2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #33785

An Introductory Course on the Design of IoT Edge Computing Devices

Mr. Matthew McConnell, Case Western Reserve University

Matthew McConnell has been a hardware design engineer building networked, embedded Linux devices primarily in the industrial Test and Measurement market for the past twenty years. He earned a Bachelor of Science in Electrical Engineering and Applied Physics and a Masters of Science in Electrical, Computer, and Systems Engineering at Case Western Reserve University. He currently collaborates with the Institute for Smart, Secure, and Connected Systems (ISSACS) to further IoT education and engineering programs at Case Western Reserve University.

Dr. Kenneth A. Loparo, Case Western Reserve University

Kenneth A. Loparo is the Arthur L. Parker Professor in the Department of Electrical, Computer and Systems Engineering, holds academic appointments in the Departments of Biomedical Engineering and Mechanical and Aerospace Engineering in the Case School of Engineering and the Faculty Director the Institute for Smart, Secure and Connected Systems. 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.

Mr. Nicholas A. Barendt, Case Western Reserve University

Nick Barendt is the Executive Director, Institute for Smart, Secure and Connected Systems (ISSACS) at Case Western Reserve University, in Cleveland, Ohio. He is also an Adjunct Senior Instructor in the Department of Electrical, Computer, and Systems Engineering and the Department of Computer and Data Sciences at Case Western Reserve University. He has worked in a variety of industries: Industrial Automation, Robotics, Data Acquisition, and Test and Measurement. He has lead technologies teams as well as been an entrepreneur. He consults with industry and academia. He is a Senior Member of the IEEE.

An Introductory Course on the Design of IoT Edge Computing Devices

Abstract

Edge Computing Devices are becoming increasingly important in the Internet of Things (IoT) ecosystem as they serve to bridge local IoT networks to Cloud resources while improving overall system performance by optimizing bandwidth usage, reducing decision latency, and minimizing costs. Due to these enhanced capabilities, IoT Edge Devices require more sophisticated designs than typical IoT Sensor Nodes that in turn require more sophisticated Design Engineers to build them. To prepare our students for these new challenges, we developed a hands-on laboratory course focused on the development tools, system components, and design paradigms used when building IoT Edge Devices. In this paper we describe the development of the course, our educational objectives, course syllabus, project assignments, results and suggestions for future course improvements.

Keywords

Internet of Things, IoT Education, Remote Learning, Edge Computing, Embedded Linux

Introduction

Over the past year, we have developed a new university-level Internet of Things (IoT) course primarily focused on providing students with a hands-on laboratory experience featuring the development tools, network topologies, and design paradigms needed for building real-world IoT products, primarily *IoT Edge Devices*. This paper describes the development process, lessons learned, and exemplar student outcomes from our work.

The IoT sector has grown rapidly in the past few years to become a critical infrastructure that affects our daily lives in many ways.[1] IoT sensor networks deployed throughout our homes, offices, hospitals, factories, cities, power grid, and beyond provide the opportunity for greater security, safer environments, reduced energy consumption, higher levels of comfort, and many other benefits. Managing these ever-expanding networks using only Cloud-based services limits their efficiency and prevents many real-time applications from being realized due to network latency and cost. To solve this problem, IoT Edge Devices distribute services to the local networks ("edges" of the network), reducing the reliance on the Cloud and enabling new, highly responsive applications.

Features of an IoT Edge Device vary depending on application needs—some include general computing or specialized computing resources (e.g., GPUs or TPUs), some have storage capability, others provide advanced networking capability or a combination of these. A general definition would be: a gateway that provides smart management and enhanced capability for the local IoT network. Due to this enhanced capability requirement, IoT Edge Device designs are more complicated than typical IoT sensor nodes and require the convergence of many different technologies.

The course breaks these Edge technologies into the following categories:

- Networking Communications
- Embedded Systems
- Sensor Hardware Interfacing
- User Interface
- Distributed Computing

These technologies are presented to students through lectures and tutorial examples, then reinforced by practical, hands-on projects, initially as standalone technologies, but with increasing integration throughout the semester. For the projects, students are allowed to choose their own end-applications, within set guidelines, to improve motivation and promote creativity.[2] For the most part, the end-applications themselves are less important than the students experience learning how to use the presented technologies to implement their design. Working through a realistic design process prepares students for future professional work on the next generation of IoT products, one of our desired goals.

