ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #36959

Traffic Lights Engineering Academy: A Remote Online Education Solution for Creating K-12 STEM Projects Featuring Microcontrollers

Hechuan Wang

Hechuan Wang received his B.S. degree in Automation from Hefei University of Technology, China, in 2014 and received his Ph.D. in Electrical Engineering at Stony Brook University (SBU) in 2021. His research interests are primarily in statistical signal processing, machine learning, and their applications. During his Ph.D. study, he actively participated in the university outreach program, where he designed two remote STEM academies that taught Electrical Engineering to 6-8th grades online. He is now an Assistant Professor of Practice at SBU, working on the Outreach Program. His educational interest is in exposing young students to engineering according to the Next Generation Science Standards (NGSS) and converting the traditional in-lab experiments into remote activities offered online.

Kathleen Ann Dinota

Kathleen Dinota received her B.S. in Marine Science and M.S. in Secondary Education, retiring in 2017 after teaching in public schools on Long Island, NY for 31 years. During her career, she taught earth science, biology and chemistry as well as science research. Kathleen has also worked in test development at NY State Education Department as an Education Specialist for the Regents Physical Setting Chemistry exam for the past 20 years. She is a former NYS Master Teacher. Kathleen currently serves as the Engineering Education Project Director and Outreach Coordinator at Stony Brook University. Kathleen helps to develop engineering experiences for students from elementary to high school, ensuring alignment to state education standards and use of appropriate pedagogy and managing related logistics.

Monica Bugallo (Dr)

Traffic Lights Engineering Academy: A Remote Online Education Solution for Creating K-12 STEM Projects Featuring Microcontrollers

Abstract

Microcontroller programming is an essential part of K-12 Science, Technology, Engineering, and Mathematics (STEM) education. Experience with microcontroller programming is a gateway to many topics in this discipline, such as electrical engineering and programming. Hands-on experiences using microcontrollers are critical for student engagement and deeper understanding. However, as classes and field trips transitioned online due to the COVID-19 pandemic, educators encountered many difficulties adapting the microcontroller experiments to remote online education. One challenge is that traditional computer software for microcontroller experiments is not easy to set up. In remote education, students cannot be expected to install the software and do the configurations on their own computers at home. The second problem is that it is hard to monitor the students' progress and give feedback in real-time. Even though there are many online collaborative coding platforms, none of them support microcontrollers. In this paper, we introduce a comprehensive solution for remote education featuring microcontrollers. An online education platform was developed that allows the students to program the microcontroller using CircuitPython with no software installation or configuration. It also allows instructors to monitor students' work remotely in real-time. In addition, a microcontroller development board for experiments in which students apply programming knowledge to the function of traffic lights was designed. A CircuitPython module for the development board was also developed, which allowed the students to focus more on the logic of the traffic lights and less on potential hardware issues. This online education solution can also be adapted to meet different needs by designing different development boards for different scenarios, including breadboard experiments to focus on circuits, adopting more powerful microcontrollers for advanced programming, and a variety of other applications for use in differentiated instruction. The proposed traffic lights engineering academy was provided to a local school district and got positive feedback. The experiences and best practices are also discussed in this paper.

Introduction

Microcontroller education is an essential topic in K-12 STEM (Science, Technology, Engineering, and Mathematics) education. Microcontrollers are the core of robotics and control systems. It is also a combination of electrical engineering and programming. Applications of microcontrollers can also be used to introduce different science and engineering fields. Classes featuring microcon-

trollers encourage the students to have hands-on experiences, advance their problem-solving skills and to develop logic-based arguments.

Due to the Covid-19 pandemic, the need for remote education increased dramatically. Many schools adapted to conducting remote lessons or utilized a hybrid model of in-person and remote learning. While many schools continued to hold in-person classes, field trips to the universities were discontinued. Because most K-12 teachers are unfamiliar with microcontroller education, field trips and summer camps have traditionally been a common way to expose students to classes featuring microcontrollers. Thus, it became very important to find out a way to offer microcontroller education through remote classes. However, when trying to offer microcontroller classes remotely, many challenges exist.

