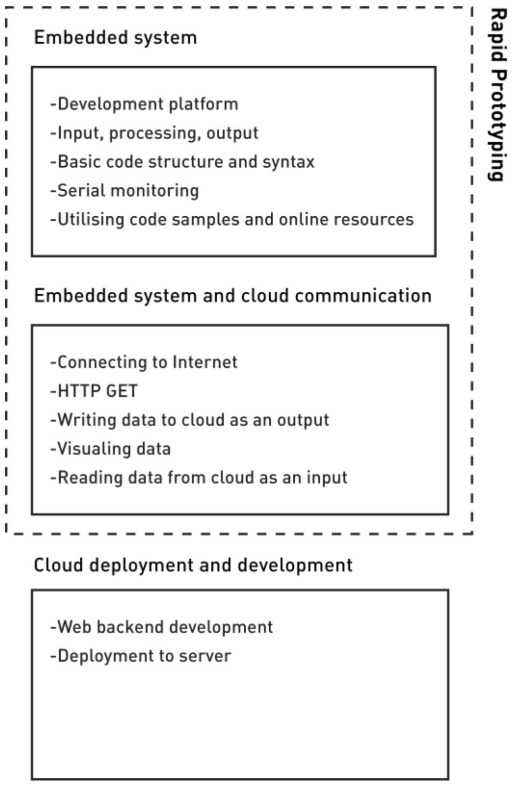

Fig. 2. IoT prototype building basics.

Each student computer equipped with Arduino Uno needs to have Python 3 and Arduino IDE installed. Needed software is free and open source. In addition, a suitable selection of sensors, actuators, LEDs, breadboards and jumper wires is needed. The component assortment should be diverse enough to allow students to realize their innovations rather than building a device based on available parts. In the case workshop, students were encouraged to first define what kind of IoT device they would like to build, and then guided on component selection and realistic implementation. During the workshop, students successfully used three different operating systems: Linux, Windows and OS X. The diversity of operating systems was mandated by the diversity of university computers and devices brought by the students.

3. Setup for prototyping

Embedded system is built on the Arduino platform, a popular open source, free software hobby and rapid prototyping platform. Our implementation consisted of Arduino Uno development board, the accompanying Arduino integrated development environment and libraries. Components chosen for implementation are listed in Table 2.

All student developed code runs in the embedded system, Arduino. Arduino does not have an operat-

Table 2. Components chosen for implementation

Hardware	Arduino Uno	Desktop computer	Virtual Private Server
OS	No OS	Linux/Windows/Mac	Linux
Software	Student's code	BotBookEspy	http://one.api.botbook.com, Apache, Flask, PostgreSQL
Language	C++	Python 3	Python 3, SQL

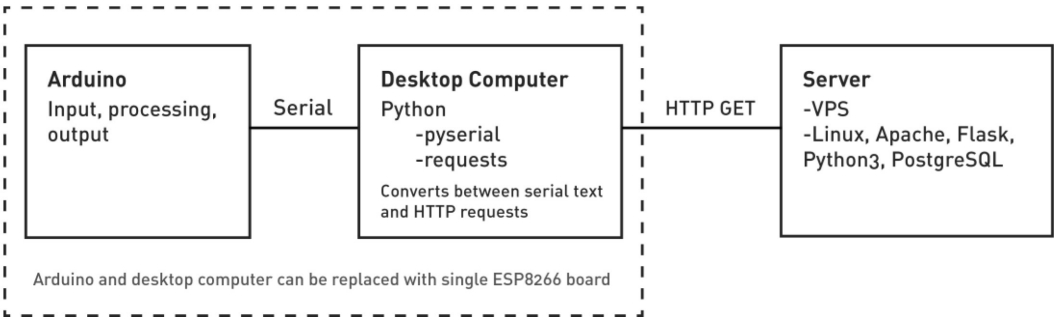

Fig. 3. Setup for prototyping.

ing system, so the only thing running is the code written by the student. Being able to see all of Arduino's code on a single screenful makes it easy for student to feel in control of the system.

