

The Beauty and Joy of Physical Computing

*Deanna Gelosi
Dan Garcia, Ed.
Eric Paulos, Ed.*



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-156

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-156.html>

May 20, 2022

Copyright © 2022, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

A big thank you to the BJC Sparks team for your support with curriculum development: Brian Harvey, Lauren Mock, Mary Fries, Michael Ball, and Pamela Fox. And a special thank you to Dan Garcia for your encouragement throughout this Master's degree, and to Michelle Cheung for your activity development work over Summer 2021.

Thank you to my Tinkering Studio colleagues and friends at the Exploratorium: Anna Guardiola, Casey Federico, Claudia Caro, Ernest Aguayo, Jake Montano, Karen Wilkinson, Luigi Anzivino, Mike Petrich, Ryoko Matsumoto, Sebastian Martin, and Steph Muscat.

Last but not least, thank you to my family for your love and support through it all: Travis Uhrig and Maggie Gelosi, Valerie and Dean Uhrig, and Breana and Patrick Castonguay.

The Beauty and Joy of Physical Computing

by

Deanna Gelosi

A master's report submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Teaching Professor Dan Garcia, Chair

Professor Eric Paulos (reader)

Spring 2022

The master's report of Deanna Gelosi, titled The Beauty and Joy of Physical Computing, is approved:

| | | | |
|-------|-------|------|-------|
| Chair | _____ | Date | _____ |
| | _____ | Date | _____ |
| | _____ | Date | _____ |

University of California, Berkeley

The Beauty and Joy of Physical Computing

Copyright 2022
by
Deanna Gelosi

Abstract

The Beauty and Joy of Physical Computing

by

Deanna Gelosi

Master of Science in Computer Science

University of California, Berkeley

Teaching Professor Dan Garcia, Chair

This Master's report captures the creation and initial pilot feedback on a new computing unit **BJC Sparks** to introduce middle school students to hardware and physical computing through hands-on, tinkering projects. The goal of this curriculum was to design an open entry point for students to see themselves in computing through meaningful, engaging projects. The unit consists of eight labs and is one part of the new **BJC Sparks** curriculum, and was designed around one piece of technology: the micro:bit, a low-cost microcontroller created for educational use. Using the micro:bit, students can control an on-board LED array, servo motors, external LED lights, and more using input sensors such as tilt, light, and pin connections. In addition to the micro:bit, projects include use of craft materials and toys to encourage creative material use with technology. Five middle school CS teachers tested this unit in their middle school classrooms in Spring 2022. In feedback surveys, they shared that they appreciated that their students were able to be creative while also learning about hardware. Some also commented that they would have liked more structure through example projects and suggesting timings. This feedback will be incorporated into future iterations of this unit to best meet the needs of students from a variety of computing backgrounds.

Contents

| | |
|---|------------|
| Contents | i |
| List of Figures | ii |
| List of Tables | iii |
| 1 Introduction | 1 |
| 2 Prior Work | 2 |
| 2.1 The Beauty and Joy of Computing | 2 |
| 2.2 Middle School CS Curricula | 2 |
| 2.3 Making and Tinkering | 3 |
| 3 Background and Motivation | 4 |
| 3.1 AP CS Principles and Visual Programming | 4 |
| 3.2 Middle School Hardware Curriculum | 5 |
| 3.3 Tinkering with Technology | 7 |
| 4 Methodology | 8 |
| 4.1 Curriculum Development | 8 |
| 4.2 Hardware Curriculum Overview | 11 |
| 5 Results | 16 |
| 5.1 Teacher Feedback | 17 |
| 6 Conclusion | 20 |
| 7 Future Work | 21 |
| 8 Appendix A - Student Pages | 22 |
| 9 Appendix B - Teaching Guide | 52 |
| Bibliography | 67 |

List of Figures

| | | |
|-----|--|----|
| 3.1 | Overview of micro:bit hardware [16] | 6 |
| 3.2 | MicroBlocks programming environment | 7 |
| 4.1 | Sample LED display animation. | 11 |
| 4.2 | An interactive cat example project. | 12 |
| 4.3 | A simple homemade switch. | 13 |
| 4.4 | A BitBooster breakout board and servo motors. | 13 |
| 4.5 | An LED connects to a micro:bit with copper tape. | 14 |
| 4.6 | Two micro:bits can communicate via radio. | 14 |
| 4.7 | A multi-stage chain reaction. | 15 |
| 5.1 | Student project for Lab 3. | 16 |

List of Tables

| | | |
|-----|--|----|
| 4.1 | The number of students and location of each participant teacher. | 10 |
| 4.2 | BJC Sparks: Hardware labs and approximate timings. | 11 |

Acknowledgments

A big thank you to the BJC Sparks team for your support with curriculum development: Brian Harvey, Lauren Mock, Mary Fries, Michael Ball, and Pamela Fox. And a special thank you to Dan Garcia for your encouragement throughout this Master's degree, and to Michelle Cheung for your activity development work over Summer 2021.

Thank you to my Tinkering Studio colleagues and friends at the Exploratorium: Anna Guardiola, Casey Federico, Claudia Caro, Ernest Aguayo, Jake Montano, Karen Wilkinson, Luigi Anzivino, Mike Petrich, Ryoko Matsumoto, Sebastian Martin, and Steph Muscat.

Last but not least, thank you to my family for your love and support through it all: Travis Uhrig and Maggie Gelosi, Valerie and Dean Uhrig, and Breana and Patrick Castonguay.

Chapter 1

Introduction

BJC Sparks: Hardware is a physical computing unit designed for middle school students as an introductory experience into computing with a block-based environment. The foundation for the larger curriculum is the Beauty and Joy of Computing (BJC), an introductory computer science curriculum intended for non-CS majors at the high school junior through undergraduate freshman level [4].

The hardware curriculum introduces students to the micro:bit through hands-on, project-based activities. The BBC micro:bit is a low-cost microcontroller with on-board sensors, I/O pins, an LED array, and buttons used in a variety of educational technology settings. Every lab in this unit uses micro:bit and each lab is scaffolded to introduce students to new features and functionalities of the tool, including additional outputs like servo motors and LEDs. All of the projects have a physical making component and builds on the ideas of *constructionism*, stating that learning is most effective when learners actively construct tangibles in the real world [19].

The goal across this unit is for students to make—both through programming in Snap! as well as with physical materials like cardboard, craft supplies, and hot glue. Students start with activities that only use the micro:bit and are slowly introduced to additional components like servo motors and LEDs over subsequent weeks. Each activity prompt is open-ended for students to bring their unique interests and passions as project inspiration. The unit culminates in a large scale, collaborative chain reaction machine [9] where students can apply their understanding of programming many aspects of a micro:bit into practice.

In Spring 2022, five teachers shared Unit 3 with their students. Participating teachers were generous with their feedback. They shared what worked well and offered suggestions for improvement. While some complemented the curriculum for offering opportunities for students to express themselves creatively while learning about hardware, others found the open-endedness unmooring and asked for more structure. This feedback will be folded into future iterations of the curriculum to better serve students and their path toward computing creatively through tinkering.

Chapter 2

Prior Work

2.1 The Beauty and Joy of Computing

The Beauty and Joy of Computing (BJC) is an Advanced Placement CS Principles (AP CSP) course whose guiding principle is to meet every student where they are at in the computing education journey. The course is designed for high school and college students and covers computational thinking skills covered by AP CSP as well as advanced topics such as higher order functions and recursion. Every programming assignment uses Snap!, a block-based programming environment based on Scratch [3]. The course is designed around students having fun and experiencing that code itself can be beautiful.

BJC consists of a series of sequential labs that walk students step-by-step through new computer science content. This approach supports students tinkering with code before requiring them to formalize their knowledge. Daily *For You To Do* tasks offer a way for students to demonstrate what they've learned to peers or their teacher, and *Take It Further* tasks are a chance for students to go deeper with a particular topic.

BJC celebrates creativity, collaboration, and sharing work with one another. It offers opportunities for students to uplift one another rather than directly compete. Students take pride in demonstrating what they've created, and the act of sharing projects helps recruit future students [13].

2.2 Middle School CS Curricula

There exist a number of curricula for teaching computer science to middle school. **Code.org** created a free online curriculum designed for the middle school level that can be taught over the course of one semester or year called CS Discoveries [7]. The course maps to CSTA standards and includes a wide breadth topics, including problem solving, programming, physical computing, user-centered design, and data. Students learn while creating their own websites, apps, games, and physical computing devices.

Bootstrap teaches text-based functional programming through online materials [25]. Their emphasis is on how studying mathematics can be leveraged in learning about computing. Bootstrap offers lessons ranging from hour-long lessons through Hour of Code to Bootstrap: Reactive in which students take a deeper dive into programming to build more sophisticated programs. The design of their program supports non-CS teachers in delivering research-backed CS curriculum.

Alice is a block-based environment that allows students to create narratives, games, and animations in 3D [24]. Alice centers creative exploration for students as they learn logical and computational thinking skills, foundational principles in programming, and object-oriented programming. It supports a wide range of prior computing backgrounds through their lesson plans, including those who are learning programming for the first time.

2.3 Making and Tinkering

Making as a playful form of learning and inquiry aligns with the teachings of educators including Friedrich Froebel [12], Maria Montessori [18], and Seymour Papert [19]. Today, researchers see making as a way to engage learners in personally meaningful and creative projects, which positions them to be *producers* instead of *consumers* as they connect with their environment and community [5, 11].

Making and maker education can often fall into the pattern of offering step-by-step instructions to guide learners along a predefined path. While learners are making something, their path is limited. *Tinkering* is learner-centered and inquiry-driven pedagogy that takes the ideas of making one step further. The tinkering approach is defined as playful, experimental, and iterative, where learners can engage in an open-ended exploration [22].

Tinkering as an approach to making and learning draws upon *constructionist* theories of pedagogy and is based on a view of learning as the process of being, doing, knowing, and becoming [14]. Tinkering moves beyond traditional in-school knowing and traditional constructivist ideas of doing to include the importance of growth and identity through being and becoming [20].

The design for tinkerable experiences consists of three pillars: *environment*, *activity*, and *facilitation*. The environment is the place where the learning is taking place, which can vary from in-school contexts to out-of-school spaces like afterschool programs, libraries, or museums. Within these spaces, details like the table layout, where to access materials, and the flow of how people move throughout the space are up for consideration. The activity itself is also a designed experience, which includes the theme or subject matter, materials for making, tools and technologies, as well as expected outcomes. The facilitation is determined by the people who introduce and support the learning of tinkerers through asking questions, making suggestions, and guiding their learning path [20].

Chapter 3

Background and Motivation

BJC Sparks takes inspiration from the design principles and lessons from *The Beauty and Joy of Computing*, and is motivated by the goal to bring BJC to middle school students. The original BJC curriculum was inspired by Seymour Papert’s idea that programming is “hard fun.” It empowers students to see the creative and expressive qualities of programming, while mastering computing skills like recursion. Students of all backgrounds see themselves as programmers and see the power that computing has in their lives.

3.1 AP CS Principles and Visual Programming

The Beauty and Joy of Computing satisfies the requirements for AP Computer Science Principles (CSP) and is built around seven Big Ideas and six Computational Thinking Practices. The Big Ideas are things that students learn about computer science, including *abstraction*, *data and information*, *algorithms*, and *programming*. Students engage in programming deeply throughout the course with Snap!, a visual block based programming language, and learn about the importance of abstraction in programming. They develop their own algorithms and analyze existing ones. Additionally, students learn about *creativity*, *the internet*, and *the global impact* of computing. Students exercise creativity through projects, and global impact is introduced through readings and classroom discussion.

Computational thinking is a practice that benefits not only computer scientists and engineers, but is a fundamental skill for everyone [29]. *Connecting computing* means making connections between computing and other areas of students lives, like hobbies, possible future careers, and social impacts of computing. *Computational artifacts* are the videos, slide decks, blogs, programs, music, spreadsheets, anything that can be created with a computer, that students engage in making. *Abstraction* is core to programming, and students learn how to structure their programming projects using layers of abstraction. *Analyzing programs* includes debugging, predicting code behavior, and thinking about efficiency, all serving the goal of creating programs that work. *Communication* and *collaboration* is supported through pair programming and classroom discussions on the social implications of computing.

All of BJC is taught using Snap!, a visual, block based programming language based on MIT’s Scratch. Snap! provides tools for students to create their own abstractions like functions and explore CS ideas like recursion, while the visual representation of blocks makes the programmatic procedure more concrete for students.

BJC Sparks

BJC Sparks began as the middle school equivalent to BJC. Students in grades 6-8 would be introduced to functional programming through Snap! and prepare them for future computer science courses in high school and beyond. However, this idea evolved over time to be more expansively redesigned. First, teachers noted that they were interested in offering this middle school curriculum to students in grades 9-11 who were not quite ready for AP CSP. Since writing curriculum for a pre-AP CSP audience was the mission for this team, it was decided to rename the series. BJC Sparks was chosen because the curriculum is “sparking” or igniting interest in computing for students.

There is some overlap between AP CSP and BJC Sparks, specifically in the basics. We recognise that we should revise AP CSP so that a student who takes BJC Sparks does not repeat the same material. One way to address this is to amend the BJC Teacher Guide to identify which labs and activities students can skip if they learned BJC Sparks.

3.2 Middle School Hardware Curriculum

Teachers today have a variety of physical computing curricular options. Birdbrain Technologies created teaching guides to support the use of their Hummingbird and Finch Robotics kits [6]. Scratch partnered with the Harvard School of Education to design creative computing curriculum that supports a wide variety of projects, though does not currently support physical computing [8]. Scratch does offer coding cards for micro:bit, though they are designed as example projects to try instead of a complete curriculum [10]. Other online platforms offer micro-curriculum like Tinkercad [1] and MakeCode [15], though their projects are designed to supplement other CS curricula. Tinkercademy [27] offers a kit for sale that pairs with their curriculum, but also offers supplemental support from MakeCode and Instructables.

micro:bit and MicroBlocks

The BBC micro:bit fig. 3.1 is an affordable microcontroller designed as an educational technology for physical computing. The compact board features an LED light display, buttons, sensors and I/O options that can be programmed using a variety of web-based block-based programming environments. The new micro:bit (V2) also includes a built-in microphone, speaker, touch input button, and a power button.