We encountered many external challenges throughout the development of this course, including a pandemic, that drastically changed many of our original plans. This paper documents our experiences, shares the positive results we achieved and outlines future plans for course improvements.

Motivations for the Course

The conceptual framework for the course originated through conversations with industry partners who were concerned with the technical know-how of recent college graduates. Students may graduate with theoretical knowledge but have little hands-on experience, requiring on-the-job training before becoming effective team members. A hands-on, project-based course in Edge Computing provides students the opportunity to develop expertise with many of the technologies and development tools they may encounter after graduation. The technologies brought together in an Edge Computing application are broadly applicable on their own, so students would gain valuable experience regardless of the field or research area they choose to enter.[3]

Laboratory projects were designed to have a complexity similar to what an entry-level engineer would encounter. The focus on IoT Edge Devices adequately fits this complexity level given that most devices require more advanced networking, computation, and development strategies than

regular IoT nodes.[4] Greater emphasis is placed on the build tools and development processes than in other IoT courses in order to prepare students for real-world development.

We were particularly concerned with our students' experience level using Linux, one of the most common platforms for IoT Edge Devices. While an IoT sensor node can often function with a simple "super-loop" software design, the software complexity needed by an IoT Edge Device requires a multi-processing operating system to support the many functions required of these devices (networking, computation, data acquisition, etc.). Linux is the ideal choice due to its highly customizable kernel, wide application software support, and open-source ecosystem.

Although many students have used Linux platforms before, such as the Raspberry Pi, most have only interacted at the application level with a stock distribution (i.e. Raspbian or Ubuntu) and have not significantly delved into the operating system itself. Developing embedded Linux products with custom hardware (as is the case with most IoT Edge Devices) requires in-depth knowledge of the Linux kernel, system services, and development tools in order to build customized, product-specific, Linux distributions (including both kernel and root filesystem with required services and applications). IoT products must also deal with network security issues, deployment models, and in-field update strategies that are becoming increasingly important with the proliferation of billions of IoT devices throughout the world.[5]

We designed the course to meet these needs and fill the curriculum gap for students who would not otherwise encounter these topics through regular course work.

Laboratory Station Equipment

For the spring semester 2020, a new on-campus laboratory was assembled for this course with a generous donation from the Intel Corporation. The initial floor plan consisted of 15 laboratory stations each with a workstation computer, customized DE2i-150 Development System[6], sensor breakout boards, and other test equipment. Unfortunately, we were only able to use this laboratory in-person for a few weeks before the campus closed due to the pandemic.

Over the summer, the laboratory stations were revamped adding a remote access capability.[7] The current configuration of a student lab station (shown in Figure 1) is:

- Ubuntu Linux Workstation
- Intel® DE2i-150 Development System
- Custom Sensor Board
- Raspberry Pi Web Camera

Figure 1: Laboratory Station Equipment

The Linux Workstation acts as the main access point for the laboratory station, allowing students to log-in through a Secure Shell (SSH) connection using their campus Single Sign-On (SSO) credentials. On-campus students may connect directly while remote students connect through the campus VPN.

The DE2i-150 Development System connects to the Linux Workstation directly using a secondary Ethernet port, isolating it from the main campus network to provide greater security (both for the development system and the network at-large).

Each laboratory station has a camera providing students a remote view to monitor their work (i.e. development board power status, LCD image, LEDs, etc.). These cameras are attached to a Raspberry Pi connected to the Workstation via USB for power and communication. A web interface provides both the camera view and power controls for the DE2i-150.

Evolution of the Course

The course was originally designed as an in-person, hands-on laboratory where students would work on projects using the computer workstations, development boards, sensor hardware, and test equipment assembled in a newly furnished laboratory space. The open laboratory model allows students to work on projects on their own schedule or with a teaching assistant during scheduled times. An initial pilot offering in the spring semester 2020 started out following this model but unfortunately had to change direction when the campus was shut down due to the Coronavirus pandemic.