Setup for prototyping is presented in Fig. 3. The prototype is built with Arduino Uno which is connected to a desktop computer running Arduino IDE. The only cable needed is the USB cable connecting Arduino to a computer, while all data is written to serial port and read from the serial port by a Python program running on the computer. By limiting the number of physical components and wires needed for the initial setup we were able to reduce points of possible user error.

Arduino can send two kinds of GET requests: send value or ask for value. All requests include an API key, a secret string used in place of login name and password. BotBookEspy, a Python serial proxy, forwards requests from the Arduino to the server. Asking for data is identical to asking for a single web page from the server. URL includes a value and API key which is then parsed by the Python proxy, returning only the value to Arduino. From Arduino programs' point of view, there is a little difference in this process from using a value read from a sensor or outputting a value to a component.

Performing all requests, both sending and receiving data, using GET requests with the device as the client is mandated by practical network architectures. Typically, both wireless and wired networks are protected by a firewall. In many IPv4 networks, the network also uses network address translation (NAT). The conflict of low number of available IPv4 addresses and the increasing need for new addresses required by IoT systems have raised worries about running short of addresses and calls for faster IPv6 adoption [7]. However, making the IoT nodes clients in the client-server architecture can work behind NAT on existing IPv4 networks without any modifications to the infrastructure. In both cases, it is impossible to open a listening port visible to the Internet, and thus, impossible to create servers. A server in the Internet can facilitate communication both from devices to the server and between the devices. This approach also reduces attack surface, as the IoT devices do not need to answer arbitrary connections from the Internet.

The same backend server that interconnects the devices can work as a backend for any web page. A static web page can use JavaScript to perform AJAX (asynchronous JavaScript and XML) requests to the server. For example, a group of students created a service where shirt sleeves could be controlled by clicking a button on a web page. For mobile experience, this web page could be viewed on a cell phone. If an app is required, this could be further packaged with a web based app making system such as Cordova or PhoneGap. An app packaged this way can be distributed in popular cell phone app markets, such as Apple App Store and Google Play.

3.1 Student development process

During the initial prototyping, students do all their development on Arduino. Using a single platform for development reduces context switching between languages and environments, and limits the number of technologies initially taught to the student. Prototype development is similar to any embedded development on the Arduino platform. The student codes in a simple and limited subset of C++ using the Arduino IDE. Code is uploaded with a single click.

To be an IoT device, by definition the thing must

communicate over the Internet. Arduino controls the communication by sending an HTTP GET URL to BotBookEspy, the Python proxy.

For example, the Arduino program written by the student might want to store current temperature, 3.5 degrees centigrade, to the server. The Arduino program would use the built-in serial functions to print the string "GET http://one.api.botbook.com/add/h9aIcPe7p8Y/?x=3.5" to computer over serial. BotbookEspy, the Python proxy running on the desktop computer, would then perform the actual HTTP GET request to the server one.api.botbook.com. If the API key is correct, this would store the data point [current time; 3.5] to the database. Current time is automatically added by the server, which avoids running real-time clock (RTC) on the embedded devices and guarantees single source of truth for time and time zones. When the server (one.api.botbook.com) answers, the first line of the answer body is returned to the Arduino program by the BotBookEspy Python proxy. As the student program only prints one line of text, this complexity is effectively hidden from him/her at this point.

Reading data is done using a similar GET request and the chain of events is similar to saving data to the server. Because the BotBookEspy proxy only returns the first line of HTTP response body, only minimal parsing is required to extract the floating point value, which is then explicitly cast to float for further processing. To let students ignore the implementation details at first, we provided serialHttpGetFloat() function to perform these steps.

The serial-HTTP proxy BotBookEspy is provided for the student. It requires no configuration or modification by the student. The proxy performs HTTP GET requests according to the instructions from the serial. Simple debugging is another feature of the proxy. All requests, both failed and successful, are shown on student computer. This allows students to catch common problems, such as typos, incorrect API keys and other trivial errors. When problems occur, it also helps splitting the problem domain at a central point.

