Raspberry Pi as a Platform for the Internet of Things Projects: Experiences and Lessons

Stan Kurkovsky, Chad Williams
Central Connecticut State University
Department of Computer Science
{kurkovsky,cwilliams}@ccsu.edu

ABSTRACT
The Internet of Things (IoT) represents a new computing paradigm that may soon add a new dimension to the skillset expected from a well-rounded computing professional. Computer Science education is addressing these demands by adding IoT-centric courses to the curriculum and including relevant content into a broad range of existing courses. This paper presents the experience of incorporating IoT projects into an existing Systems Programming course. We examine several suitable hardware platforms, provide a sampling of student projects implemented using the Raspberry Pi with a variety of sensors, and discuss a number of lessons learned that could benefit other educators planning to incorporate the IoT material into their coursework.

CCS Concepts
• Social and professional topics ~ Computer science education • Computer systems organization ~ Embedded and cyber-physical systems

Keywords
Internet of Things; Raspberry Pi; sensors; student projects.

1. INTRODUCTION
The term 'Internet of Things' (IoT) became truly prominent in 2005 when the International Telecommunication Union published a report on this subject [12]. The IoT concept encompasses devices, sensors, and services existing within an interconnected infrastructure with an efficient access to ample computational facilities. This paradigm enables tangible objects to gather and transmit information about the physical world, thus creating usable services and interfaces allowing these objects to intelligently interact with their users and other systems [28]. The IoT systems have been successfully implemented in many areas, including home automation, surveillance, transportation, and healthcare.

An IHS report from March 2016 indicates that the number of installed IoT devices will grow from 15.4 billion in 2015 to 30.7 billion devices in 2020 and to 75.4 billion in 2025 [11]. Consequently, the number of IoT devices is expected to exceed the number of mobile phones by 2018 becoming the largest category of connected devices [4]. A McKinsey report estimates that IoT has a potential economic impact between $3.9 trillion to $11.1 trillion a year by 2025 [19]. It is unsurprising that many major industry players, including IBM, Samsung, Intel, and ARM, to name just a few, are showing a very significant interest in the IoT market showcasing their relevant products at many recent technology exhibitions.

ITiCSE '17, July 03-05, 2017, Bologna, Italy

DOI: http://dx.doi.org/10.1145/3059009.3059028

Unlike a new technology or a framework du jour, the IoT represents a new paradigm of computing, much like mobile computing did a decade ago. Given such a tremendous growth in the number of deployed devices and services, the IoT is positioned to make a significant impact on the skillset of professionals working in the software and hardware industries. Computer Science educators began to respond to these demands by adding IoT-centric courses in the curriculum and including relevant content into a broad range of existing courses. This paper reviews a number of such initiatives and describes the authors' experiences of incorporating IoT projects into an existing Systems Programming course. We examine several hardware platforms suitable for student projects, provide a brief sampling of the projects completed by the students, and conclude by discussing a number of our experiences and lessons that could benefit other educators planning to incorporate the IoT material into their coursework.

2. BACKGROUND WORK
A draft version of Computer Engineering Curricula 2016 does mention the IoT in the Computer Engineering Body of Knowledge (Strategies for Emerging Technologies unit, Applied Emerging Technologies area) as a core outcome: *"Describe the internet of things (IoT) and its effect on computer engineering."* We believe that the role of the IoT is significantly downplayed here because other items in the same list of outcomes also include *"Describe the manner in which a 3D printer would reproduce a miniature model sailboat in a bottle"* and *"Explain ways in which Web 2.0 and social networks might affect the field of computer engineering."*

Lacking specific guidance from the current Computer Science Curricula [7], colleges and universities have begun to introduce IoT topics in many different ways, some of which are sampled below.

Jeong et al [13] describe an IoT-centric course designed around the Lab of Things platforms developed by Microsoft Research [3]. With the IoT being the main topic of the course, the authors used a project-based learning approach to provide students with a solid experience of building IoT systems using industry-standard frameworks such as Microsoft Azure.