The mid-semester transition to remote learning was particularly difficult when students lost access to the laboratory equipment. Remote lectures were fine but student projects were greatly hindered. For the remainder of the spring semester we sent custom hardware kits to students

containing a Raspberry Pi with an accelerometer add-on. With these kits students were able to cover much of the original course content but some of the key components (such as Linux build systems and boot-loaders) had to be dropped from the laboratory projects, although all topics could still be covered in the lectures.

Over the summer we regained access to the lab to reconfigure it for completely remote access, enabling fall semester 2020 students to log into the laboratory workstations to build and run their project applications from anywhere in the world. Revamping the laboratory space and equipment for completely remote access was a challenge but luckily the necessary networking technologies (mainly SSH) were the same ones being taught in the course so, instead of obstructing students, the remote access was simply integrated with the project assignments providing students with a practical usage of the technology.

The course continues to evolve as feedback from the previous semester is evaluated and improvements devised to improve future offerings of the course for hybrid delivery (in-person and remote). Planned improvements are detailed later in the Future Work section.

Course Prerequisites

The course enrollment targeted undergraduate students in the Electrical Engineering, Computer Engineering, or Computer Science majors in their junior or senior years. A strong programming background in Python or C languages was required and prior experience using Linux was suggested. Python was used for the bulk of the application programming while all Linux Kernel device driver development required C. The course also includes basic web development requiring some knowledge of HTML, CSS, and Javascript.

Course Pedagogy and Structure

The learn-by-doing approach, augmented with lectures covering many finer details, provide students with multiple avenues for learning the presented technologies. Laboratory projects progressively build upon one another allowing students to create more complex projects over the semester while reinforcing earlier lessons.

The course was designed for a 14-week semester featuring eight laboratory projects with two 75-minute lectures per week. The laboratory assignment periods vary, starting out with three one-week projects, then moving to four two-week projects, and finishing off with a three-week final project. A schedule summary with project descriptions is shown in Table 1.

The early one-week projects allow many smaller topics to be covered rapidly in order to acclimate students with the basic technology and procedures used throughout the course. Each two-week lab focuses deeper on a technology area and includes a mini-project where students build a given project incorporating some custom elements of their own choosing. These guided mini-projects demonstrate the presented technology as well as allow the students to re-imagine how they may use the technology in their own designs. The course ends with a three-week final project in which students choose their own project (within certain limits and approved by the instructor) based on the culmination of the technologies and concepts acquired from earlier projects.

Week	Lab	Description
1	Lab 1	**Introduction to the Remote Lab** Setup remote lab accounts and run through a simple programming exercise.
2	Lab 2	**Working with IoT Sensors** Build a basic IoT sensor application using Python, MQTT, and AWS DynamoDB.
3	Lab 3	**Embedded Linux Development Fundamentals** Learn to use the Yocto Project build system to create a custom Linux image.
4	Lab 4	**IoT Communications**
5		Web application using WebAPI data, MQTT notifications, and dashboard visualization.
6	Lab 5	**Linux Device Drivers for Sensor Communication**
7		Develop a low-level Linux kernel driver to interface with sensor hardware.
8	Lab 6	**Edge Device User Interface Development**
9		Explore UI/UX design by building an application for the LCD panel.
10	Lab 7	**Edge Computing Applications**
11		Distributed computing applications using TensorFlow-Lite.
12	Lab 8	**Final Project**
13		
		Thanksgiving Break
14		Final Project Presentations

Table 1: Laboratory Assignment Schedule

Courseware Delivery

Instructional content was delivered through synchronous-remote lectures and through project guides accompanying each laboratory assignment. Important information were covered by both to ensure students had multiple exposures to learn and retain the knowledge presented. In addition, supplemental support was offered through Slack[8] discussion boards and through the Canvas Learning Management System (LMS)[9] to ensure students were successful throughout the course. The following sections provide further details for each of these mechanisms.

Lectures

In-person lectures were planned for the spring 2020 pilot course but had to transition to remote (via Zoom[10]) halfway through the semester and remained remote for the fall semester. We found a number of benefits using the remote lecture format over in-person lectures and a few disadvantages.

On the positive-side, remote screen-sharing proved superior to the traditional classroom overhead projector for lecture slides and live demonstrations. Students are able to see the material better and slides with finer details are possible. This was particularly helpful for this course because many of the lectures include demonstrations using the Linux shell or other applications which are best viewed at native resolution. Also, remote lectures are easily recorded and automatically posted to Canvas with minimal effort providing students the ability to re-watch lectures at their leisure which was not an option with the in-person sessions.