Challenges in remote microcontroller education

The first challenge is in establishing the coding environment for microcontrollers. Usually, schools will have dedicated labs with computers set up by teachers or technicians who are very familiar with microcontroller development. That is because setting up the programming environment for microcontrollers requires knowledge of microcontrollers. However, in most remote education situations, students are using their own devices at home and cannot be expected to set that up without assistance.

Further compounding the problem is that students have different devices at their homes, running different operating systems. In practical class design, every student would use the same software, so that the instructor can help them out if they ran into any technical difficulties. However, in remote education, students are using their own devices, which means there are a variety of operating systems. The most popular ones are Windows, Mac, Linux, and Chromebooks. Software installation procedures are different for all of these devices. In some school districts, every student have a Chromebook, distributed by the school, which can resolve the problem of diverse devices. However, Chromebooks do not support most microcontroller programming software.

Although there are some efforts to program microcontrollers on the Chromebook, such as the Arduino online editor and MakeCode for micro:bit, there are still some challenges met while teaching the class. Students can only get meaningful help if the code is reviewed by the teacher instantly. Usually, that is not possible in remote classes, unless students are promptly sharing their screens and asking for help, which seldom happens. There are some cooperative coding education platforms, which help the teachers check the student's work in real-time, but, as far as we know, all of these products target coding classes only, and none support microcontrollers.

Other than software issues, microcontroller experiments are complex, involving many parts that can be problematic, such as circuit connections and choice of component values. These issues are further complicated by a remote instructional model as it is very difficult for the instructor to rectify any problems that may arise.

Contributions of this paper

In this paper, we provide a comprehensive solution to remote microcontroller education, including the software, experiment material, and class structure. Our solution allows students to work on

their own computers with zero-setup, and also helps the instructor to monitor the students' work in real-time. We provide an example of this solution in a remote course in which students create a small traffic light model. This solution can be extended in different ways to meet the need of students of different levels and interests.

Note that the goal of this paper is to discuss an optional strategy for microcontroller education that can be used remotely, and not to introduce a new educational framework. It was not conducted as part of a research study but as an effort to continue to provide engineering outreach programs, while in-person activities were prohibited.

Alignments with the science standards

Three of the NGSS Science and Engineering practices are "Developing and Using Models" and the "Use of Mathematics and Computational Thinking" and "Obtaining and Evaluating Information". This project actively uses all of these practices in its instruction and application. The Traffic Light PCB models an intersection where traffic and pedestrians may meet. Students must consider the civil engineering aspects of timing a traffic light at an intersection that would be most functional and provide the greatest level of safety for motorists and pedestrians when coding the timing of the light at the intersection. Students must consider the behavior of humans at an actual intersection in order to correctly create a functional model.

While mathematics has been a part of the middle school curriculum since the onset of formal education, the use of computational thinking is relatively new and is not commonplace in most math and science classrooms. Students "develop computational thinking when they approach a new situation with an awareness of the many ways that computers can help them visualize systems and solve problems" [1]. In this program, students are exposed to the use of a computer to effectively program a useful tool for public safety. "The use of digital tools to test and compare solutions to an engineering design problem" [2] is demonstrated as students use Python coding to program the traffic lights and as the instructor is able to provide real-time feedback to the students in a remote environment.

In our course, students were provided technical documents and were instructed on how to best find information in them in order to complete asynchronous assignments independently. This helped students to complete assignments without direct instruction from the instructor or TAs and encouraged independent experimentation and exploration.

The lessons presented in this course were aligned with the Middle School NGSS standards in the Engineering Design and Physical Science components. Specific standards include:

- **MS-ETS 1-1.** Define the criteria and constraints of a design problem with sufficient precision to ensure a successful solution, taking into account relevant scientific principles and potential impacts on people and the natural environment that may limit possible solutions.
- **MS-ETS 1-2.** Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.
- **MS-ETS 1-3.** Analyze data from tests to determine similarities and differences among several design solutions to identify the best characteristics of each that can be combined into a

new solution to better meet the criteria for success.

- **MS-ETS 1-4.** Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved.
- **MS-PS 4-1.** Use mathematical representations to describe a simple model for waves that includes how the amplitude of a wave is related to the energy in a wave.
- **MS-PS 4-2.** Develop and use a model to describe that waves are reflected, absorbed, or transmitted through various materials.

