

Springfield Public Schools
Springfield, New Jersey 07081

PLTW Computer Science for Innovators and Makers

Grade 6

Curriculum Guide

2022

This curriculum may be modified through varying materials, techniques, and strategies as per an individual student's IEP.

Springfield Board of Education
Approved July 25, 2022

Springfield Public Schools
Springfield, New Jersey

Dr. Rachel Goldberg
Superintendent of Schools

Erica Scudero
Assistant Superintendent

Curriculum Developed By:

Tim Kielty ~ Middle School Principal

Gregory Salmon ~ STEM Supervisor

Christina Rodriguez ~ Mathematics Teacher

Springfield Board of Education
PO Box 210
Springfield, New Jersey 07081

"Cultivating compassionate and extraordinary learners"

Rachel Goldberg
Superintendent of Schools
(973) 376-1025

Michelle Calas
School Business Administrator/
Board Secretary
(973) 376-1025

Resolution: School Government
Introduced By: Mr. Miller
Seconded By: Mrs. Gamarekian
Adopted: July 25, 2022

Curriculum Adoption

To adopt the following curriculum and courses for the 2022-2023 school year:

- Revised Elementary Curriculum due to revised state standards.
  - K-5 Social Studies
- Revised Middle School Curriculum due to revised state standards.
  - Grades 6-8 Social Studies
- Revised High School Curriculum due to revised state standards.
  - AP Macroeconomics
  - AP European History
  - AP US Government & Politics
  - AP US History
  - Contemporary Global Studies
  - Diversity Studies
  - Holocaust & Genocide
  - Sociology
  - US History to 1900
  - World History
  - World History Honors
  - 20th Century & Beyond
- Revised Middle School Courses due to new standards in Computer Science
  - PLTW Gateway: Computer Science for Innovators and Makers
  - PLTW Gateway: App Creators
- Revised High School Courses due to new standards in Computer Science
  - Introduction to Java
  - AP Computer Science Principles
  - Advanced Computer Science - Data Structures: Java
- New Elementary Courses
  - Kindergarten Computer Science
  - Grade 1 Computer Science
  - Grade 2 Computer Science
  - Anticipated - moving PLTW Launch down to Grade 3
  - PLTW Launch Grade 4
  - PLTW Launch Grade 5
- New Middle School Elective Course
  - Forensics

- New Elective Courses at FMG
  - Creative Writing
  - Drama/Theater/Playwriting
  - Journalism
  - Mock Trial
  - Model UN
  - Mysteries in History
  - Project Citizen
- New ELA Courses at FMG
  - Accelerated Communication Arts Grade 7
  - Accelerated Communication Arts Grade 8
- New Elective Courses at JDHS:
  - African American Studies

I, Michelle Calas, School Business Administrator/Board Secretary of the Springfield Board of Education, hereby certify that the extract from the minutes of a meeting of the Board of Education duly called and held on July 25, 2022 has been compared by me with the original minute book of said Board of Education, and is a true, complete and exact copy therefor, as far as the same relates to the subject matter referred to said contract.

IN WITNESS THEREOF, I have hereunto set my hand and seal of the Springfield Board of Education, this 26th of July 2022.

SEAL

Michelle Calas
School Business Administrator/Board Secretary

Table of Contents

New Jersey Student Learning Standards: Pages 4-5

General Preparation and Consideration: Pages 6-13

Course Outline: Page 14

Lesson 1 – Blink!: Pages 15-58

Lesson 2 – The Ins and Outs: Pages 59-112

Lesson 3 – Program the Physical World: Pages 113-130

PLTW Curriculum Revisions: Pages 131-134

PLTW Curriculum Framework: Pages 135-153

New Jersey Student Learning Standards

8.1 Computer Science

Computing Systems

People interact with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form.

- 8.1.8.CS.1: Recommend improvements to computing devices in order to improve the ways users interact with the devices.
- 8.1.8.CS.2: Design a system that combines hardware and software components to process data.
- 8.1.8.CS.3: Justify design decisions and explain potential system trade-offs.
- 8.1.8.CS.4: Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.

Networks and the Internet

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world.

- 8.1.8.NI.1: Model how information is broken down into smaller pieces, transmitted as addressed packets through multiple devices over networks and the Internet, and reassembled at the destination.
- 8.1.8.NI.2: Model the role of protocols in transmitting data across networks and the Internet and how they enable secure and errorless communication.
- 8.1.8.NI.3: Explain how network security depends on a combination of hardware, software, and practices that control access to data and systems.
- 8.1.8.NI.4: Explain how new security measures have been created in response to key malware events.

Impacts of Computing

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and, in turn, computing influences new cultural practices.

- 8.1.8.IC.2: Describe issues of bias and accessibility in the design of existing technologies.

Data & Analysis

Computing systems exist to process data. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important. Data is collected and stored so that it can be analyzed to better understand the world and make more accurate predictions.

- 8.1.8.DA.1: Organize and transform data collected using computational tools to make it usable for a specific purpose.
- 8.1.8.DA.2: Explain the difference between how the computer stores data as bits and how the data is displayed.
- 8.1.8.DA.3: Identify the appropriate tool to access data based on its file format.

Algorithms & Programming

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems.

- 8.1.8.AP.1: Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.
- 8.1.8.AP.2: Create clearly named variables that represent different data types and perform operations on their values.
- 8.1.8.AP.4: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 8.1.8.AP.6: Refine a solution that meets users' needs by incorporating feedback from team members and users.
- 8.1.8.AP.9: Document programs in order to make them easier to follow, test, and debug.