On the down-side, the remote sessions tended to be less interactive than the in-person meetings, partly due to the video conference format but also students were able to use the discussion boards outside of class very effectively to ask questions and receive answers. We found most students preferred asking questions on Slack where longer conversations could be held anytime of the day. Perhaps this points to a paradigm shift for the classroom.

Laboratory Project Guides

Each laboratory assignment includes a set of project guides that provide instructions, tutorials, and background information for the assignment. These project guides include all necessary information to complete the assignment while the lectures reinforce important concepts, adding additional details not necessarily required for the laboratory assignment but useful for student awareness.

The laboratory assignments are distributed using GitHub Classroom[11] (each student receives their own private GitHub repository for each assignment). The project guides were written in Markdown, the standard documentation format for GitHub. Students are able to make notes in the guides as they wish and save their updates in their GitHub repository for later reference.

The laboratory assignments are the students' primary work output for the course and the main graded assessments. All student work is captured in their GitHub repository and submitted by tagging the repository for assignment submission. A new repository is created for each assignment to simplify the organization and grading of the assignments.

The instructor, teaching assistants, and graders have access to student GitHub repositories because they are owned by the same GitHub Organization that the instructor has control over. Students do not have access to other students' repositories, although GitHub Classroom does have the option to create shared repositories for group projects. A new GitHub Organization is created each semester to simplify management.

The project guides for the first few assignments contain step-by-step instructions to complete required tasks such as setting up SSH keys or installing compiler tools. These operations are not complicated but difficult to remember if only done occasionally so students are encouraged to bookmark these instructions for later use in other courses or on the job.

Slack Discussion Boards

Slack is a web-based team messaging application that provides group message channels as well as direct messaging. The free-tier service (10k messages) was adequate for a one-semester course of this type.

A dedicated Slack Workspace was set up for the course to facilitate nearly all course communication, with E-mail and Canvas handling the remainder for specific purposes. The key factors for choosing Slack were the group messaging capability and the ease of use. Additional details about students usage of Slack is discussed in the Student Communication and Support section.

Canvas LMS

The university uses the Canvas LMS for nearly all offered courses so students are well-versed in using the system. For this course, Canvas was mainly used for announcements, assignment management, lecture recordings, grading rubrics, and grade tracking.

For assignment management, Canvas serves as the initial distribution point and main submission mechanism. Canvas sends students an assignment posting including the GitHub Classroom link and the grading rubric. Once finished with the assignment, students upload their video reports to Canvas, while all other work is committed to the GitHub repository.

The grading rubric mechanism Canvas provides simplifies grading by allowing the instructor to assign point amounts to various parts of the lab, then a grader simply has to check the work and assign points for each category. Video reports can be viewed directly from the Canvas website and grade tracking is automatically calculated per student.

Canvas has an optional Zoom integration that provides a calendar with Zoom links for scheduled lecture meeting times. The Zoom lectures are automatically recorded and posted on Canvas shortly after the class ends with little instructor effort needed.

Project Submission and Demonstration

The course originally planned for in-person project demonstrations plus a written report documenting students' work. This model has been used in previous laboratory courses successfully but after the campus closed, due to the pandemic, this model was no longer feasible. Presentations over Zoom were attempted but we found this to be ineffective since video conferencing capability greatly varied among students making it difficult and time-consuming to see what students were demonstrating.

We moved to a recorded laboratory video report that combined both the demonstration and the written report into a short video. This generated far better results with students putting more thought into both the demonstrations and the descriptions of their work. Minimal guidance was given in this first trial and, based on these first results, we compiled a best practices guide to set a minimum standard for future classes.

Some key best practices we found were:

- Keep reports concise and on topic to minimize the video length.
- Videos should be no more than 5-8 minutes (for these projects anything longer was repetitive and unnecessarily impacted grading times).
- When possible, use a screen recording app rather than an iPhone to take video of terminal operations so text can clearly be read.
- Group presentations are possible using a Zoom conference call with screen sharing.
- If an action will take a while, such as running a compile, use that time to talk about what is going on or about other aspects of the project. Avoid silence.