In addition, there is an explicit requirement in the NGSS to make connections to Engineering, Technology and Applications of Science. This Engineering Academy illustrates the influence of Science, Engineering and Technology on Society and the Natural World "Technologies extend the measurement, exploration, modeling, and computational capacity of scientific investigations" [2].

When considering engineering activities to be presented as extra-curricular or in-classroom programs, it is important to consider alignment to the NGSS as these standards continue to be widely adopted across the United States. Teachers have limited confidence in teaching engineering due to a lack of pre-service preparation [3]. Efforts must be made to support teachers and expose students to the engineering design process and critical thinking skills. At the university, programs can be developed and taught by engineering faculty and staff that can engage students with technology and foster engineering skills. This can be done in person or as a remote experience [4] [5].

Related solutions

When the Covid-19 pandemic forced the closure of so much of society, we were faced with a need to adapt our programs. As providers of extra-curricular activities and field trip opportunities in engineering education, we needed to find a way to offer our programs in a remote format, since outside groups were not permitted to visit our university. We transitioned our outreach programs to remote offerings and continued to find ways to engage students in electrical engineering and computer science.

At the time when all field trips were in-person, there were efforts to create microcontroller courses based on Arduino Uno [5]. As this education model requires the installation of Arduino IDE on the lab computers, it is not practical for remote education.

Some providers of remote learning opportunities have noted that microcontroller programming education is difficult in fully remote settings [6]. As a result, they switched to programming education such as software and website development, removing the focus on microcontrollers.

A variety of asynchronous learning methods create possible solutions, such as those offered by Arduino [7]. However, because these options do not include synchronous learning methods in conjunction, students lack the direction and assistance to learn the content systematically.

It is common for many providers of remote education to use off-the-shelf video conferencing and classroom management applications [8]. These apps are not optimized for microcontroller education and are not able to provide timely feedback to the students.

Another option for remote microcontroller education is to use a simulation platform such as TinkerCAD, which can provide timely feedback on students' work [4]. However, simulations are not vivid enough, especially simulations of sensors and motors. Remote Laboratory [9] is another option where the experiments are performed remotely, and the students can only see the results in the video stream, which is a little better than simulation, but it is sub-optimal if the students cannot operate the hardware directly.

Structure of the paper

In this paper, we first provide a general solution to remote microcontroller education which contains the core essential components for all classes under this category. Then we provide an example of a class in which students learn how to simulate a traffic light and conduct related experiments. The results and feedback are given in the following section, and we also discuss our experiences. Finally, we discuss other academies that we are currently working on and use them as examples to share how the proposed solution can be extended and adapted to fit different needs. The conclusion is given at the end of this paper.

Solution to remote microcontroller education

In this section, we provide a comprehensive solution for creating a remote class for microcontroller education. University, middle school, and high school teachers can create remote classes according to this solution.

Language: CircuitPython

Python is the most popular high-level programming language in the year 2021 [10]. Compared to other high-level programming languages, it is easy to learn and understand but powerful and wildly used. CircuitPython, which is developed by Adafruit Inc, is a special version of Python that can run on microcontrollers. When a supported microcontroller is connected to the computer by a USB cable, the microcontroller will behave like a USB drive with a Python script file. Results of running that Python script are sent to the computer by serial communication, which is also through the USB cable.

Unlike traditional microcontroller programming languages, such as C++ and Assembly, which will need the installation of a driver, a compiler, a code editor, and special hardware and software for uploading the binary files to the microcontroller, the requirement for programming CircuitPython is minimal. On the hardware side, other than the microcontroller itself, only a USB cable is needed. And on the software side, we only need a text editor for code editing and a serial console for displaying the result and debugging information. This software requirement can be satisfied with the CircuitPython Online IDE we developed.