MicroBlocks fig. 3.2 is a free, block-based programming environment for learning physical computing with educational microcontroller boards such as the micro:bit [17]. Inspired

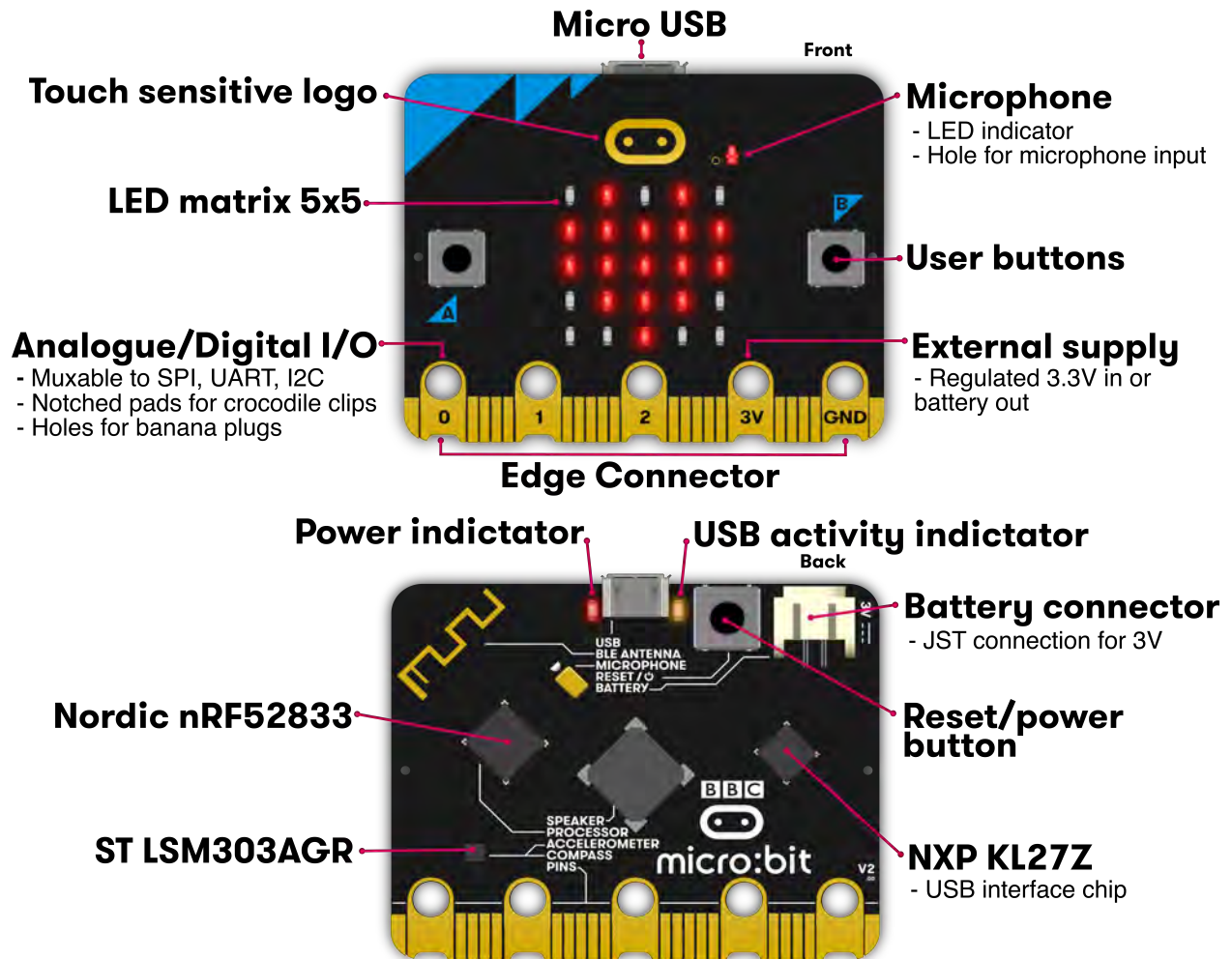


Figure 3.1: Overview of micro:bit hardware [16]

by Scratch [23] and from the developers of Snap!, MicroBlocks features both interactive programming and autonomous operation. Interactive programming allows for live edits to code and the ability to immediately see the changes. In contrast, Microsoft MakeCode [15] requires building and uploading the program before seeing changes. Autonomous operation allows for the board to run independently without the need for the programming environment open and running on the computer or for the board to be plugged into the computer. In Scratch, micro:bit projects only function when the program is live on the computer.

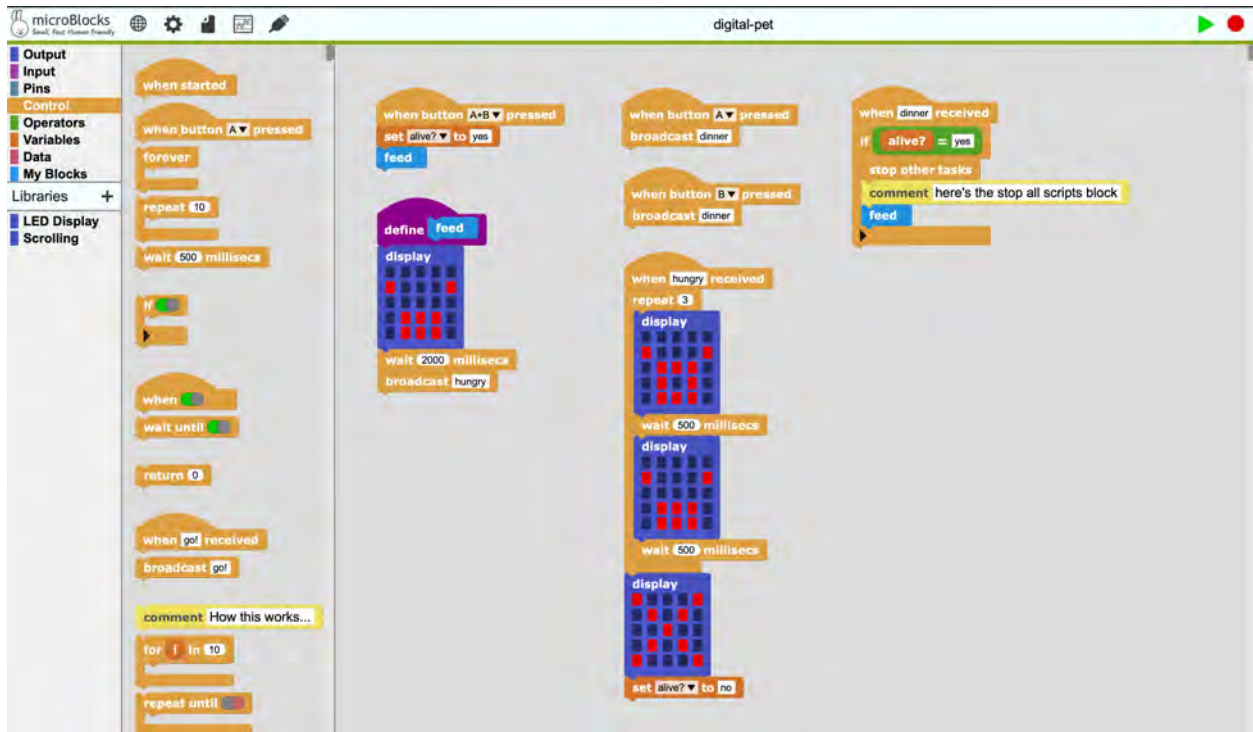


Figure 3.2: MicroBlocks programming environment

3.3 Tinkering with Technology

To inform the creation of a hardware unit for BJC Sparks, I drew upon my background as an activity designer and developer at the Exploratorium’s Tinkering Studio in San Francisco. Our work in the Tinkering Studio has parallels to that of BJC: we create rich environments for tinkering with phenomena that center learner agency and creativity. We design experiences that are rich in science, technology, engineering, art, and mathematics (STEAM) using familiar tools and everyday materials. This work happens in the museum the Exploratorium, in the community through the Boys and Girls Club of San Francisco, and through collaborations within both research and practice, both in the United States as well as abroad.

One strand of work is *computational tinkering*, a hands-on, materials-rich approach to exploring programming and technology where code becomes a material for making. We also use block-based programming environments and emphasize the ways physical materials can connect to and support digital tools and environments. One technology we used was the micro:bit, which is where the inspiration for BJC Sparks first originated. It easily connects to block-based programming environments, can be functional with just a few blocks, and also affords the opportunity for rich complexity.

Chapter 4

Methodology

BJC Sparks: Hardware is designed for teachers across the country and world. The hardware curriculum was created in 2021-22 and was piloted with teachers during the 2021-22 academic year.

4.1 Curriculum Development

Fall 2020: Hardware Scoping

A variety of hardware was considered for the middle school curriculum. Other strong contenders for the unit featured robotics with BirdBrain Technologies wireless Finch (2.0) and micro:bit, internet of things with Plezmo, programmable sewing machine with Turtlestitch, and programmable 3D printing with Beetle Blocks [26, 21, 28, 2]. This range of tools and technologies fits nicely into any makerspace and would be a wonderful introduction to the breadth and depth that physical computing offers .

The reason we chose to exclusively feature micro:bit as the core technology is twofold. The first is the accessibility of the curriculum. A micro:bit is a low-cost microcontroller at less than fifteen dollars a board. This brings the cost-per-student for the curriculum down substantially when compared with purchasing large pieces of equipment like 3D printers, programmable sewing machines, and robotics kits. More schools and students would be able to have access to micro:bits than expensive fabrication equipment and robotics. Some schools may also be able to purchase one micro:bit per student and allow them to keep them afterwards.

The second was the depth of understanding. By focusing solely on micro:bit, students are able to learn many aspects of the tool and then culminate in a final project (Lab 7: Collaborative Chain Reaction) that celebrates the many things that the microcontroller can do. It would be challenging to go into as much depth while jumping between four or five tools over the course of a semester or quarter.

Spring 2021: Software Scoping

After selecting micro:bit as the hardware, the choice of what software to use was the next task. Snap! was the obvious and preferred choice since BJC is designed exclusively around using the block-based environment. However, Snap! does not currently offer key features that would be necessary for the hardware unit. The first is that Snap! requires tethering of the micro:bit to the computer at all times. Untethered micro:bits allow for taking a Tamagotchi digital pet on the go or making a more complex chain reaction machine without worrying about laptops cluttering the space. Another important feature is how the blocks in the visual programming environment are designed. Snap!'s blocks prioritize screen-based interactions and do not have the focus on hardware that's needed for a physical computing curriculum.

MicroBlocks is a hardware-first version of Snap! that prioritizes features like untethering the micro:bit from the computer and designing blocks specifically for hardware. For instance, it has a block that looks like the micro:bit screen so students can click individual LEDs to turn them on or off. MicroBlocks does not have all of the features of Snap!, including cloud storage of projects, a digital stage for making computer-based animations, or the ability to create higher order functions. However, these lacking features do not outweigh the advantages of designing a curriculum exclusively around MicroBlocks.

A third software solution was introduced as a viable alternative in spring 2022. Project ExCITE (Exploring Computation Integrated into Technology and Engineering) teaches BJC AP CSP curriculum along with computer control and robotic activities. The introduction of this new option came late in the hardware curriculum development timeline and was not fully explored as an alternative to MicroBlocks, but their block-based programming language is at least comparable to MicroBlocks.

Summer 2021: Curricular Development Sprint

The hardware curriculum team had a three-month sprint with developing labs and activities during the summer of 2021. An undergraduate researcher joined in the efforts and prototyped activity introductions and projects. The curriculum development continued into the fall and spring semesters.

During the same summer, the BJC Sparks team also lead a week-long PD for the participating teachers to introduce and prepare them to teach Unit 1 starting in Fall 2021.

Fall 2021 - Spring 2022: Pilot Program with Teachers

BJC Sparks was piloted during the 2021-22 academic year. Twenty teachers from around the world participated in the pilot in the fall of 2021 with Unit 1: Functions and Data. The original plan was to cover units 1-2 during the 9-month academic year. However, due to the active nature of curriculum development alongside teachers piloting the curriculum, Unit 1 extended past the fall semester into the beginning of the spring. Given it was a priority to

| Teacher initials | Total number of students | Location |
|------------------|--------------------------|-----------------------|
| DB | 90 | Lafayette, CA |
| LH | 12 | Shenandoah, PA |
| BK | 14 | Pacific Palisades, CA |
| DD | 61 | Fremont, CA |
| ZA | 14 | Oakland, CA |

Table 4.1: The number of students and location of each participant teacher.

pilot Unit 3 in Spring 2022, the team broke Unit 2: Sequencing and Iteration into two parts: 2a and 2b. The former included labs that were important for students to complete before Unit 3, and the latter were labs that could be completed afterwards.

For the pilot of the hardware unit, five of the original twenty teachers were able to teach Unit 3 in Spring 2022. The participating teachers are at middle schools across the country, though they are mostly concentrated in California (table 4.1). They work with a wide variety of students, ranging from 14 to 90 students.

In January 2022, we hosted a two-hour professional development (PD) workshop for all five teachers. To introduce and prepare them for the hardware unit, two curriculum developers walked through the logistics of the second half of the pilot year and then taught Lab 1. Lab 1 introduces students to micro:bit, so the PD workshop allowed teachers to also be learners and walk through some of the challenges their students might face in connecting their hardware to the computer and running their first commands.

| Lab Number and Name | Timing range |
|-------------------------------------|-------------------|
| Lab 1: Meet micro:bit | 1-2 class periods |
| Lab 2: Interactive Pet | 1-2 class periods |
| Lab 3: Game Play | 2-3 class periods |
| Lab 4: Marvelous Movements | 2-4 class periods |
| Lab 5: Paper Stories | 2-4 class periods |
| Lab 6: Making with Multiples | 2-4 class periods |
| Lab 7: Collaborative Chain Reaction | 3-5 class periods |

Table 4.2: BJC Sparks: Hardware labs and approximate timings.

4.2 Hardware Curriculum Overview

BJC Sparks: Hardware (Unit 3) consists of seven labs (table 4.2), each one introducing a new feature of the micro:bit and culminating in a collaborative chain reaction machine. Each lab includes a project for students to make both with physical materials and with code. Activities within labs are open-ended and offer many possible directions for students to bring in their own personal interests to projects.

Lab 1: Meet micro:bit

The first micro:bit experience is to make an animation on the LED screen. Students are introduced to the core blocks in MicroBlocks and learn how to make an animation by switching between at least two different images. Animations can be as simple (e.g. a two-frame animation, see fig. 4.1) but also increase in complexity when introducing new inputs like light levels.

One big idea with Lab 1 is that students walk away with an understanding of the key fundamentals of their software and hardware. They know how MicroBlocks works, how to connect their micro:bit to MicroBlocks, and how to save a project locally (unlike Snap! which saves projects on the cloud). Students also learn now to make an animation as a series of frames. This can be as simple as two frames alternating back-and-forth, or a more complex series of frames. The most important take-away experience for students with this lab is that they have an early success with MicroBlocks and see themselves as capable of using the software to interact with their micro:bit.

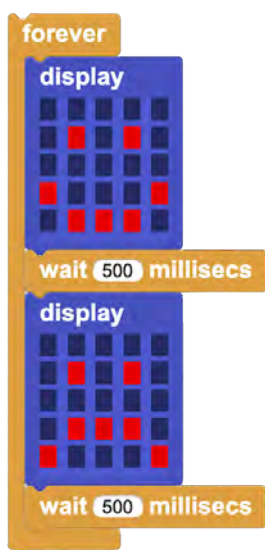


Figure 4.1: Sample LED display animation.

Lab 2: Interactive Pet

A continuation on animations from Lab 1, students make an interactive pet using the LED screen and various inputs from the micro:bit. The lab consists of four activities: designing, building, adding interactivity, and storytelling. In the first activity, students start by making a plan for what type of interactive pet they want to make. While this is a good practice for physical computing in general, crafting and carrying out a plan that combines software and hardware also aligns with *CSTA Standard 2-CS-2: design projects that combine hardware and software components to collect and exchange data*. The activity supports students planning process by encouraging them to sketch their ideas on paper and to be specific about their inputs and outputs.

The second and third activities focus on the physical and digital aspects of making an interactive pet. *Physical making* includes using everyday craft materials to construct a holder for the micro:bit in the shape of the interactive pet. Example ideas include making a shape like an animal (fig. 4.2) or a digital pet like a Tamagotchi.

The final activity supports students in sharing their pets to their classmates and beyond. This storytelling activity also ties in key computer science principles of documenting projects in such a way that others can use their creations as described in *CSTA Standard 2-AP-19: document programs in order to make them easier to follow, test, and debug*. Students explain to other users how to interact with their pet by identifying its inputs and outputs.



Figure 4.2: An interactive cat example project.

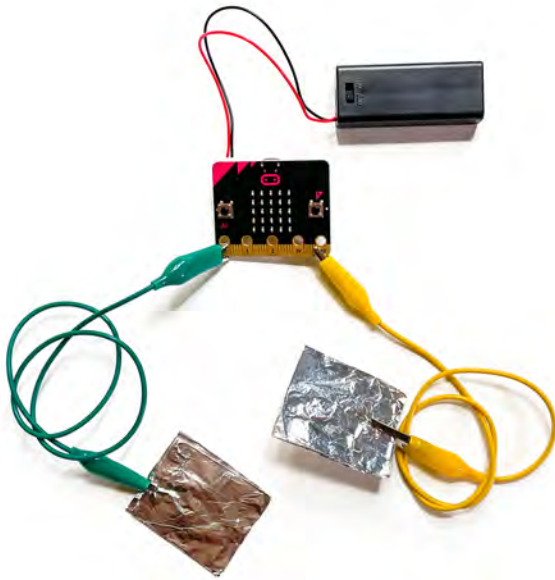


Figure 4.3: A simple homemade switch.

Lab 3: Game Play

In this lab, students create their own games collaboratively with a partner. The game can be single- or multi-player and will use some combination of inputs and outputs introduced in Lab 2. The lab is split across three activities that invite students to brainstorm project ideas collaboratively, build the physical and digital components of their game, and then play test their games with classmates for feedback and iteration.