Osipov and Riliskis [21] conducted a 'triple-run experiment,' in which three quarter-long graduate and undergraduate courses were taught in parallel to provide a holistic view on the IoT concept. The work in each of these three network-centric courses covered network applications, programming, and wireless sensor networks. It was rooted in different aspects of the same research project for

non-intrusive detection and monitoring of motor vehicles traveling along the motorways equipped with a road sensor network.

Perhaps the largest course to introduce the IoT concepts has been My Digital Life offered by the Open University in the UK [15]. First offered in 2011 and scheduled to be discontinued in 2017, this 9-month introductory course attracted many thousands of students. Each student in this course received a SenseBoard, a custom-designed device based on the Arduino microcontroller, along with a collection of sensors that can be attached to the SenseBoard. SenseBoard runs Sense, a visual programming environment based on MIT's Scratch that has been extended with special features to interact with the sensors.

Given that many existing CS programs have little or no room for adding another required course into their curriculum, integrating IoT topics into other existing courses is a viable option. He et al [9] describe a broad framework of laboratory modules for an embedded systems design course, which incorporates many elements of IoT systems, their design and programming. Using Arduino and Raspberry Pi kits, each module is designed for using independently from others, but the courses in which they can be adopted are greatly limited by their embedded systems context.

Zhong and Liang developed an IoT-focused course for Computer Science majors using the Raspberry Pi as the hardware platform [29]. The course focuses on a series of progressively complex projects, in which student teams design and prototype devices constructed by pairing Raspberry Pi with a range of sensors.

Whether educators used IoT systems built out of a single device [2,15,29] or as a part of a larger infrastructure combining cloud-based services with local hardware [9,13,21], there is a strong consensus that Computer Science graduates need more exposure to the IoT concepts and that IoT systems offer an excellent foundation for hands-on student projects.

Incorporating IoT studies in the Computer Science curriculum may have a number of additional positive impacts. Using tangible toolkits for physical computing that incorporate tinkering with processor boards, wires, sensors, and other devices support pedagogies of production and collaboration [5]. In general, as a constructionist activity, creating tangible objects has been shown to help develop critical thinking skills and solidify the learners' understanding of the subject matter [22]. Not being constrained to a single computing environment, such as a desktop computer, tangible components used to build IoT devices embedded in everyday objects provide learners with ample opportunities to construct and share their mental models through design and prototyping of interactive objects.

3. IOT HARDWARE PLATFORMS

There is an ongoing debate aimed to standardize the reference architecture common to all IoT systems [27]. As shown in Figure 1, generally, the architecture of most IoT systems is comprised of four functional layers: a *sensing layer* to collect the data about the physical world, which may also include actuators that make a tangible impact on the environment or other objects; an *access layer* to collect and transmit the data; a *service layer* to create and handle usable value-added services and applications, and an *interface layer* to interact with the users and other systems.

As a rule, the sensing and networking layers of most IoT systems are implemented within the hardware, while the load of the service and interface layers may be balanced between the device itself and the related infrastructure of various web-based services. A framework that supports designing, building, and experimenting with an IoT system in any setting, including student learning, must support some, if not all of the above layers inside of a hardware device rather than being implemented entirely on a desktop computer or any other general purpose computing device. Therefore, studying, let alone building an IoT system, requires a dedicated hardware platform capable of implementing some of the four layers within a relatively small physical form factor.

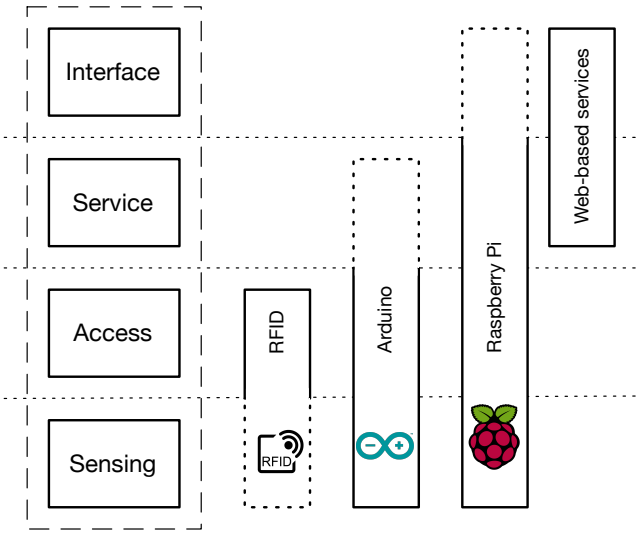

Figure 1. Hardware platforms for the Internet of Things systems and functional layers of their architecture.