Coding Environment: CircuitPython Online IDE

The software for microcontroller coding is called Integrated Development Environment (IDE). IDEs for remote microcontroller education need to have the following features:

- Zero-setup: there should not be any installation of software at all, and no special hardware is needed.
- High compatibility: the IDE should work on computers with any operating system, especially Chromebooks, and behave the same on all operating systems.
- Remote code monitoring: teachers should be able to see the code via the internet and thus able to give timely feedback.
- Clean and clear: the software should focus on basic functions and avoid overwhelming the students with too many details and professional functions.

Unfortunately, we were unable to find any IDE for microcontrollers that had all the features mentioned. We decided to develop such a coding platform by ourselves.

The education coding platform contains two parts: CircuitPython Online IDE, which is a browser-based IDE, and Code Monitor, which is a browser-based application that helps the teachers monitor the students' work. The CircuitPython Online IDE is able to run as a website on any computer with Chrome Browser, including the Chromebooks, which satisfies the requirements of zero-setup and high compatibility. In Figure 1, you can see the page of the Online IDE. And In Figure 2, you can see the page of the Code Monitor for the teachers.

Figure 1: CircuitPython Online IDE

Figure 2: Code Monitor

From the student's perspective, the first step is to connect the microcontroller to the computer by a USB cable. The microcontroller will appear as a USB drive with a Python code file named 'code.py' inside. Students will then access the CircuitPython Online IDE website and log in with their username. In the IDE, a student should first click on the 'connect' button to establish communication with the microcontroller. Then the student needs to click on 'open' button to open the Python script file on the microcontroller USB drive. The file content will appear inside the code editor on the left. After they edit the code, they can click on the 'save and run' button to run the code. This will also save the modified code to the microcontroller. The results will print out in the serial console on the right. They can also use the command window to send messages to the microcontroller.

In the past, we have found it very important to have multiple ways to monitor students' work in a remote educational model, because video conferencing offers less direct teacher-student interaction [5]. Especially in microcontroller coding experiments, instructors need an easy way to see the students' code to help the students in real-time. From the instructor's perspective, the teacher can open the Code Monitor, click on 'add student' to add students by their username. Students' names will appear as buttons on the top. The teacher can click on the buttons to monitor students' code, commands and also access their results in the serial console.

The entire educational platform, including CircuitPython Online IDE and the Code Monitor, is published on GitHub via copyleft license GPL3.0. In order to access and use it, please follow the instructions in the GitHub repository (https://github.com/urfdvw/CircuitPython-online-IDE-for-classes) and use it according to the terms in the license.

Experiment Circuits

Microcontroller experiments usually involve the construction of the circuit. Students of younger age usually have difficulties constructing the circuit, so for younger students, it is a better idea to provide a pre-built circuit by PCB board. Another option is to give them some circuit modules, and let them connect the modules instead of connecting the circuit components. Older students, who have better fine motor skills, can construct the circuit on the breadboards [5].

Traffic lights academy design

In this section, we provide one sample class design called Traffic Lights, developed using the solution proposed in the previous section.

Instructors and students

We, the instructors that created and taught this class, are faculty from the Electrical and Computer Engineering Department in Stony Brook university. We reached out to a group of students in 7^{th} and 8^{th} grades. They attended the class from home without assistance from their middle school teachers. These students had some experience with Python coding, so they knew basic Python syntax, and programming concepts such as variables, if-else, and loops. However, most of them did not have experience with microcontrollers. We designed the class difficulty and content to fit their level.

Experiment Kit

We decided to use Seeeduino Xiao development board as the core of the experiment kit, which is a minimum system of SAMD21 microcontroller. This development board is the perfect fit for our introductory experiments to microcontrollers. It is not expensive, because SAMD21 is an entry-level chip and there are not any unneeded on-board peripherals. However, it is still powerful enough to run CircuitPython programs and it supports many input/output functions such as General Purpose Input/Output (GPIO), Pulse Width Modulation (PWM), Inter-Integrated Circuit (I2C), and touch-buttons, which are needed for the traffic light experiments.

Because this is an entry-level class, we decided to have most of the circuits pre-built on the PCB board. The microcontroller board and the Organic Light-Emitting Diode (OLED) screen, which can be reused in other projects, are used as separable modules.