Students have the option to make a game that exists entirely on the micro:bit board, or a physical game that could use the board's I/O pins. To accomplish this, they are introduced to homemade switches (fig. 4.3) and use the `read analog pin` block to determine whether the switch is open or closed. Both styles of games also include variables as a means to track and share the current score of the game.

Lab 4: Marvelous Movements

Students learn now to control servo motors to create a precise moving contraption. This can be a game or something else that moves in an interesting way. This lab prepares students for their collaborative chain reaction machines (Lab 7), where precise movements could be very useful for continuing the momentum of their machines.

While the micro:bit can run on two AA batteries, using additional outputs like servos requires more power than these batteries can provide. The BitBooster breakout board (fig. 4.4) uses two AA batteries to control servos and sometimes the board also needs to be powered through its USB micro cable.

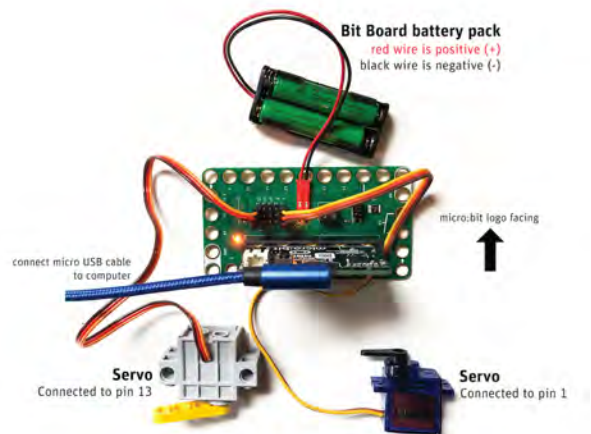


Figure 4.4: A BitBooster breakout board and servo motors.

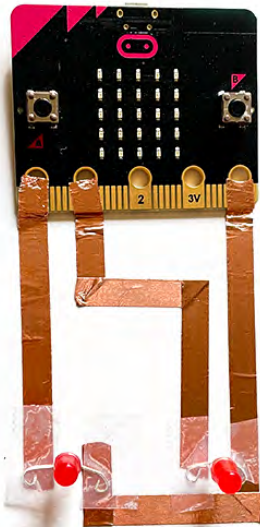


Figure 4.5: An LED connects to a micro:bit with copper tape.

Lab 5: Paper Stories

Copper tape is featured as a conductive material for circuit making. Students construct circuits to turn on 5mm LEDs by laying down conductive tape onto paper and wiring these connections to a micro:bit. Those who made homemade switches in Lab 3 may find the programming familiar since this method also uses the micro:bit's I/O pins.

Planning and troubleshooting is of utmost importance with this lab. Students design their circuits to be able to power more than one LED (fig. 4.5). They are careful that no wires cross unintentionally and troubleshoot any challenges with code as well as the physical materials. This process aligns with *CSTA Standard 2-CS-03: systematically identify and fix problems with computing devices and their components.*

One takeaway from this lab is for students to be introduced to unusual yet familiar conductive materials that can be used for making circuits. For some, this may be the first time experiencing making circuits using non-traditional circuit materials. This is also an opportunity to explore paper craft as engineering, and construct paper mechanisms like pop-ups and switches using materials on hand.

Lab 6: Making with Multiples

The penultimate lab is a collaborative project in which students learn about radio communication. Two micro:bits can wirelessly communicate with one another by sending strings and integers via radio (fig. 4.6). Students can use what they've learned over the last five labs and use radio communication to add complexity to their servo, LED, and micro:bit sensor projects.

One important detail is defining what radio channel to use for communication. In a classroom full of project partners, defining radio channels for each group will be crucial.

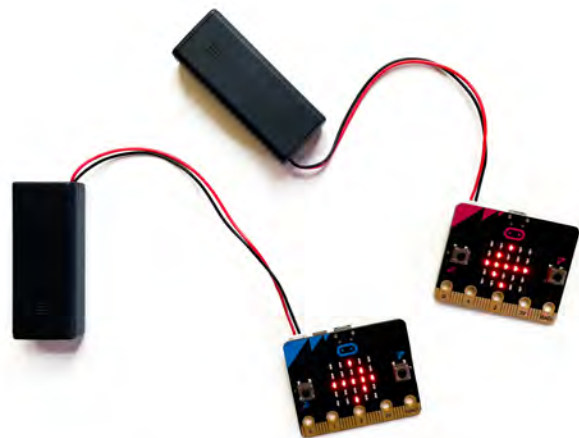


Figure 4.6: Two micro:bits can communicate via radio.



Figure 4.7: A multi-stage chain reaction.

Lab 7: Collaborative Chain Reaction

The final lab is a celebration culminating in the variety of ways to use micro:bit in combination with physical materials. Together, students build a collaborative chain reaction machine in which pairs of students work on segments that link-up together and create one continuous cause-and-effect contraption. Student pairs use both of their micro:bits in their section alongside any analog materials and techniques of their choosing. This event can be shared with other students, friends, or family and is a way of demonstrating all that has been learned through this unit.

Chapter 5

Results

“I’m noticing that to keep the middle school students engaged, you need to show them a new technique / blocks / functions of using the micro:bit at the beginning of every class. I’m seeing a lot of success in running unit 3 because of the plan in covering one lab per week. The students are excited to see something new and are going back to their previous week’s projects and incorporating a new functionality or their eager to begin another unique project.”

- DD, participating teacher

In Fall 2021, twenty middle school computer science teachers joined the BJC Sparks pilot program. In Spring 2022, only five teachers remained to implement Unit 3. This attrition is due to the challenges of teaching at the middle school level. It’s rare to have the same group of students for the entire academic year, so teachers who taught Unit 1 in the fall were not able to work with the same group of students in the spring. Only five teachers taught the same group of students since Fall 2021.

At the time of writing this report, four of the five teachers indicated that they had started teaching Unit 3 to their classes. All of the teachers had taught Lab 1 (the lab that was introduced to them during the January PD workshop). One teacher (DD) taught up through Lab 4 and one taught up to Lab 3 (LH) (see fig. 5.1 for one of the student projects). When given a survey to give feedback on what labs they had taught, two of the five teachers completed a portion of the questionnaire. The following feedback is a summary of their takeaway thus far with the curriculum.



Figure 5.1: Student project for Lab 3.

5.1 Teacher Feedback

We collected pre- and post-surveys from teachers who completed the hardware unit. Before teaching Unit 3, teachers average rating for comfort teaching with micro:bit as 5.8 out of 10. After teaching Unit 3, the score increased to 7.8. Due to the small sample size, these results only indicate that by working with micro:bit, teachers became more familiar and consequently more comfortable teaching with the hardware.

Timing

Teachers commented that the suggested timing did not always align with what they needed (see table 4.2 for original timings). In some cases, the activities took less time than anticipated, and in others, they took longer. Their first-hand experiences teaching these activities will help the curriculum development team refine their suggested timings. Of the two respondents, one teacher (DD) had students weekly and taught the entire lab in one ninety minute session. Students were able to complete the core activity but did not have time to explore other directions or tangents. The other teacher (ZA) had their students for a longer amount of time and self-reported that their students were able to complete optional tasks and projects.

Specifically, DD noted that they covered Labs 1 and 2 in one class and Labs 3 and 4 took one class per lab. They noted that, “*Some students want to spend more time on each lab while others are eager to move forward. Covering the concepts and learning objectives of the lab in the first 20 minutes of class is beneficial to all and then letting the students create new projects or continue a big project keeps the students fully engaged in their learning.*” This pacing is a bit faster than predicted, but it’s notable that DD is teaching in an after school setting and that packaging each lab per weekly session is quite beneficial to the overall pacing of the program. It would be much harder to pick up a project after putting it down one week prior.

Alternatively, ZA said their students spent much longer on Lab 2, noting that the “*Making the pet took a little longer, but that was more of students wanting to make their project perfect and they were happy to get to some hands on activities so I let them work a little longer on it.*” While it’s not ideal that the lab timing didn’t match up with reality, it’s wonderful feedback to hear that students wanted to continue working on the project they had started.

Student Successes and Challenges

Participating teachers shared both the successes and challenges for students in the pilot version of Unit 3. DD noted that they appreciated that the activities were short and allowed time to cover basic computing concepts and learning objectives as well as time to build their own artifacts. DD also shared that the unit was very successful with their SPED (special education) student, sharing that “with a little help was able to successfully complete the interactive pet activity.” ZA also commented that they appreciated that students enjoyed

being creative with Unit 3 while also learning how to use hardware. This feedback is met with great enthusiasm, since the hardware unit was designed to meet a wide variety of programming backgrounds and personal interests.

One piece of feedback shared informally in an email was that some of the labs did not meet the interests of some students. BK noticed that Lab 2 (Interactive Pet) was “too crafty” and that their students would hopefully enjoy Lab 3 (Game Play) more than Lab 2. This feedback is met with mixed reactions. While it is important that there are ways for students to meaningfully engage in exploring the micro:bit board’s input sensors and LED screen, craftiness is seen as an asset by the curriculum team and not as something that would disengage students. Further conversation would be required to unpack how students engaged in the project and if there are opportunities to expand the definition of what an interactive pet could look like to meet student interests.

The most prominent challenge that DD identified was the inconsistency of the homemade circuit in Lab 3 (Game Play). “It was a little buggy for some student[s]” they noted in their feedback, “so they didn’t enjoy it as much but it could have also been due the the limited time we had to work on the games”. This is a known issue with homemade switches, and could be addressed with a dedicated troubleshooting guide. BK also noted that their students needed more support in adding sound files to MicroBlocks. Another challenge was students had trouble thinking of Lab 3 projects that could incorporate a micro:bit. DD noted that showcasing more example projects could help students see different game possibilities.

Hardware and Software

We asked in pre- and post-surveys for teachers to self-report their comfort using micro:bit. Before teaching Unit 3, the average self-reported score was 5.8 on a scale of 1-10 across all five teachers. After teaching some of Unit 3, the average score increased to 7.5 for the two teachers who completed the survey. The increase in average score is not reflective of all five participating teachers since we have not yet collected their scores, but it’s still notable that there was a noticeable increase among the two teachers.

One teacher asked whether Unit 3 could include additional sensors, such as an ultrasonic sensor, for measuring distance. Additional sensors could become part of the students’ extended toolkit for building more complex contraptions into their projects. They could also be the jumping-off point for a subsequent hardware unit.

We also asked teachers to comment on Snap! and MicroBlocks. For Units 1 and 2a, teachers used Snap! and Unit 3 used MicroBlocks. ZA and DD shared that MicroBlocks was easy to use and program and that the environment overall environment was satisfactory. When asked if they would use Snap! over MicroBlocks if given the option, both teachers responded that they would like to use Snap! in the future. DD said that they would use Snap! so students could save projects to their accounts rather than to the computers themselves. MicroBlocks does not support cloud saving and this is a known difference between the two environments. ZA commented that students would be able to better build off their prior knowledge by continuing with Snap! and add their new hardware knowledge to their toolkit.

They would also have access to live data using *Snap!* and could program their micro:bit based off this live data. *Snap!* affords students access to higher-order functions and live data, which could sustain and deepen their programming understanding.

Chapter 6

Conclusion

The BJC Sparks Hardware Unit (Unit 3) was developed to complement the function-first curriculum for pre-AP CSP students in middle school and early high school. The curriculum design principles included centering affordable hardware, hands-on learning, and simultaneous physical and digital making. The unit was created to work with a variety of student programming backgrounds and offer possible extensions to continue learning. The unit used micro:bit, an affordable piece of hardware, as the core educational technology. The BBC micro:bit features a variety of on-board inputs and can be augmented to use external outputs. In concert with micro:bit, a variety of readily-available craft supplies and materials were used for students to build creative and personally-meaningful projects.

In Spring 2022, five BJC Sparks teachers participated in a pilot program to test out Unit 3 with their classes. During the pilot, students used the Unit 3 labs and explored micro:bit across seven different experiences. Teachers shared feedback about the curriculum and noted both the successful aspects as well as areas for improvement. They appreciated the creativity students showcased while making and learning hardware concepts and the approachability of the activities. Challenges identified included troubleshooting I/O pins and switches, activity length and timing, and finding ways for all students to engage in physical making. BJC Sparks is still under development and will take this feedback into account for future iterations.

Chapter 7

Future Work

BJC Sparks is still under development and will continue to pilot their curriculum for teachers. Teachers will continue to implement Unit 3 during the remainder of the 2021-22 school year. Feedback that has already been identified includes adjusting the suggested timing for activities and creating a troubleshooting guide for homemade switches. During Summer 2022, a week-long professional development series will take place for another cohort of teachers online.

Unit 3 may ultimately be taught in Snap! instead of MicroBlocks, and future work would involve translating the activities into Snap! blocks. As identified by both curriculum developers and by pilot teachers, there are significant advantages to stay in the same programming environment throughout the three units.

Chapter 8

Appendix A - Student Pages

The following pages are the student facing curriculum available at <https://bjc.berkeley.edu/bjc-r/course/middle-school.html>.

Lab 1: Meet micro:bit

- 1. Get Ready
- 2. Tiny Animation

Lab 2: Interactive Pet

- 1. Designing Your Pet
- 2. Building Your Pet
- 3. Adding Interactivity
- 4. Telling a Story

Lab 3: Game Play

- 1. Game Design
- 2. Build Your Game
- 3. Game Testing

Lab 4: Marvelous Movements

- 1. Make It Move
- 2. Servo Project

Lab 5: Paper Stories

- 1. LED It Glow
- 2. Blink
- 3. Paper Story

Lab 6: Making with Multiples

- 1. Send a Signal
- 2. Interactive Communication

The remaining labs are under development.

Lab 7: Collaborative Chain Reaction

- 1. Make a Plan
- 2. Build Connections
- 3. Test Sequences
- 4. Launch Day



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1837260; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Get Ready

In this activity, you'll set up your micro:bit and write your first program in MicroBlocks.

Collect Materials

1. Gather materials.

- micro:bit
- USB cable
- Battery pack (optional)



Set Up Your micro:bit with MicroBlocks

For this unit, you'll use a program called MicroBlocks, a Snap!-like programming language designed to control tools like micro:bit.



2. Open [MicroBlocks](#) in a Chrome browser.

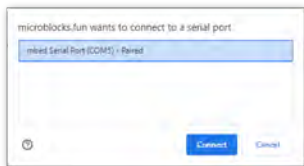
Why Chrome?

Chrome supports connecting a micro:bit to your web browser.

3. Connect your micro:bit to your computer with a USB cable.



4. Select the USB icon in the upper left corner (shown right), then select your board from the dialog box.



5. A green circle should automatically appear behind the USB icon, showing that the micro:bit is connected. If not, ask your teacher for help.



Adding Blocks

Your micro:bit is now connected, but you'll need to add more blocks to program it. You'll do this by adding a couple of *libraries* to MicroBlocks.

Vocabulary

A **software library** is a collection of procedures (blocks) that can be used in programs.

6. Click the + symbol next to Libraries.



7. Select "Basic Sensors.ubl" and then click "Open."





7. Select "Basic Sensors.ubl" and then click "Open."



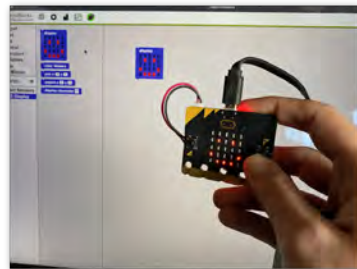
8. Do the same to load the "LED Display.ubl" library.

Using the Display Block

One block that's unique to micro:bit is the **display** block. It's designed to look like the grid of LEDs on the front side of the board. There are 25 LEDs that you can turn on and off individually.




9. Drag the **display** block into the Scripts area (the large open area on the right), and design your own pattern by clicking the LED rectangles to turn them on or off.
10. Click on the block once and a your pattern should appear on your micro:bit! (If not, work with your classmates or teacher to fix the problem.)



Saving Your Project

Saving in MicroBlocks is different from saving in Snap!.

11. Download your project:
 - a. Click the MicroBlocks "File" menu ().
 - b. Choose "Save."
 - c. Selecting a location on your computer.