Career Readiness, Life Literacies, and Key Skills (Standard 9):

- 9.2.8.CAP.5 - Develop a personal plan with the assistance of an adult mentor that includes information about career areas of interest, goals and an educational plan.
- 9.2.8.CAP.10 - Evaluate how careers have evolved regionally, nationally, and globally.
- 9.2.8.CAP.12 - Assess personal strengths, talents, values, and interests to appropriate jobs and careers to maximize career potential.
- 9.2.8.CAP.19 - Relate academic achievement, as represented by high school diplomas, college degrees, and industry credentials, to employability and to potential level
- 9.4.8.CT.1 - Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).
- 9.4.8.IML.4 - Ask insightful questions to organize different types of data and create meaningful visualizations.
- 9.4.8.IML.7 - Use information from a variety of sources, contexts, disciplines, and cultures for a specific purpose (e.g., 1.2.8.C2a).
- 9.4.8.TL.3 - Select appropriate tools to organize and present information digitally.
- 9.4.8.TL.5 - Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.

TEACHER EDITION

PLTW GATEWAY

Computer Science for Innovators and Makers

General Preparation and Considerations

FEEDBACK ON ACTIVITIES, PROJECTS, AND PROBLEMS

Thank you for being a PLTW Computer Science for Innovators and Makers teacher! Below are the links to two surveys that you can use to provide feedback about your teaching experience.

The top link should be used to provide feedback for each lesson.

The bottom link is to provide feedback on the course overall, and should be completed at the conclusion of the course. Your feedback is confidential and helps inform improvements to PLTW curriculum and professional development.

Problem Feedback

PLTW Gateway Teacher Survey

COURSE PREFACE

Computer Science for Innovators and Makers (IM) teaches students that programming goes beyond the virtual world into the physical world. Students are challenged to creatively use sensors and actuators to develop systems that interact with their environment. While designing algorithms and using computational thinking practices, students code and upload programs to microcontrollers that perform a variety of authentic tasks. The unit broadens students' understanding of computer science concepts through meaningful applications. Teams select and solve a personally relevant problem related to wearable technology, interactive art, or mechanical devices.

COURSE MATERIALS

- PLTW Gateway Notebook
- Computer with internet

COURSE RESOURCES

- Design Process
- Pair Programming
- Pair Programming Poster
- Input Output
- Input Output Poster
- Microbit Controller
- Microbit Controller Poster

NATIONAL AND STATE STANDARDS ALIGNMENT

For the most up-to-date Standards Alignment, please visit **PLTW's Alignment to Standards**.

Safety Considerations

Refer to the **safety guidelines** provided by the Micro:bit Educational Foundation.

Ethical Considerations

Whether communicating in a virtual or physical environment, students should demonstrate positive digital citizenship skills. The projects students create in this unit should be appropriate and inoffensive to others. Students are responsible for the digital content and programs they create.

If your school has an Acceptable Use Policy or Responsible Use Policy regarding the use of school technology, you may choose to review it with your students and discuss the importance of becoming responsible digital citizens.

Student Accommodations

- Students with visual needs may benefit from a **Google Chrome browser extension** to convert the website to high-contrast color schemes. Other web browsers may have similar extensions to assist with converting the programming environment to high-contrast color schemes.
- Consider pairing students who have difficulty distinguishing colors or are color-blind with students who are not color-blind. You might also suggest to color-blind students that they wrap small strips of tape around each input/output component's wires and write the color of the wire on the tape to label it.

Unit Preparation

If you have multiple classes or individuals using the same computers, previous student work will be visible in the programming environment. To limit this, consider instructing either the outgoing or incoming student to go projects and select new project.

Tech Tips

As your students use their web browser to access the Computer Science for Innovators and Makers curriculum and the Microsoft MakeCode programming environment, they may encounter browser loading issues. Selecting the refresh icon in their web browser's toolbar may resolve the following situations:

- Curriculum - If the images within an APB do not appear to load, refresh the page in the browser.
- Microsoft MakeCode programming environment - The drawers or buttons within the programming environment's user interface may occasionally become unresponsive. Refreshing the page in the browser may resolve the issue. In most cases, the code will be restored and the open program will reload. Students will likely not lose their code when the browser is refreshed.

A reset button is located on the back of your students' micro:bits. Unlike reset buttons on most microcontrollers, this button will not wipe the firmware on the micro:bit. It's safe for your students to press the reset button if they want to restart the current program stored on the micro:bit's processor. For more information and cautions regarding the use of the reset button, review the **micro:bit device resource**.

There is more than one version of the micro:bit. You may have the micro:bit that is shown in Figure 1, where the compass and accelerometer sensors are separate components. Or you may have the version shown in Figure 2, where the compass and accelerometer sensors are combined as a motion sensor that can perform both functions. The user interaction with both versions is not affected by this difference.