Encapsulating Internet access to separate functions also works as preparation for optional later miniaturization. When the program works with the quickly iterable, reliable and wired initial prototyping setup, the student can simply replace the contents of these functions to allow standalone operation without the BotBookEspy Python proxy and without a computer.

This prototyping system allows student to develop his software on a single, simple platform while performing real requests to a real server. Supporting parts can later be replaced when the system is more mature and needs to be packaged and miniaturized.

3.2 Server

To engage student creativity and imagination, the prototyping setup here allows students to work on aspects that are special and unique in their setup. In contrast, the prototyping platform hides much of the boilerplate code and other work that is similar between many projects. One of the parts similar between projects is the server backend.

On one hand, students want to get started quickly and concentrate on the unique parts. On the other hand, there needs to be a clear and scalable path for reaching clients, obtaining customers and publishing the end results. To meet these criteria, we wrote a simple backend software on top of a free stack. The licenses of our software and the required stack all meet the criteria for free software as defined by the Free Software Foundation [16].

The tasks fulfilled by the server are data logging, embedded device connectivity, web client connectivity and data monitoring. Data logging, embedded device connectivity and web client connectivity are performed using an HTTP GET API. Web clients, such as mobile apps and static web pages on other servers, require special security features such as Cross-Origin Resource Sharing (CORS), which were implemented in our server and used by students in the pilot workshop. Data monitoring is created using a regular JavaScript application on a web page, using the same APIs as other clients. User authentication is implemented as a server side feature.

Learning a new API can be a significant challenge for programmers on any level, resulting in wasted effort [17]. To make APIs easier to learn, emphasis should be placed on API usability [17]. To make our API stable (not requiring frequent changes) and simple to learn for our students, we decided to leverage familiar web technologies and use only three types of GET requests.

The GET API only provides three functions: add, last and json as shown in Table 3. The two API keys for each user are created at registration time, and they are shown on user page and on the two example pages. View API key cannot modify data, so it can be used in public web pages, cell phone apps and digital signage in public areas. Two different view functions are required by different clients: full browsers often want to see many data points, for example to draw a graph. To facilitate a common

Table 3. API endpoints provided by BotBookAPI v 1.0.0

Path	Feature
/add/<addkey>/?x=	Store a data point [currentTime, x]
/last/<viewkey>	Return x value for last data point
/json/<viewkey>/index.json	Return all stored data points

use case of one device reacting to an event from another, geographically distant device, we implemented the last endpoint. Even though the same data could be obtained by parsing the output of json endpoint, the last endpoint reduces the strain of parsing on the limited embedded environments and also makes it much simpler for students to develop common use cases.

In addition to GET API, the BotBookAPI web application provides login related functions (register with invite, login, user page, logout) and API usage examples (send.html and getlast.html). On the user page, user can retrieve his API keys (viewkey and addkey) and monitor the latest data points. Monitoring the data points allows instant results for beginners and helps debugging by splitting the problem space.

Students can use any web browser with JavaScript support to retrieve the data. For example, students could embed data points on their static homepage. To allow the data to be used this way, the BotBookAPI server application injects Cross-Origin Resource Sharing (CORS) headers to /json/ responses. Namely, it sends Access-Control-Allow-Origin * and Content-Type application/json. This feature was used by many projects, for example controlling shirt sleeves over the Internet with mobile phone, and letting social media site Facebook users control LED lights.

The students using these features do not need to be familiar with the implementation details, such as CORS. Instead, they could use their web development skills (HTML, CSS and JavaScript) in combination with modifying the given examples.

In the pilot workshop, the server was provided to students. To allow advanced students to create their own solutions, we provide the BotBookAPI server code as an example [18]. As it runs on a completely free stack (Linux, Apache, PostgreSQL, mod_wsgi, Python 3, Flask), students are able to build a whole IoT solution they control. In the workshop, one advanced student implemented a similar backend from scratch using similar architecture and LAMP (Linux, Apache, MySQL, PHP) stack. As most students in the pilot workshop started with no programming skills at all, it was convenient that the server side could be used as an extra task for the advanced student. This also shows that it's possible for students to obtain the tools in addition to the skills.