A file will download to your computer that you can open using MicroBlocks later.

In this activity, you learned how to set up a coding environment for micro:bit and made a pattern on the micro:bit LED display.



BACK NEXT



EDC Education Development Center

Berkeley UNIVERSITY OF CALIFORNIA

The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Tiny Animation

In this activity, you'll write a script make an animation appear on the LED screen by switching between two screens.

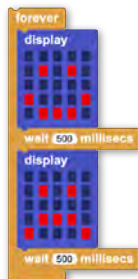
To start, you'll make an animation flipping between a smile and a frown.

1. If it isn't already, open [MicroBlocks](#), and connect your micro:bit.
2. Create two face designs on two copies of the **display** block.



Design your LED pattern by clicking on individual rectangles to light them up.

3. Create a loop to cycle through both expressions.



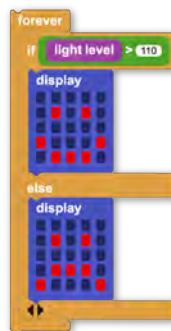
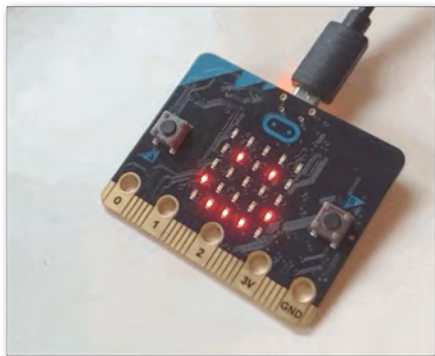
Tip: Animations are a rapid sequence of images displayed in a row. To bring your face to life, you'll need to switch between **display** blocks quickly, but not too quickly. To achieve this, use a **wait** block after each **display**. Adjust with the amount of time the program waits to what you like.

4. Once your animation is playing on the micro:bit, create your own animation by switching between two **display** blocks.

Make an Interactive Animation

So far, your animation changes on its own, but you can control when it changes for example, by using the micro:bit's LEDs as light sensors. The LEDs can detect different levels of light, and you can program the micro:bit to change the animation depending on how much light they detect.

5. Use an **if** block together with a **light level** block to change the image on the LED screen based on the light detected.



Tip: Light levels vary greatly depending on the room that you're in, so you'll need to experiment with values for your program considers bright or dark, and you may need to change it later for a different room. One way to do this is to use the **say** block. [Click for a picture.](#)

If There Is Time...

6. Make a more complicated animation by adding more **display** blocks or varying the **wait** time.

In this activity, you used **display** blocks together with **forever**, **wait**, and **light level** to create animations controlled by pauses and by light.

Designing Your Pet

In this activity, you'll design an interactive, imaginary creature that uses the micro:bit LED display.

Collect Materials

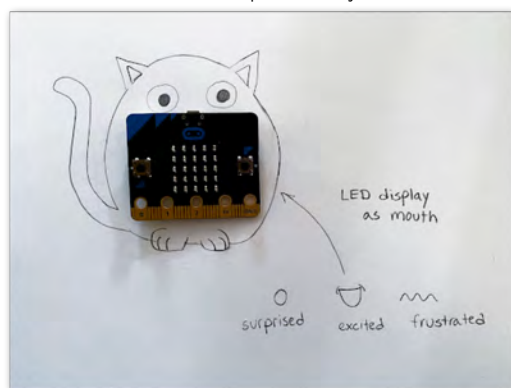
1. **Gather materials.**

- micro:bit
- USB cable
- battery pack (optional)

Plan Your Pet


Your pet can be real, fictional, or an entirely new creature of your own creation. It will detect its environment using micro:bit's sensors and inputs, and the LED display will let your pet respond.

2. Brainstorm your creature. Think about how it spends its day or the activities it enjoys doing. Use a pencil and sketch what it will look like.



3. Write some notes about how your pet will look and behave:

- What will its body look like? What materials do you need?
- Where will the micro:bit go? How will it be used to enhance the pet?
- What will be shown on the LED display?

You've already seen how to control the 25-LED display depending on light level detected by the micro:bit. The board can also respond to its two buttons (labeled "A" and "B"), and it can play sounds with a block that's like .

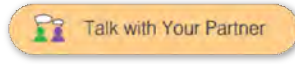
4. Your pet should respond to at least one of these interactions:

- Pressing a button
- Tilting the micro:bit
- Changing the light level (see [L1: Meet micro:bit](#))

Write some notes about how your pet will be respond to your actions.

5. Share your ideas with a partner, and offer feedback on their plan.

- Do you have ideas for making it more interesting?
- Do you think they may run into any challenges with their plan?



In this activity, you decided on a pet to create and thought about how to make it interactive using micro:bit.



Building Your Pet

In this activity, you'll create your pet using craft supplies.

Collect Materials

Gather materials.

- micro:bit
- Craft supplies, such as:
 - Scissors
 - Markers
 - Glue stick
 - Colorful paper
 - Cardboard
 - Pencils
 - Popsicle sticks
 - Decorative materials (googly eyes, pipe cleaners, etc)



Make a Pet

Create a body that will house your micro:bit. Consider the following design details:

- How will you integrate the micro:bit into the pet body?
- Do you want your pet to be flat or three-dimensional?
- Do you want parts of the micro:bit accessible? (LED screen, A/B buttons)

It's okay if you try a design that doesn't work at first. Think about if there are ways to modify the design to make it fit your vision, or if you need to change your original design. Your pet may evolve as you're making it.

Examples

The cat (left) builds off of the design seen in [Activity 1: Designing Your Pet](#). Notice how it looks similar but not quite the same as the drawing. The LED screen became the cat's mouth and the A/B buttons are accessible on either side.

The digital pet (right) is a new design that takes a different approach to making an interactive pet. Its body is an egg-shaped home that houses a pet that lives on the screen. To interact with the pet, one button is visible below the LED screen.



1. Create your body for your pet that integrates the micro:bit.

You can move between [Activity 2: Building Your Pet](#) and [Activity 3: Adding Interactivity](#).

In this activity, you created the physical body for your interactive pet.

Adding Interactivity

In this activity, you'll bring your pet to life with code.

To make your pet interactive, you will need to use at least two different **inputs** that cause at least two different **outputs**.

Vocabulary: Input and Output

In computing, the **input** is the action that tells the computer to do something. The **output** is the resulting action that occurs after the input command is received.

The micro:bit has different types of inputs: light (bright or dark), button pressed, or certain movement. The output we're currently exploring is displaying a picture on the LED display.

Consider the following inputs and decide which one to use to enhance your pet's personality or story.

- **light level** (like from Lab 1)
- **button A** or **button B**
- **tilt x** or **tilt y** or **tilt z**

1. Open [MicroBlocks](#) in a Chrome browser and start a new project. Load the additional libraries by selecting **Library**, then **LED Display** and **Basic Sensors**.
2. Use a **when** block to start your script.



3. Select an input for your interactive pet. Note that you'll need to include an inequality with **light level** and **tilt x**. Experiment with different values to trigger the script (100 may not be the right value). For a reminder about inequalities, check out 1.6.2, 1.6.3, and 1.7.



4. Create an animation using the **LED display**, **wait**, and **repeat** blocks.



In this activity, you created an interactive pet using code, paper, and storytelling.



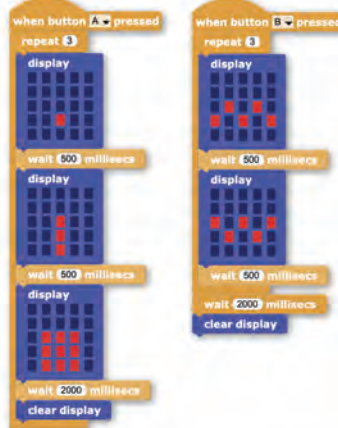
Telling a Story

In this activity, you'll bring your pet to life using cardboard, paper, and code.

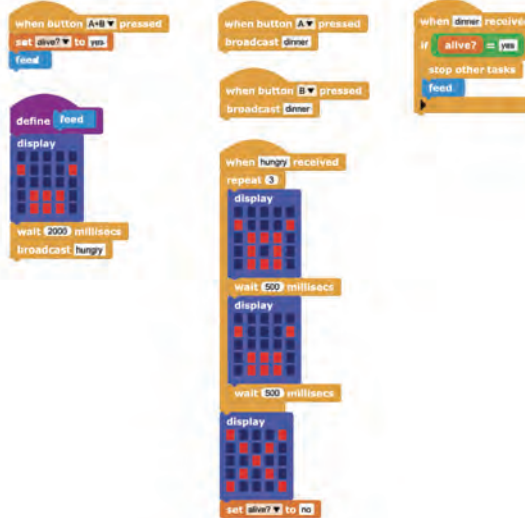
Now that you've made your interactive pet, document its interactions so that a classmate can know how to interact with it.

1. Describe your pet, how you interact with it (the inputs and outputs), and a story about your pet.
2. Write up your description to place next to your pet. When classmates or others visit your pet, they should know how to interact with it based on your text.

Meet Orange! He's a very vocal cat. When the A button is pressed, he meows loudly. When the B button is pressed, he purrs softly.



My digital pet is very hungry. Feed it by pressing the button, but don't wait too long or else it will die. You can bring the pet back to life by pressing both buttons at the same time.



If There Is Time...

Talk with Your Partner

It's helpful to share your designs with others for design feedback, to share ideas, and to test that your project works as intended. Share your pet with a partner by telling its story of how to interact with it. If something doesn't work as intended, make your desired changes to your design.


In this activity, you added code to make your pet interactive.

Game Design

In this activity, you'll start making a game (or reimagine an existing one) that uses micro:bit using light level, buttons, or tilt to be played either individually or with others.

Collect Materials

For each activity, we'll provide you with a list of materials to gather and some suggested materials.

-  Gather materials.
 - micro:bit and USB cable
 - Homemade switch materials (optional)
 - Aluminum foil
 - Alligator clip wires
 - Craft supplies

For a list of craft supplies you may find helpful, see [Lab 2 Activity 2: Building Your Pet](#).

Get Started

Think of a game you've played many times, and imagine how adding code could enhance the experience. You'll make a game that uses the micro:bit only or uses the micro:bit as a part to the game alongside other general craft materials.

- Brainstorm the game that you would like to bring to life. Use micro:bit inputs and outputs that you know to incorporate into game play, such as:

Inputs:

- Light level
- A or B button
- Tilt x, y, or z

Outputs: LED display

-  **Talk with Your Partner** Think first on your own and then discuss with a partner:
 - What are the rules to the game?
 - How will the game start? How will players know when it ends?
 - Is it single player, or can it be played collaboratively?

When making your game, you'll likely want to use a **variable** to keep track of data like a score. We first learned about variables in [Unit 1 Lab 7: Dealing with Data Dots, Activity 2](#). Here are some key pieces of information to remember:

Vocabulary

A **variable** is like a labeled box that can store a value, such as a word, a number, or a list.

Create a new variable in MicroBlocks by selecting the Variables tab on the left. Select **Add a variable** and give it a name that's meaningful to your project. You will then have access to set and use your variable in your script.



Before building, decide whether you would like your game to be played with the micro:bit only or if you want to include additional materials. The following is an example of a micro:bit only game.

Example 1: micro:bit only

Fast Click

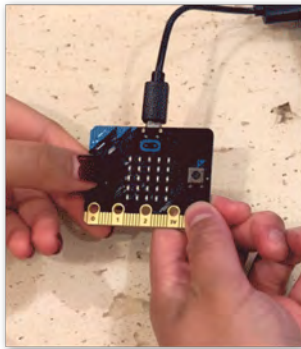
In this game, players click the A and B buttons as quickly as they can. After the micro:bit displays "GO", Player A and B will try to click their respective buttons as fast as possible. The player able to click their button 10 times the fastest wins. The LEDs will then display the letter of which player (A or B) won the round.



Example 1: micro:bit only

Fast Click

In this game, players click the A and B buttons as quickly as they can. After the micro:bit displays "GO", Player A and B will try to click their respective buttons as fast as possible. The player able to click their button 10 times the fastest wins. The LEDs will then display the letter of which player (A or B) won the round.



```
when started
  broadcast go!
  forever
    if A Score = 10
      display character A
      wait 1000 millisecs
      broadcast go!
    if B Score = 10
      display character B
      wait 1000 millisecs
      broadcast go!

when button A pressed
  change A Score by 1

when button B pressed
  change B Score by 1

when go! received
  set A Score to 0
  set B Score to 0
  clear display
  scroll text GO
```

In this activity, you started to brainstorm your game design and rules and decide whether your game is made with a micro:bit only or with additional materials.

BACK NEXT



EDC
Education
Development
Center

Berkeley
UNIVERSITY OF CALIFORNIA

The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Game Play

In this activity, you'll build your game and play it with classmates.

In the previous activity, you were introduced to a game that uses the micro:bit only. The following is an example game that uses additional materials. We'll introduce read analog pin, a new input type, and use the **read analog pin** blocks to determine whether an action has occurred or not using a switch.

Vocabulary

An **analog input pin** can read a voltage level that ranges of voltages provided to your micro:bit.

A **switch** is an electronic device that disconnects or connects an electrical path, resulting in a circuit turning off or on.

Make a switch using your micro:bit by connecting alligator clip wires to Pin 0 and GND. Start by connecting these two alligator clips together and you've got a switch! Use another conductive material like aluminum foil to add more complex shapes to your game design.

Example 2: micro:bit + materials

Tube Ball

A toilet paper tube transforms into a basket for catching a ball. Alligator clips to make a homemade switch, so when the ball is inside the tube, it completes the circuit. I really like this technique and hope we can use it more! I'm also using a counter to keep track of every time the ball goes in the hole. To reset the game, press the A button.



```

when started
  set score to 0
  forever
    say score
  when button A pressed
    set score to 0

when read analog pin 0 > 150
  display character score

when read analog pin 0 < 10
  wait 500 milliseconds
  clear display
  change score by 1
  scroll text SCORE!
  display character score
  wait until read analog pin 0 > 150
  wait 3000 milliseconds
  
```

Tip: Add scrolling text to your micro:bit by adding the Scrolling library in MicroBlocks. [Click for a picture.](#)

Build a Game

After brainstorming and seeing games that use only a micro:bit as well as one that uses physical materials, it's time to start bringing your game to life.

1. Design and build your game in MicroBlocks and with physical materials, if applicable.
2. Play your game yourself or ask a classmate to play with you. If you find that something does not work as intended, make necessary changes.
3. Write down the game and its rules to communicate to those who are new at playing the game or are new to micro:bit.

If There Is Time...

After playing your game with classmates, ask them for feedback on your game. Was it easy to play? Did the rules make sense? Consider their feedback and make appropriate changes to your game design.

If you have completed your game early, make a game of the opposite type. For example, if you made a micro:bit only game, make a game that uses additional materials or visa versa.

In this activity, you built your own game and tested it with classmates.

Make It Move

In this activity, you'll learn about servo motors, how to connect one to your micro:bit, and use MicroBlocks to control its movement.

Collect Materials

1.  Gather materials.

- micro:bit and USB cable
- Crazy Circuits Bit Board
- Servo motor
- Craft supplies

For a list of craft supplies you may find helpful, see [Lab 2 Activity 2: Building Your Pet](#).

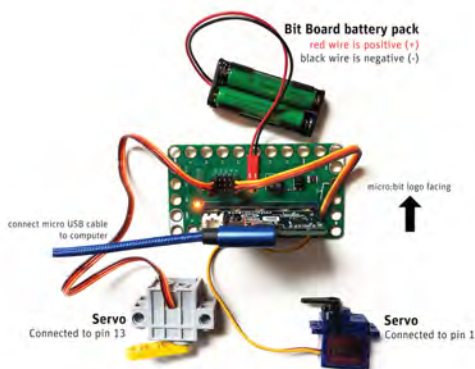
Get Started

Motors are one way to add movement to your projects. There are many different types of motors, including one called a *servo motor*.

Vocabulary

A **servo motor** provides position control, so it can be told to move into an exact spot. Its position can be selected from 0 to 180 degrees.

1. Insert your micro:bit into the Bit Board, pins facing down.
2. Connect a servo to the pins on the Bit Board, making note of which pins. Use the following image as reference for setup.



