A New Course for Teaching Internet of Things: A Practical, Hands-on, and Systems-level Approach

Mr. Nicholas Barendt, Case Western Reserve University

Nick Barendt is an Adjunct Senior Instructor in the Department of Electrical Engineering and Computer Science at Case Western Reserve University, in Cleveland, Ohio. He earned his Bachelor of Science and Master of Science in Electrical Engineering and Applied Physics at Case Western Reserve University, in Cleveland, Ohio, in 1995 and 1998, respectively. He has worked in a variety of industries, including Industrial Automation, Robotics, Data Acquisition, and Test and Measurement. He has lead technology teams, professional service firms, and startups. He consults with industry and academia on business and technology. He is a Senior Member of the IEEE.

Dr. Nigamanth Sridhar, Cleveland State University

Nigamanth Sridhar is the Dean of the College of Graduate Studies and Professor in Electrical Engineering and Computer Science at Cleveland State University. His research interests are largely focused on computer science education, with specific attention to issues of equity in computer science courses taught in the K-12 school system. This work is supported by grants from the NSF and the Cleveland Foundation. He holds a Ph.D. in Computer Science from Ohio State University.

Dr. Kenneth A. Loparo, Case Western Reserve University

Kenneth A. Loparo is the Nord Professor of Engineering in the Department of Electrical Engineering and Computer Science and holds academic appointments in the Departments of Biomedical Engineering and Mechanical and Aerospace Engineering in the Case School of Engineering. He has received numerous awards including the Sigma Xi Research Award for contributions to stochastic control, the John S. Diekoff Award for Distinguished Graduate Teaching, the Tau Beta Pi Outstanding Engineering and Science Professor Award, the Undergraduate Teaching Excellence Award, the Carl F. Wittke Award for Distinguished Undergraduate Teaching and the Srinivasa P. Gutti Memorial Engineering Teaching Award. He was Associate Dean of Engineering from 1994 -1997 and chair of the Department of Systems Engineering from 1990 -1994 and the Department of Electrical Engineering and Computer Science from 2013-2017.

Loparo is a fellow of a Life Fellow of the IEEE and a fellow of AIMBE, his research interests include stability and control of nonlinear and stochastic systems with applications to large-scale systems; nonlinear filtering with applications to monitoring, fault detection, diagnosis, prognosis and reconfigurable control; information theory aspects of stochastic and quantized systems with applications to adaptive and dual control and the design of distributed autonomous control systems; the development of advanced signal processing and data analytics for monitoring and tracking of physiological behavior in health and disease.

A New Course for Teaching IoT
- a Practical, Hands-On, and Systems-Level Approach

Nicholas Barendt
Case Western Reserve University
Cleveland, Ohio
nab2@case.edu

Nigamanth Sridhar
Cleveland State University
Cleveland, Ohio
n.sridhar1@csuohio.edu

Kenneth Loparo
Case Western Reserve University
Cleveland, Ohio
kal4@case.edu

Abstract

This paper describes a one-semester course developed to address a gap in undergraduate engineering education – preparing students for creating and maintaining Internet-of-Things (IoT) products and services. The principles that drove the course content and organization are explained, along with a novel courseware delivery mechanism and organization to facilitate repeatability, as well as some additional tools the authors have found useful. The two-part organization of the IoT course content – building a complete IoT system, and then investigating system properties, behaviors, and concerns of that system – is explained in some depth. A detailed course outline illustrates the wide variety of technologies students gain hands-on experience with during the course, including embedded, web/cloud, mobile, analytics, load testing, security. A novel application of DevOps tools to incrementally deliver multi-platform (systems) solutions each week is discussed. Finally, lessons learned from several offerings of the course are presented, along with challenges, opportunities and successes, and directions for future work.

Keywords

Internet of Things, IoT Education, Raspberry Pi, MQTT, Security, Analytics, DevOps

Introduction

Forecasts for the growth in the number of connected devices are staggering. One report [1] predicts 8.4 billion connected things in use worldwide in 2017 and 20.4 billion by 2020. The scale of this growth makes it critical that computer science and engineering students and others in related fields are instilled with the core concepts, technologies, architectures, and system-level perspective of this IoT revolution. The authors are not the first to identify this need; both [2] and [3] have pointed out the need and the challenges. This paper describes the design of a single, undergraduate course that addresses this need, as well as a description of how this course can be replicated at other institutions.

Background & Motivation

The course described in this paper was developed to address a perceived need in the undergraduate engineering curriculum to prepare students for building, deploying, and managing Internet of Things (IoT) products and services. Building IoT products requires connecting different platforms, often with a variety of technologies and programming languages, into a system. Both the variety of platforms and the "systems" nature of IoT often make it hard to approach. Undergraduate students in Computer Science (CS), Computer Engineering (CE), and Electrical Engineering (EE) typically have limited hands-on experience with multiple platforms, and rarely any experience with systems. Many current undergraduate IoT courses focus individually on embedded systems [4] [5], hardware [6] [7], or networking [8] , which are important, but possibly miss the educational opportunity of exposing students to the greater breadth of technologies required to deliver an IoT product or service. Similarly, cloud computing courses focus on the cloud infrastructure [9], and less on systems. Other undergraduate IoT educational opportunities appear as undergraduate research opportunities [10] [11], also important, but not necessarily scalable. A few graduate courses exist, such as described in [12], but we believe this topic needs to be more broadly accessible to undergraduates. [13] describes a course that introduces students to embedded systems, sensors, networks, and cloud computing, but does not appear to tackle system-level concepts such as analytics and security, both critical to modern IoT systems.

This course was intended to address these gaps in as comprehensible a manner as possible, subject to the constraints of a 1-semester, 3 credit-hour course. The course was developed as an undergraduate technical elective, targeting juniors and seniors in CS, CE, and EE.