We developed a printed circuit board (PCB) to perform traffic light simulations and related experiments. On this experiment board, we included eight addressable RGB LEDs to simulate the lights on the traffic light, six touch buttons to mimic car sensors and pedestrian crossing buttons, an SSD1366 I2C OLED screen to display count down numbers, a buzzer to give sound feedback, and a photoresistor that detects the ambient light brightness. The PCB board with the microcontroller and OLED screen connected is shown in Figure 3.

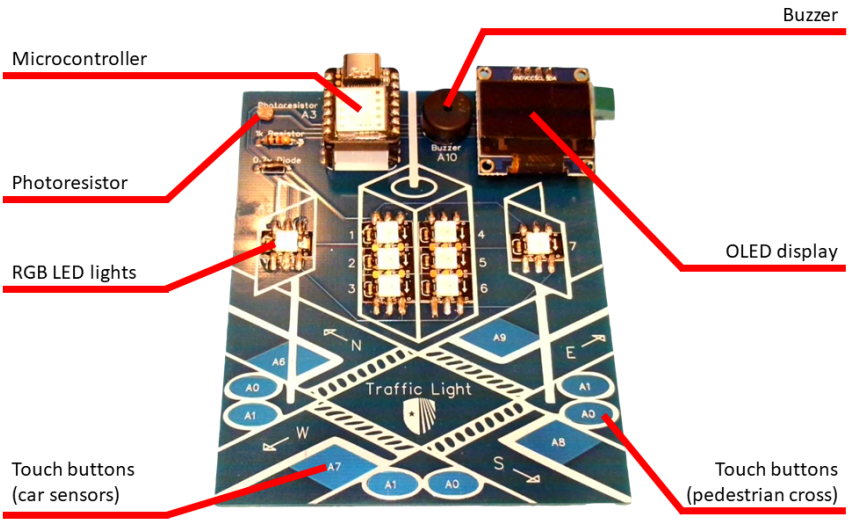


Figure 3: Traffic Light PCB

There are many native CircuitPython modules for the hardware on the traffic light PCB. However, some advanced coding concepts need to be understood for the students to use these modules, such as indexable objects and setter functions. These modules are more closely related to the hardware than to the behavior. To offer the camp to entry-level students, we wrote a series of modules that requires much less coding experience to use. With these modules, students were able to focus more on the behaviors of the circuits than hardware implementations. For example, instead of

writing code for the communication protocol for RGB LEDs, students could write the code as "leds.turn_on(ind=3, color=color.red)", which is much easier to understand.

We recognize that teaching students how to learn is more important than teaching content, so we also wrote a technical document for all the modules we developed and instructed them on how to use them to find the information they may need. We posted the documents for the modules on a website, so students are able to access these documents during and after the class.

Experiment planning

There are a great variety of experiments that can be done with the proposed setting. However, due to the limited time and the level of students, we chose the following experiments. These experiments either teach the students some knowledge about microcontroller programming, or are sufficiently interesting to engage the students, or both.

On the first day, we introduced the software and the hardware platform, including the PCB board, the Online IDE, and also the process of microcontroller development. This was achieved by a mini-lecture and two hands-on experiments. In the first experiment, students uploaded a pre-written Simon Game program to the microcontroller. Students practiced using the board and the online IDE and also get a sense of what microcontrollers are capable of doing. In the second experiment, we guided the students to print out 'Hello World' in the online IDE, and understand the relation between script and serial console outputs.

On the second day, we introduced Object-Oriented Programming (OOP) to the students. We also introduced additive color and RGB LEDs. Then, the students practiced OOP by blinking the RGB LEDs lights. After introducing more details of the RGB LEDs, students practiced OOP by writing codes simulating the red-yellow blink of a traffic light. Students were also taught to read technical documentation of Python modules in order to discover other functions of RGB LED independently.

On the third day, we discussed the relationship between sound and waveform. Students practiced this by writing code to produce different sounds with the buzzer. After they learned how to display text on the OLED display, they did a comprehensive project simulating the start signal of car race using RGB LED, buzzer, and OLED display.

On the fourth day, we reviewed how to use functions and loops in Python, and then created a function that will display a count-down on the OLED screen in conjunction with a buzzer beep. In the hands-on experiment, students used the countdown function to construct a fully-functional traffic light model, where three colors are used for the light signal, OLED is used for the countdown for the green signal, and buzzer beeps are made together with the count down.