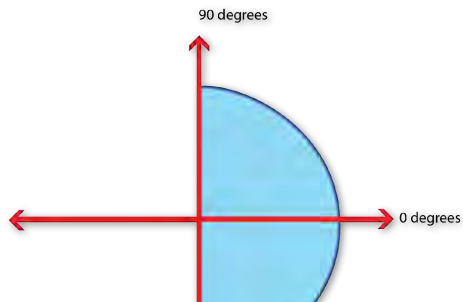
3. Add the servo library in MicroBlocks by selecting Library, and then Servo.ubl.

Control the servo motor by using the servo motor blocks. After adding the servo library, you'll have the following blocks available:



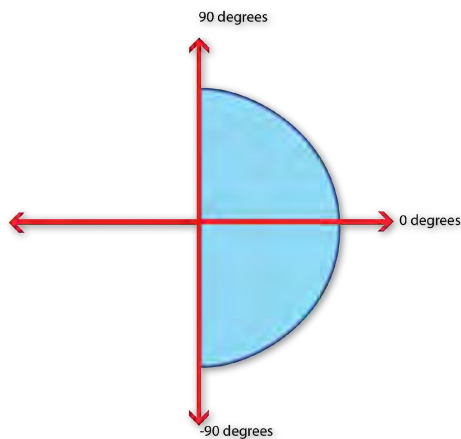
- **set servo to degrees**

Moves the servo arm to a particular position in degrees. The range of -90 to 90 degrees is equivalent to 180 degrees. Be sure to address the right servo (check your Bit Board to see which port your servo is connected to, and use the following chart as reference for determining servo position in degrees.



- **set servo to degrees**

Moves the servo arm to a particular position in degrees. The range of -90 to 90 degrees is equivalent to 180 degrees. Be sure to address the right servo (check your Bit Board to see which port your servo is connected to, and use the following chart as reference for determining servo position in degrees.



- **set servo to speed**

Determine the speed at which the servo moves. The speed range is from 0 to 100, and the sign (positive or negative) determines the direction it turns.

- **stop servo**

Stop the servo's movements.

Tip: Each block asks to specify which the servo number. Check which port your servo is plugged into on the Bit Board. The available options are 0, 1, 13, 14, or 15.

Add Movement

Now that you've set up your micro:bit and servos, make your first movement. Use the **set servo to degrees** block to make the servo move, and change the sign (positive and negative) to switch the direction.

4. Make your servo move using at least two blocks.

In this example, the servo motor moves twice with a **wait** block to add a pause between movements. This script starts by pressing the A button.

```
when button A pressed
set servo 1 to 45 degrees (-90 to 90)
wait 500 millisecs
set servo 1 to 0 degrees (-90 to 90)
wait 500 millisecs
```

Add Another Type of Movement

There are many ways to make a servo move: big or small, quick or slow, with pauses or rapid succession, the choice is yours!

5. Create a second servo movement.

This example uses the B button to reset the servo's position.

```
when button B pressed
set servo 1 to 90 degrees (-90 to 90)
```

If There Is Time...

Now that you've made two different movements with your servo, consider making something more complex using different micro:bit **inputs**. Or add more than one servo to your Bit Board to control multiple motors simultaneously.


In this activity, you learned how to connect a servo motor to your micro:bit and make it move in MicroBlocks.



Servo Project

In this activity, you'll create your own servo project using the Bit Board, MicroBlocks, and general making materials.

Before diving into your servo project, take some time to brainstorm independently and discuss your project idea with a classmate.

1. Brainstorm how you want to incorporate motion into a project and what supplies you'll need to accomplish your idea.
2.  **Talk with Your Partner** Talk with a nearby classmate about your idea, and listen to theirs. Ask yourselves the following questions:
 - Describe the story of what will happen. What's the order of events?
 - Does a person need to interact with the story? If so, how?
 - Describe the movement? How will it start? How will it stop?

Example 1: Goal!

Make a game using servos, like this soccer game. The servo controls the player's leg, which can then kick the ball into the goal. Keep track of the score on the micro:bit by pressing the A button the mark each point.

Tip: To keep track of the score, this project uses **variables**, which were introduced in Lab 3 Activity 1: Game Design.



```

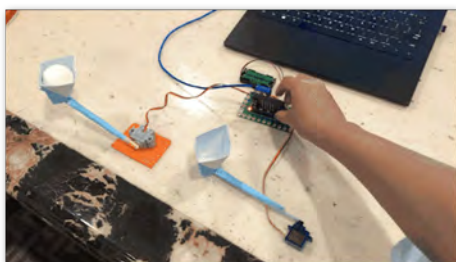
when button B pressed
  wait 500 millisecs
  set servo 1 to 0 degrees (-90 to 90)
  wait 200 millisecs
  set servo 1 to 45 degrees (-90 to 90)
  wait 200 millisecs
  stop servo 1

when button A pressed
  change count by 1
  display character count

when button A+B pressed
  set count to 0
  display character count
    
```

Example 2: Pass the Ball

Move a ball from one end of a table to the other by passing it between servos. These servo arms are extended and modified with a cup on the end to hold a small ball. Start the sequence of events by pressing the B button on the micro:bit, which then initiates the servo arms to pass the ball from one cup to another.



```

when button B pressed
  wait 1000 millisecs
  set servo 13 to 0 degrees (-90 to 90)
  wait 500 millisecs
  set servo 13 to 45 degrees (-90 to 90)
  wait 500 millisecs
  stop servo 13
  wait 500 millisecs
  set servo 1 to 0 degrees (-90 to 90)
  wait 500 millisecs
  set servo 1 to 90 degrees (-90 to 90)
  wait 500 millisecs
  stop servo 1
    
```

3. Create your servo project using MicroBlocks and physical materials.
4. Share your project with others and make changes, if desired.

If There Is Time...

Make another project with a servo motor. If possible, add a second servo and make a project that incorporates both.

In this activity, you created your own unique project using servo motors and controlled them with code.

LED It Glow

In this activity, you'll learn about LED lights and how to turn them on both with and without code.

Collect Materials

1.  Gather materials.

- micro:bit and USB cable
- Copper tape
- LEDs
- Coin cell battery
- Pencil
- Needle nose pliers (optional)
- Craft supplies



For a list of craft supplies you may find helpful, see [Lab 2 Activity 2: Building Your Pet](#).

LEDs and Circuits

In this activity, we'll explore programmable light by using LEDs.

Vocabulary

An LED contains a light emitter inside of a plastic bulb. This light emitter can be made from different materials, and when electricity runs through it, it shines different colors. However, electricity can only flow in one direction, and the name for electronic parts with this quality is called a diode. Thus, an LED stands for "light-emitting diode."

An LED only works when oriented in one direction (it's not reversible), so it's important to be familiar with its structure. LEDs have two legs with different lengths. The long leg is the positive side of the LED, and the short leg is the negative side.

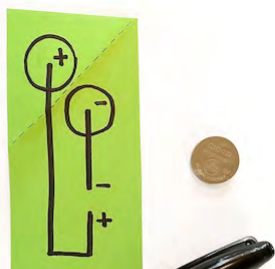


First, we will illuminate an LED with only a battery (no code). The simplest way is to place the positive (+) side of your LED on the positive side of the battery.

1. Grab an LED and a coin cell battery. Place the long leg on the positive (+) side of the battery, and the short leg on the negative side (-) of the battery].

Next, we'll walk through how to make a circuit on a piece of paper.

2. Recreate this drawing on a quarter sheet of paper. The circles are created by tracing a coin cell battery twice in the corner of the piece of paper. The dashed line is where the paper folds. Take note of where the positive (+) and negative (-) signs are located.

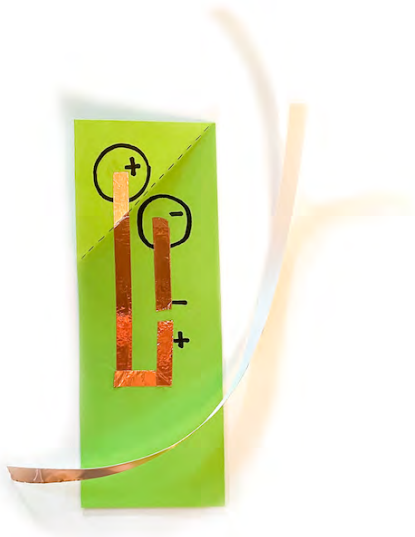


Next, we'll walk through how to make a circuit on a piece of paper.

2. Recreate this drawing on a quarter sheet of paper. The circles are created by tracing a coin cell battery twice in the corner of the piece of paper. The dashed line is where the paper folds. Take note of where the positive (+) and negative (-) signs are located.



3. Use copper tape to cover the drawn path, leaving a gap in the middle. Careful to use one continuous piece of tape, if possible, as the underside of the tape is less conductive and may lead to the light not turning on.

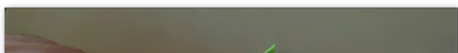


Tip: Use this folding technique to make corners with copper tape. [Click for a video.](#)

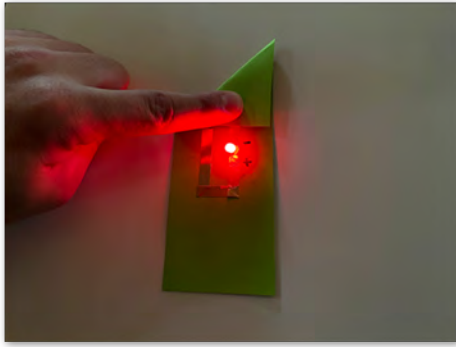
4. Place the coin cell battery face up so that it's on top of the (-) circle.



5. Place your LED onto the copper strips in the correct (+)/(-) orientation and secure it using copper tape or clear tape. When the corner of the paper is pressed on top of the battery, the light should illuminate.



5. Place your LED onto the copper strips in the correct (+)/(-) orientation and secure it using copper tape or clear tape. When the corner of the paper is pressed on top of the battery, the light should illuminate.



In this activity, you learned how to complete a circuit on paper using copper tape, a coin cell battery, and an LED.



BACK NEXT



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1136596, 1441075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Blink

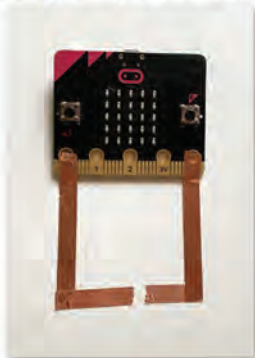
[Toggle developer todos/comments \(red boxes\)](#)

In this activity, you'll learn how to make your LED light blink.

micro:bit Set-up

Before writing code in MicroBlocks, set up your micro:bit on a piece of paper. Use two pieces of copper tape to make your connections. One piece of tape will start at pin 0 and the other will start at GND.

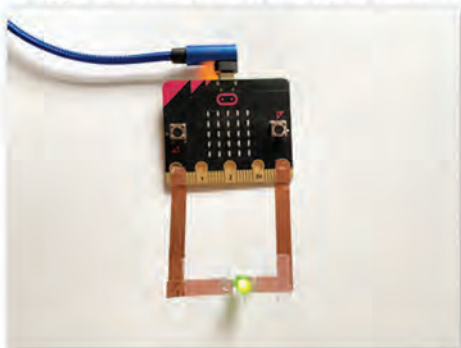
1. Place two pieces of copper tape on the micro:bit on a piece of paper.



Test Code

Test your set up by adding an LED with the following starting code. Make sure that the positive side of the battery is connected to pin 0 and the negative side of the LED is connected to GND.

2. Add an LED to your copper tape circuit and test that it turns on with this example code.



Blink Code

After testing that your LED turns on, you'll modify to the code so it blinks on and off. The code from the test uses a block `set digital pin ... to ...` which controls when electrical current is sent into a specified pin. Choose which pin to send current (0, 1, or 2) and whether there should be current (green toggle for on, red toggle for off).

3. Modify the test code to turn the LED on and off.



4. Test with micro:bit by plugging it into your computer. [Click for a video.](#)

If There Is Time...

Explore other blink patterns. Create custom on-and-off sequences using the `set digital pin` block. Once you have a custom blink pattern, use one of the micro:bit inputs from a previous lab to start turn on your light.

In this activity, you learned how to make an LED blink with code.

Paper Story

[Toggle developer todos/comments \(red boxes\)](#)

In this activity, you'll make your own paper creation that uses programmed lights to tell a story.

Plan Your Project

Brainstorm what story to tell using paper and lights. Use a pencil and sketch what it will look like.

1. **Talk with Your Partner** Think first on your own and then discuss with a partner:

- What story do you want to tell? What materials do you need to create your story?
- Where will the micro:bit go? Where will the copper tape and LEDs go?
- What pattern will the LEDs display? Will they use an input?

2. Place your micro:bit on a piece of paper and draw in pencil where to place your copper tape and LEDs. Take a look at the tips below for how to add multiple LEDs and how to layer pieces of copper tape. In this drawing, the lines are copper tape placements and the dots are LED locations.



3. Add copper tape and LEDs. Check that your LEDs are placed in the correct orientation (positive leg on the pin side, negative leg on the GND side).

Tip: It's okay to add more than one LED to a piece of copper tape. Note that both LEDs touch the copper tape that's attached to GND.



Create with Code and Paper

After planning, start building your project both in MicroBlocks and out of paper.

4. Write the code that will control the LEDs. In this example, the A and B buttons are used to control each individual LED, as well as together when both buttons are pressed.

```
when button A pressed
  set digital pin 0 to on
  wait 500 milliseconds
  set digital pin 0 to off
```

```
when button B pressed
  set digital pin 1 to on
  wait 500 milliseconds
  set digital pin 1 to off
```

```
when button A+B pressed
  set digital pin 0 to on
  set digital pin 1 to on
  wait 500 milliseconds
  set digital pin 0 to off
  set digital pin 1 to off
```

5. Use paper and any other materials to communicate your story.

5. Use paper and any other materials to communicate your story.



Tips

- **Multiple LEDs, Same Pin**

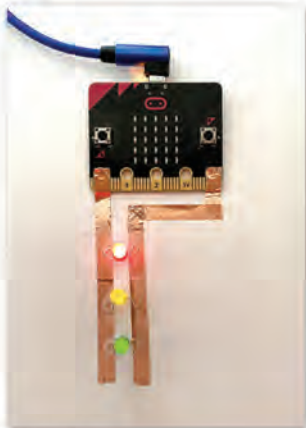
It's okay to have multiple LEDs attached to the same copper tape. This type of circuit design is called **parallel circuit**.

Vocabulary

A **parallel circuit** has multiple branches and allows for easier addition of multiple LEDs. In parallel, the LEDs retain most of their brightness.

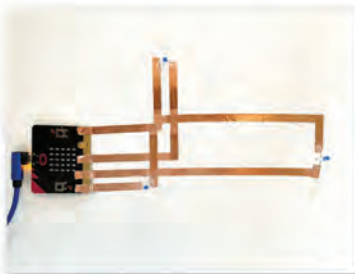
A **series circuit** has all elements in a chain, one after another and makes a complete circle. In series, the more LEDs added, the dimmer they become.

In this example, three LEDs are all attached to the same pin. This design works well for LEDs that all use the same code. If you want to control LEDs individually, attach them to separate pins.



- **Multiple LEDs, Different Pins**

In this example, three LEDs are connected to three different pins. This allows for each LED to be programmed individually.



```
set pins to 100
forever
  set digital pin 2 to
  wait pause millisecc
  set digital pin 3 to
  wait pause millisecc
  set digital pin 4 to
  wait pause millisecc
  set digital pin 5 to
  wait pause millisecc
  set digital pin 6 to
  wait pause millisecc
  set digital pin 7 to
  wait pause X millisecc
```



Tip: It's okay to have pieces of copper tape cross over one another. The key is to place something in between the two pieces of copper tape, like paper or clear tape.

In this activity, you created a story out of paper and programmed lights.



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1325596, 1441075, and 1637280; the U.S. Department of Education under grant number S411G200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.

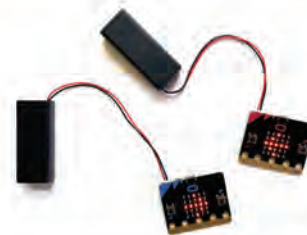


Send a Signal

In this activity, you and a partner will connect your micro:bits and send messages back and forth.