Some educators pioneering the use of IoT paradigm in education as early as a decade ago, started experimenting with radio-frequency identification (RFID) technology to communicate the physical proximity of tangible objects to each other. RFID tags store and transmit identification numbers of physical objects they are embedded in or attached to, which can be read by RFID readers using radio communication [8]. Although today RFID is widely adopted in transportation, logistics, supply chain management, pharmaceutical, medical, and many other industries, RFID technology only supports the networking layer of the IoT paradigm. In a way, RFID also implements the sensing layer, but only to the extent of being able to detect the presence of the tagged object, without the possibility to collect any information about the environment. The functionality of all other layers typically must be implemented on a separate computing system equipped with one or more RFID readers.

Arduino, originally introduced in 2005, is a programmable microcontroller board that can be extended by connecting to other devices, sensors, or special purpose expansion boards ('shields'). Low cost, flexibility, and the ease of programming quickly made Arduino a popular choice for hobbyists, individual learners, and educators seeking a platform for teaching concepts ranging from introductory programming to embedded computing [10]. All Arduino boards can be programmed with a language based on C/C++ using a cross-platform IDE. WiFi networking has been added to recent versions of Arduino boards, while Ethernet ports were supported early on. Given the open architecture of Arduino to connect virtually any possible sensors, actuators, or networking interfaces, Arduino makes a great choice for building an IoT system with the full support for the sensing and networking architectural. However, given its limited processing power, Arduino may not be able to efficiently support the implementation of the service and interface layer components.

Raspberry Pi is an inexpensive credit-card sized computer originally developed to promote teaching computer science basics to school children [25]. Like Arduino, the Raspberry Pi encourages experimenting with its hardware configuration. The purpose behind the development of the Raspberry Pi project was to create a replacement for an increasingly complex 'closed box' computer that would encourage kids to code and tinker with it. The Raspberry Pi includes a processor, system memory, network interfaces, and a memory card slot, along with the ports for attaching sensors, peripherals, and other devices. A number of add-on board, including the Sensorian shield [18] and the Sense HAT (http://www.raspberrypi.org/products/sense-hat), combining several types of sensors have been developed for the Raspberry Pi. Typical setups include connecting Raspberry Pi to the same kind of peripherals that can be attached to a traditional desktop, or remotely logging in to it from another computer, in which case Raspberry Pi does not need to be connected to anything other than a power supply and a wired or wireless network. Raspberry Pi runs Linux-based Raspbian, which includes a number of programming environments such as Python, Scratch, BlueJ, and Greenfoot [14]. The Raspberry Pi platform is a natural fit for implementing the sensing and access layers of the IoT system architecture. In the case of designing a single-device IoT system, Raspberry Pi has enough flexibility and processing power to implement the service layer by running data- and sensor-driven applications on the device. At the same time, its IO and networking capabilities would easily allow one to extend it in order to implement the interface layer to make these applications easily accessible by the users and its services accessible by remote systems.

Many commercially available IoT devices and IoT projects described in the literature increasingly rely upon a cloud-based middleware infrastructure partially or fully implementing the functionality of the service and interface layers. This infrastructure can enable IoT devices to access a broad range of web services that can leverage much higher processing power, provide access large data sets, and enable a rich interaction with other systems and services [20]. For example, ThingSpeak enables collection, storage, analysis, and visualization of data received from sensors connected to Arduino, Raspberry Pi and other IoT hardware platforms [23]. ThingSpeak provides APIs for a number of programming languages including C, Java, and Python, as well as MATLAB support for data analytics and visualization. A survey of other middleware platforms for IoT can be found in [26].