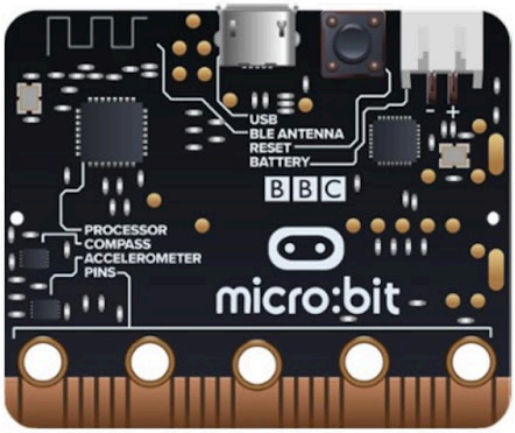


Figure 1. micro:bit Back with Compass and Accelerometer Separate

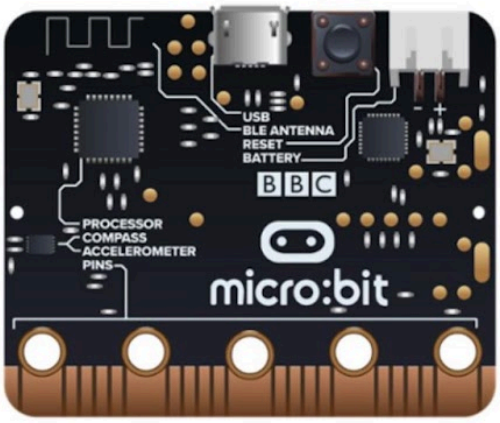


Figure 2. micro:bit Back with Compass and Accelerometer Combined

Coding Outside of Class

- Your students may be interested in creating additional programs or projects outside of class using the Microsoft MakeCode programming environment. You can encourage them to use this free tool and provide them with the website's address: **https://makecode.microbit.org**
- You may choose to send home a note or email to parents and families explaining the Microsoft MakeCode programming environment and include the web address. Students can also share their programs in the programming environment as a web link or embedded code. You can find instructions for sharing code in the curriculum as well as at: **https://makecode.microbit.org/share**

Whitelisting Sites

If your school district restricts the use of the Microsoft MakeCode websites, contact your system administrator and request to whitelist the web addresses posted at: **https://makecode.microbit.org/faq**

PLTW Gateway Notebook

For each APB, determine where you would like your students to document their work, answer formative assessment questions, and answer conclusion questions. Which items you collect and assess is up to you. The curriculum assumes students will document all work in their PLTW Gateway Notebooks, as indicated by PLTW Gateway Notebook call-outs. However, where and how students document their work is up to you. For example, you may choose to use other documentation tools, such as a digital notebook, a Word document to be shared with you, or an online collaboration service (such as Google Drive). Communicate your plan to your students when facilitating the APB.

Professional Development Opportunities

Professional development opportunities are available to you from the Professional Development page of myPLTW.org. To begin learning, visit myPLTW.org and select the **Professional Development** icon.

myPLTW.org 〉

Community Forums

The Community tool allows users to create posts, ask questions, and share files within their Community groups. As a PLTW teacher, you will automatically be subscribed to Community groups based on the course(s) for which you are certified.

As a Community group member, you have the opportunity to post comments, tag posts, bookmark, and follow other members of your group. The Community environment is provided to facilitate a professional learning community among teachers who teach the same course across the country. Use this forum to ask questions, share best practices and lessons learned, and provide instructional support to your community of professionals.



Instructional LEAP Award 2019

Pair Programming with Code Tracing Charts - Part 1 (Video) 〉

LEAP Video Community Guide 〉

If you have a topic that could benefit from a LEAP video, you can submit a request using this **LEAP Video Suggestion "Box."**

To access the Community tool, visit myPLTW.org and select the **Community** icon.

myPLTW.org 〉

PROJECT LEAD THE WAY PLTW

PLTW Gateway

Computer Science for Innovators and Makers | Course Outline

Have you ever wondered how code can be used in wearable tech, art exhibits, or mechanical devices?

Students learn about programming for the physical world by blending hardware design and software development. Using microcontrollers with inputs and outputs, they develop code that brings their physical designs to life.

It's time to become an innovator and maker using physical computing!

Computer Science for Innovators and Makers teaches students that programming goes beyond the virtual world into the physical world. Students are challenged to creatively use sensors and actuators to develop systems that interact with their environment. Designing algorithms and using computational thinking practices, they code and upload programs to microcontrollers that perform a variety of authentic tasks. The unit broadens students' understanding of computer science concepts through meaningful applications. Teams select and solve a personally relevant problem related to wearable technology, interactive art, or mechanical devices.

Computer Science for Innovators and Makers Lesson Summary

Lesson 1	Blink!
Lesson 2	The Ins and Outs
Lesson 3	Program the Physical World

Lesson 1: Blink!
Students begin to explore the capabilities of physical computing systems with The Digital Dive game, an engaging, live-action activity where students "become" computer parts and transmit commands. They learn to use algorithmic thinking as they prepare to code. Students use block-based coding to create, download, and upload programs to the micro:bit microcontroller. They learn processes and gain skills to debug programs starting with pre-bugged programs. They apply these skills to their own project where they code a blinking message that includes text, images such as emojis, and animation.

Lesson 2: The Ins and Outs
In this lesson, students explore a variety of sensors and actuators to use as inputs and outputs in physical computing projects. Using different materials to transfer electrical signals, such as conductive thread, alligator clips, conductive paint, and copper tape, students create their own input device—a sensor or switch—to interact with a program they develop on the microcontroller. They use these skills in the lesson's project to design, develop, and program a system to protect safes and secrets.

Lesson 3: Program the Physical World
Within teams, students become innovators and makers. Teams apply their physical computing knowledge and skills as they design and create one of three problem options:

- A wearable safety device someone might use when completing a physical activity outside at night
- An engaging art installation to help improve a community space
- A useful mechanical dispenser for a person or animal who needs assistance to retrieve an object

Teams collaborate and learn that solving authentic problems involves the unit content knowledge, as well as skills from other disciplines, such as communications, mathematics, and science.

TEACHER EDITION

Lesson 1

Blink! – Quick Guide

Preface

In Lesson 1, students begin to explore the capabilities of physical computing systems with The Digital Dive game, an engaging, live-action activity where students "become" computer parts and transmit commands. They learn to use algorithmic thinking as they prepare to code. Students use block-based coding to create and download programs to the micro:bit microcontroller. They learn processes and gain skills to debug programs starting with pre-bugged programs. They apply these skills to their own project where they code a blinking message that includes text, images such as emojis, and animation.

Essential Questions

1. How do you express yourself and your creativity through computer science?
2. How can algorithmic thinking skills be used across multiple disciplines?
3. How can computer programs solve problems?