1. Gather materials

- Two micro:bits
- Two USB cables
- Two battery packs
- Any additional craft materials, if desired



A micro:bit can use **radio waves** to wirelessly communicate to another micro:bit. One micro:bit can send transmit or a message, the other will receive and can then be programmed to send a message back.

Vocabulary: Radio Communication

Radio is a way of transmitting and receiving information over a distance.

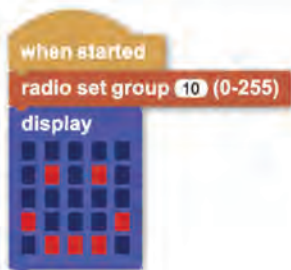
Many micro:bits in the same room can cause interference and confusion when sending and receiving messages. To avoid this, pick a group that you and your partner will join that's unique from other groups. Start by sending a message back and forth with a partner, but know that you can add more than two micro:bits to a group for communication.

radio set group 0 (0-255)

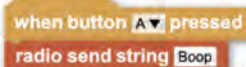
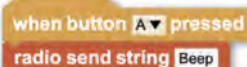
Text Messaging

Together with a partner, follow along with this example and then make your own.

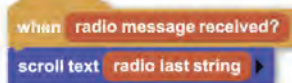
- Add the radio communication library by choosing **Library**, then **Radio** and **Scroll Text**.
- Both partners will create messages to send. It's important that these messages are different from one another to start, but this can be changed later on.
- Both Partners:** Add your group number to the **when started** block. The smiley face is a visual check for you to see that you are ready to receive radio messages.



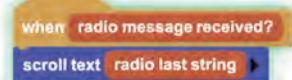
- Each Partner:** Pick a unique word or phrases to send to your partner. You can use the examples below as a test. In this case, Partner 1 has the message of **Beep** (left) on their micro:bit and Partner 2 has the message **Boop** (right).



- Both Partners:** Add this final script that checks for whether your micro:bit has received a message. To create it, start with the **when** block and **radio last received?** by dragging it on top of the **green toggle**.



- Start the project by pressing the green play button in the top right hand corner. The new **when radio last received?** block should illuminate.



- Test out your code by pressing the **A** button on each micro:bit and see your message scroll by.

7. Start the project by pressing the green play button in the top right hand corner. The new `when radio last received?` block should illuminate.

```
when radio message received?  
  scroll text radio last string
```

8. Test out your code by pressing the **A** button on each micro:bit and see your message scroll by.

Send an Integer

In addition to sending text, it's also useful to know how to send numbers using radio. This process is very similar to sending strings, though some of the blocks are slightly different.

```
radio last number
```

```
radio send number 123
```

- 8. Review how the original messages were sent and created.
- 9. Update the code to send and receive messages with numbers (integers).

If There Is Time...

Begin brainstorming with your partner about a larger project you would like to build that uses radio communication. Consider bringing in tools and techniques that you've already learned to enhance your project (servo motors, LEDs, etc).

In this activity, you learned how to connect two micro:bits wirelessly using the radio feature and sent and received messages from a classmate.

[BACK](#) [NEXT](#)



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138896, 1441075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



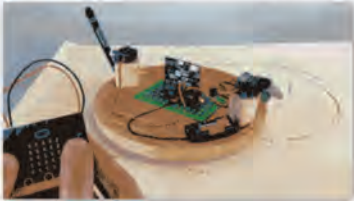
Interactive Communication

In this activity, you and your partner will make a project that uses two micro:bits in communication with one another.

Now you can use radio to communicate between two micro:bits and make an interactive project. Here are two different pathways to consider when making a collaborative project with your partner.

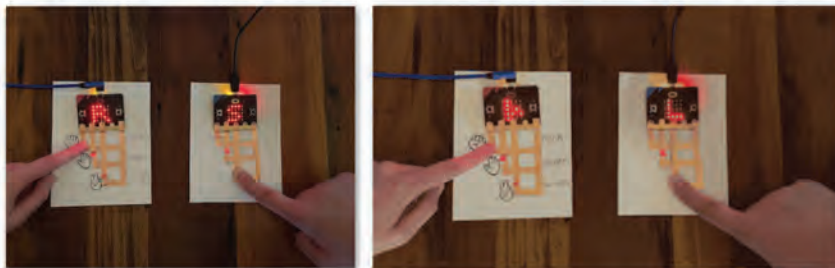
Remote Controller

Use one micro:bit as a remote control that takes different inputs like button, tilt, light, or pin. Then use a second micro:bit to control outputs like LED display, servo motors, LEDs, and music. This drawing machine uses one micro:bit to control two servo motors attached to a second micro:bit which make marks on a piece of paper. Refer back to [Lab 4: Marvelous Movements](#) for a refresh on how to control servo motors.



Two Player Game

Consider using each micro:bit as a controller for playing a two-person game. In this example, the game Rock Paper Scissors is made into a physical game using copper tape like in Lab 6: Paper Stories. Each of the three pins on the micro:bit are used as the three options in the game. The copper tape is used to make small switches to indicate the player's selection. Radio communication between micro:bits determines if the player wins, loses, or draws.



```

when button A pressed
  radio set group 0 (0-255)
  set me to Rock
  set partner to Rock
  pluck MIDI key 60 for 100 msecs
  wait 4000 milliseecs
  pluck MIDI key 60 for 100 msecs
  wait 4000 milliseecs
  pluck MIDI key 60 for 100 msecs
  wait 3000 milliseecs
  broadcast check winner

when check winner received
  if me = partner
    scroll text DRAW
  else if me = Rock and partner = Scissors
    scroll text WIN
  else if me = Paper and partner = Rock
    scroll text WIN
  else if me = Scissors and partner = Paper
    scroll text WIN
  else if
    scroll text LOSE

when button A+B pressed
  clear display

when radio message received?
  set partner to radio last string

when read analog pin 0 < 150
  display character R
  set me to Rock
  radio send string me

when read analog pin 1 < 150
  display character P
  set me to Paper
  radio send string me

when read analog pin 2 < 150
  display character S
  set me to Scissors
  radio send string me
    
```


Create a Custom Message

Now that you've successfully sent a message, craft your own with a partner.

- Talk with Your Partner First think on your own and then discuss with a partner:
 - What message do you want to send?
 - What input will send your message? Buttons pressed, tilt, light levels, or something else?
 - What will be your output when a message is received? Text, servo motor moves, light illuminates, or something else?

Create a Custom Message

Now that you've successfully sent a message, craft your own with a partner.

-  **Talk with Your Partner** First think on your own and then discuss with a partner:
 - What message do you want to send?
 - What input will send your message? Buttons pressed, tilt, light levels, or something else?
 - What will be your output when a message is received? Text, servo motor moves, light illuminates, or something else?
- Create your messages and send them back and forth.

If There Is Time...

Select a new group radio channel with more than two students and think of messages to send that would work for a larger group. Consider using other physical outputs like you used in Lab 4 or Lab 5.

In this activity, you made a project that used radio communication to send and receive messages between you and your partner.

[BACK](#) [NEXT](#)



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1637280; the U.S. Department of Education under grant number 5411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Make a Plan

In this activity, you and a partner will make a plan for your section of the chain reaction machine.

What is a Chain Reaction?

A chain reaction is a sequence of causes and effects that often accomplishes a task in a somewhat impractical manner. They often feature everyday household items that are transformed into parts of a machine that carry momentum forward and help the reaction unfold.

In this lab, your class will build a chain reaction machine together. You and a partner will take one section of the chain reaction, which will consist of multiple parts. Your team can use whatever materials and tools in your section, but it must set off the contraption of the next pair. The result will be a continuously flowing chain reaction machine.



Collect Materials

1. Gather materials.

- Any materials from Labs 1-6
- General Building Materials
 - String
 - Cardboard tubes
 - Rubber bands
 - Clothespins
 - Paper cups
 - Push pins
 - Skewer sticks
 - Washers
- Interesting Building Materials
 - Plastic car tracks
 - Dominoes
 - Pulley
 - Feathers
 - Balloons
 - Fan
- Glue and Adhesives
 - Blue tack
 - Masking tape
 - Hot glue



Elements to a Chain Reaction

A chain reaction can consist of both analog and digital materials. Consider including a variety of materials when building your segment.

Use both micro:bits to help move the chain reaction along, and consider using elements such as:

- The LED display
- Basic Sensors, such as *tilt*, *acceleration*, and *light level*
- Servo motors
- External LEDs
- Radio communication

Here are some example physical, no-tech elements:

- Dominoes falling over
- A ball rolling down a ramp

You may also include some low-tech elements, such as:

- A homemade switch that turns something on/off
- A fan blowing air

Plan Your Segment

2. With a partner, pick a section of the chain reaction to complete.
3. Make a plan with your partner what you want to create that incorporate the following elements:

- Two micro:bits
- A sensor or actuator

Plan Your Segment

2. With a partner, pick a section of the chain reaction to complete.
3. Make a plan with your partner what you want to create that incorporate the following elements:
 - Two micro:bits
 - A variety of analog materials
 - A story or theme
4. Also, plan out with neighboring pairs how each of your segments will connect to one another. Place a piece of tape on the part of the table where your two chain reaction segments will connect.

In this activity, you made a plan with your partner for what elements of the chain reaction you want to include in your segment.



PAGE NEXT



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1136596, 1441075, and 1637260; the U.S. Department of Education under grant number 541C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Build Connections

In this activity, you and a partner make you segment of the chain reaction using physical and digital materials.

Assemble Your Segment

Begin constructing your segment of the chain reaction and remember to include the following elements:

- Two micro:bits
- A variety of analog materials
- A story or theme

1. Decide with your partner which parts of the chain reaction segment you will work on. Each segment is made up of many parts, so you both will be able to work on parts separately as well as together.

Create a rough plan of the sequence of events before diving deep into one aspect of the segment. If something isn't working as intended, take a step back and see if there's another way to accomplish the same movement or task.

2. As components of your segment become more finalized, secure them to the surface to avoid accidentally moving any of the pieces.
3. Do small tests as you build. You might be surprised to find that something that worked well initially has stopped working after adding a few more pieces. Smaller tests will help with troubleshooting as you're building.

It's okay if your plan shifts as you're building. Communicate any changes with your partner and with neighboring groups, if necessary.

If There Is Time...

Consider adding a finale to the end of the chain reaction. What will be the final celebratory act? Or final task accomplished?

In this activity, you created your segment of the chain reaction with your partner.



Test Sequences

[Toggle developer todos/comments \(red boxes\)](#)

In this activity, you and a partner test with your chain reaction segment and troubleshoot any missed connections.

Troubleshooting

After making a complete segment for your class' chain reaction machine, it's important to test it as many times as time allows. It's often the case a chain reaction segment is inconsistent, or it tends to fail in the same spot. Take this time to really study your connections, both the physical materials as well as your micro:bit components, for opportunities to strengthen your design.

1. Set off your chain reaction segment at least five times consecutively. It's alright if the cause-and-effect chain doesn't run as expected. Take note of the following each time you try:
 - Does the chain reaction segment run from start to finish? If not, where does it end?
 - Were there any inconsistencies between trial runs? If so, where were they?
 - Did the micro:bit perform as intended? If not, are there ways to modify the MicroBlocks code to improve its performance?
2. Take this information and make adjustments to your segment accordingly. Find any opportunity to make a part of your machine more consistent.
3. After making these changes, return to question 1 and 2 for another round of testing. You may find that some of the previously problematic areas are now resolved, some may still be an issue, and new challenges may have arisen. Take note of these points in your segment and adjust them accordingly.

Take time to also check in with your neighboring teams. Run tests with both pairs (if applicable) and make similar changes.

If There Is Time...

Run larger tests with two or more groups and see if adding more segments changes how the chain reaction performs.

In this activity, you tested your chain reaction segment with your partner and made improvements to your design.

BACK  NEXT

The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1328596, 1441075, and 1837280; the U.S. Department of Education under grant number 5411C000074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Launch Day

[Toggle developer todos/comments \(red boxes\)](#)

In this activity, you and your class will launch your chain reaction machine.

Final Preparations

This is your last opportunity to make small adjustments to your segment design before the finale.

1. Finalize the set-up for your segment. Use the following checklist as reference:
 - Check every cause-and-effect. Are your objects positioned as needed? Does anything need to be secured?
 - Test your micro:bit components. Do they have batteries and react as intended?
 - Confirm your connections to neighboring pairs. Is everything in alignment?
2. Move away any extraneous materials to clear a path for the chain reaction.

In this activity, you tested your chain reaction segment with your partner and made improvements to your design.

[BACK](#) [NEXT](#)

The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1128596, 1441075, and 1637280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Chapter 9

Appendix B - Teaching Guide

The following pages are the student facing curriculum available at
<https://bjc.berkeley.edu/bjc-r/course/middle-school-teacher.html>.

Purpose

This unit extends student work with sequencing and iteration into the realm of physical computing using the micro:bit controller. Students use craft materials to build a toy and game; use e-textile materials (e.g., conductive thread, LEDs, and sensors) to create circuits; use neopixels to program creations to change colors, twinkle, and pulse; and program micro:bits to send and receive messages. The unit culminates with building collaborative chain reaction (Rube Goldberg) machines.



[Submit End-of-Lab or General Feedback](#)

CSTA Standards:

- 2-CS-01: Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
- 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
- 2-CS-03: Systematically identify and fix problems with computing devices and their components.
- 2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
- 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
- 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
- 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
- 2-AP-17: Systematically test and refine programs using a range of test cases.
- 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
- 2-IC-22: Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.

For more micro:bit resources, check out the activities available on [Learn MicroBlocks](#).

Teacher Guides for Student Labs

Lab 1: Meet micro:bit

- approximately 1–2 class periods

Lab 2: Interactive Pet

- approximately 1–2 class periods

Lab 3: Game Play

- approximately 2–3 class periods

Lab 4: Marvelous Movements

- approximately 2–4 class periods

Lab 5: Paper Stories

- approximately 2–4 class periods

Lab 6: Making with Multiples

- approximately 2–4 class periods

Lab 7: Collaborative Chain Reaction

- approximately 3–5 class periods



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138598, 1441075, and 1637280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Lab 1: Meet micro:bit

Students set up their micro:bit controller and begin using it with the MicroBlocks interface. MicroBlocks is similar enough to Snap!, that students should adapt quickly. In this lab, they create a simple animation using MicroBlocks' **display** block, which controls the micro:bit's LED array; MicroBlocks' **light level**, which reports the amount of light detected by a sensor; as well as loops, conditionals, and predicates.

Pacing

This lab is designed for 1–2 class periods (**25–70 minutes**).

- **Daily Activity:** *Computing in the News* 5–15 minutes each day
- **Activity 1:** *Get Ready*: 10–30 minutes
- **Activity 2:** *Face Creator*: 10–20 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read *Computing in the News* for additional details and suggested news sources.

- **Prepare:**
 - Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Unit 3 Materials

This unit requires a number of **materials** that you should to begin gathering before teaching the unit. Amazon links are provided for convenience, but you can get these materials from any provider you prefer.

Activity 1: Get Ready.

[↑ Back to Top](#)

- **Materials:** *Get Ready* student page
- **Prepare:**
 - Work through the student activity page and connect your own micro:bit before leading this activity with students.
 - MicroBlocks does not offer accounts and cloud storage for projects. Decide where you want students to save their MicroBlocks files. For example, you may have them save to a networked drive or download files and upload them to a folder on Google Drive.
- **Learning Goals:**
 - Set up the micro:bit with MicroBlocks.
 - Control the micro:bit controller using the MicroBlocks interface.
 - Save a MicroBlocks project file to the local computer.
- **Activity Plan:**
 - **Collect Materials and Set Up Your micro:bit with MicroBlocks:** Students connect their micro:bit to the computer. (about 5 minutes)
 - **Adding Blocks:** Students add the "Basic Sensors.ubl" and "LED Display.ubl" libraries. (less than 5 minutes)
 - **Using the Display Block:** Students explore the **display** block and use it to control the micro:bit. (about 5 minutes)
 - **Saving Your Project:** Students save their MicroBlocks project locally. (less than 5 minutes)
- **Tips:**
 - You may wish to have the micro:bits and USB cables set out for students before they arrive.
 - If students struggle to get their micro:bits connected...
 - Invite them to follow all of the setup instructions again.
 - Ensure that their micro:bit controller is properly connected to their computer.
 - Try a different USB cable.
 - Try a different micro:bit with that computer and try a different computer with that micro:bit.
 - Additional resources about micro:bit can be found at <https://microbit.org/get-started/first-steps/introduction/>
 - Additional resources about MicroBlocks can be found at <https://microblocks.fun/get-started>
- **Standards:** None covered.