Class organization

We organized the class in units of approximately 40 minutes. One day's activities can contain two units with a short break in the middle.

Each unit included a 15 to 20-minute mini-lecture and the remaining time was used for experiments. Some more complex experiments took the entire unit of time. However, we were careful to

ensure that the length of mini-lectures did not exceed 20 minutes, because students did not retain attention.

The lecture largely focused on two aspects. One was the knowledge that is necessary for finishing the experiments, such as circuits and code. The other aspect was some related knowledge that can be applied to many different areas, such as sound frequency, computer science concepts, and engineering fields. This was to give the students a bigger picture.

There were three kinds of experiments. The first kind was referred to as 'demonstration'. This kind of experiment was performed by the instructor. Because the demonstrations were performed step-by-step, students followed the instructor and simultaneously repeated the instructor independently. This kind of experiment enabled the students to learn new coding techniques.

The second experiment was referred to as simply 'experiment'. They were performed in Zoom breakout rooms. In each breakout room, there was a teaching assistant (TA) or instructor. The experiment was led by the TA by asking questions or giving instructions. Students answered the questions or tried to do the steps together with discussing with the TAs. Because the breakout rooms were smaller groups than the main room, the students were more willing to talk and engage in the experiments. This kind of experiment was our most-utilized form of the experiments, because it allowed the students to have hands-on experience and explore independently, while assisted and guided by experienced TAs.

The third kind of experiment was called 'task'. It was used in-class or after-class. Students were given a problem to solve, and some clues to follow, but they first needed to try to develop their own solution without the assistance of TAs. The objective was for students to practice what they have learned, and digest their new knowledge. The solution of the tasks was given and explained afterward. Students were also encouraged to present their solutions to the class and receive constructive criticism and assistance from their peers. Student confidence was increased by these interactions.

We realized that some of the students would have trouble during the class, especially in the experiments where they wrote their own code. However, because we had limited time for them in class, we also had office hours for the students before the class on each day. Students who came early could debug their code from the previous day and catch up with their peers.

Results and feedback

While the class consisted of a small number of students, the feedback was encouraging. The results of an anonymous feedback survey showed that all participants expressed a better understanding of and increased interest in either civil engineering, programming or circuits. While perception of the difficulty of concepts varied, most students felt that the level of instruction was appropriate and that they had a understanding of most of the concepts presented in the course. Students expressed an interest in learning more advanced concepts such as car sensors and a pedestrian crossing in any future classes to be offered.

Students were also asked to complete pre- and post-surveys to assess their background with Python coding and microcontrollers and to determine if the learning objectives for the class were achieved. The learning objectives were mastered by most of the class.

It is noteworthy that this course was offered in the evening in December at a time when the COVID variants were surging. Students still attended all classes and retained attention and engagement, despite the distractions of their daily life.

One student showed especially keen interest in the program and reached out to the instructor after the completion of the course to share his unique code for the traffic light and inquired about future offerings. Another student continued to work on the traffic light one month after the conclusion of the course, as evidenced by the CircuitPython Online IDE system log.

Insights and discussions

Prerequisites

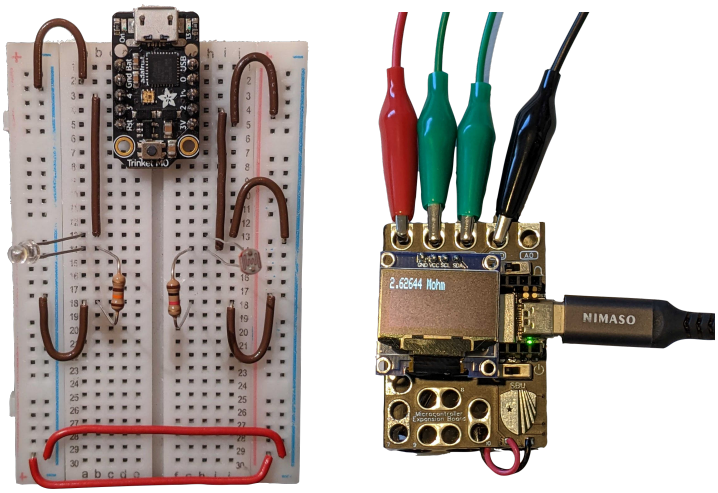
This camp focused on programming for microcontrollers, which means basic Python programming experience was required. However, we found that not many students had enough experience with coding, so we have the following suggestions. If this is a long-term program that contains multiple topics, this camp is better positioned immediately after an entry-level Python course. If this microcontroller course is offered as a stand-alone activity, the recruitment requirements should clearly mention a basic understanding of programming concepts. The instructor should still explain the Python programming details when demonstrating example code. Complete solutions to all coding experiments should also be given for students to follow. Although this is sub-optimal because the students did not come up with the code independently, they still get adequate hands-on experiences writing code.