Activity, Project, or Problem	Days
Activity 1.1 The Brain	2–3
Activity 1.2 What To Do	3
Activity 1.3 How To Do It	3
Activity 1.4 Crush the Bug	2
Project 1.5 The Blinking Message	4
Lesson 1 Assessment	1
Total	**15–16 Days**

TEACHER EDITION

PLTW GATEWAY

Activity 1.1

The Brain

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 2.1A Collect, process, and analyze real or simulated data.

LO 2.5A Describe the hardware components of an electronic device and how they interact with software and the environment.

LO 2.5B Identify methods in which electronic devices communicate with each other.

LO 3.5B Describe the role, connections between disciplines, and impact of engineering, biomedical science, and/or computer science on society.

KNOWLEDGE AND SKILLS

KS

KS 2.1A1 Understand that computers enable rapid processing of information.

KS 2.5A1 Describe the modular components of computer hardware, particularly input devices, processors, memory, and output devices.

KS 2.5A2 Describe the interaction between computing devices and the surrounding environment.

KS 2.5B1 Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth® technology.

KS 3.5B2 Describe the impact of engineering, biomedical science, or computer science on invention and innovation.

ESSENTIAL QUESTION

EQ

EQ 1.1 How can algorithmic thinking skills be used across multiple disciplines?

RESOURCES

Activity 1.1 Answer Key

Preparation

- Identify or prepare a clear, open space for students to do The Digital Dive.
- Sort the Digital Dive cards according to icon and by color.
- Have the PLTW Gateway Notebooks ready for distribution.
- For details regarding common preparation needs across multiple APBs, see **Unit Preparation**.

Safety Considerations

Before you begin The Digital Dive, choose an open area for the activity or move all possible obstacles away from the teams and "data transfer" area. As students complete The Digital Dive, remind them that while they should try to send information from one team to the next as quickly as possible, they need to be careful to avoid obstructions and classmates in the area.

Student Accommodations

Students who are unable to move throughout the room during The Digital Dive, can still participate in various teams by helping select appropriate cards or performing an output, such as vocalization as a member of the Speakers team.

Activity Time: 2–3 Days

Possible Student Misconceptions

- Computer science is just programming or coding.
- Computer science knowledge is not appropriate for "average" students.
- Computer scientists or software engineers just make video games.
- Computer science equates to computer literacy - the understanding of using basic computer functions.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

What's a Computer Scientist

Students will begin the activity thinking about what computer science means to them. This will help them activate prior knowledge. The course provides a list of prompts for students to think about. The prompts will help students consider what a computer scientist is and what they do. Allow students time to reflect on their thoughts about computer science and computer scientists by responding to the prompt in the APB. If you plan to implement a flipped class, students could complete their response to this prompt outside of class.

You may also choose to do a KWL (Know-Want to Know-Learned) chart as a class or individually. This could be done on large chart paper with sticky notes or on KWL chart handouts. Place the student responses to the opening prompt under the Know column. Allow them to include anything they currently know about computer science. This will help identify misconceptions and prior knowledge. If you use the KWL approach, then you will want to ask students what they want to learn about computer science. Complete the Want to Know column on the chart as a class or individually.

Discussion: After students have completed their responses to the prompt, invite them to share their thoughts as you lead the class in a short discussion.

- Encourage students to share their thoughts without correcting possible misconceptions they have about computer science or computer scientists.
- You may choose, however, to ask them additional probing questions to get them to think deeper and expand on their current understanding of computer science.
- Students will explore these terms throughout the lesson and later reflect on what they've learned compared to these initial thoughts.

Video: Unit Introduction

When viewing the video as a whole class, you may choose to provide students with more introduction than the APB text. Ask students what they think physical computing might involve. Before they view the video, they can share their guesses to start thinking about the term.

Discussion: Discuss the video as a class and have students read the text below it.

- Ask students to explain what physical computing means.
- How does this compare to virtual computing?
- Begin asking students clarifying questions about computer science and physical computing to start rethinking possible misconceptions that you identified from their responses to the first question in the APB.
  For example, besides developing computer games, what else might be part of computer science? How do you think computer science impacts our daily lives?

Discuss the video as a class and have students read the text below it.

The Digital Dive

After students have read the introduction for this activity, complete The Digital Dive as a class.

To review the organization and flow of the digital dive activity, view the Digital Dive video.



The Digital Dive video

Digital Dive Instructions

1. Assign students, as evenly as possible, to six teams.
   a. **User**
   b. **Mouse**
   c. **Keyboard**
   d. **Processor**
   e. **Speakers**
   f. **Monitor**
2. Four of the teams get a set of cards.
   a. **User** - all Action Cards (include the Bluetooth® Connection cards only when the activity is played with a Bluetooth® connection)
   b. **Mouse** - Signal Cards
      i. 1 Play Music Card
      ii. 1 Pause Music Card
      iii. 1 Increase Volume Card
      iv. 1 Decrease Volume Card
      v. 5 Click Emoji Cards
   c. **Keyboard** - Signal Cards
      i. 26 Letter Key Press Cards
   d. **Processor** - all Program Cards
3. The **Speakers** team will hum or whistle a song of their choice as output.
4. The **Monitor** team will write their output on a whiteboard or paper with a marker.
5. You can give a variety of tasks to the **User** team to start the sequence, but here is a suggestion: Type "HELLO WORLD" with a smiley face emoji at the end and play a song when done. Have the user turn the volume up and down while the speakers keep humming/whistling for fun.
6. The **User** team must have one person run the Action card to either the **Keyboard** or **Mouse** teams.
7. Depending on the card, the next team runs their Signal card to the **Processor** team.



- To further illustrate components of a computer, you might dissect an old computer as a class to show the inside components, ports where cables are connected, and processors.
- After students have responded and/or submitted their responses, you may choose to open the questions up for a class discussion. Then proceed with the Wireless Connections portion of The Digital Dive.