This course was developed as part of an unusual partnership between higher education and industry. Expert practitioners, with deep, specialized technical knowledge from a software consultancy, LeanDog, Inc., partnered with Case Western Reserve University (CWRU) in Cleveland, Ohio, to build the initial courseware in 2015. The course was initially offered in the Fall of 2015 and again, with updates, in the Spring of 2017. The third course offering was a special joint-offering, with students from both CWRU and Cleveland State University (CSU) in the Spring of 2018.

Basic principles for creating the class include:

- Class lectures and assignments should be experiential (hands-on)
- The class should focus on state-of-the art technologies, protocols, and platforms in current industry use
- The class should provide students with authentic experiences with real-world software systems, including IoT, using a variety of languages, protocols, etc.

- The class should avoid the use of turn-key IoT Platforms, so that students understand the individual technology subsystems, and how those subsystems connect into a complete system
- The class should demonstrate evolutionary, incremental product/system design, and allow students to gain an understanding of how complicated products/systems can be evolved from simple cores
- The value of pairing for hands-on, experiential learning should be integrated throughout the class experience
- The class should emphasize the value of weekly "show-and-tell" demonstrations of new, working functionality, along with the recognition that product evolution rests on incremental development of complete, working versions of the product
- The class should encourage students with different backgrounds to participate, by minimizing prerequisites
- Students should gain experience with both functional and non-functional systems (e.g., scaling, security, etc.) requirements in assignments

The description of the course content, structure, etc. is based on the most recent offering. Later in the paper we provide examples of how the course has evolved.

Course Prerequisites

Given the breadth of technologies involved in this course, there is no simple set of course prerequisites; making the prerequisites too restrictive defeats the overall intention of the course. To date there are no prerequisites, but instructor consent has been required to register. During information sessions and individual student discussions and phone calls, the instructors have explained the course and useful experience; i.e., some programming experience, experience working at a Unix/Linux command-line as opposed to only working in a graphical Integrated Development Environment (IDE), etc.

Pedagogy and Methods

A key goal of the course is to create an environment where students can learn the skill of being introduced to a variety of new technologies in a short period of time, as well as how to integrate these different technology components together to build a complex system. This skill is representative of what is expected by software development organizations but is one that is difficult to include as a learning objective in a traditional lecture-based course. The collection of technologies that the authors have chosen to use for this course are not presented as the *only* ones to be used in IoT development; rather, students are provided with a basis in the underlying technical infrastructure and recognize that the particular choices are simply representative.

The course is delivered using a combination of Lectures and hands-on Project Assignments, similar to other Project-Based Learning courses [14][15]. The Project Assignments rely on the Courseware, described in detail below.

Courseware

The Courseware (equivalent to a digital textbook, plus software for the various platforms) is delivered via Git, a popular Distributed Version Control System (DVCS). The Courseware has evolved with each offering of the course, incrementally improving each time, similar to the concepts articulated in [16]. The Courseware resides in an instructor-only repository, and the content is incrementally released to students, via a second repository for students that is used only for the duration of the semester term. Each week (a) the solution to the previous week's assignment, (b) new textbook content for the current week, and (c) the current week's assignment are incrementally released to the students. The textbook content is written with Markdown [17], a simple text format that allows for rich content to be easily created and delivered to students (including basic styling: headings, bold, italics, URL links, cross-reference links within the courseware, images, etc.). GitHub, where the repositories are hosted, automatically renders Markdown to HTML when the repository is viewed with a web browser.

The organization of the courseware itself is designed using the principles of DevOps that students learn about during the semester, and which are used as standard practice in the software development industry. To facilitate the incremental delivery of content, the instructor repository has a somewhat novel branch organization. The chapter for each week has two branches associated with it: one for the assignment and one for the solution, of the form "chapter_XX_assignment" and "chapter_XX_solution" (where XX is in [00-14]). Everything in the "_assignment" branch appears in the related "_solution" branch, plus the assignment solution (and Ansible, the DevOps portion of the solution, as discussed below). Each week after the first week builds on the branch that comes before it, so, for example, everything in "chapter_04_solution" appears in "chapter_05_assignment"; since "chapter_04_solution" also contains the content from "chapter_04_assignment" and "chapter_03_solution", etc. back to the start, "chapter_05_assignment" also does. This branching organization requires some thought and discipline to manage (e.g., an update to the content requires the change to be made on the branch that contains the earliest instance of that file), but experience has shown that this structure greatly facilitates the overall management of the course. Some simple automated scripts have been developed to help manage this structure and organization, keeping content synchronized across branches, as well as to perform some automated checks on the Courseware (e.g., compliance to coding standards, integrity of internal cross-reference links, etc.).

Courseware Delivery

Each week of the course addresses a new concept or technology with a lecture, including live-coding demonstrations, and accompanying Courseware including the assignment for the week

(see Course Syllabus, below, for details). Courseware, including the solution for the previous week and the new content for the current week, is merged from the instructor-only repository into the student repository. The nature of the solutions for this course are novel enough that they are addressed below in **DevOps for Solution Sets**.

The Courseware has detailed instructions for the new assignment, walking students through installing and configuring any needed software components or tools for the week's assignment, followed by a demonstration of the concept or technology for the students to perform on their own, and the project assignment for the week, explaining the requirements to be delivered, often including screenshots, animated-GIFs, etc., as well as links to additional material that might be of help.

Project Assignments

Students work in pairs for each assignment. The value of pairing for improving student programming efficiency and quality, particular for students in the lowest GPA quartiles, has been reported in [18], [19], and [20]. Pairing has also been associated with students starting projects earlier [18], which we believe is important for student success in this fast-paced course.

