6-8-2015

A Framework Architecture for Student Learning in Distributed Embedded Systems

William L. Honig

Konstantin Läufer

George K. Thiruvathukal

A Framework Architecture for Student Learning in Distributed Embedded Systems

William L. Honig, Konstantin Läufer, George K. Thiruvathukal
Department of Computer Science
Loyola University Chicago
Chicago, Illinois, United States of America
whonig@luc.edu, laufer@cs.luc.edu, gkt@cs.luc.edu

***Abstract*—Academic courses focused on individual microcomputers or client/server applications are no longer sufficient for students to develop knowledge in embedded systems. Current and near-term industrial systems employ multiple interacting components and new network and security approaches; hence, academic preparation requires teaching students to develop realistic projects comparable to these real-world products. However, the complexity, breadth, and technical variations of these real-world products are difficult to reproduce in the classroom.**

This paper outlines preliminary work on a framework architecture suitable for academic teaching of modern embedded systems including the Internet of Things. It defines four layers, two of which are at the edges of the network, and not adequately covered in academia. For each layer of the architecture, specific technology and suitable devices are identified. Desired academic outcomes for courses using projects based on the architecture are identified. Feedback and comparison is sought on how effective student course and research activities based on the framework will be to real-world embedded systems developers.

Keywords—embedded systems education; distributed architecture; personal area networks; wearable devices; Internet of Things; Bloom's Taxonomy

I. INTRODUCTION

Computer science academic education needs to evolve to meet the needs of organizations producing real-world, industrial embedded systems [1]. Changes in computing technology that drive this development are described in Section II.

Section III presents a broad framework architecture to establish a common approach and set of technology solutions applicable to multiple courses. The architecture can support broad student learning without needing a different physical laboratory for each individual project; it spans a broad range from networked systems to local and wearable devices.

The framework architecture presented here is novel and an attempt to improve academic preparation of students for the embedded systems industry. It is intended to be more diverse in terms of hardware and devices than some other approaches [2] and to prepare students for development of applications for the Internet of Things. Section IV discusses preliminary assessment ideas for evaluating student outcomes across multiple courses and projects based on the architecture with Bloom's Taxonomy [3] and is an alternative to other models for evaluating learning [4, 5]. Section V briefly describes example projects using the architecture.

II. CHANGING NEEDS

For some time, academic preparation for computer science and engineering students has focused on individual embedded systems using microcontrollers [6, 7] and hardware prototypes at the level of individual stand-alone systems. More complex distributed systems were primarily studied through higher-level networked and distributed systems using client/server architectures, distributed databases, and the internet [8, 9]. Although still important, these approaches are no longer sufficient for students to gain skills in more sophisticated embedded systems being developed in industry today.

First, many real-world products include multiple interacting intelligent elements. These elements are often microcontrollers or custom hardware elements that need to interact to complete services or transactions in order to perform properly. To control product cost these elements may be quite different from each other with disparate timing, interface, and power needs.

Second, sometimes these elements are connected using network technologies (wired or wireless) that cover fairly small areas (at most a few meters using technologies such as Bluetooth). For cost, power, and possibly security reasons, these interconnections are often different than the usual internet and wide area networks used at higher levels.

These two new knowledge areas must be added to student learning to prepare students for successful careers. It is necessary for academic course work to address these areas in addition to traditional client/server, internet, and database system development. However, it is difficult to know how to adequately cover such topics in a typical academic course.

In fact, several courses may be required to address the range of computing, network, and security aspects of these modern distributed systems. Determining what theory, architectures, technologies, hardware, and software systems to use on a course-by-course basis is inefficient at best. Hence, a framework architecture seems a reasonable approach to ensure some consistency of approach across courses and to allow course, project, and research work to build upon each other.

978-1-4673-7711-9/15/$31.00 ©2015 IEEE

III. Framework Architecture and Key Technologies

In this section, we outline a broad framework architecture for academic use in preparing students for successful entry into embedded systems careers. The architecture is a work in progress; it is a physical implementation of the more general IoT-A framework [10]. Feedback is sought from industry and academia on its contents and how to best apply it to ensuring good learning outcomes for students.

The distributed embedded system framework architecture includes four main layers, of which two are newly identified to address the needs for multiple interacting intelligent devices and networks shown in section II. Fig. 1 introduces the four layers of the architecture and highlights the characteristics of each layer. Each layer is described here starting with the lowest or most specialized device layer.

A. Functional Device Layer 4

At the lowest level, the architecture provides devices that provide one or more functions to the user or other parts of the architecture. These functional devices interact with the real world using sensors and actuators, they measure and monitor physical values, and may act autonomously and in real time once initiated.