Wireless Connections

- Ask students to read this topic's introduction individually or as a class. Ask them to consider what hardware in The Digital Dive activity might use wireless connections and discuss the possibilities as a class.
- Complete The Digital Dive again, but include the wireless connection Action cards this time. The card shows five, vertical radio signal bars. To "pair" wireless components, teams should pass the Connection card along with their Action and Signal cards. If the Connection card is not passed between these teams, then the Keyboard or Mouse is not considered "paired" and the sequence must restart. Make sure students pass the wireless Connection card.
- Do one of the same sequences from the first time your students did The Digital Dive. Then proceed with new tasks for the teams to perform that include the wireless Connection card.
- You may also choose to repeat the activity using a Connection card between the Processor team and the Speakers team. This would be necessary if the user had Bluetooth speakers connected to the computer.

Compare the Hardware

To assist students with transferring what they've learned about computer hardware to the micro:bit microcontroller's hardware, have them analyze the micro:bit, front and back sides, in pairs.

Note: There is more than one version of the micro:bit. You may have the micro:bit that is shown in Figure 1, where the compass and accelerometer sensors are separate components. Or you may have the version shown in Figure 2, where the compass and accelerometer sensors are combined as a motion sensor that can perform both functions. The user interaction with both versions is not affected by this difference.

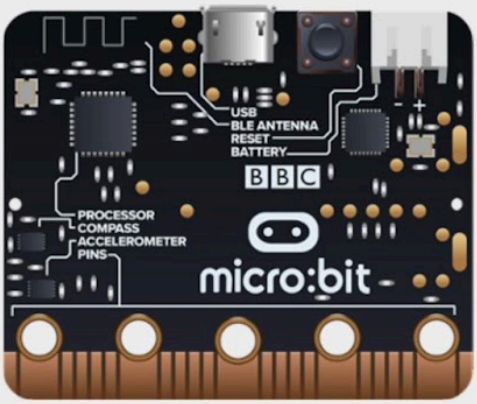


Figure 1. micro:bit Back with Compass and Accelerometer Separate

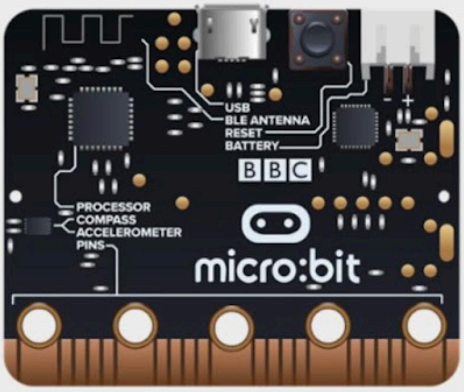


Figure 2. micro:bit Back with Compass and Accelerometer Combined

Reflect on Learning

If you're using a KWL chart for this activity, you may use the first conclusion question to complete the chart. Have students complete their individual or class chart with their new computer science definition in the Learned column. Ask students to share their new definitions and how it compared to their initial definition from the first question of this APB.

TEACHER EDITION

PLTW GATEWAY

Activity 1.2

What to Do

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.1C Describe how models are used to make predictions and learn about a phenomenon, situation, or design.

LO 2.2A Analyze and create algorithms.

LO 3.1A Collaborate effectively on a diverse team.

LO 3.2A Communicate effectively for specific purposes and settings.

LO 3.2B Document a process appropriately.

KNOWLEDGE AND SKILLS

KS

KS 1.1C1 Identify various models that may be used, which include but are not limited to physical models (prototypes), mathematical models, simulations, schematics, code, or 2D and 3D representations.

KS 2.2A1 Create an algorithm or sequence of steps to accomplish a task.

KS 2.2A2 Identify different algorithms that can be used to solve the same problem.

KS 2.2A3 Analyze, break down, and explain the logic of an algorithm.

KS 3.1A1 Describe how diverse perspectives in collaboration typically produce the best results in a process.

KS 3.1A3 Recognize individual strengths when defining roles and responsibilities.

KS 3.2A1 Distinguish technical communication artifacts that capture a process.

KS 3.2A2 Communicate to meet the needs of the audience and be appropriate to the situation.

KS 3.2B1 Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations.

ESSENTIAL QUESTIONS

EQ

OA 1 How do you express yourself and your creativity through computer science?

EQ 1.1 How can algorithmic thinking skills be used across multiple disciplines?

EQ 1.2 How can computer programs solve problems?

RESOURCES

Flowchart Infographic

Activity 1.2 Answer Key

Preparation

- Print the **Flowchart Infographic** for each student.
- Prepare a drawing for the mirror drawing activity, or use the one included in the teacher resources. You will have to describe the drawing to your students. Be prepared to show an enlarged version of the drawing for students to see after they complete that portion of the activity.

Activity Time: 3 Days

Possible Student Misconceptions

- Computer scientists start a program by sitting at a computer and writing code.
- Preplanning using algorithms and flowcharts is a waste of time.
- Algorithms are all linear, meaning they only follow one set sequential path.
- There is only one algorithm to solve a problem.
- Computers can think or perform actions independently.
- Coding is best done as an individual.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Execute a Task

- Students will begin the activity thinking about what computer scientists have to think about before creating code. This will help them activate prior knowledge.
- Allow students time to reflect on their thoughts about devices that require human input to work by responding to prompts a, b, and c in the activity. If you plan to implement a flipped class, students could complete their response to these prompts outside of class.