The necessity of monitoring students' code

Monitoring students' work is necessary for remote teaching. When students' code does not work, they are mostly confused. As their debugging experience is limited, it is not easy for them to describe the issue to their instructor clearly. Furthermore, middle school students are reluctant to share their screens in a video conference because it might cause some privacy concerns. To solve this problem, in the monitor app we developed, instructors can see all information about the student's work, including the code, console returns, commands file status, and serial connection status, which will help the instructors identify the problem quickly. A collaborative editor could also be another solution, where the instructors can also edit the student's code directly. However, that is not the highest priority because it is crucial for the students to be able to identify code errors and correct them autonomously. The role of an effective instructor is to support students by helping them identify the code error and provide hints and clues to solutions.

Firewalls and blocks

Some schools have restrictions on the devices given to the students, which suppose to protect the students from unwanted content. However, these protections could fail the CircuitPython Online IDE. For example, some white-list systems will block websites that are not on the list. One of the schools we worked with also blocked access to USB drives on the student's Chromebooks which is needed when connecting the microcontroller to the computer. It is important to contact the school



(a) Smart Night Light (b) Sensor Interface

Figure 4: Circuits of other programs designed according to the the proposed solution.

district technical support before the program begins in order to unlock any restrictions placed by the district.

Extensions and future work

This paper provides a solution for the microcontroller curriculum and the *Traffic Lights* course is used as an example. The same solution can be used to create other microcontroller courses. At this time, we are also offering other microcontroller courses. In one course, called *Smart Night Light* (shown in Figure 4a), students built and programmed a night light that turns on in the dark with a light sensor and LED lights. In another course we developed, called *Sensor Interface* (shown in Figure 4b), students wrote code for experiments about a variety of sensor applications in daily life, engineering, and science. In this section, we use the *Smart Night Light* and *Sensor Interface* as examples to show different ways to extend the provided solution.

Different ways to construct the circuit

In the *Traffic Light* course, we used PCB boards and reusable hardware modules to simplify the circuit connection. However, there could be other ways to construct the circuit. In the *Smart Night Light* course, students built the circuit with a breadboard, which required the fine motor skills to put all components on the breadboard manually. This gave the students a better understanding of circuits, and how to use standard lab tools to perform electronic experiments.

Another popular way to connect the circuit is using alligator clips. This is an easy way to connect a small number of components to the microcontrollers, which is used on popular microcontroller development boards such as the *BBC micro:bit* and *Adafruit Circuit Playground*. In the *Sensor Interface* course, students used alligator clips to connect the sensors and resistors, which is very convenient.

Different focus

There are a variety of aspects of microcontrollers on which teachers can choose to focus their instruction.

In the *Traffic Lights* course, we focused on how to write programs to drive different circuits, which is suitable as an introduction to microcontrollers. In a more programming-focused class, teachers can use CircuitPython as the tool to teach Python programming concepts.

The use of microcontrollers is an essential topic in electrical engineering, and the experiments usually involve concepts like voltage, current, resistance, and capacitance. Microcontroller experiments can also focus on these concepts and include circuit construction, control, and measurement. For example, in the *Smart Night Light* course, we introduced Ohm's law, and also taught the students how to calculate resistance by writing code to solve a binary linear equation set.