Activity 2: Face Creator.

[↑ Back to Top](#)

- **Materials:** *Face Creator* student page
- **Learning Goals:**
 - Use a loop to create an animation.
 - Use a conditional together with the micro:bit light sensor to control program behavior.
- **Activity Plan:**
 - **Animate a Face:** Students use two instances of the **display** block together with the **forever** and **wait** blocks to create a simple animation. (5–10 minutes)
 - **Make an Interactive Face:** Students use two instances of the **display** block together with the **if** and **light level** blocks to control their with light animation. (5–10 minutes)
- **Standards:** None covered.
- **Solutions:** Intended code structures are included on the student activity page.

Lab 2: Interactive Pet

Students plan and design their own interactive pet using the micro:bit and learn about inputs and outputs using hardware. They use the micro:bit's light and tilt sensors as inputs (changes to the pet's environment) and the LED display as outputs (the pet's response). Students use "hat blocks" such as `when ()` and `when button () pressed` to initiate code in response to inputs so events in the pet's environment trigger the pet's response.

Alongside programming, students craft a physical body for their pet that houses the micro:bit while keeping the LED screen and buttons accessible. The body should communicate something about the pet that then is reinforced through its programmed behaviors.

[Submit Feedback](#)

Pacing

This lab is designed for 1–2 class periods (50–110 minutes).

- **Daily Activity: Computing in the News** 5–15 minutes each day
- **Activity 1: Designing Your Pet** 20–35 minutes
- **Activity 2: Building Your Pet** 15–30 minutes
- **Activity 3: Adding Interactivity** 10–20 minutes
- **Activity 4: Telling a Story** 10–30 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read [Computing in the News](#) for additional details and suggested news sources.

- **Prepare:** Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: Pet Inventor.

[↑ Back to Top](#)

- **Materials:** [Pet Inventor](#) student page
- **Prepare:**
 - ◊ Create an Interactive Pet of your own to show to students. Consider using a variety of inputs and outputs, and craft a body that uses materials that are readily accessible to students.
 - ◊ Lay out pens and paper for students to use while brainstorming.
- **Learning Goals:**
 - ◊ Create a plan for making an Interactive Pet.
- **Activity Plan:**
 - ◊ Gather pen and paper for brainstorming and sketching.
 - ◊ Interact with example Interactive Pets and their behavior.
 - ◊ Individual brainstorming and planning of Interactive Pet design.
 - ◊ Give and receive feedback from a partner on Interactive Pet plans.
- **Tips:**
 - ◊ Be as specific as possible with their plans, which will help them as they move into the building portion of the project.
 - ◊ Consider how detecting light level or pushing buttons can create storytelling opportunities for the pet.
 - ◊ When sharing plans with a partner, encourage incorporating some of their feedback into the design.
- **Standards:**
 - ◊ 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
- **Solutions:** Example pets and their code and behavior are included on the student pages.

Activity 2: Building Your Pet.

[↑ Back to Top](#)

- **Materials:** [Building Your Pet](#) student page
- **Prepare:**
 - ◊ Gather a variety of craft supplies for students to use in building their pet. There is a list on the student activity page.
- **Learning Goals:**
 - ◊ Craft a body that supports the Interactive Pet code.
- **Activity Plan:**
 - ◊ Revisit plans made in Activity 1 for making an Interactive Pet
 - ◊ Gather materials for making Interactive Pet bodies.
 - ◊ Build the pet body.
 - ◊ If time, move onto Activity 3 and begin programming pet behavior.
- **Tips:**
 - ◊ Make the LED screen visible and the buttons accessible by cutting holes in the body.
- **Standards:** None covered.

Activity 3: Building Your Pet.

[↑ Back to Top](#)

- **Materials:** [Adding Interactivity](#) student page

- Standards: none covered.

Activity 3: Building Your Pet.

[↑ Back to Top](#)

- **Materials:** [Adding Interactivity](#) student page
- **Prepare:**
 - ◊ Make available student's Interactive Pet bodies and craft materials for finalizing designs. (less than 5 minutes)
- **Learning Goals:**
 - ◊ Create inputs and outputs using micro:bit.
 - ◊ Experiment with different inputs and outputs and pick ones that best fit the designed pet.
 - ◊ Connect the inputs and outputs to a physical body.
- **Activity Plan:**
 - ◊ Gather materials and Interactive Pet bodies.
 - ◊ Program Interactive Pet behavior based on planning in Activity 1.
 - ◊ Make changes to pet body as needed.
 - ◊ Encourage sharing and testing with others.
- **Tips:**
 - ◊ Check ambient light level in the room with the [say](#) block.
 - ◊ Encourage making animations using multiple [LED display](#) blocks followed by [wait](#) blocks.
- **Standards:**
 - ◊ 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
- **Solutions:** Example pets and their code and behavior are included on the student page.

Activity 4: Telling a Story.

[↑ Back to Top](#)

- **Materials:** [Telling a Story](#) student page
- **Prepare:**
 - ◊ Gather a variety of craft supplies for students to use in building their pet. There is a list on the student activity page.
- **Learning Goals:**
 - ◊ Document Interactive Pets by communicating its inputs and outputs.
- **Activity Plan:**
 - ◊ Allow time for finalizing any physical making or programming tasks.
 - ◊ Document Interactive Pet inputs and outputs through writing or video.
 - ◊ Share documentation with a partner, small group, or entire class.
- **Standards:**
 - ◊ 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
- **Solutions:** Example pets and their code and behavior are included on the student page.

Activity 4: Telling a Story.

[↑ Back to Top](#)

- **Materials:** [Telling a Story](#) student page
- **Prepare:**
 - ◊ Gather a variety of craft supplies for students to use in building their pet. There is a list on the student activity page.
- **Learning Goals:**
 - ◊ Document interactive Pets by communicating its inputs and outputs.
- **Activity Plan:**
 - ◊ Allow time for finalizing any physical making or programming tasks.
 - ◊ Document Interactive Pet inputs and outputs through writing or video.
 - ◊ Share documentation with a partner, small group, or entire class.
- **Tips:**
 - ◊ Encourage storytelling beyond listing inputs and outputs. Describe an encounter with the Interactive Pet. What is the experience like? What happens when you try to communicate with the pet?
 - ◊ Sharing is an important part of the process. It's an opportunity for students to see different ways of creating interactions with micro:bit. Consider what mode of sharing works best for your students. One idea is to create an interactive petting zoo where students can go and meet each of the interactive pets alongside their documentation to know how to interact with them.
- **Standards:**
 - ◊ 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
 - ◊ 2-AP-19: Document programs in order to make them easier to follow, test, and debug.
- **Solutions:** Example pets and their code and behavior are included on the student page.

Feedback

Your feedback is critical to the success of this pilot. Please let us know how each lab went. For this lab, you might provide feedback such as:

- Were Lab 2 student projects distinct from their Lab 1 projects? If so, in what ways?
- Did students engage in the storytelling aspects of the lab? If so, how did they document and share their Interactive Pets?
- Did students build their bodies and then program their micro:bits, or move back-and-forth between the two? Describe their workflow.

[Submit Feedback](#)

Correlation with CSTA Standards

Correlation with CSTA Standards

- **2-CS-02: Design projects that combine hardware and software components to collect and exchange data.** Collecting and exchanging data involves input, output, storage, and processing. When possible, students should select the hardware and software components for their project designs by considering factors such as functionality, cost, size, speed, accessibility, and aesthetics. For example, components for a mobile app could include accelerometer, GPS, and speech recognition. The choice of a device that connects wirelessly through a Bluetooth connection versus a physical USB connection involves a tradeoff between mobility and the need for an additional power source for the wireless device.
- **2-AP-19: Document programs in order to make them easier to follow, test, and debug.** Documentation allows creators and others to more easily use and understand a program. Students should provide documentation for end users that explains their artifacts and how they function. For example, students could provide a project overview and clear user instructions. They should also incorporate comments in their product and communicate their process using design documents, flowcharts, and presentations.



BACK NEXT



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1837280; the U.S. Department of Education under grant number 5411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.





Lab 3: Game Play

Students pair up to create their own games to play with other classmates. They can just the micro:bit or additional craft materials (see [Activity 1](#) for details). Students are introduced to **variables** and the input/output (I/O) pins on the hardware. After completing their games, students introduce their projects to one for feedback to iterate on their design.

[Submit Feedback](#)

Pacing

This lab is designed for 2–3 class periods (85–155 minutes).

- **Daily Activity: Computing in the News** 5–15 minutes each day
- **Activity 1: Game Design**: 20–35 minutes
- **Activity 2: Build Your Game**: 30–60 minutes
- **Activity 3: Game Testing**: 30–45 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read [Computing in the News](#) for additional details and suggested news sources.

- **Prepare:**
 - ◊ Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: Game Design.

[↑ Back to Top](#)

- **Materials:** [Game Design](#) student page
- **Prepare:**
 - ◊ Provide physical example projects for students to play to help in their own brainstorming process.
 - ◊ Make available pens and paper for student pairs to use when brainstorming.
- **Learning Goals:**
 - ◊ Collaborate with a partner to create a project design.
 - ◊ Document the plan by sketching and writing on a piece of paper.
- **Activity Plan:**
 - ◊ Introduce the project by showcasing examples for students to try themselves.
 - ◊ Pair up students or ask students their own partner.
 - ◊ Encourage brainstorming and identifying multiple possible projects before selecting the one that they want to pursue.
 - ◊ Sketch and write down the game and its rules, its inputs and outputs, and any physical build that's required.
- **Tips:**
 - ◊ As much as possible, it's helpful to be explicit with project planning at this stage. It's okay if projects evolve as students build, but the next activity will be much easier if there's a clear plan identified early.
- **Standards:** None covered.

Activity 2: Build Your Game.

[↑ Back to Top](#)

- **Materials:** [Build Your Game](#) student page
- **Learning Goals:**
 - ◊ Use variables to keep track of score.
 - ◊ Create physical switches using micro:bit I/O pins, if applicable.
- **Activity Plan:**
 - ◊ Students review their plan from Activity 1 and make any changes.
 - ◊ Create your game with code and physical materials.
 - ◊ Encourage testing while building.
- **Standards:** None covered.
- **Solutions:** Intended code structures are included on the student activity page.

Activity 3: Game Testing.

[↑ Back to Top](#)

- **Materials:** [Game Testing](#) student page
- **Learning Goals:**
 - ◊ Give and receive feedback on game design.
 - ◊ Iterate on game design based on feedback from classmates.
- **Activity Plan:**
 - ◊ Finalize any game building before launching into feedback.
 - ◊ Student groups pair up to give and receive feedback.
 - ◊ Incorporate feedback from classmates into final game design.
- **Standards:**
 - ◊ **2-CS-01:** Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.

✖ Incorporate feedback from classmates into final game design.

• **Standards:**

✖ **2-CS-01:** Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.

Feedback

Your feedback is critical to the success of this pilot. Please let us know how each lab went. For this lab, you might provide feedback such as:

- In what ways did your students successfully collaborate during this lab? In what ways was this paired project challenging?
- Was student feedback on projects successful? If so, how? If not, please explain.
- Did students choose to make switches as seen in the Skeeball example? If so, about how many students chose this path?
- How many students did micro:bit-only games? How many used physical materials?

Submit Feedback

Correlation with CSTA Standards

2-CS-01: Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. The study of human-computer interaction (HCI) can improve the design of devices, including both hardware and software. Students should make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design their own components or interface (e.g., create their own controllers). Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and learnability. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.

BACK NEXT



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1338596, 1441075, and 1837260; the U.S. Department of Education under grant number 5411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Lab 4: Marvelous Movements

Students create unique moving creations using servo motors. They learn how to program them using servo commands, including `set servo to degrees`, `set servo to speed`, and `stop servo`. Then, students will make their own projects inspired by servo movement that builds on their learning in previous labs.

[Submit Feedback](#)

Pacing

This lab is designed for 2–4 class periods (65–135 minutes).

- **Daily Activity:** *Computing in the News* 5–15 minutes each day
- **Activity 1:** *Make It Move*: 25–50 minutes
- **Activity 2:** *Servo Project*: 35–70 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read *Computing in the News* for additional details and suggested news sources.

- **Prepare:**
 - Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: Make It Move.

[↑ Back to Top](#)

- **Materials:** *Make It Move* student page
- **Prepare:**
 - Set up a zone for all of the crafting materials that's easy for students to access.
 - Consider supplementing materials based on what you have available in your space.
 - Clear off work surfaces so students have space to program and build in the same location.
- **Learning Goals:**
 - Create movement by programming a servo motor.
 - Use a breakout board to control a servo motor using micro:bit.
- **Activity Plan:**
 - Walk through setting up the Bit Booster board as a class.
 - Make one servo move and attach that movement to a micro:bit input.
 - Make another servo movement connected to another input.
- **Tips:**
 - When troubleshooting the servos, check the following:
 - The direction of the servo wires (orange is positive, brown is negative).
 - The direction of the battery wires (red is positive, black is negative).
 - The direction the micro:bit is facing.
- **Standards:** None covered.

Activity 2: Servo Project.

[↑ Back to Top](#)

- **Materials:** *Servo Project* student page
- **Learning Goals:**
 - Learn how to control a servo motor using code.
 - Make an interactive project that uses servo motion.
 - Incorporate previous learning goals into a multi-faceted project.
- **Activity Plan:**
 - Students decide what project to make and share their ideas with a partner for feedback.
 - Students build their project using code and physical materials.
 - Students share their projects for feedback, and incorporate changes into the next version of their design.
- **Standards:** 2-CS-01. Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.

Feedback

Your feedback is critical to the success of this pilot. Please let us know how each lab went. For this lab, you might provide feedback such as:

- Did you experience any challenges working with servo motors? If so, what were they?
- Did students make a variety of projects with servos? If so, what were they? If not, why do you think this was the case?
- Were students able to move the servo motor as intended? If not, please explain their intended movement.

[Submit Feedback](#)

Correlation with CSTA Standards

Correlation with CSTA Standards

2-CS-01: Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. The study of human-computer interaction (HCI) can improve the design of devices, including both hardware and software. Students should make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design their own components or interface (e.g., create their own controllers). Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and learnability. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.



BACK NEXT



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1338596, 141075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Lab 5: Paper Stories

Students use the micro:bit I/O pins to wire up LEDs using copper tape. They first learn how to make a copper tape LED circuit without code. Then, they learn how to control the lights using **digital pin** blocks by making an LED blink and a custom paper craft project.

[Submit Feedback](#)

Pacing

This lab is designed for 2–4 class periods (90–175 minutes).

- **Daily Activity:** *Computing in the News* 5–15 minutes each day
- **Activity 1:** *LED It Glow*: 20–35 minutes
- **Activity 2:** *Blink*: 20–35 minutes
- **Activity 3:** *Paper Story*: 45–90 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read *Computing in the News* for additional details and suggested news sources.

- **Prepare:**
 - ◊ Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: LED It Glow.

[↑ Back to Top](#)

- **Materials:** *LED It Glow* student page
- **Prepare:**
 - ◊ Lay out materials for students to make simple copper tape circuits. Make sure that they are plentiful and not precious, as part of learning how to make a copper tape circuit is making mistakes.
 - ◊ Keep LEDs in a small container to keep them from dispersing widely.
- **Learning Goals:**
 - ◊ Make a simple circuit using an LED, coin cell battery, and copper tape.
 - ◊ Practice manipulating copper tape and making electrical connections.
- **Activity Plan:**
 - ◊ Students gather the materials they need to make a copper tape circuit.
 - ◊ Students map out and make a circuit.
 - ◊ If time, students make circuits with more than one LED.
- **Tips:**
 - ◊ Copper tape can be finicky. If possible, keep one continuous piece for each segment.
 - ◊ If two pieces need to be connected to one another, the top of the copper tape is slightly more conductive than the adhesive side so connecting two top pieces might make for a stronger connection.
 - ◊ If LED does not light up, try smoothing out the tape or checking the battery and LED direction. Also try a different LED. If looking to invest in tools, a multimeter and soldering iron can make troubleshooting easier.
- **Standards:** None covered.