Discussions

1. After students have completed their response to the prompts, invite them to share their thoughts as you lead the class in a short discussion.
   - Encourage students to share their thoughts without correcting possible misconceptions they have about programming devices.
   - You may choose to, however, ask them additional probing questions to get them thinking deeper and expand on their current understanding—for example, what type of devices should or should not be programmed and why?
   - Students will explore how to organize these thoughts throughout the lesson and later reflect on what they've learned compared to their initial thoughts.
2. Facilitate a class discussion about problem-solving and tools that may be used to solve problems—that is, flowcharts, graphic organizers, concept maps, diagrams, and so on. As students discuss, ask them, "How did that tool help you with your problem?"
3. Discuss how setting the table may be different in different homes based on the number of people in the family, what people are eating (for example, do you need a grapefruit spoon or steak knife?), cultural differences (such as chopsticks instead of silverware), and so on. Ask students, "How would the algorithm look different in your home?" Ask for a volunteer to share.

PLTW Gateway Notebook - Reflect on Learning

Have students answer the reflection questions. If short on time, students may complete questions 1 and 2 outside of class time and then work with a partner to complete question 3 during the next class period.

Discussions: Ask students, "What would happen if a computer or robot were to follow your steps from question 3?" Since computers can't think on their own, would their steps provide enough detail for the computer to complete the task of brushing teeth?

Writing an Algorithm

When viewing the video as a whole class, you may choose to provide students with more introduction than the APB text.

- Ask students what they think algorithmic thinking might involve. Write student responses on the whiteboard. Having their predictions written down provides an opportunity for students to reflect on what they learned from the video, during the class discussion after watching the video.
- Students may do this portion of the APB outside of class, if your school schedule allows it and you choose to assign it.

Show the Algorithmic Thinking video. Pause the video as necessary for students to take notes. At a minimum, students should write the definitions of algorithmic thinking and algorithm.

Discuss the video as a class

- Ask students to explain what algorithmic thinking means. Begin asking students clarifying questions about the importance of planning and algorithmic thinking to start rethinking possible misconceptions identified from their responses to the first questions in the APB.
  For example, stress the importance of planning before "diving in" to coding or prototyping the solution. Share the Benjamin Franklin quote, "If you fail to plan, you are planning to fail!"
- Discuss where algorithms are used and the advantages of using them.
- Discuss what students plan: for example, what they are going to wear this weekend, what they are going to do at their birthday parties, where they are going and what they are doing on vacation, and so on? So how is a school assignment any different? Shouldn't we plan how we are going to solve this problem?
- Reference the final unit problem options that were introduced in Activity 1.1 The Brain.

PLTW Gateway Notebook - Reflect on Learning

- www.draw.io is an online flowchart maker that your students may want to use.
- You may have students write their algorithms as homework or give time in class.
- Assign partners and have students read their partners' algorithms and think about how they could be improved. Is there enough detail in the algorithms?
- Facilitate teams working on improvements to their individual algorithms. Have a couple of students volunteer to read them aloud and have a class discussion on how they could be improved.

> **Discussions**
> Lead a discussion comparing writing algorithms to The Digital Dive game played in The Brain activity.

Flowchart

- Ask students to read the text in this topic individually or as a class.
- Ask them to consider situations where they may have used a flowchart before. Students may have used a flowchart in math class or a graphic organizer in History or English class to help graphically display information.
- Lead the students into an explanation about the process of creating algorithms with conditions.
  - Explain the role of the rhombus in the flowchart.
  - Explain how a conditional flowchart works using the flowchart for setting the table.
- Be sure students can download the Flowchart Infographic and provide directions on where to include it in their notebook to access it as a resource.

Discussions: Facilitate a discussion about the similarities and differences between flowcharts and algorithms. For example:

- A similarity could be that both provide all the steps to complete a task.
- A difference could be that a flowchart shows the steps in a pictorial representation where the shapes or symbols help to explain what is happening, whereas the algorithm is typically a list of the steps written in natural language.

Mirror Drawings

You may come up with your own idea for students to draw or use the drawing shown below.

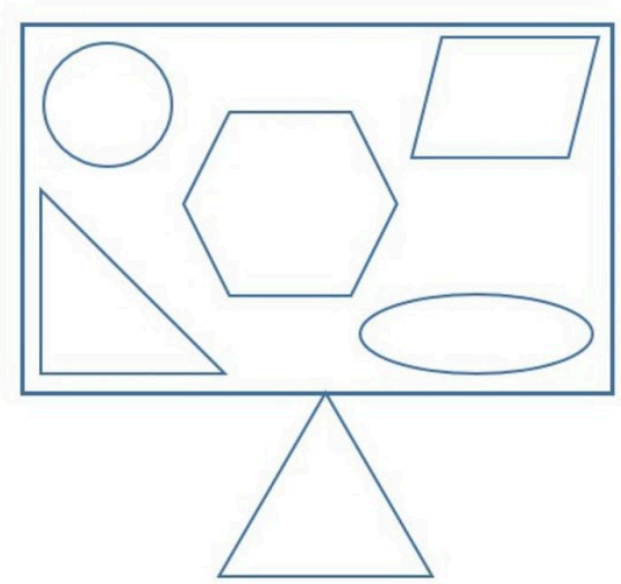


Figure 1. Mirror Drawing Example

Have students draw a picture to describe to their partners.

Discussions: Discuss their experiences as a class. With a show of hands, ask who is better at giving directions? Who is better at listening? Ask students to share why they are better at one role than the other. Discuss how they could improve at either or both roles.

TEACHER EDITION

PLTW GATEWAY

Activity 1.3

How to Do It

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.2B Apply user-centered design principles when creating a solution.
LO 2.3A Develop and apply abstractions.
LO 2.4A Analyze the structure and functionality of a program.
LO 2.4B Create programs by developing and testing code in a modular, incremental approach.
LO 2.4C Adapt or improve existing code.
LO 2.5A Identify methods in which electronic devices communicate with each other.
LO 3.2A Communicate effectively for specific purposes and settings.
LO 3.3A Demonstrate the ability to manage multiple resources throughout a project.