Modern web frameworks combine multiple components, all of which must work together to provide a scalable and secure service. To standardize this configuration, our example BotBookAPI server configuration was described in Puppet. Puppet is a modern (idempotent, infrastructure-as-code) configuration management system. Idempotence means the goal, the target state of the system, is described. Puppet only does modifications to system state to fix any deviations. Infrastructure-as-code approach means using versionable plain text for this configuration. Using Puppet makes it much easier for an administrator to install the system, but it does not eliminate the need for basic administration skills. In the workshop, only one student installed his own backend, probably due to the strict time limit.

As noted above, the whole environment is based on free software. While the server end could be built relying on third party services, free software solution has some significant advantages. IoT's future ubiquity combined with efficient storing and accessing data raises serious privacy concerns [7]. Inference techniques and access to large amounts of data makes anonymization more unreliable. The need for accepting several privacy policies makes it even more complicated for users to prevent unwanted data usage. [19] Open source server code could solve some of these issues as it enables development of services that are not entirely controlled by third party commercial operators.

4. Miniaturization with ESP8266

The ESP8266 is a low-cost Wi-Fi capable microprocessor, designed by Espressif systems and there are breakout boards available with various features and layouts based on ESP8266 [20]. With small size, features and the low price combined, ESP could be potential platform for IoT prototypes and products. At the time of writing, an inexpensive ESP8266-12E development board with pin headers connected and USB cable included sold for 7 EUR for one-unit orders, making it an affordable IoT prototyping platform [21].

It is common to add external components to Arduino to enable wireless connectivity. For example, Sarkar and Das [22] mention Arduino Wi-Fi Shield but end up using an ESP8266 as an additional component to Arduino. Adding external components increases complexity, size and price. It could also add to power consumption and thus further increase size for battery powered devices. The improvements of ESP8266 family chips and related devboards have made it feasible to miniaturize the whole system into an ESP8266 board and completely drop the requirement for an Arduino board.

ESP8266 is a family of chips, including barebones ESP8266-01 (ESP-01) and more featureful ESP8266-12E. Many development boards have been published for each chip by different third-party manufacturers. Typically, the smallest ESP-01 requires many support components, provides many low-level challenges and is slow to work

with. Some ESP8266-12E devboards are simple to use, as they only require a USB cable and Arduino IDE for programming.

We tested ESP-12E and ESP-01 to find out how fast code uploads would they enable. For testing, we used 9 lines of code. Using Arduino IDE with ESP-12E, the upload took approximately 30 seconds including time to compile the program.

Uploading code to ESP-01 using Arduino IDE requires using a FTDI Serial TTL USB Cable and 3.3V power source. 3.3V connected to ESP-01 VCC has to be disconnected and reconnected each time before uploading. In our tests, loading the code to ESP-01 took approximately 34 seconds. This includes the compiling time but excludes disconnecting and connecting the needed wire, and user errors with the wires. Using Arduino Uno with the same code example, the upload time was under 4 seconds. Even though the difference might seem insignificant, the time adds up quickly as during a normal coding process, the program is uploaded repeatedly. Also, user errors related to reconnecting wires are greatly reduced when working with Arduino boards.

Arduino and desktop computer setup presented above can be replaced with a single ESP8266-12E. The only code change needed is replacing serial read/write functions with WLAN functions and adding WLAN initialization function. This way the prototype can be ported to an inexpensive and small package while still having the advantages of the fast and reliable prototyping phase in wired environment. As the three WLAN functions are the same for any program using the proposed setup, we can provide ready-made functions for students.