Students "clone" the student Git repository into their own copy of the repository where they and their partner for the assignment do their work for the project assignment, alongside the Courseware. This might involve extending source code files provided as templates to be filled out, creating new source code and/or configuration files, capturing data from system tests, or related tasks. As the assignments become more complex, requiring more than one computing platform/node, they use this single repository to capture all needed software and configuration files for submission.

Project Submission and Demonstration

Before the assignment due date, students must submit the following:

- A Git "pull-request" of their working solution for the assignment for their pair
- A brief text write-up for the assignment, describing what each student learned and what surprised them in completing their assignment
- A video demonstrating the required functionality for the assignment

At the beginning of the next class period, students are required to demonstrate their working solution for the assignment (the video serves as a backup in case of technical issues, as well as requiring students to think through the process of demonstrating their solution before class).

Student Support

Students have access to the instructors and TAs during the class time and office hours. Additionally, and in our experience a critical aspect of the course delivery, students and instructors use a real-time collaboration tool, *Slack*, to communicate. Instructors and TAs make important announcements, share suggestions and pointers for the assignment, announce last-minute updates to the courseware, answer student questions, provide diagnostic advice, and share information about related events and talks. Students are able to pose questions, ask for assistance, share screenshots when they have issues, etc. Most important for scaling this course, and gratifying for the instructors, we have found that students begin to support each other using this tool, answering each other's questions, providing tips, etc.

For many assignments, students are required to install and configure third-party libraries and services (e.g., web servers, message brokers, etc.) on their Raspberry Pi 3 B (RPI3B) and Amazon Web Services (AWS) Elastic Compute Cloud (EC2)virtual machines. With these machines being geographically dispersed, and, in the case of the AWS EC2 virtual machines, managed by student accounts and cryptographic keys, it can be difficult to assist students with diagnosing installation and configuration issues. To help address these issues, students have access to a "call for help" software application that when executed provides instructors temporary, secure access to the student machine as a "superuser", using a "jump server" architecture [21].

Instructor and TA Coordination and Organization

Once again, we emphasize that not only does this class teach students how to develop complex IoT systems, but it also models and demonstrates a variety of best practices that are used in the software development industry. These best practices are usually mentioned or discussed in a course on software engineering, for instance, but students rarely get to see them in action. In this course, the instructors and TAs make these best practices "come alive" for students to experience first-hand.

The instructors and TAs use a number of tools to help organize and support the class:

- *Git*, as mentioned above, for the Courseware, as well as for storing and versioning grading rubrics, lecture demo scripts for live-coding, and other meta-content
- *Trello*, a web-based project management tool, based on a "card" being a unit of work, is used to coordinate and track updates to the Courseware as well as per-week tasks (delivering updated Courseware to students, preparing for the week's lecture and live-coding, grading assignments, etc.)
- *Slack*, in addition to the student-instructor and student-student interaction mentioned above, a private, instructor-only channel is used for instructor-instructor and instructor-TA communication and coordination

- *Google Documents*, including lecture presentations (*Google Slides*), hardware order tracking (*Google Sheets*), and a survey that is sent to students at the beginning of the semester to gather information on their technical experience (*Google Forms*)

Course Syllabus

At a high-level, the course consists of two parts:

1. Constructing the IoT System (the "product")
2. Instrumenting, extending, and evolving the IoT System - Properties, Behaviors, and Concerns of Systems

The course begins by incrementally constructing an IoT System: constructing a device and then connecting that device to the "cloud" and to mobile devices, constructing multiple User Interfaces and integrating the various platforms and technologies with a publish-subscribe architecture (see Figure 1). After the IoT System is completed, approximately midway through the semester, the course then shifts focus to the behaviors, properties, and system-level concerns of that system (e.g., analytics, remote firmware update, load testing, security, etc.) by instrumenting, extending, and evolving the IoT System built during the first part.

This two-phase approach – constructing the IoT System and then evolving that IoT System – allows students to understand how the various components are individually built and how they are connected into a useful system, and then, further, how that system behaves (e.g., observing it under load), how to evolve it (e.g., remote firmware update), how to instrument and understand the system and user behavior (analytics) and how system-level concerns impact the various components (e.g., security). Note that there are several IoT products available in the market that do all these things out of the box; this course does not use any of these products, because the central point of the course is to enable students to "go into the box" to see all the individual components, and more importantly, how these components integrate with one another.

Starting with the "Thing"- the Hardware Kit

It would have been possible to construct a course or IoT platform with a required collection of features and services to connect to a generically wide range of devices. We decided early on that making the course more concrete and less abstract would facilitate students' understanding and experience, especially for an undergraduate course.

After a number of brainstorming sessions early-on to decide what "thing" (device) to build the course around, we chose a simple desk lamp. The decision to use a desk lamp was chosen based on several criteria:

- The device's use and utility should be obvious to most any user; avoid devices requiring substantial domain knowledge (e.g., motor control), particularly since the target audience of students is broad (EE, CE, CS)
- The device should have some state to manage, but that state should not be overly complicated
- The device should be easily portable, so students can develop and use it in the classroom, home, dorm room, laboratory, or wherever they might be, easily fit into their backpack, and easily powered
- The device should be low-cost so that each student can have their own kit