4. IOT AND SYSTEMS PROGRAMMING

Our Systems Programming course has been a required component in our ABET-accredited undergraduate program in Computer Science for many years. The course originated as a low-level programming course aimed to study the inner workings of Linux while writing system-level programs in C. It is a 3-credit semester long course with a typical class size of 25 students. With Data Structures being the only prerequisite for this course, it is aimed mainly at sophomores and juniors.

We made a decision to add IoT material to this course to achieve the following objectives: expose students to the basics of programming hardware/software interfaces; review recent trends in IoT applications, especially in home automation; and provide students with an opportunity to design their own IoT system using the Raspberry Pi as the hardware platform.

At the same time, we had to operate within a number of constraints. All IoT material had to fit into the existing course without compromising its integrity. We needed to continue using C as the main programming language for Linux systems programming. Given the target audience, we had to assume that in many cases student experience beyond Data Structures may be limited. We also needed to present a substantial amount of sample IoT applications while leaving enough room for students to suggest their own ideas for the course project.

Raspberry Pi was chosen as the hardware platform for this course because its operating system is Linux-based and it comes equipped with the tools needed for programming in C making it an excellent fit for teaching all Systems Programming concepts. For the duration of the semester, each student received their own Raspberry Pi kit, which included Raspberry Pi 3 model B with WiFi and Bluetooth connectivity, a USB power supply, a 16GB Micro SD card with Raspbian OS, an enclosure case, two heatsinks, and an HDMI cable.

5. IOT PROJECTS

A semester-long project is an integral part of our Systems Programming course. With the IoT topics being integrated into the course, it only became natural to give the course project the focus on building an IoT system. Given the possible complexity of the resulting system, the importance of early design choices in shaping the structure and features of the system, and given a possibly limited student exposure to hardware design and advanced topics, we structured the course project into the following three incremental phases.

Simple IoT device/sensor. Students were first asked to design and implement a system that would enable the Raspberry Pi to interact with one or more sensors. This would help students break the barrier of designing a system comprised of both hardware and software, something that most of them likely have never done before. This phase implemented the sensing and access layers of the IoT system architecture outlined in Section 3. It was also important to ensure that the logic of interfacing and communicating with the sensors or other hardwired devices was implemented and debugged before moving to the next phase. For example, consider creating a single-device system to monitor the weather. This device can be designed using the Raspberry Pi with several sensors (such as temperature, pressure, and humidity) or a Sense HAT attached to it. This phase would involve enabling the Raspberry Pi to read the data from the sensors.

Smart IoT device. At this phase, students focused on the logic of processing the data received from the sensors. Based on this logic, students also implemented any value-added service that this system would be able to provide if it were implemented on a single device. This phase implemented the service and interface layers of the architecture. In the weather monitoring example, this phase would involve programming the Raspberry Pi to query the sensors on schedule or on demand, storing the readouts, and developing an interface (perhaps with an attached touchscreen monitor) to display the current readout values and, possibly, reviewing the readout history.

Networked IoT device. Once the main functionality of the previous phase is implemented, the students are ready to distribute the logic of this functionality across the network so that the services of this IoT system can be accessed from multiple locations, aggregated with data and/or services provided by other systems. This phase would extend or provide a new implementation of the service and interface layers. For example, instead of storing historical sensor readouts locally, the weather monitoring system could transfer them to a cloud-based service such as ThingSpeak [23], which can provide more advanced computational tools and data visualization

capabilities and combine the data from multiple weather monitoring IoT nodes.

The remainder of this section provides a brief sampling of student projects implementing IoT systems with the Raspberry Pi on our course. All of them were implemented following the three incremental phases described above.

5.1 Smart Camera

This system implements a motion-activated camera that takes a picture any time when a motion of a warm body is detected. The system is configurable so that the photo can be stored locally in the Raspberry Pi's memory, sent to an email address, and/or uploaded securely to a remote server. A system of this nature may have a number of application areas including security/surveillance cameras, wildlife/game photography, motion detection for home automation systems, license plate capture at gate/toll, baby monitoring, etc.