KNOWLEDGE AND SKILLS

KS

KS 1.2B2 Describe the importance of involving prospective users early and often during the design process.

KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.

KS 2.3A1 Explain the purpose of an abstraction.

KS 2.4A1 Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures.

KS 2.4A2 Describe the appropriate code blocks and conventions used in the programming language.

KS 2.4A3 Identify possible events that can occur during runtime and select the appropriate event handler blocks to respond to these events.

KS 2.4B2 Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable).

KS 2.4B3 Test code frequently as it is being developed.

KS 2.4C2 Improve readability of code by creating or improving documentation, using descriptive variable and procedures names, and using comments.

KS 2.4D1 Describe the components of a mobile app.

KS 2.4D2 Navigate and use an app development software to design, develop, and test a mobile app.

KS 2.4D3 Design an app's user interface using appropriate tools.

KS 2.4D4 Select and use appropriate features to meet the desired functionality of an app, such as user interface components, layout arrangements, media objects, drawing and animation, sensors, location awareness, accessing phone features, and data storage.

KS 2.4D5 Deploy an app using the emulator or a mobile device.

KS 2.5A1 Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth.

KS 2.5A2 Use a systematic process to identify the source of a problem within individual and connected devices.

KS 2.6A2 Give examples of technologies that assist or extend human capabilities.

KS 3.2A2 Use accurate and appropriate terminology.

KS 3.3A2 Manage digital files appropriately.

KS 3.5A2 Explore a variety of careers related to engineering, biomedical sciences, or computer science.

ESSENTIAL QUESTIONS

EQ

OA 1 How do you express yourself and your creativity through computer science?

EQ 1.1 How can algorithmic thinking skills be used across multiple disciplines?

EQ 1.2 How can computer programs solve problems?

RESOURCES

- 1.3 Grids Formative Assessment
- 1.3 Grids Formative Assessment Answer Key
- Activity 1.3 Answer Key

Preparation

- Ensure students can download and access the activity's .hex files. You may choose to download them and place them in an accessible location for students, such as a shared network drive or student flash drives.
- If you have multiple classes or individuals using the same computers, previous student work will be visible in the programming environment. To limit this, consider instructing either the outgoing or incoming student to go projects and select new project.

Activity Time: 3 Days

Possible Student Misconceptions

- Algorithms are all linear, meaning they only follow one set, sequential path.
- Computer scientists start a program by sitting at a computer and writing code.
- Preplanning using algorithms and flowcharts is a waste of time.
- When creating a physical computing system, the code and system will always function properly the first time tested.
- Computers read and execute code within programs only in a linear manner, from top to bottom.
- Errors in computer programs are the fault of the computer and not the human who programmed it.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Micro:bit Scavenger Hunt

- Guide students to answer the scavenger hunt questions in complete sentences by including the question in their answer.
- Consider whether you want students to take some class time, or maybe time out of class, to work on several of the tutorials from the Home page.
- Guide students in the use of the programming environment to create simple programs. See what projects may interest your students. Consider assigning some of these with a flipped classroom approach or as an out-of-class activity. Later in the unit, one of these projects may be an excellent extension for a student or group of students who are very advanced.
- As students explore the Home page and the Hardware page, consider this as an orientation to the content that is there, not a time to learn the details of everything. Make sure your students see the big picture of what they can find here WHEN they need it.
- For example, on the Hardware page, students can become more aware of features on the micro:bit, such as the compass or accelerometer. This page can also be a help in troubleshooting when they first connect their micro:bit and USB cable, and more.

Flowcharting the "Blink"

- Students should have a copy of the Flowchart Infographic from Activity 1.2 to refer to.
- While students begin their brainstorming, organizing of ideas, and then flowcharting, circulate around the room, listening, observing, and questioning their work.
- Listen to students' discussions. Consider posing the following questions as you move from group to group.
  - How is one step and shape different from another and why?
  - Why did you choose that particular shape for that step?
- If time allows, consider having students share their flowchart with other students and see whether they can understand or even suggest corrections or improvements to each other's flowcharts.

Programming the "Blink"

- If students have not spent any time on their own working on the projects from the Make area, they may benefit from watching you demonstrate how to grab and drag a block or two within the Microsoft MakeCode programming environment.
- Have students work on translating their own flowchart into code on their own computer.
- If students ask where a specific command is, consider asking them what it does and which drawer they might find it in.
- Explain to students how to submit their program to you, possibly on a flash drive or to a shared network location. You may need to review proper naming conventions for saving files with your students (i.e. A1_3_Blink_LastNameLastName).

> **Note**: For additional assessment, you may choose to have students add their documentation throughout the activity in their PTLW Gateway Notebooks or on an exit ticket to submit as they leave class.

Connections and Board Confirmation

- Make sure that each pair of students has a micro:bit and USB cable. All students will use their own computer, so make sure all are on.
- Assign roles to your pairs of students; one will be “Innovator” and one will be “Maker”.
- While Maker completes the cable connections, Innovator can access the file in the teacher-directed location and copy it onto their computer. When Maker has completed all cable connections, Innovator can assist the Maker in locating the file and copying it to their computer.
- After the Innovator completes the download and the connection test, they should disconnect the cabling from the computer and micro:bit and pass both to the Maker. The Innovators coach the Makers in cable connections, and then the Makers run the Maker test file.

Note: If students are using Chromebooks, you will need to configure Chrome to ask where to save the file before students can send their .hex files to their micro:bits. You only need to do this once.

1. Open the *Settings* for Chrome.
2. Click *Advanced* at the bottom of the page.
3. Find the *Downloads* settings.
4. Enable the setting *Ask* where to save each file before downloading.