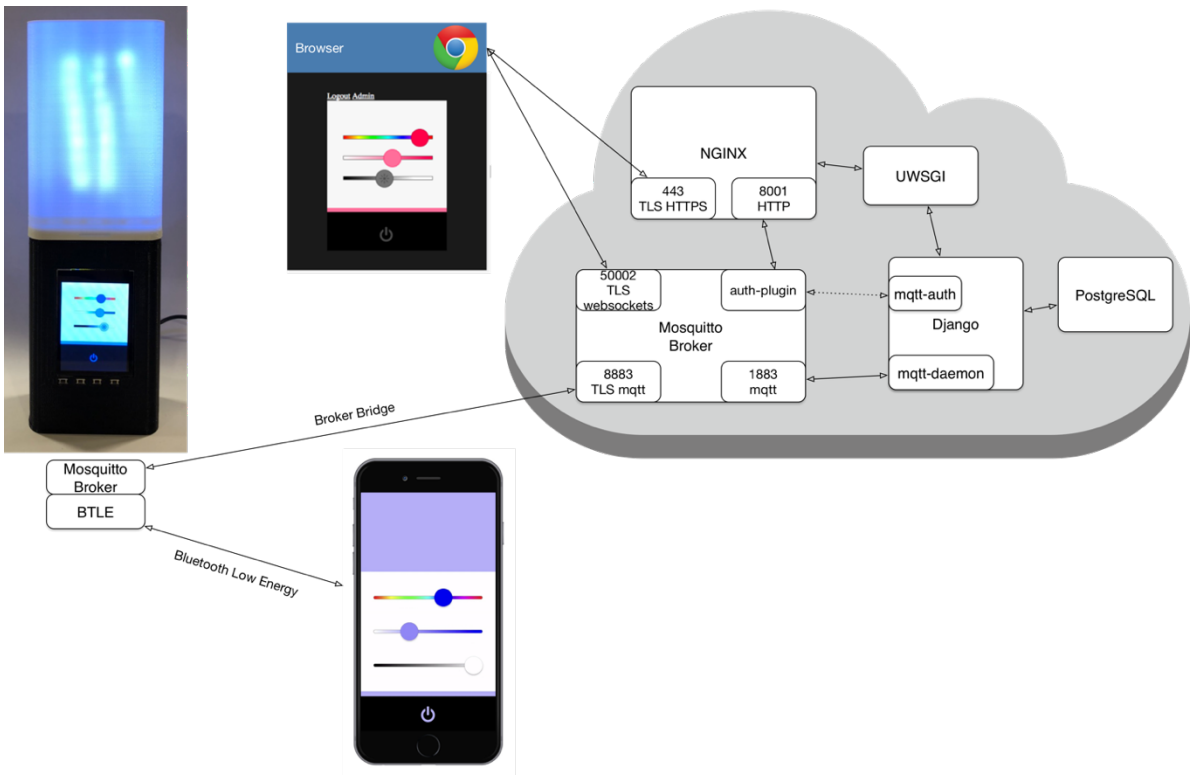

Figure 1: An overview of the system, showing the various technology components that students work with, and how these components come together in the course.

A desk lamp's operation should be evident to most people. To make the device and state interesting, the lighting element is a low-cost Red-Green-Blue (RGB) LED strip, driven from a simple, low-cost circuit board. The lamp's state consists of:

- Color (Hue and Saturation)
- Brightness
- On/Off

This design lends itself to surviving even when bounced around in student backpacks, and has proven to be relatively robust.

Whether the world needs an internet-connected desk lamp, with Wi-Fi and BLE, and full analytics support, is discussed during class, somewhat tongue-in-cheek.

LAMPI

We decided to name the custom-designed desk lamp to foster a more comprehensive IoT system, with a little branding, logo, common User Interface (UI), etc. We settled on "LAMPI" as the product name (see Figure 2).

Embedded System

Deciding on the embedded system (processor family, memory, I/O, etc.) involved a compromise between a few constraints

- Cost and form-factor of available Single-Board Computers (SBCs) and related development tools (embedded development often requires special hardware debuggers, cross-compiler toolchains, etc.)
- Sufficient computing resources for networking (e.g., TCP/IP, 802.11 Wi-Fi, Bluetooth, etc.) and adequate security (e.g., SSL/TLS)
- Active development community (software and hardware accessories)
- Limited class time available to ramp-up students on embedded systems- this is not an Embedded Systems Course (see course principals, above)

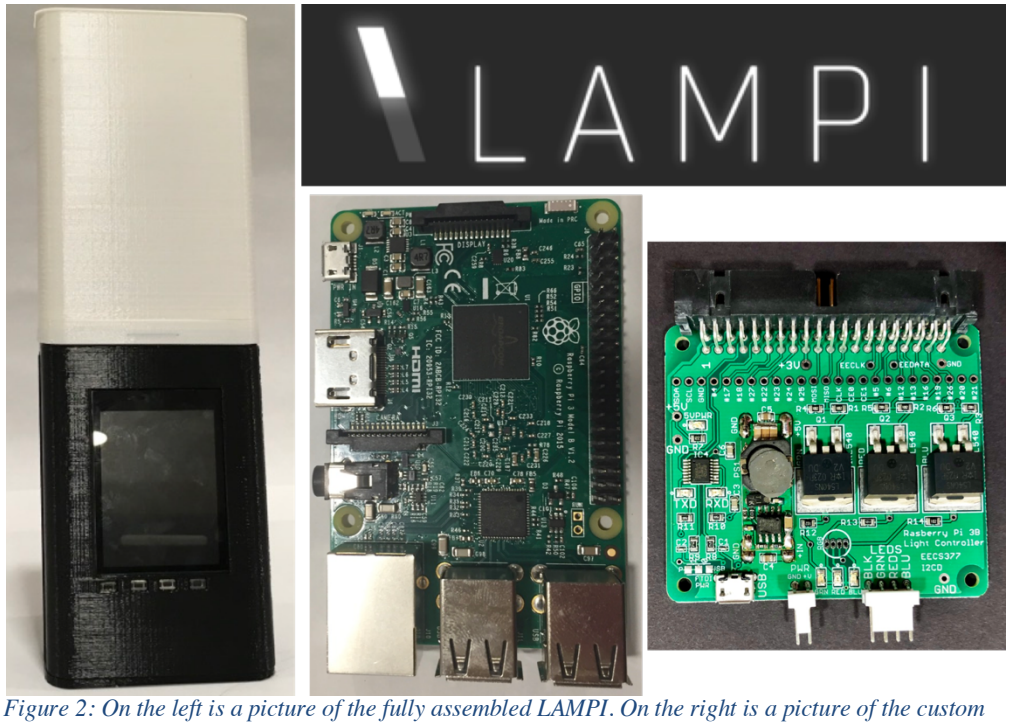

Figure 2: On the left is a picture of the fully assembled LAMPI. On the right is a picture of the custom PCBA that we use in the LAMPI, and the Raspberry Pi 3 B