The Raspberry Pi Camera Module is attached via a ribbon cable to a camera serial interface (CSI) included on board of the Raspberry Pi. The case provided with the Raspberry Pi kit used in this project has a special cutout for the camera lens, so the camera module can be securely enclosed. The camera draws power from the Raspberry Pi, so no additional power cables are needed. The passive infrared sensor (PIR) is a simple device that sends a high (3V) signal when infrared radiation is detected. The sensor hardware itself controls sensitivity and delay settings for detection of motion and can be adjusted as needed. The PIR uses pulse width modulation (PWM). The PWM wires can be connected directly to the Raspberry Pi's general-purpose input/output (GPIO) pins as shown in Figure 2.

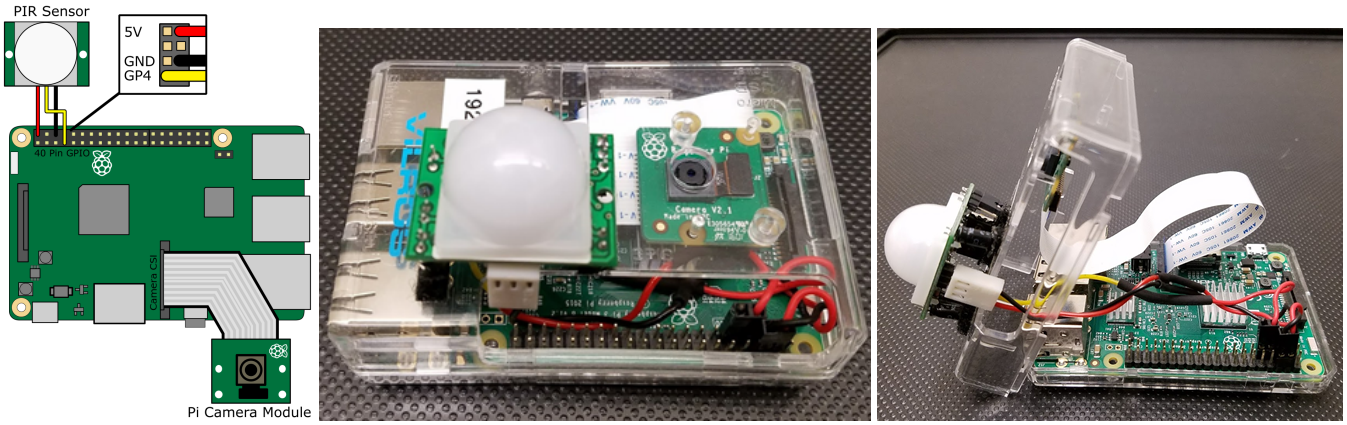

Figure 2. Design and implementation of the Smart Camera project.

5.2 Automatic Door Lock

A prototype of an automatic door lock was developed to unlock a door whenever a user of the previously approved mobile device is in the vicinity. This system used Bluetooth addresses associated with every Bluetooth-enabled mobile device as a label uniquely identifying its owner. Each approved device must be enrolled into the system, either locally, or remotely. When a Bluetooth-enabled device is in the vicinity of the lock, the Bluetooth module onboard the Raspberry Pi can identify its presence, compare it with a list of approved devices, and unlock the door if there is a match.

This device should be located behind the locked door to prevent tampering. Enrollment of approved Bluetooth devices and logging of unlocking attempts can be done locally, as well as remotely via a network connection to an external server. Remote operation also enables building this system without any additional devices attached to the Raspberry Pi. A solenoid-based linear actuator driving the door lock (http://www.adafruit.com/products/1512) requires an additional power supply to activate the electromagnet inside the actuator.