- Single take videos are preferable over many cut-jumps, unless the project has multiple parts where different setups are necessary.

Student Communication and Support

Student communication is important in a normal class setting and doubly so for a remotely delivered course. We made multiple communication methods available ensuring students had the necessary support for their project work while acclimating to the new distance learning model. We will briefly describe what we found worked well, what did not, and what was unnecessary.

The four main communication services available to students were the university E-mail system, the Canvas LMS, Slack discussion boards, and Zoom video conferencing. Each type of service has pros and cons depending on the mode of communication necessary to either deliver course content or to resolve student issues.

For course content, Zoom and Canvas were used as described in previous sections. Additionally, the E-mail system provided the initial course announcements, both before and during the first week of the semester, and as a backup for Canvas. Any announcements posted on Canvas were cross-posted on Slack to ensure student visibility. These were the typical usage modes which we found worked well.

Resolving student issues remotely posed a more challenging problem for which we tested a number solutions. Ideally, being able to respond to a student's problem immediately would be best but not always feasible. The two most important factors we found were timeliness of instructor issue responses and the ability for students to help each other solve common problems.

The best solution we found was setting up a Slack discussion board with channels for each laboratory assignment as well as a tech support channel for general problems. This was a little surprising because similar message boards were used previously for other in-person laboratory courses but with far less student engagement. This was likely due to remote students inability to socialize directly in the laboratory or between classes on campus. Many of the conversations about problems that would have normally occurred in-person instead happened on the message boards.

Having problem solving discussions take place in group messages instead of in-person conversations led to an important side benefit for the overall class. On the message board, all students could see and learn from the problem solving discussions rather than just the small group of students who happened to be present for the in-person conversation. The message board also broadened the instructor's perspective on the issues students were working through so minor issues could be corrected before becoming larger problems.

Direct messages on Slack also largely replaced e-mails when private conversations were needed between the instructor and a student (and presumably in between students themselves). This was mainly due to the convenience of having all communication for the course in one dedicated workspace rather than spread over many e-mail chains interspersed among messages from other courses.

Grading

Grading for the course was based primarily on the laboratory assignments. Students were evaluated on their project ideas, design work to implement the ideas, and presentation of the results in their video report.

Assignments and grading rubrics were managed through Canvas. Students are able to upload videos directly to Canvas for assignment submissions. They were also required to commit all code and other work files to their GitHub repository for each assignment.

Results

The pilot offering in spring semester 2020 had mixed results due to the large disruption caused by loosing access to the laboratory. Students did well in adapting but the fully envisioned course material could not be presented. We did find that many students struggled using Linux from the command-line so, in the second offering, more emphasis was placed on reviewing common Linux utilities and shell scripting in the first few lectures to bring students up to a functional level necessary for the laboratory assignments.

The revamped remote course in the fall semester 2020 was far more successful. The laboratory remote access scheme had very few problems and required minimal on-site instructor visits to keep the student projects running smoothly. Remote administration was possible to fix nearly all problems reported by students.

As gauged by student questions, activity in Slack, and project submissions, students remained engaged throughout the semester, many exceeding the project requirements because they were invested and passionate about their project ideas for the laboratories.

Video reports worked well and will likely remain the first choice even when in-person laboratory courses resume. We believe students more effectively presented their projects and conveyed their thought processes using the video format.

Here are a selection of applications students produced for their final projects. All projects used the DE2i-150 Development System as the main IoT Edge Device for the application and used either real-time data pulled over the network or simulated data to simply prove the concept.

- A home IoT Environmental Sensor Network using an IoT Edge Device to aggregate data before pushing it to a Cloud database and host a visualization web interface.
- A system to display artwork for the currently playing Spotify song with background lighting computation to adjust accent lighting to best match the artwork.
- A fall detection system for senior care centers with an alert messaging mechanism.
- Ocean vessel fleet tracking system with weather alerts based on vessel course.
- Stock market monitor with alerts based on AI prediction model.
- Campus shuttle bus estimated time-of-arrival system with notifications for when to leave the house to catch the bus with minimal wait-time based on a prediction model.

- Smart garden system using local sensor data and weather forecasts to make smart watering decisions and statistics collection to optimize growth patterns.