Given all of these, we, along with many others [22] [23] [24], settled on the ubiquitous Raspberry Pi 3 B (RPI3B) (see Figure 2). The RPI3B is relatively low-cost (US$35), physically small, readily available from multiple distributors, is "self-hosting" with well-supported GNU/Linux development tools, typically runs the Linux OS that offers a complete, high-level OS (networking, multi-processing, etc.), and has more-than-sufficient computing resources (1.2GHZ Quad Core ARM processor, 1GB of DRAM, and both 802.11 Wi-Fi and Bluetooth) for our needs, allowing students to develop in a high-level language (avoiding the learning curve of C/C++, which is common in embedded systems). Right on day one of the class, we are clear with students that the RPI3B may not be a realistic choice for a consumer device, like a desk lamp, given the cost, but that it was chosen for the course for the reasons above.

Touchscreen

Fortunately, there are a variety of touchscreens available for the RPI3B, including several small form-factor options. We chose a capacitive touchscreen from Adafruit [25].

Custom PCBA

Low-cost, analog RGB LED strips were chosen to be the light-source for the desk lamp, given their low cost and wide availability. To source enough power to drive these LED strips with the RPI3B, however, a small, simple custom Printed Circuit Board Assembly (PCBA) (see Figure 2) was required. The PCBA has three Field Effect Transistors (FETs), one to drive each color channel of the RGB LED strip. General Purpose Input/Output (GPIO) pins on the RPI3B control each FET, and Pulse-Width Modulation (PWM) allows the brightness of each color channel to be controlled independently. The PCBA provides power to the RPI3B and touchscreen, containing a 9VDC to 5VDC step-down converter. The PCBA also includes an FTDI USB-Serial interface chip and a Micro-USB connector to simplify connecting to the serial port on the RPI3B. The PCBA is powered by a 9VDC "wall-wart" style power supply.

Enclosure

The desk lamp enclosure was modeled in a CAD package with 3D printing of the parts in mind. The enclosure consists of 4 parts: base front, base rear, LED column, shade. All of the parts are printed on hobbyist-grade Fused Deposition Modeling (FDM) 3D printers with commodity, low-cost Polylactic Acid (PLA) filament. The base parts are printed in black material. The LED column and shade are printed in white, translucent material (see Figure 2).

Hardware Kit Logistics

Ordering components, custom PCBAs, materials for 3D printing, coordinating the 3D printing, building wiring subassemblies, etc. can be time consuming, and needs to start months before the course begins. Final "kitting" occurs a day or two before the beginning of class. (In our most

recent offering, for example, we had all the parts ordered by Thanksgiving of the previous year in order for everything to be ready for a Spring semester course.)

Semester Plan

Table 1: : The list of topics introduced in each week, along with the specific technology components that students use

WEEK	TOPIC	SCOPE	TECHNOLOGIES
PART 1 – BUILDING THE LAMPI IOT SYSTEM			
1	IoT and Devices; Hardware Kit	Embedded	Raspbian, UART, Wi-Fi, PWM
2	User Experience & User Interfaces	UI & Embedded	Kivy
3	Pub/Sub Architectures	Pub/Sub	MQTT, Mosquitto
4	Connecting Devices to the Cloud	Cloud	AWS EC2, Mosquitto
5	Web User Interfaces	Web	HTML/CSS, JavaScript, WebSockets
6	Web Frameworks	Web	Python Django, SQL
7	User-Device Association & Production Deployment	Web	NGINX
8	Mobile	Mobile	iOS
9	Bluetooth Low Energy	Mobile & Embedded	BLE, Core Bluetooth
PART 2 – SYSTEM PROPERTIES, BEHAVIORS, AND CONCERNS			
10	Analytics & Dashboards	System	Keen.io
11	Secure Remote Update	System	PKI, Cryptographic Keys & Signing
12	User Modeling, Load Testing, Scaling	System	AWS EC2, Locust.io
13	Essential Security	System	SSL/TLS, CA, HTTPS, MQTTS, NGINX
14	Rules-Based Access Control	System	HTTPS, MQTTS, Mosquitto
15	Final Projects	System	*various*

As outlined above, the course is divided into two parts: constructing the IoT System (the "product"), and instrumenting, extending, and evolving the IoT System - Properties, Behaviors, and Concerns of Systems and the supporting courseware is divided into week-long modules, where each module tackles a new technology or concept. Details are provided in Table 1.

Course Sample

Here is an example of one week of the course, Week 10: Analytics and Dashboards. Prior to this, students have incrementally constructed the LAMPI IoT System (including: the device with a touchscreen UI, connectivity to the cloud, a Web UI, and an iOS application with BLE connectivity to the device).

At the beginning of Week 10, students are provided a complete solution for the system in the Courseware up through and including the previous week (including Ansible configurations to deploy the solution to their RPI3B device and AWS EC2 virtual machine server). The lecture for the week introduces the concepts of analytics and the importance of data in IoT and features live-coding to demonstrate the analytics platform used for the course (*Keen.io*, a SaaS analytics platform). The Courseware provides step by step instructions for creating a free account and installing required libraries for each environment (RPI3B, AWS EC2 Ubuntu, Python Django, and iOS), how to capture events on each platform, and how to query the analytics database and generate graphs. The assignment is given in two parts: instrument each of the UIs to capture and record common events, and then construct a specified Web dashboard (Figure 3).