5.3 NFC Tracking

All student and faculty IDs at our university have a built-in RFID tag, which is used to unlock office doors and access parking garages. A device built with the Raspberry Pi connected to an appropriate RFID reader and placed at a room entrance could inconspicuously register all people entering or leaving the room. Such a system may be particularly useful in a computer lab to gather statistics about the utilization of the facility. At the very minimum, the collected data could show the amount of through traffic and its distribution over time. A minimal analysis of the data could tell how much time each individual spends in the lab. Connecting the system with the university's LDAP server would enable getting the names of individuals associated with their ID tags. Similar systems employing near field communication (NFC) technologies such as RFID for tracking people and objects are currently among the most broadly deployed implementations of IoT systems in the industry.

6. LESSONS LEARNED

Choice of the hardware platform. As shown in Section 2, both Raspberry Pi and Arduino have been successfully used to build IoT devices and systems in student projects. With the costs of the boards and sensors aside, both platforms are fully capable of implementing rich functionality for the sensing and access layers of the IoT system architecture shown in Figure 1. The choice between the two platforms, in our opinion, should be dictated mainly by the complexity of the other two layers: how much data processing and other logic is needed at the service layer to implement a meaningful application, and what are the requirements for the system's interaction with the users and, possibly, with other networked systems at the interface layer. The Raspberry Pi is very capable of implementing all four layers of functionality on the device itself. However, both platforms can be extended with web-based services such as ThingSpeak to fully implement the service and interface layers of functionality.

Teamwork. Given the target audience of sophomores and juniors, students may have a varied and possibly limited exposure to different Computer Science topics. Special care must be taken

when forming the project teams to ensure that each team is comprised of students with a varied knowledge background and skillsets that include some computer networking, databases, and distributed computing. Our experience also shows that it was much easier to debug the hardware issues when students were working on it as a team, which resulted in a division and exchange of responsibilities related to hardware.

Version control. As with any team environment, it is crucial to ensure that all team members are working with the same code base. Once the Raspberry Pi devices were added to the mix, this created an additional layer where mistakes could be made as many students chose to mix development on their computers and then transfer code to the Raspberry Pi devices. To help alleviate this issue, basic Git version control instruction was embedded in lessons and GitHub repositories were created for student teams to assist with distributed version control. Instructing students how to integrate Git version control into their workflow and distribute code between environments via GitHub rather than direct file transfer smoothed out this complexity that many students were initially encountering. Similar to the experience of other programs, this exposure also helped students see first-hand how versioning issues can arise in practice and the benefit of using tools to manage them [17].

Regression testing. A common issue students encountered in their debugging were small differences in hardware configuration of sensors from device to device causing code to execute differently between devices. One of the trends that emerged from this was that, without prompting, students started developing simple regression test suites to allow them to quickly verify that the problematic areas of code still worked with their configuration. This demonstrates that while students might have experienced frustration related to these scenarios, it also led to the highest level of understanding in Bloom's taxonomy – reflection, and a far deeper understanding of what the issue was and how to create basic regression testing in real systems [1].

Connection logistics. Raspbian does not properly support WiFi networking with enterprise-level certificate-based encryption, which is deployed on our campus. For classroom use, we chose to use a 'headless' setup for each individual Raspberry Pi, in which it would connect to a student's laptop via an Ethernet cable. The only other connection required is power, which can be supplied via a USB cable connected to the same laptop. This setup enables a straightforward terminal connection to the Raspberry Pi via SSH. While a terminal connection to Raspberry Pi is sufficient for writing C code with the *GNU nano* editor and *gcc* compiler, some students chose to use a VNC (virtual network computing) client to have full access to the Raspbian graphical desktop on their laptops.

Importance of design. Although we provided every student with a Raspberry Pi kit, we made a decision not to provide students with sensors or sensor kits. By doing so, we encouraged students to be more thorough in investigating the hardware features of the IoT systems they designed. Because of the time constraints of the project, these design decisions were crucial for a successful implementation of the project.

Real-world hardware. Hardware can sometimes break, wired connections may fail (especially if the entire Raspberry Pi plus sensors assembly is handled without a special care), sensors are not always precise, and the sensors may behave differently depending on the external factors such as network infrastructure and physical environment. These issues can add a lot of frustration and difficulty to debugging the code. As a result, students eventually learn to resolve these situations, which brings them closer to the real-world experiences that are much different compared to many traditional classroom projects.