Activity 2: Blink.

[↑ Back to Top](#)

- **Materials:** *Blink* student page
- **Prepare:**
 - ◊ Lay out materials in an easy to access manner.
 - ◊ Make accessible the practice circuits from Activity 1 as reference.
- **Learning Goals:**
 - ◊ Use code to control the LED by turning it on and off.
- **Activity Plan:**
 - ◊ Students make a copper tape circuit using a micro:bit.
 - ◊ Students program the light to turn on and off.
 - ◊ Students create custom light blink patterns and add inputs to control these patterns.
- **Standards:**
 - ◊ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Activity 3: Paper Story.

[↑ Back to Top](#)

- **Materials:** *Paper Story* student page
- **Prepare:**
 - ◊ Make available paper supplies and tools.
 - ◊ Clear surfaces so programming and physical making can happen side-by-side.
- **Learning Goals:**

Activity 3: Paper Story.

[↑ Back to Top](#)

- **Materials:** [Paper Story student page](#)
- **Prepare:**
 - ✦ Make available paper supplies and tools.
 - ✦ Clear surfaces so programming and physical making can happen side-by-side.
- **Learning Goals:**
 - ✦ Create light patterns using code.
 - ✦ Control multiple LEDs simultaneously.
 - ✦ Craft a narrative around the lights.
- **Activity Plan:**
 - ✦ Students can build off of their projects from Activity 2 or start an entirely new circuit.
 - ✦ Encourage planning and mapping out LED placement.
 - ✦ Students create with code and paper to make their stories.
 - ✦ Once complete, consider wiring up their projects for display.
- **Standards:**
 - ✦ **2-CS-03:** Systematically identify and fix problems with computing devices and their components.

Feedback

Your feedback is critical to the success of this pilot. Please let us know how each lab went. For this lab, you might provide feedback such as:

- What types of projects did students make with paper circuits?
- What challenges did students encounter when building their paper circuits?
- What challenges did students encounter when programming their LEDs?

[Submit Feedback](#)

Correlation with CSTA Standards

- **2-CS-03: Systematically identify and fix problems with computing devices and their components.** Since a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to devices connected to it. Just as pilots use checklists to troubleshoot problems with aircraft systems, students should use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.

[BACK](#) [NEXT](#)



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138598, 1441076, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Lab 6: Making with Multiples

Radio communication is one of micro:bit's features, allowing two or more micro:bits to send and receive messages to one another. In this lab, students will learn how to send and receive messages by sending strings and integers, and then make a project with a partner that uses two micro:bits in communication with one another.

[Submit Feedback](#)

Pacing

This lab is designed for 2–4 class periods (65–110 minutes).

- **Daily Activity: Computing in the News** 5–15 minutes each day
- **Activity 1: Send a Signal** 20–35 minutes
- **Activity 2: Interactive Communication** 40–60 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read [Computing in the News](#) for additional details and suggested news sources.

- **Prepare:**
 - Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: Send a Signal.

[↑ Back to Top](#)

- **Materials:** [Send a Signal](#) student page
- **Prepare:**
 - Students will partner up for the activity, so either decide student pairs or ask students to find a partner.
 - This project is focused on learning how to use radio commands with micro:bit, through students are welcome to incorporate physical materials if it adds to their project idea.
- **Learning Goals:**
 - Communicate between two micro:bits via radio.
 - Make an interactive project with two communicating micro:bits.
- **Activity Plan:**
 - Students pair up and work through radio communication examples.
 - Students practice sending string messages and then send integers.
 - Students can begin planning their collaborative project if time allows.
- **Tips:**
 - Ensure each pair is on a separate group channel to avoid sending mixed signals.
 - Micro:bits shouldn't be too far away from one another to be able to communicate via Bluetooth.
- **Standards:** None covered.

Activity 2: Interactive Communication.

[↑ Back to Top](#)

- **Materials:** [Interactive Communication](#) student page
- **Prepare:**
 - Lay out materials in an easy to access manner.
- **Learning Goals:**
 - Implement a unique project that uses radio communication.
- **Activity Plan:**
 - Students build their radio communication projects using code and physical materials.
 - Students test their projects and share them with classmates.
 - If time allows, students expand the number of micro:bits on a group channel and make a more complex project.
- **Standards:** None covered.

Feedback

Your feedback is critical to the success of this pilot. Please let us know how each lab went. For this lab, you might provide feedback such as:

- How did students incorporate radio communication in to a project?
- Did students extend beyond working in pairs and make projects with more than two people? If so, what did they make?
- Was radio communication consistent and reliable? If not, please explain.

[Submit Feedback](#)



Lab 7: Collaborative Chain Reaction

[Toggle developer todos/comments \(red boxes\)](#)

A chain reaction (or Rube Goldberg machine) is a series of causes and effects that often results in the accomplishment of a simple task. The sequence of events is often absurd or convoluted, but that is the beauty in creating one of these machines. In this lab, students will work together to create a chain reaction machine that blends micro:bit with analog materials. Students map out a plan, run tests on their segments, and perform collectively the chain reaction machine.

[Submit Feedback](#)

Pacing

This lab is designed for 3–5 class periods (65–110 minutes).

- **Daily Activity: Computing in the News** 5–15 minutes each day
- **Activity 1: Make a Plan**: 20–35 minutes
- **Activity 2: Build Connections**: 40–90 minutes
- **Activity 3: Test Sequences**: 20–35 minutes
- **Activity 4: Launch Day**: 40–60 minutes

Daily Activity: Computing in the News.

[↑ Back to Top](#)

Every day of BJC should begin with a 5-minute student presentation on a recent news article about computing. This helps build a foundation for considering the impacts of computing technologies on employment, issues of bias and accessibility, and tradeoffs between public access to information and security. Read [Computing in the News](#) for additional details and suggested news sources.

- **Prepare:**
 - ✦ Assign to one student or a pair of students a recent news article about computing for them to read and then summarize for the class during the next class period.

Activity 1: Make a Plan.

[↑ Back to Top](#)

- **Materials:** [Make a Plan](#) student page
- **Prepare:**
 - ✦ Pick a place for students to build their chain reactions. This can take up quite a lot of room, so pick a space with ample room for building as well as additional materials.
 - ✦ Place chain reaction materials nearby so students have easy access to balls, ramps, and other building materials.
 - ✦ Set a date for the chain reaction finale to invite friends, family, and other students and teachers to join in the celebration.
- **Learning Goals:**
 - ✦ Make a plan for a multi-step project with their partner.
 - ✦ Collaborate with neighboring teams on how segments will connect.
- **Activity Plan:**
 - ✦ Students see what materials are available to them before making a plan.
 - ✦ Pairs articulate what materials they will need for their segment, how they will utilize both micro:bits, and how their segment will connect with neighboring segments.
- **Tips:**
 - ✦ Tables are great building surfaces, and each pair of students can use one table for building. Place tables in a zig-zag (non-linear) formation to add more complex movements between segments.
 - ✦ If possible, keep student projects undisturbed for the duration of their building. If that isn't possible, consider having them build on pieces of cardboard that then can be moved at the end of class and stored safely.
 - ✦ Make tape marks on the table where connections will occur between two pairs of students.
- **Standards:**
 - ✦ 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.

Activity 2: Interactive Communication.

[↑ Back to Top](#)

- **Materials:** [Build Connections](#) student page
- **Prepare:**
 - ✦ Make available any materials gathered from Activity 1, along with any planning materials student pairs created.
- **Learning Goals:**
 - ✦ Students use computation in at least two different ways.
 - ✦ Students integrate their computational elements with physical materials in their chain reaction segments.
- **Activity Plan:**
 - ✦ Students gather building materials to make into their chain reaction segments.
 - ✦ Students make their segments and connections to other team segments.
 - ✦ Students test their segment components to see if their cause-and-effect works as intended.
- **Standards:**
 - ✦ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Activity 3: Test Sequences.

[↑ Back to Top](#)

- **Materials:** [Test Sequences](#) student page
- **Prepare:**
 - ✦ Make available student chain reaction sequences from Activity 2, along with any planning materials student pairs created.

Activity 3: Test Sequences.

[↑ Back to Top](#)

- **Materials:** [Test Sequences student page](#)
- **Prepare:**
 - ✦ Make available student chain reaction sequences from Activity 2, along with any planning materials student pairs created.
- **Learning Goals:**
 - ✦ Students test their chain reaction segments and identify inconsistencies in performance.
 - ✦ Students make changes to their chain reaction based on the inconsistencies identified during testing.
 - ✦ Students perform similar testing at connection points between partner groups.
- **Activity Plan:**
 - ✦ Students finalize details of their chain reaction.
 - ✦ Students run tests on their chain reaction segments, taking note of where there are inconsistencies or incomplete tests.
 - ✦ Students use this information to make changes to their design, and then run more tests to see if new issues arise.
 - ✦ Students also test at connection points between two different pairs to ensure their handoff is consistent.
- **Standards:**
 - ✦ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Activity 4: Launch Day.

[↑ Back to Top](#)

- **Materials:** [Launch Day student page](#)
- **Prepare:**
 - ✦ Make available student chain reaction sequences from Activity 2, along with any planning materials student pairs created.
 - ✦ Celebrate the chain reaction finale with friends, family, and the school. Invite others to the event.
- **Learning Goals:**
 - ✦ Students test their chain reaction segments and identify inconsistencies in performance.
 - ✦ Students make changes to their chain reaction based on the inconsistencies identified during testing.
 - ✦ Students perform similar testing at connection points between partner groups.
- **Activity Plan:**
 - ✦ Students finalize details of their chain reaction.
 - ✦ Students run tests on their chain reaction segments, taking note of where there are inconsistencies or incomplete tests.
 - ✦ Students use this information to make changes to their design, and then run more tests to see if new issues arise.
 - ✦ Students also test at connection points between two different pairs to ensure their handoff is consistent.
- **Tips:**
 - ✦ Before launching the chain reaction, ask all students to clean up their workspaces of any extra materials.
 - ✦ Find a place in the room where students and others who are watching can see the entire chain reaction. If that's challenging to do, consider filming the chain reaction set-off so that others can see it at a later time.
 - ✦ Consider adding an announcer who will narrate the chain reaction finale. This can be you, a student, or someone else. It will help for those who can't see the small up-close details to hear what's happening.
 - ✦ Ask each student pair to present what will happen at each stage of their chain reaction segment.
 - ✦ If a segment of a student's chain reaction doesn't work as intended, ask them to help their segment along by making a small change.
- **Standards:**
 - ✦ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

[Submit Feedback](#)

Correlation with CSTA Standards

- **2-CS-03: Systematically identify and fix problems with computing devices and their components.** Since a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to devices connected to it. Just as pilots use checklists to troubleshoot problems with aircraft systems, students should use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.
- **2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.** Collaboration is a common and crucial practice in programming development. Often, many individuals and groups work on the interdependent parts of a project together. Students should assume pre-defined roles within their teams and manage the project workflow using structured timelines. With teacher guidance, they will begin to create collective goals, expectations, and equitable workloads. For example, students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then distribute tasks and roles among members of the team and assign deadlines.

[BACK](#) [NEXT](#)



The Beauty and Joy of Computing by University of California, Berkeley and Education Development Center, Inc. is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. The development of this site has been funded by the National Science Foundation under grant nos. 1138596, 1441075, and 1837280; the U.S. Department of Education under grant number S411C200074; and the Hopper-Dean Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or our other funders.



Bibliography

- [1] Autodesk. *Explore Micro:bit With Tinkercad*. URL: <https://blog.tinkercad.com/explore-microbit-with-tinkercad>.
- [2] *Beetle Blocks*. URL: <http://beetleblocks.com/>.
- [3] UC Berkeley. *Snap!: Build Your Own Blocks*. URL: <https://snap.berkeley.edu>.
- [4] UC Berkeley. *The Beauty and Joy of Computing*. 2022. URL: <https://bjc.berkeley.edu/>.
- [5] Paulo Blikstein. “Digital fabrication and ‘making’ in education: The democratization of invention”. In: *FabLabs: Of machines, makers and inventors* 4.1 (2013), pp. 1–21.
- [6] *Curriculum planning tool*. Aug. 2019. URL: <https://www.birdbraintechnologies.com/resources/curriculum-planning-tool/> (<https://www.birdbraintechnologies.com/resources/curriculum-planning-tool/>).
- [7] Code.org Middle School: CS Discoveries. 2022. URL: <https://code.org/educate/curriculum/middle-school>.
- [8] Harvard School of Education. *ScratchEd Creative Computing Curriculum Guide*. URL: <https://scratched.gse.harvard.edu/resources/scratch-curriculum-guide.html>.
- [9] The Tinkering Studio at the Exploratorium. *Chain Reaction*. 2017. URL: <https://www.exploratorium.edu/tinkering/projects/chain-reaction>.
- [10] Scratch Foundation. *Scratch micro:bit Coding Cards*. URL: <https://resources.scratch.mit.edu/www/cards/en/microbit-cards.pdf>.
- [11] Paulo Freire. *Pedagogy of the oppressed*. New York: Continuum, 1970.
- [12] Friedrich Froebel. *The education of man*. New York: Appleton, 1887.
- [13] Dan Garcia, Brian Harvey, and Tiffany Barnes. “The Beauty and Joy of Computing”. In: *ACM Inroads* 6.4 (Nov. 2015), pp. 71–79. ISSN: 2153-2184. DOI: 10.1145/2835184. URL: <https://doi.org/10.1145/2835184>.
- [14] Idit Ed Harel and Seymour Ed Papert. *Constructionism*. Ablex Publishing, 1991.
- [15] MakeCode. *Courses*. URL: <https://makecode.microbit.org/courses>.

- [16] BBC micro:bit. *micro:bit Overview*. URL: <https://microbit.org/get-started/user-guide/overview/>.
- [17] *MicroBlocks*. URL: <https://microblocks.fun/>.
- [18] Maria Montessori. *The montessori method*. New York: Frederick Stokes Co., 1912.
- [19] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. USA: Basic Books, Inc., 1980. ISBN: 0465046274.
- [20] Mike Petrich, Karen Wilkinson, and Bronwyn Bevan. “It looks like fun, but are they learning?” In: *Design, make, play*. Routledge, 2013, pp. 68–88.
- [21] *Plezmo*. URL: <https://www.plezmo.com/>.
- [22] Mitchel Resnick and Eric Rosenbaum. “Designing for tinkerability”. In: *Design, make, play: Growing the next generation of STEM innovators* (2013), pp. 163–181.
- [23] Mitchel Resnick et al. “Scratch: programming for all”. In: *Communications of the ACM* 52.11 (2009), pp. 60–67.
- [24] Susan H. Rodger et al. “Engaging Middle School Teachers and Students with Alice in a Diverse Set of Subjects”. In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. SIGCSE ’09. Chattanooga, TN, USA: Association for Computing Machinery, 2009, pp. 271–275. ISBN: 9781605581835. DOI: 10.1145/1508865.1508967. URL: <https://doi.org/10.1145/1508865.1508967>.
- [25] Emmanuel Schanzer et al. “Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap”. In: *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. SIGCSE ’15. Kansas City, Missouri, USA: Association for Computing Machinery, 2015, pp. 616–621. ISBN: 9781450329668. DOI: 10.1145/2676723.2677238. URL: <https://doi.org/10.1145/2676723.2677238>.
- [26] BirdBrain Technologies. *Finch Robot*. URL: <https://www.birdbraintechnologies.com/finch/>.
- [27] Tinkercademy. *Micro:bit Tutorials and Store*. URL: <https://tinkercademy.com/microbit/>.
- [28] *Turtlestitch - Coded Embroidery*. URL: <https://www.turtlestitch.org/>.
- [29] Jeannette M Wing. “Computational thinking”. In: *Communications of the ACM* 49.3 (2006), pp. 33–35.