Such functional devices often have small size and weight, may be internally powered, and may or may not be aware of other devices in the same layer. However, they all typically expect either periodic, on-demand, or continuous communication with devices or systems at higher layers of the architecture. They may report data to these higher levels, or receive inputs and information to use.

Internal software for functional devices is often specialized, may be open or closed to inspection and modification, and usually requires stringent real time performance to properly measure, record, or respond to real world events and changes. Although some devices may have sophisticated and extensible software, others may be limited and highly optimized to reduce memory needs or power usage.

For academic use in hands on courses, "tinkering" is highly effective for student learning. At this layer some mechanism is required to allow interesting sensors or actuators to be connected to and controlled by software students create. Prototyping boards that allow general analog and digital connections allow this type of tinkering.

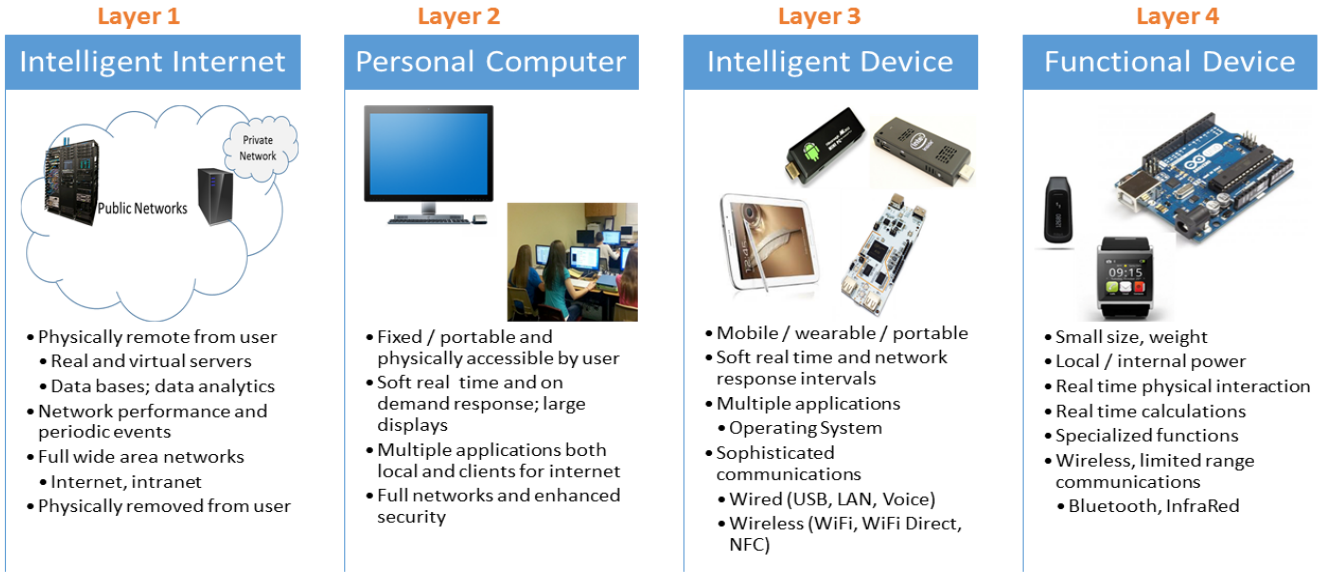
Functional devices are the focus of much ongoing innovation including wearable devices and the Internet of Things. One way to envision these devices is that they are often designed for "set and forget", once turned on, connected, and running, the user may forget about them (at least until the user wishes to use information they produce).

B. Intelligent Device Layer 3

The next layer includes larger, more visible, and more capable devices. Devices at the Intelligent Device level have more general purpose or expandable capabilities than those in Layer 4. Typical devices at this level include mobile phones, tablets, wearable computers, and other programmable computer devices; all typically have a separate operating system, are extensible by adding applications, and support various communication protocols for interacting with each other and across layers of the architecture.

Devices at this layer of the architecture tend to be larger and have a more obvious presence than those at Layer 4. They may be mobile or fixed (possibly mounted on walls or inside automobiles). They are usually capable of autonomous operation and can conduct meaningful activities on their own without communication to other devices. However, in many applications they are expected to share information or results with other devices at various levels of the architecture. One specific role for devices at this layer is to manage and control a set of devices in Layer 4, often in a hierarchical architecture (e.g. a Bluetooth host device and its currently connected paired devices).

Fig. 1. Academic Framework Architecture



Software for Layer 3 devices is often open or extensible to enable easy addition of applications and interfaces to other devices. For academic use, some extensibility of this type is important to ease the development of meaningful student projects without requiring a complete build of a system from hardware to software.

C. Personal Computer Layer 2

The next layer of the architecture includes the traditional personal computer. Layer 2 (and Layer 1) of the architecture are more familiar and long established parts of many academic environments.