Big data. By their very nature, IoT systems are designed to collect the data about their environment, their users, and the surrounding objects. Cloud-enabled IoT systems are capable of aggregating and analyzing this data. It would seem natural to add a discussion about big data analytics into an IoT-centric course. Similarly, any course focusing on big data would also benefit from added coverage of the IoT systems, their architecture, as well as the kinds and amounts of data they can generate.

Security and privacy. Proliferation of IoT devices naturally leads to an increased number of incidents where their possible security weaknesses are exploited. For example, in October 2016, a massive distributed denial of service attack was launched against Dyn, an Internet infrastructure company that provides critical services to many top companies, such as Amazon, Twitter, Netflix, and Spotify. This attack was launched using thousands of compromised IoT devices, such as digital video recorders and video cameras [16]. The IoT paradigm offers an excellent context for discussions about the tradeoffs between convenience and added value of the IoT services vs their possible threats to security and privacy. A summary addressing many aspects of security, privacy, and trustworthiness of many existing and plausible IoT system is presented in [24].

Ethical and societal issues. Related to security and privacy are the issues of eavesdropping and surveillance enabled by the IoT devices and services, the data they collect, and the resulting possibly compromising information that could be mined out of that data. These issues offer a fertile ground for discussions about the impact that the proliferation of IoT systems will have on the society and the individuals.

7. SUMMARY

As a computing paradigm, the IoT has already made very significant inroads in the industry. This is resulting in new demands on Computer Science programs to prepare graduates ready to build and work with the IoT systems, create tools to process and analyze the data they generate, and understand and address many aspects of the societal impact of the IoT ecosystem. Furthermore, IoT-centric projects offer an excellent framework for exposing students to many issues of hardware/software design, implementation, and testing. Students seem to embrace this approach, which is reflected in their feedback:

It was very beneficial to be able to learn what we did and see it in action in the IoT projects on the Raspberry Pi and devices. It really helped to understand the material better than just writing "ordinary" programs. I honestly can't say one bad thing about using the Raspberry Pi, it was a lot of fun and very educational.

Experimenting with IoT and RPi gave me a true understanding that computing systems can be built and used not only for personal computers and mobiles.

Raspberry Pi works well for introducing students to software/hardware interaction. Forces students to collaborate.

While students did note that additional complexity was added by incorporating the Raspberry Pi, the types of challenges reported related to requiring a more in depth knowledge of the operating system and hardware integration. Thus, while some additional effort may need to be made to simplify certain aspects of this type of IoT project to be incorporated in other courses to avoid adding overhead to the curriculum; for a Systems Programming class,

these challenges actually further emphasized the curriculum and led to deeper student learning. Our goal in future work will be to build upon these successes and recognize potential problem areas so that Raspberry Pi IoT project-based learning can be integrated into other existing courses in our curriculum.

8. ACKNOWLEDGMENTS

Providing each student in our course with a Raspberry Pi kit was made possible by a CCSU AAUP Curriculum Development grant.

9. REFERENCES

[1] Anderson, L.W., Krathwohl, D.R. and Bloom, B.S. 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Allyn & Bacon.

[2] Bruce, R.F., Brock, J.D. and Reiser, S.L. 2015. Make space for the Pi. In *SoutheastCon 2015*. IEEE.

[3] Brush, A.J., Filippov, E., Huang, D., Jung, J., Mahajan, R., Martinez, F., Mazhar, K., Phanishayee, A., Samuel, A., Scott, J. and Singh, R.P. 2013. Lab of things: a platform for conducting studies with connected devices in multiple homes. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication* (pp. 35-38). ACM.

[4] Cerwall, P., Ed. 2016. *Ericsson Mobility Report*. June 2016. https://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf.

[5] Charlton, P. and Avramides, K. 2016. Knowledge Construction in Computer Science and Engineering when Learning Through Making. *IEEE Transactions on Learning Technologies*, *9*(4), pp.379-390.