ESP (and by extension, typical ESP devboards) use 3.3 volt logic levels. In comparison, Arduino uses 5 volt logic, a voltage that would break ESP input pins. When using ESP devboards providing 5 V output, often the only change needed is a level converter or a voltage divider on ESP input pins. For ESP output, 3.3 V is high enough to consistently register as high in many 5 V sensors and actuators.

5. Assessment

In our case workshop, all student groups were able to design and build a working prototype based on their own ideas. Final prototypes are listed on Table 4.

Workshop had a range of art students from different departments, such as art education, industrial design and interior & textile design. During the workshop, the most prominent issue was lack of technical skills needed for successful prototyping.

Table 4. University of Lapland final prototypes—4 day case workshop

Project	Description
Far away candle	An imitation RGB-LED candle that lights up when a flame sensor detects real candle being lit up in a different location. Concept is that you can give an artificial candle to someone close to you and light it up even when they are on the other side of the world.
Baiting box	A box that offers a candy bar when someone comes near, but pulls it away when a person approaches. Movement and distance are measured with an ultrasonic distance sensor. Servo motor moves candy bar in and out. Cloud service updates and presents the number of people tricked on a website.
Henhouse curtain	Curtains that can be controlled with webpage interface. Opening and closing can also be timed by cell phone calendar events by loading HTTP API. The curtain mechanism is built with continuous rotation servo moving the blinds along the rail and microswitches to detect when the edges are near. The project was used as a prototype for one student's urban henhouse.
Busy owl	A felt owl that reacts with movement and color to the number of Bluetooth devices present in another space. The owl's animatronics are constructed with servo motors and color changing feature is made by combining RGB-LED to optical fibers. Bluetooth devices are detected by using an external Bluetooth module. The number is shown on a web page with animated presentation of the owl. Professional looking felt owl was also designed and built from the scratch during the workshop. The group implemented their own LAMP (Linux Apache MySQL PHP) HTTP backend to similar to BotBook HTTP API.
Pattern riddle	A box with large buttons and a riddle to help to find the right pattern. Correct and incorrect attempts are logged and shown on a website.
Care about you	A light installation made with LEDs that is lit when someone clicks sharable "care about you button" on a website. The link was distributed on the social media during the workshop and received a large number of clicks from people who were not course participants.
Secret knock	A webpage that reacts on people knocking on a table. Right knocking pattern opens the next page.
Sleeves of distant thoughts	A shirt with sleeves that can be raised and lowered with a cell phone user interface. Sleeve length is adjusted with a continuous rotation servo and a reel attached to the servo horn. Successful combination of fabric and garment design was a large part of this project. The shirt that was fabricated during the course, was both functional and professional looking.
Dancing garden	Miniature garden of trees creating shadows on the background. Trees attached to servo motors can be rotated to desired positions via website. Concept is that trees can be adopted by several people, creating a moving social garden installation.

As the student group was diverse, the base knowledge varied from students with experience in programming and building embedded systems to students that had near zero knowledge in those areas. This issue could be alleviated by arranging a basic embedded system course prior to IoT workshop. In the preliminary knowledge review, over half of the students evaluated their programming skills with the lowest available number 1/5 while the average rating was 1.5 (n = 19). In the feedback, the average was raised to 2.5/5 and no one selected the lowest rating.

Reception for technical subject presented for mainly non-technical audience was positive. In the feedback questionnaire of the case workshop, students scored the course 4.3/5 on average. Every participant would also recommend the course to other students.

The workshop was tested again with a different audience but similar results. A diverse group of ICT students (n = 27) in Haaga-Helia University of Applied Sciences, Helsinki, participated the course in spring 2017. During the five-day workshop, the groups created, prototyped and published a total of 13 projects. All ideas from the projects were from the students themselves. All groups successfully built a working prototype during the workshop and published a documentation of their project on the Internet [23]. Final prototypes are listed on Table 5.