Layer 2 devices may be mobile or fixed and are connected to an intranet within an organization or to the public internet. They have been the hands-on component of the majority of applications for decades. While increased use of cloud computing applications has reduced the importance of applications that run completely on the personal computer, most users expect to be able to run complete applications within the architecture of their device.

While it could be argued that Layer 2 and Layer 3 devices are really the same thing (somewhat general purpose computers with operating systems that allow extensions), the framework architecture distinguishes them. Some applications (e.g. a web browser) may be able to run on devices at both levels and perform similar functions. However, the demand of distributed systems will increasingly require network, performance, and security distinctions between these two levels. For example, a Layer 3 controller of several wearable body or environment sensors at Layer 4 may need to operate in environments such as airplanes when a networked personal computer will not be useable.

D. Intelligent Internet Layer 1

The top layer of the architecture is the familiar world of the Intelligent Internet including the ability to have broad access to multiple traditional servers, data base systems and cloud resources. The Intelligent Internet will continue to evolve new applications and services including some to interconnect, serve, and integrate devices at layers 2 through 4 of the architecture.

While many academic courses provide students an opportunity to learn and develop systems across layers 1 and 2, few now span all four layers of this proposed architecture. Thinking about architecture solutions, designing, programming, testing, and characterizing of applications across all four layers is key to student understanding of the current world of distributed embedded systems.

E. Key Technologies and Typical Devices

For successful use in academia, especially at institutions that do not have extensive hardware laboratories, the architecture must be readily realized using a combination of commercially available devices. Table I shows key technologies and example devices for Layer 3 and Layer 4 of the architecture. Layers 2 and 1 use the well-established computing and internet hosted devices that are common to academic courses today.

The requirements for Layer 3 and 4 devices are driven by the learning goals described in the next section; the key requirements are:

- Low Cost: reasonably low hardware and operational costs to ensure adequate numbers of devices can be maintained on hand without requiring support staff.
- Public Interfaces: published (at minimum) or open source definition of interfaces to allow devices to be incorporated into the architecture and interwork with other devices at one or more layers.
- Development Tool Availability: one or more tool sets or tool chains must be available to allow creation of student software on the devices and customization of existing software. It is acceptable to have a unique tool set for each device and different programming languages for different devices. If fact, dealing with disparate languages, design approaches, and tools may add important student skills.

Many of these requirements can be met by devices provided with various open source licenses. Open source hardware [11] and software is well suited to the needs of academic use as it allows access to implementation details.

Table I distinguishes fixed commercial devices ("Commercial Off The Shelf", or COTS) and extensible devices that may be easily customized by students. Much successful student project work can be done by incorporating COTS devices into a broader architecture.

However, COTS devices may be of limited use in projects where their existing interfaces cannot be extended or modified. Students need to understand how to meld together different kinds of devices to fully appreciate the complexities of modern systems. There are now several available prototyping systems that can provide accessible customization similar to the industrial environment (e.g. pcDuino, RaspberryPi, Arduino.)

TABLE I. TECHNOLOGY AND DEVICES

Layer	**Framework Embedded Architecture**		
	Key Technologies	***COTS Devices***	***Custom Devices***
3 – Intelligent Devices	Touch screens, speech recognition, gesture recognition, motion sensing	Mobile phones, tablet computers, heads-up displays / glasses, flexible keyboards	Extensible computer boards, compute sticks, wearable computers
4 – Functional Devices	Diverse physical world sensors and actuators, real-time computation for analysis	Fitness trackers, smart watches, portable headsets, smart buttons, RFID tags, biometric sensors	Microcontroller controller prototyping boards, wearable controller boards, programmable devices

IV. Learning Outcomes

To evaluate the architecture's value for student learning, a formal assessment of learning outcomes is intended, based on Bloom's Taxonomy. Bloom's Taxonomy was conceived to improve communication and comparison of test results by giving better precision to terms such as "thinking" and "problem solving" [3] and later updated to support standards-based curriculum planning and evaluation tools [12]. It has recently been used increasingly in computer science education [13, 14].

The six cognitive categories in Bloom's Taxonomy represent increasingly complex and more sophisticated learning. Example assessment topics for each category are:

1) Remembering: ability to recognize communication software functions used in one or more layers of the architecture (e.g., BlueTooth, WiFi Direct).
2) Understanding: ability to explain and contrast memory allocation methods at one or more layers of the architecture (e.g. RAM, flash memory, dedicated program memory).
3) Application: skill to apply suitable implementation techniques at one or more layers of the architecture (e.g. when to use software events or hardware interrupts).
4) Analysis: skill to select appropriate communication techniques between two or more layers of the architecture (e.g. when to use parity or other error detection / correction).
5) Evaluating: ability to make judgements about alternative approaches using multiple devices (e.g. able to critique proposed architectural solutions for a given distributed application).
6) Creating: ability to define and construct custom applications using multiple devices and multiple levels of the architecture (e.g. ability to define and develop a complete capstone project).