[6] Computer Engineering Curricula 2016. Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering. ACM/IEEE CS, October 2015. https://www.computer.org/cms/Computer.org/professional-education/curricula/ComputerEngineeringCurricula2016.pdf.

[7] Computer Science Curricula 2013. Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM/IEEE CS, December 2013. https://www.acm.org/education/CS2013-final-report.pdf.

[8] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), pp.1645-1660.

[9] He, J., Lo, D.C.T., Xie, Y. and Lartigue, J. 2016. 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* (pp. 1-9). IEEE.

[10] Hodges, S., Taylor, S., Villar, N., Scott, J., Bial, D. and Fischer, P.T. 2013. Prototyping connected devices for the internet of things. *Computer*, *46*(2), pp.26-34.

[11] IHS Markit. 2006. *IoT platforms: enabling the Internet of Things*. March 2016. https://www.ihs.com/Info/0416/internet-of-things.html.

[12] International Telecommunication Union. 2005. The Internet of Things. *ITU Internet Reports*. November 2005. https://www.itu.int/net/wsis/tunis/newsroom/stats/The-Internet-of-Things-2005.pdf

[13] Jeong, G.M., Truong, P.H., Lee, T.Y., Choi, J.W. and Lee, M. 2016. Course design for Internet of Things using Lab of Things of Microsoft Research. In *Frontiers in Education Conference (FIE), 2016 IEEE*.

[14] Kölling, M. 2016. Educational programming on the Raspberry Pi. *Electronics*, *5*(3), p.33.

[15] Kortuem, G., Bandara, A.K., Smith, N., Richards, M. and Petre, M. 2013. Educating the Internet-of-Things generation. *Computer*, *46*(2), pp.53-61.

[16] Krebs, B. 2016. Hacked Cameras, DVRs Powered Today's Massive Internet Outage. Krebs on Security, October 21, 2016. https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/

[17] Lawrance, J., Jung, S., and Wiseman, C. 2013. Git on the Cloud in the Classroom. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (SIGCSE '13). ACM, pp. 639-644.

[18] Mahmoud, Q.H., Qendri, D. and Lescisin, M. 2016, February. The Sensorian Shield: Transforming the Raspberry Pi into an IoT Platform. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education,* p.162. ACM.

[19] Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J., and Aharon, D. 2015. The Internet of Things: Mapping the Value Beyond the Hype. June 2015. http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/the-internet-of-things-the-value-of-digitizing-the-physical-world.

[20] Mashal, I., Alsaryrah, O., Chung, T.Y., Yang, C.Z., Kuo, W.H. and Agrawal, D.P. 2015. Choices for interaction with things on Internet and underlying issues. *Ad Hoc Networks*, *28*, pp.68-90.

[21] Osipov, E. and Riliskis, L. 2013. Educating innovators of future internet of things. In *2013 IEEE Frontiers in Education Conference (FIE)*, pp. 1352-1358. IEEE.

[22] Papert, S. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[23] ThingSpeak API. http://thingspeak.com, accessed January 12, 2017.

[24] Voas, J. 2016. Demystifying the Internet of Things, *Computer*, vol. 49, no. 6, pp. 80-83, June 2016.

[25] Vujović, V. and Maksimović, M. 2015. Raspberry Pi as a Sensor Web node for home automation. *Computers & Electrical Engineering*, *44*, pp.153-171.

[26] Wang, F., Hu, L., Zhou, J., Wu, Y., Hu, J. and Zhao, K. 2015. Software toolkits: practical aspects of the internet of things—A survey. *International Journal of Distributed Sensor Networks*, *2015*.

[27] Weyrich, M. and Ebert, C. 2016. Reference Architectures for the Internet of Things. *IEEE Software*, *33*(1), pp.112-116.

[28] Xu, L. D., He, W., and Li, S. 2014. Internet of Things in Industries: A Survey, *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233-2243. 2014.

[29] Zhong, X. and Liang, Y. 2016. Raspberry Pi: An Effective Vehicle in Teaching the Internet of Things in Computer Science and Engineering. *Electronics*, *5*(3).