All projects use HTTP APIs, most of them used the BotBook HTTP API described in this paper. In amusement park queue estimator, students implemented their own HTTP backend based on the same concept, giving indication that this method and API is easy to understand. One project read its data from a public HTTP API of a third-party weather service, and two sent email using third party HTTP API. Internet web interface was obviously important for projects that needed to be controlled from great distance, but in some cases web also served as a very convenient cell phone remote control for devices in the same room. These examples indicate that the concept of the HTTP API is easy to understand on a

Table 5. Haaga-Helia University of Applied Sciences final prototypes—5-day workshop

Project	Description
Internet of window blinds	A curtain that can be controlled with a cell phone or any other web browser. Full size window blinds controlled by two servos.
GSR lie detector	Stress level is estimated from galvanic skin resistance measured with capacitor load time. Stress level is relayed to web for display on a video projector or viewing as additional information during a video call.
Amusement park queue estimator	Shows real time map of queues on a web page. Queueing visitors are detected from RFID badges. Custom HTTP backend with JavaScript Node.js using the Express framework, frontend with HTML and AJAX (Asynchronous JavaScript and XML).
Room fill counter	Estimates the number of persons in the room. By-passers and their direction (in or out) is measured with two active infrared sensors. This count is sent to HTTP backend. A web page reads this count and draws a graph of total persons in the room.
Weather lamp	Lamp changes color according to temperature in chosen city. Temperature is read from a public API of an external public service.
Medicine alarm	Show on the web if medicines are taken, so caretakers can monitor the situation remotely and non-intrusively. If a pressure (weight) sensor has not been disturbed, a very simple web page indicates forgotten pills by changing a number.
Cool down gamer	A fan that can be activated using the web, to be used as a gag in online games. A relay and an additional power source was used for running a 12 V fan.
Two-way door monitor	Alarm device that detects entries but ignores exits, with web remote control. It uses two microswitches to monitor a typical Finnish apartment building double door that has inner and outer door near each other. A box for Arduino was 3D printed using a third-party model.
Tomato watchman	Email user if tomato living conditions are out of pre-defined limits. Measured conditions are temperature, light, air humidity and soil humidity. A Python program checks conditions once an hour using Linux cron, then sends email using HTTP API to third party email service.
Body heat for health	Body heat is measured and graphed on a web page. Prototype required a wire, but initial attempts for porting to ESP8266-12E were made for miniaturization and wireless operation.
Guitar humidity measurement	Conditions inside guitar box are measured, wirelessly transmitted to backend using WIFI and graphed on the web. These measurements help guarding expensive instruments when they are borrowed or stored. Successful miniaturization was achieved using ESP8266-12E. A business model based on this device was being evaluated after the course.
Lock using NFC debit cards, travel cards and student cards	A lock that can be opened a card already carried by the user. Optionally, a verification email could be sent with a link click required for opening the lock.
You hit like a programmer	A heavy punching bag in public area measured the hit strength with an accelerometer and showed the result on the web.

level where students can apply it to new situations and services.

Background varied from novices (0 credits) to final-year students. One student initially visited the workshop but did not start the project and did not provide feedback, and thus is not included in the population n. Course feedback was excellent, with mean 4.8/5, mode and median 5/5. The results of this workshop are in line with the experiences of the workshop we described earlier. The results give initial indication that the proposed method is suitable for audiences with diverse backgrounds and different levels of technical expertise.

6. Discussion

Using Arduino with a desktop computer and a Python program as a proxy was confirmed to be a suitable way for rapid prototyping IoT devices. Wired environment combined reliable prototyping with fast development cycle.

The phase when students design their projects and choose the features that will be implemented is crucial. Without guidance, there is a risk that they will start a project that is impossible to carry through in a given time, or make a too simple safe plan with not much space for innovation.

The option to use ESP8266 to turn prototype into small autonomous package was not carried out by any group in the case workshop. There was a prebuilt small battery operated ESP8266 device used as an example of how the whole device can be ported to a small, wireless and inexpensive package. It is important that students are not left with an impression that IoT devices are computers combined with embedded systems, but the setup is used for effective prototyping purposes only.