V. Example Projects

The Android Wall Project [15] explores the application of Layer 2 communication clusters of commodity tablet devices to problems spanning a "trilogy" of concerns: sensing, computation, and visualization. The conjecture is that these clusters may provide a low-cost, energy-efficient, flexible, and ultimately effective platform to tackle a wide range of problems within this trilogy. Applications include environmental and security monitoring, calculating and visualizing the energy footprint of an organization, facial recognition involving multiple cameras, visualizing relationships among versions of a text, etc.

The Smart Watch Integration project develops applications across several layers: programmable Android smart watch communicating locally with a tablet computer using Bluetooth; tablet connected to GPS and public internet to access location and map information. Interacting programs created to: receive periodic updates of local address text, display to user, and vibrate when arriving at desired address (on watch); to use current geo-location to retrieve nearest address from internet and send address to watch (on tablet). This project uses a (still rare) COTS smart watch that is extensible with user written programs [16].

References

[1] W. Wolf, "What and why about architecture for embedded systems," Proceedings WCAE '00 Workshop on Computer Architecture Education, ACM, 2000, doi>10.1145/1275240.1275243.

[2] A. Azzarà, D. Alessandrelli, M. Petracca, P. Pagano, "Demonstration abstract: PyoT, a macroprogramming framework for the IoT", Proceedings 13th International Symposium on Information Processing in Sensor Networks, IEEE, April 2014.

[3] B. Bloom, M. Engelhart, E. Furst, W. Hill, and D. Krathwohl, Taxonomy of Educational Objectives Handbook 1: Cognitive Domain, Longman, New York, 1956

[4] A. Schaefer, et al., "The empirically refined competence structure model for embedded micro- and nanosystems," Proceedings 17th Annual Conference of Innovation and Technology in Computer Science Education, pp. 57-62, ACM, doi>10.1145/2325296.2325314.

[5] S. Jaschke, et al., "Competence research: teaching embedded micro/nano systems," WESE '11, Proceedings 6th Workshop on Embedded Systems Education, pp. 17-24, ACM, doi>10.1145/2077370.2077373.

[6] L. Sousa, S. Antao, J. Germano, "A lab project on the design and implementation of programmable and configurable embedded systems," IEEE Transactions of Education, vol. 56, issue 3, pp. 322-328, August 2013, doi>10.1109/TE.2012.2222411.

[7] E. Brand, W. Honig, and M. Wojtowicz, "Intelligent systems development in a non engineering curriculum", Proceedings 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE '11), pp.48-52, ACM, doi>10.1145/1999747.1999764.

[8] G. K. Thiruvathukal, Distributed Sysems Course Materials, http://distributed.cs.luc.edu/html/ , retrieved 4/11/2015.

[9] R. Brandon (ed), "Requirements for internet hosts – communication layers", Internet Engineering Task Force, 1989, https://tools.ietf.org/html/rfc1122 , retrieved 4/11/2015.

[10] NEW: IoT-A, Internet of Things Architecture, http://www.iot-a.eu/public, retrieved 5/12/2015.

[11] R. Stallman, "Why we need free digital hardware designs", Wired, March 2015, http://www.wired.com/2015/03/need-free-digital-hardware-designs/, retrieved 4/11/2015.

[12] L. Anderson., D. Krathwohl, P. Airasian,, K. Cruikshank, R. Mayer, P. Pintrich, J. Raths, and M. Wittrock, Taxonomy for Learning, Teaching, and Assessing. Pearson, New York, 2000.

[13] W. Honig, "Teaching and assessing programming fundamentals for non majors with visual programming", ITiCSE '13, Proceedings 18th ACM Conference on Innovation and Technology in Computer Science Education, pp. 40-45, ACM, doi>10.1145/2462476.2462492.

[14] C. Starr, B. Manaris, and R. Stalvey, "Bloom's taxonomy revisited: specifying assessable learning objectives in computer science", SIGCSE Bull. 40, 1 (March 2008), 261-265, doi>10.1145/1352322.1352227.

[15] T. Delgado Dias, X. Yan, K. Läufer, and G. K. Thiruvathukal, "Building capable, energy-efficient, flexible visualization and sensing clusters from commodity tablets: position statement and preliminary progress report", 2nd Greater Chicago Area System Research Workshop (GCASR), May 3, 2013, Evanston, IL, USA, http://ecommons.luc.edu/cs_facpubs/66/ , retrieved 4/11/2015.

[16] i'm SpA, i'm Watch, http://www.imsmart.com/en , retrieved 4/11/1015.