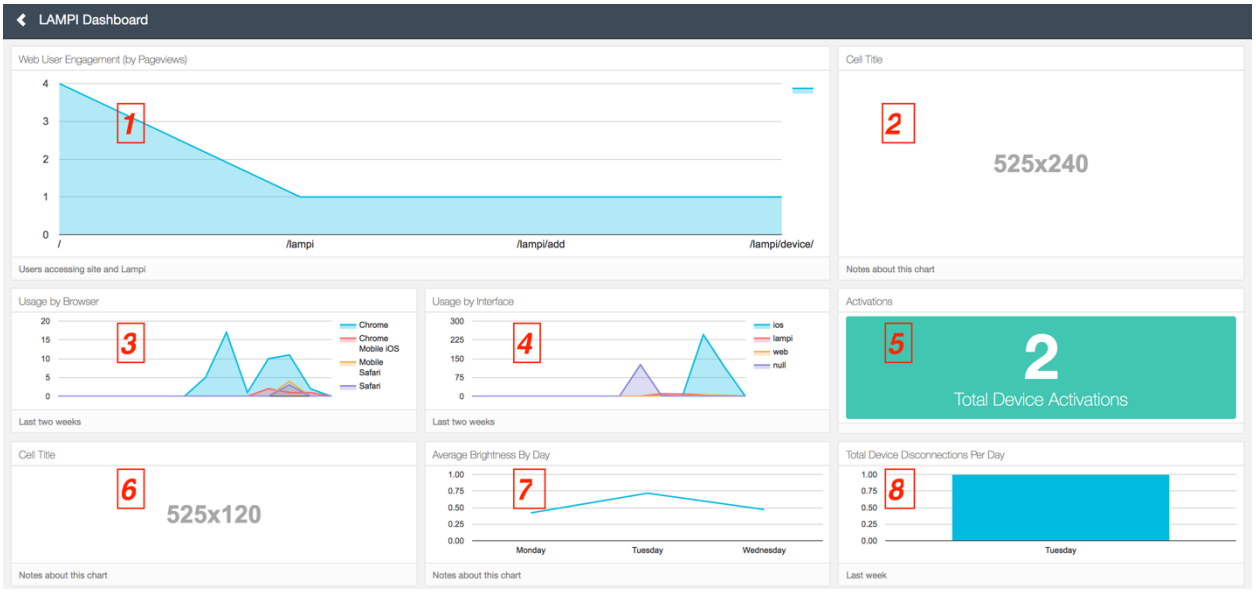

Figure 3: Week 10 Analytics Dashboard Assignment

DevOps for Solution Sets

Unlike a typical programming course, where the solution is captured in one or a handful of source code files that can be shared with the students along with some explanatory text, solutions for this course involve deployed source code along with system configuration (installed services and packages, configuration files, etc.), particularly as the semester progresses and the solutions involve configuring and deploying code to multiple computers. Further, given the incremental nature of this course, where each week's assignment literally builds upon the previous week's

work, student systems must be updated to a known working solution before the current week's assignment can begin.

This course uses a DevOps [26] tool to deliver solution sets to students as part of the Courseware. DevOps system automation tools, like Chef, Puppet, and Ansible, allow a computer's configuration to be specified like source-code, and make the configuration process repeatable (Chef uses the term "recipe" to refer to the configuration file, an apt analogy). This configuration includes installation of third-party packages (typically libraries and services), creation of accounts, file system directories, configuration of services, as well as deploying custom applications and source code. This system configuration, captured in text files, can be versioned alongside custom software, often within the same version control repository.

Ansible was chosen for use in this course. Each week a portion of the new Courseware delivered to students includes a complete solution for the previous week's assignment as a set of Ansible "playbooks" that will update student systems to a known working solution state. Students execute these playbooks against their computing environment (e.g., their RPI3B and AWS EC2 virtual machine) before beginning the new assignment. These playbooks recreate the set of operations the students should have completed in the assignment.

Building and maintaining these playbooks is a non-trivial effort, but has proven critical to keeping students from falling behind if they struggle on a particular assignment. Given the incremental structure of the solution sets, built to support the Courseware, it also allows a "vanilla" OS installation to be "fast-forwarded" to any given point-in-time for the semester. This makes it straight-forward for students (and instructors!) to quickly recover from a hardware glitch or system misconfiguration - start with a clean slate, run the playbooks for the current week, and within minutes have a complete, from-scratch, working implementation of the previous week's assignment.

Grading

Grading for each assignment is based on a rubric. Each assignment's rubric addresses point allocations for demonstrating aspects of functionality required by the assignment, as well as student submission of a demonstration video and short write-up, and an in-class demonstration.

Results

The course has been offered three times: At CWRU in Fall 2015 and Spring 2017, and a joint offering between CWRU and CSU in Spring 2018, with increasingly larger classes. Approximately 55 students have completed the course so far. Grades and student feedback (formal course evaluations and ad hoc discussions) indicate that students are generally mastering the material covered. Students are particularly vocal in their praise for the hands-on nature of the course and its coverage of current industry technologies and practices. Surveys have shown that

most students are registering for the course because another student that took the course recommended it. The authors intend to introduce more rigorous qualitative and quantitative assessment into the course in the next offering.

Student comprehension may best be judged by the Final Projects, where they must work independently from any text or courseware. The Final Project demonstrates the students' ability to synthesize the material from the semester and solve a new, novel engineering problem – success at synthesizing the course material results in a successful Final Project while a failure to synthesize the course material results in a failed Final Project.