While some students lacked technical skills, the case class excelled in the tasks where innovation was needed. When asked to make a plan of an IoT-device and present it to everyone, students instantly started forming surprisingly unique ideas. Several groups also managed to combine their existing expertise to embedded system design. Comparing the art students' case workshop to the second one with ICT students shows how educational background effects on the results. The first group put a lot of effort to make devices aesthetically pleasing while the second group developed more advanced technical solutions.

Arduino is sometimes seen as a tool for hobbyists which skips many low-level challenges. Easier process should not be overlooked by engineers and experts as it supports effective and more creative prototyping process [24]. While being accessible, Arduino can be utilized for serious projects and advanced applications [25].

7. Conclusions

Basic IoT prototyping skills that enable students to design and build simple devices based on their own innovation can be taught in a four-day-workshop. Prior embedded system experience is not compulsory, but lack of it calls for smaller workshop size and more personal support from a teacher or teachers for each student.

The setup and the architecture used were suitable for teaching IoT and building rapid prototypes. As there was no need for server programming, students could focus on their embedded system development. After students understood the concept of input, processing and output, it was simple to move from developing embedded systems to IoT devices. The text output of the Python proxy program and Arduino IDE's serial monitor helped debugging and developing the prototype considerably. Using GET to send and receive data showed out to be easy to understand for students and bypassed many network limitations.

Utilized prototyping cycle and setup is considerably faster for developing programs for ESP8266 than uploading code directly to it. The upload time from Arduino IDE to ESP-12E was 750% of the upload time to Arduino. Devices can be first developed and tested with a computer combined with Arduino, and then easily ported to ESP. Case workshop's timeframe did not allow students to use ESPs, leaving that part of the process to be tested on future courses. Longer courses would also allow prototyping more complex and finalized IoT devices than the ones built during the case workshop.

Acknowledgements—The authors wish to express their thanks to Ville Kyrki, Tomi Knuutila, University of Lapland, Minna Kivihalme and Haaga-Helia University of Applied Sciences.

References

1. S. Madakam, R. Ramaswamy and S. Tripathi, Internet of Things (IoT): A literature review, *Journal of Computer and Communications*, **3**(5), 2015, pp. 164–173.
2. J. Gubbi, R. Buyya, S. Marusic and M. Palaniswamia, Internet of Things (IoT): A vision, architectural elements, and future directions, *Future Generation Computer Systems*, **29**(7), 2013, pp. 1645–1660.
3. Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015 [Online]. Available: http://www.gartner.com/newsroom/id/3165317. Accessed 18 May 2017.
4. V. S. Gordon and J. M. Bieman, Rapid prototyping: lessons learned, *IEEE Software*, **12**(1), 1995, pp. 85–95.
5. S. Chandrasekaran, A. Stojcevski, G. Littlefair and M. Joordens, Project-oriented design-based learning: aligning students' views with industry needs, *International Journal of Engineering Education*, **29**(5), 2013, pp. 1109–1118.
6. A. M. Agogino, S. L. Beckman, C. Castaños, J. Kramer, C. Roschuni and M. Yang, Design Practitioners' Perspectives on Methods for Ideation and Prototyping, *International Journal of Engineering Education*, **32**(3), 2016, pp. 1428–1437.