A microcontroller course can also focus on its applications. Student interest in microcontrollers can be sparked by introducing the widespread use of microcontrollers in daily life, engineering, and science. We followed this idea in the *Sensor Interface* course and introduced many sensor-related experiments, such as screen rotation control, soil moisture measurement, and laser theft alarm.

Different difficulties

All the examples we provided above target middle school students. However, the model we propose is suitable to host more advanced topics to meet the needs of high school and even college students. In the 7^{th} grade version of the *Traffic Light* course, we were not able to use the full potential of our development board. For example, experiments about touch-buttons were not included. We are currently developing a high school version of the *Traffic Light* course, which will make full use of the hardware. Advanced programming topics can also be included in this high school version, such as event-driven programming and finite state machine.

Different educational models

The *Traffic Light* course we offered was offered as a completely remote class, where the students were at home and used their own home computers. However, the proposed solution can be used in all models of instruction. We offered the *Smart Night Light* course to a school as a remote field trip, where the students were in the classroom with their classroom teacher, but the university instructors conducted the program remotely. The solution can also be offered to hybrid classes where some of the students are in the classroom and some are accessing the class remotely. Even though in-person education is outside the range of this paper, it is still worth mentioning that the CircuitPython Online IDE we developed can simplify the setup process for in-person microcontroller classes.

Conclusion

In this paper, we propose a novel comprehensive solution for remote classes featuring microcontrollers, which are an important topic in STEM education. Our solution can be adapted for use in

remote instruction, as well as simplifying in-person instruction. We assert that this is a feasible solution and have provided an example class in the traffic light application. Students that participated in our program met with great success and were enthusiastically engaged in learning about microcontrollers. Feedback in the form of a brief, anonymous survey illustrates that students enjoyed the class and that the learning objectives were met by most students. We also discussed our experience and provided possible extensions for this program that can be used from middle to high school to teach students about microcontrollers. As a consequence of the Covid-19 pandemic, educators at all levels have had to shift their methods of providing instruction to include remote education. This has presented many challenges and has also created opportunities to attempt new approaches to engage students. This innovative solution is a breakthrough in remote education, which opens the door to more possibilities and opportunities.

References

[1] C. Sneider, C. Stephenson, B. Schafer, and L. Flick, "Computational thinking in high school science classrooms," *The Science Teacher*, vol. 81, no. 5, p. 53, 2014.

[2] NGSS Lead States, *Next generation science standards: for states, by states*, 2013, oCLC: 880438242. [Online]. Available: http://site.ebrary.com/id/10863742

[3] K. Christian, A. M. Kelly, M. Bugallo, and K. Sheppard, "University-based training of high school science teachers to implement the next generation science standards," in *Proceedings of the 2018 American Society for Engineering Education Annual Conference & Exposition*, 2018.

[4] Z. K. Stuart, M. Bugallo, K. Dinota, H. Wang, and A. Esposito, "A university-designed middle school remote summer engineering academy," in *2021 ASEE Virtual Annual Conference Content Access*, 2021.

[5] Z. Stuart, "University-designed middle school science experiences aligned with ngss," in *Proceedings of the 2020 American Society for Engineering Education Annual Conference & Exposition*, 2020.

[6] "Bringing innovative experiences in education," publisher: Arduino Education. [Online]. Available: https://docs.google.com/document/d/1UW5RuLpSCRuQf4mfTMVJHA9TjzYsn1VIpXxF0ayS9GU

[7] "Arduino Education." [Online]. Available: https://www.arduino.cc/education/remotelearning

[8] "Virtual learning during the pandemic," publisher: Arduino Education. [Online]. Available: https://docs.google.com/document/d/1InEtnWxDTrfbEnXZn9cL0LrpA4aGrzdSoVOsceHAUS4

[9] J. García-Zubía, I. Angulo, U. Hernandez, M. Castro, E. Sancristobal, P. Orduña, J. Irurzun, and J. R. de Garibay, "Easily integrable platform for the deployment of a remote laboratory for microcontrollers," in *IEEE EDUCON 2010 Conference*. IEEE, 2010, pp. 327–334.

[10] "index | TIOBE - The Software Quality Company." [Online]. Available: https://tiobe.com/tiobe-index/