The course was evaluated based on students success in their final projects and on feedback received from student end-of-semester course surveys. These assessments have guided the direction of future work, detailed in the next section, to improve the course for the next iteration.

Directions for Future Work

We identified a number of course improvements for the next offering.

1. Improve remote lectures by pre-recording some content.

   Topics such as a detailed tutorial for a software tool or a recitation of review material could be pre-recorded for students to watch outside the classroom at their own pace leaving more lecture time for general discussion of more complicated topics.

2. Increase the amount of hardware interaction for students.

   We found generating interesting sensor data in an empty laboratory room quite difficult so are considering sensor kits for remote students for them to generate their own sensor data then have that data piped, in real-time, to the laboratory station.

3. Incorporate recently released AWS services into the laboratory assignments.

   Amazon is constantly improving their AWS service offering in the IoT space.

   - AWS Timestream was just released this past fall and provides better database technology for the time series data produced by most IoT sensors.
   - Demonstrations using AWS Greengrass (Lamda or "serverless" computing) were presented in lectures but integrating it into the laboratory assignments would give the students another look at moving resources to the Edge and could be coupled with the remote sensor kits.

4. Over-the-Air upgrades and configuration options for fleet deployments.

   These topics are mentioned in the lectures but adding laboratory exercises would improve the students appreciation for these complicated management issues.

5. Add additional homework assignments or quizzes to the two week labs.

   In the typical college student fashion, a two-week assignment means one week to procrastinate before rushing to finish late in the second week. Although our students were mostly better than that, there still needs to be some incentive to start on the lab earlier to prevent unforeseen issues, like network or power outages, from causing late submissions. Either a homework assignment for the first half of the two-week laboratory or a quiz covering the *technology introductions* may add motivation to start the laboratory sooner.

6. Develop protocols for students to work on team projects using the same remote laboratory stations.

   There are a few technical issues with multiple students trying to use the same laboratory workstation simultaneously which were avoided by having students work individually. Enabling multiple students to share laboratory stations (either with a time-sharing arrangement or through virtualization) would allow increasing the enrollment limit but these technical issues would first have to be solved.

Conclusions

We worked through many challenges in developing this course and created new methodologies for delivering educational content that we hope to utilize for other courses in the future. We believe our students learned a great deal throughout the course that will help them in their future engineering endeavors.

Acknowledgements

The authors would like to thank Intel Corporation for its generous support for the creation of this course, and the Institute for Smart, Secure and Connected Systems (ISSACS), and the Department of Electrical, Computer, and Systems Engineering at Case Western Reserve University for their support.

References

[1] Bansal S. and Kumar D. Iot ecosystem: A survey on devices, gateways, operating systems, middleware and communication. *International Journal of Wireless Information Networks*, 27:340–364, 2020.

[2] Asad Yousuf, Mohamad Mustafa, and Alberto De La Cruz. Project based learning. In *2010 Annual Conference & Exposition*, Louisville, Kentucky, June 2010. ASEE Conferences. doi: http://dx.doi.org/10.18260/1-2--16081. https://peer.asee.org/16081.

[3] Nicholas Barendt, Nigamanth Sridhar, and Kenneth A. Loparo. A new course for teaching internet of things: A practical, hands-on, and systems-level approach. In *2018 ASEE Annual Conference & Exposition*, Salt Lake City, Utah, June 2018. ASEE Conferences. doi: http://dx.doi.org/10.18260/1-2--29706. https://peer.asee.org/29706.

[4] K. Cao, Y. Liu, G. Meng, and Q. Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020. doi: http://dx.doi.org/10.1109/ACCESS.2020.2991734.

[5] Iot connected devices worldwide 2030, Jan 2021. URL `https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/`.

[6] De2i-150 development system, 2021. URL `http://de2i-150.terasic.com`.

[7] Matthew McConnell, Nicholas Barendt, and Kenneth A. Loparo. A framework for remote hardware lab course delivery - rapidly adjusting to 2020. In *2021 ASEE Annual Conference & Exposition*. ASEE Conferences, July 2021.

[8] Slack, 2021. URL https://slack.com.

[9] Canvas lms, 2021. URL https://www.instructure.com/canvas.

[10] Zoom, 2021. URL https://zoom.us.

[11] Github classroom, 2021. URL https://classroom.github.com.