A few recent Final Projects topics includes:

- Augmented Reality - Build and integrate the LAMPI UI for Microsoft HoloLens
- Alexa - integrate Amazon Alexa into the lamp (including adding microphone and speaker) and create an Alexa Skill to control LAMPI
- Google Home - integrate Google Home into the lamp (including adding microphone and speaker) and create a Google Home integration to control LAMPI
- Building Enchanted Objects [28] for student campus life (weather, laundry, campus transportation)
- Extend Locust.io, the tool used for HTTP load testing in the course, to load test MQTT
- Build a light-based alarm clock with LAMPI
- Build an SMS interface for LAMPI with Twilio
- Build an IFTTT integration for LAMPI

The Final Project grading rubric has 4 sections: Written Communication (15%), Presentation (15%), Demonstration (20%), and Problem Solving (50%). The first two tie to the ABET Student Outcomes [27] "(g) an ability to communicate effectively"; the third and fourth to ABET Student Outcomes "(e) an ability to identify, formulate, and solve engineering problems" and "(k) an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice".

Evolution of the course

The course has evolved considerably after each offering. Feedback and periodic retrospective discussions with students and instructors have led to dramatic improvements in the Courseware content (e.g., overhaul of analytics coverage and BLE, improving text consistency with a style guide), Courseware delivery (e.g., the Git branch structure discussed earlier and related automation tools), the hardware (complete redesign after the first offering leading to a simpler, more easily produced desk lamp), and overall course repeatability (the addition of the Ansible DevOps solutions to the Courseware, clearer grading rubrics, and creation of live-coding demonstration scripts).

Conclusions

We conclude with a brief discussion of challenges for a course of this nature, opportunities and successes so far, and directions for future work.

Challenges

One of the key principles articulated for this course, "class should focus on state-of-the art technologies, protocols, and platforms in current industry use" is critical to its relevance, but implicitly creates its most challenging aspect - to adhere to that principle requires that the course content be continually updated. Individually, the technology platforms used in the course are all under rapid development (e.g., Linux, iOS, AWS, Django, etc.). Combined, these changes require reviewing, revising, updating, and testing essentially all aspects of the course each time it is offered (Courseware, Assignments, Solutions, including the DevOps portion, Lectures, etc.), which is non-trivial. Similar challenges have been pointed out for teaching cloud computing [29], and [30] applies DevOps to overcoming some of the challenges.

Given the technical breadth of the material covered in the course, it also requires instructors and TAs with equivalent technical breadth. The TAs in particular must also have, or rapidly develop, excellent diagnostic skills on a variety of platforms, as they must support students often experiencing the languages and platforms for the first time. Augmenting instructor experience with appropriate Subject Matter Experts (SMEs) to guest lecture on particular topics (e.g., User Experience and User Interface Design, iOS, JavaScript) has proven valuable, both for the experience such SMEs can provide, but also for students still learning about the wide variety of engineering and software development careers available.

The complexity added by a hardware component creates additional complexity for a course of this nature. Designing an IoT product, including the enclosure and electronics, requires a variety of disciplines to come together. Ordering all of the components, including the custom PCBA, and coordinating 3D printing of the enclosure parts, plus kitting, requires a lot of time. The cost is approximately $150 per student kit, purchasing in low volumes (40 kits). To achieve low costs, many of the parts must be ordered months in advance of the course. Including a hardware component in the course requires supporting that hardware throughout the course, diagnosing possible hardware issues, and, invariably, replacing damaged components.

Opportunities and Successes

The need for a systems-based IoT undergraduate course seems clear. Experience with this course has demonstrated strong student interest, at least partially driven by the hands-on, experiential nature of the course. Additionally, the broad applicability beyond IoT of many of the technical concepts, including analytics, load testing, and essential security, has proven a strong student draw.

The range of technologies, languages, and platforms required to build a real-world IoT system can be daunting. By carefully structuring an incremental build path, where students can be rapidly introduced to a new technology and build something of interest, on top of their previous work, has proven viable. While the instructors are clear to point out that the one-week course assignments will not produce a mastery of any particular technology or language, students are often impressed at how much they have learned in a single semester (e.g., building a touch screen UI to control a device, connecting that device to the cloud, building a Web UI, building a BLE interface for the device, building an iOS app with BLE integration, etc.).

The value of the two-phase approach to teaching IoT systems (building the system, followed by investigating system properties, behaviors, and concerns) has been proven in our estimation - students learn how to incrementally construct the system, and are thus better prepared to investigate and extend the system, armed with a clear understanding of its architecture.

Directions for Future Work

Future offerings of the course will require the periodic updates discussed above - necessary to keep up with changes to technologies and platforms used. Incremental additions, such as building courseware to support Android BLE, along with the existing iOS support, are likely. More far reaching changes are possible, including full or partial virtualization/simulation to replace the hardware kit. This would reduce the cost of the course, and hopefully reduce barriers to broader, large-scale adoption.

Building a Continuous Integration (CI) system to support the maintenance of the Courseware is a likely future path. A CI system with appropriate tests could simplify the periodic updates required by automatically testing changes to the Courseware and testing the Courseware with updated versions of the technologies and platforms used.

Expanding this course to more institutions and more students is of special interest, including building the systems and processes necessary to support faculty at other institutions in adopting the material and tools. Beyond scaling the course to the target audience of juniors and seniors in undergraduate engineering disciplines, the instructors are considering adapting the course to reach broader audiences, including college freshmen, high school seniors, and industry practitioners. The authors invite collaborative discussions in any of these areas.

Acknowledgements

The authors would like to thank and acknowledge LeanDog, Inc., the Institute for Smart, Secure and Connected Systems (ISSACS) and the Electrical Engineering and Computer Science Department at Case Western Reserve University, the Baldwin Endowment Fund managed by the Case Alumni Association, the Electrical Engineering and Computer Science Department at Cleveland State University, and the Cleveland Foundation for their funding and support. The

authors would also like to offer their personal thanks to Gary Johnson and John Gibbons for their support and technical contributions in creating, supporting, and expanding this course.