7. L. Atzori, A. Iera and G. Morabito, The internet of things: A survey, *Computer networks*, **54**(15), 2010, pp. 2787–2805.
8. P. Laplante and S. Ovaska, *Real Time System Design and Analysis*, Wiley-IEEE Press, USA, 2011, pp. 3–4.
9. K. Karvinen, Teaching robot rapid prototyping for non-engineers—a minimalistic approach, *World Transactions on Engineering and Technology Education*, **14**(3), 2016, pp 341–346.
10. A. Soriano, L. Marin, M. Valles, A. Valera and P. Albertos, Low Cost Platform for Automatic Control Education Based on Open Hardware, *IFAC Proceedings Volumes*, **47**(3), 2014, pp 9044–9050.
11. A. Araujo, D. Portugal, M. S. Couceiro and R. P. Rocha, Integrating Arduino-based educational mobile robots in ROS, *Journal of Intelligent & Robotic Systems*, **77**(2), 2015, pp. 281–298.
12. C. Parikh, Introducing Arduino Platform to Sophomore's using an apt recipe, *Proceedings of the 2014 ASEE North-Central Section Conference*, 2014 [Online]. Available: http://people.cst.cmich.edu/yelam1k/asee/proceedings/2014/Paper%20files/aseencs2014_submission_10.pdf. Accessed 7 September 2017.
13. P. Hertzog and A. Swart, Arduino—Enabling engineering students to obtain academic success in a design-based module, *Global Engineering Education Conference (EDUCON)*, Abu Dhabi, 11th–13th April 2016, IEEE.
14. MEDEA, Arduino FAQ—with David Cuartielles [Online]. Available: http://medea.mah.se/2013/04/arduino-faq/. Accessed 7 September 2017.
15. Y-K. Lim, E. Stolterman and J. Tenenberg, The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas, *ACM Transactions on Computer-Human Interaction (TOCHI)*, **15**(2), 2008, p. 7.
16. What is free software? [Online]. Available: https://www.gnu.org/philosophy/free-sw.html. Accessed 18 May 2017.
17. B. A. Myers and J. Stylos, Improving API usability, *Communications of the ACM*, **59**(6), 2016, pp. 62–69.
18. BotBookAPI server code v 1.04 [Online]. Available: http://iot.botbook.com/. Accessed 18 May 2017.
19. J. A. Stankovic, Research directions for the internet of things, *IEEE Internet of Things Journal*, **1**(1), 2014, pp. 3–9.
20. K. K. Patel, J. Patoliya and H. Patel, Low cost home automation with ESP8266 and lightweight protocol MQTT, *Transactions on Engineering and Sciences*, **3**(6), 2015, pp 14–19.
21. ESP8266 ESP-12E Serial Wi-Fi Development Board Module [Online]. Available: http://www.volumerate.com/product/esp8266-esp-12e-serial-wi-fi-development-board-module-w-built-in-antenna-micro-usb-cable-844404647. Accessed 18 May 2017.
22. T. Sarkar and N. Das, Exploring Web of Things with embedded devices, *Int. J. Advanced Networking and Applications*, **7**(3), 2015, pp. 2719–2723.
23. Internet of Things presentations today [Online]. Available: http://terokarvinen.com/2017/internet-of-things-presentations-today. Accessed 7 September 2017.
24. P. Jamieson, Arduino for teaching embedded systems. Are computer scientists and engineering educators missing the boat? *FECS*, USA, 12th–15th July 2010, pp. 1–6. Available: http://www.users.miamioh.edu/jamiespa/html_papers/fecs_11.pdf. Accessed 18 May 2017.
25. K. Karvinen, T. Tikka and J. Praks, Using hobby prototyping boards and commercial-off-the-shelf (COTS) components for developing low-cost, fast-delivery satellite subsystems, *Journal of Small Satellites*, **4**(1), 2015, pp. 301–314.

Kimmo Karvinen is a Master of Art, from Media Lab at University of Art and Design Helsinki. He is currently working towards his D.Sc. at Department of Automation and Systems Technology, School of Electrical Engineering, Aalto University. In addition to his focus on scientific research, Kimmo has co-authored five books about embedded systems, translated to 12 languages. He is currently working as a CBO in a hardware manufacturer that specializes in IoT technology. Before that he worked as a CEO in a leading company specializing in AV automation in Finland.

Tero Karvinen is a Master of Science in Economics from Helsinki School of Economics and Business Administration. He is currently working towards his Ph.D. at Westminster Business School, the University of Westminster. Tero has co-authored five books about embedded systems, translated to 12 languages. He is currently teaching Linux and embedded systems in Haaga-Helia University of Applied Sciences, where his work has also included curriculum development and research in wireless networking.