References

[1] N. Eddy, "Gartner: 21 Billion IoT devices to invade by 2020," *InformationWeek Nov*, vol. 10, 2015.
[2] L. Laird and N. Bowen, "A new software engineering undergraduate program supporting the Internet of Things (IoT) and Cyber-Physical Systems (CPS)," in *Proc. ASEE*, 2016, pp. 26–29.
[3] G. Kortuem, A. K. Bandara, N. Smith, M. Richards, and M. Petre, "Educating the Internet-of-Things Generation," *Computer*, vol. 46, no. 2, pp. 53–61, Feb. 2013.
[4] J. He, D. C.-T. Lo, Y. Xie, and J. Lartigue, "Integrating Internet of Things (IoT) into STEM undergraduate education: Case study of a modern technology infused courseware for embedded system course," in *Frontiers in Education Conference (FIE), 2016 IEEE*, 2016, pp. 1–9.
[5] D. V. Gadre, R. S. Gaonkar, N. Prasannakumar, and S. N. Ved, "Embedded Systems and Internet of Things (IoTs) - Challenges in Teaching the ARM Controller in the Classroom," presented at the 2017 ASEE Annual Conference & Exposition, 2017.
[6] O. T. Ayodele, O. O. Kazeem, O. O. Akintade, and L. O. Kehinde, "Development of an Interface and Connectivity Platform for a Basic IoT Training Module," *ASEE Comput. Educ. CoED J.*, vol. 8, no. 3, Nov. 2017.
[7] S. Abraham and A. Miguel, "Creation of an Internet of Things (IoT)-Based Innovation Lab," presented at the 2017 ASEE Annual Conference & Exposition, 2017.
[8] S. J. Dickerson, "Preparing Undergraduate Engineering Students for the Internet of Things," presented at the 2016 ASEE Annual Conference & Exposition, 2016.
[9] D. Guo and A. Koufakou, "How Cloud Computing Is Addressed for Software Development in Computer Science Education," in *Human-Computer Interaction. User Interface Design, Development and Multimodality*, 2017, pp. 415–426.
[10] D. Turgut, L. Massi, S. S. Bacanli, and N. H. Bidoki, "Multidisciplinary Undergraduate Research Experience in the Internet of Things: Student Outcomes, Faculty Perceptions, and Lessons Learned," presented at the 2017 ASEE Annual Conference & Exposition, 2017.
[11] S. Shayesteh, M. Rizkalla, and Mohamed El-Sharkawy, "Curriculum Innovations through Advancement of MEMS/NEMS and Wearable Devices Technologies," presented at the 2017 ASEE Annual Conference & Exposition, 2017.
[12] X. Liu and O. Baiocchi, "14 An Internet of Things (IoT) Course for a Computer Science Graduate Program," *Shap. Future ICT Trends Inf. Technol. Commun. Eng. Manag.*, 2017.
[13] S. J. Dickerson, "A comprehensive approach to educating students about the internet-of-things," in *2017 IEEE Frontiers in Education Conference (FIE)*, 2017, pp. 1–7.
[14] R. Ross, J. Whittington, and P. Huynh, "LaserTag for STEM Engagement and Education," *IEEE Access*, vol. 5, pp. 19305–19310, 2017.
[15] Nansong Wu and Kaiman Zeng, "Embedded System Based First-Year Engineering Course with Aid of Online Simulation and Social Media." First Year Engineering Experience Conference Enhancing the First Year of Engineering Education.
[16] C. D. Kloos *et al.*, "From software engineering to courseware engineering," in *2016 IEEE Global Engineering Education Conference (EDUCON)*, 2016, pp. 1122–1128.

[17] "Markdown (Aaron Swartz: The Weblog)." [Online]. Available: http://www.aaronsw.com/weblog/001189. [Accessed: 02-Feb-2018].
[18] C. Kowalec and A. DeOrio, "Partnership Characteristics and Student Performance in an Introductory Computer Science Course," presented at the 2017 ASEE Annual Conference & Exposition, 2017.
[19] M. O. Smith, A. Giugliano, and A. DeOrio, "Long Term Effects of Pair Programming," *IEEE Trans. Educ.*, vol. PP, no. 99, pp. 1–8, 2017.
[20] F. J. Rodríguez, K. M. Price, and K. E. Boyer, "Exploring the Pair Programming Process: Characteristics of Effective Collaboration," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, New York, NY, USA, 2017, pp. 507–512.
[21] "Jump server," *Wikipedia*. 30-Dec-2017.
[22] X. Zhong and Y. Liang, "Raspberry Pi: an effective vehicle in teaching the internet of things in computer science and engineering," *Electronics*, vol. 5, no. 3, p. 56, 2016.
[23] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, "Raspberry Pi as Internet of things hardware: performances and constraints," *Des. Issues*, vol. 3, p. 8, 2014.
[24] G. S. Sundaram *et al.*, "Bluetooth communication using a touchscreen interface with the Raspberry Pi," in *Southeastcon, 2013 Proceedings of IEEE*, 2013, pp. 1–4.
[25] Adafruit Industries, "Adafruit PiTFT Plus 320x240 2.8" TFT + Capacitive Touchscreen." [Online]. Available: https://www.adafruit.com/product/2423. [Accessed: 02-Feb-2018].
[26] "DevOps," *Wikipedia*. 22-Jan-2018.
[27] "Criteria for Accrediting Engineering Programs, 2016 – 2017 | ABET." .
[28] D. Rose, *Enchanted Objects: Design, Human Desire, and the Internet of Things*. Simon and Schuster, 2014.
[29] S. Campbell, "Teaching Cloud Computing," *Computer*, vol. 49, no. 9, pp. 91–93, 2016.
[30] H. B. Christensen, "Teaching DevOps and Cloud Computing Using a Cognitive Apprenticeship and Story-Telling Approach," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, New York, NY, USA, 2016, pp. 174–179.