LED Grid Practice Exercises

- Students may need to be reminded of the grid location coordinates and how they differ from math class.
- The first exercise is a drag-and-drop practice to sharpen their LED location skills.
- When they have correctly completed the drag-and-drop exercise, they will then use the Microsoft MakeCode programming environment (step 4) to create a shape (the same LED coordinates they used in the exercise) using the plot block from the LED drawer.

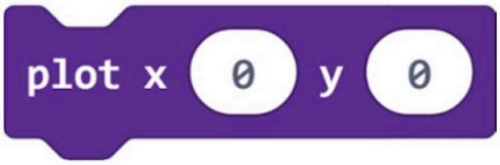


Figure 1. plot Block

- They could use the on start block, forever block, or even a when button pressed block to begin their code. A later discussion on which they chose and what computing power each of those choices requires may lead them to creating more elegant code, which is described later in the unit.

TEACHER EDITION

PLTW GATEWAY

Activity 1.4

Crush the Bug

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.2B Apply user-centered design principles when creating a solution.

LO 1.3A Apply computational thinking to solve problems.

LO 2.4B Create programs by writing and testing code in a modular, incremental approach.

LO 2.4C Adapt or improve existing code.

KNOWLEDGE AND SKILLS

KS

KS 1.2B1 Investigate the types of interactions between users and a proposed solution.

KS 1.3A2 Choose appropriate computational practices when solving a problem.

KS 1.3A3 Decompose a problem into smaller parts.

KS 2.2A4 Locate and debug errors within an algorithm.

KS 2.4B4 Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line by line, generating output, or commenting out code.

KS 2.4C1 Find existing code relevant to a problem and extend or apply it to a new purpose.

KS 2.4C2 Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments.

ESSENTIAL QUESTION

EQ

EQ 1.1 How can algorithmic thinking skills be used across multiple disciplines?

EQ 1.2 How can computer programs solve problems?

RESOURCES

Activity 1.4 Answer Key

Preparation

- Ensure students can download and access the activity's .hex files. You may choose to download them and place them in an accessible location for students, such as a shared network drive or student flash drives.
- Create a code tracing chart to use with the whole class on an interactive white board, chart paper, or a dry erase board.

Activity Time: 2 Days

Possible Student Misconceptions

- Algorithms are only linear, meaning they only include one set, sequential path.
- When creating a physical computing system, it will always function properly the first time tested.
- Errors in computer programs are the fault of the computer and not the human who programmed it.
- You cannot learn from failure; persistence is a waste of time.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

A Famous Bug

- Instruct students to read the short story about Rear Admiral Grace Hopper. You may have students read this section independently, in pairs, or as a class. If your schedule allows it, you may assign this reading to students before you begin the APB, as an out-of-class or flipped-class activity.

- Students may be interested in learning more about Grace Hopper and her achievements in computer science. You may have students informally research her life and share their findings with the class.
- Some students may be interested in her development of an operational compiler and how compilers work. This could be an informal research topic for them to explore and share with the class.

Finding a Bug

- In this section, students are presented with a bugged program and the program's intended outcomes. Instruct students to read the section independently or as a group. Independently, students use the Microsoft MakeCode programming environment to create the program shown in step 1. After testing the bugged program, either in the micro:bit emulator or on a connected micro:bit, have students reflect on their findings by answering the two questions in the APB.

> **Scaffolding Note**: Students may not yet fully understand the code blocks used in the programs they analyze in this activity. However, reading and analyzing unfamiliar code is excellent practice for students to learn new ways of programming and improving their coding techniques. Encourage students to read and analyze the code in the buggy programs throughout this activity. Remind them to use what they've learned so far about algorithmic thinking to understand each program.

Code Tracing Charts

Use the Code Tracing Charts video to review how to complete this with your students.



Code Tracing Charts video

- As a class, discuss the bug found in the program. Use a blank code tracing chart to go through the bugged program presented in this section of the APB as a class, to isolate or find the bug. A completed chart is provided below as an example.
- Ensure students add descriptive titles to the top of their code tracing charts as instructed in the APB. (Example: Repeating Smiley Face Code) Also encourage students to add any program or hardware notes and ideas beside their code tracing charts.

Code	Outcome	Correct?
on button A pressed	The LED lights up when button A is pressed.	YES
while true - do pause (ms) 500 show LEDs (center) pause (ms) 500	The center LED lights up and stays on and doesn't blink.	NO
clear screen	The LED grid is never cleared.	NO

Figure 1. Code Tracing Chart Example

- Remind students throughout the unit that code tracing charts are only part of their programming notes. They will need to make comments in their programs, add hardware notes to their Gateway Notebook, and describe desired outcomes of their programs. To facilitate student documentation of their debugging process, you may choose to modify the code tracing chart or instruct students to include additional information about their programs. For example, a fourth column might be added to their charts that describes the desired outcome of a specific set of code.
- The Code column should show the logic of the program and not each individual code block. For example, the While block encompasses three commands: two Pause commands and one Show LEDs. These commands show what is to be looped, so it is reasonable to lump this code as one part to analyze. While debugging, students should view the program as a logical sequence of procedures, rather than individual blocks of code or commands.
- After the class has completed the chart, ask the students to make predictions about what may be causing the LED to not blink. Are all of the commands necessary to make the LED blink forever included in the While code block? One command isn't being executed at all. Why is that? Ask them to consider the bug and change their program to crush the bug and fix the code.
- Observe students as they make corrections to the program. Your observations are considered a formative assessment and can inform your facilitation decisions. As you observe students, take note of how students are trying to fix the problem. You may need to ask additional questions such as "What are some ways you can fix this bug?" and "How can you get that Clear Screen code block to start working?"
- Encourage them to test their revised programs and make any additional corrections. Ask students to share their solution with you or the whole class.
- For additional assessment, you may choose to have students write descriptions of how they were able to debug the program in their notebooks or on an exit ticket to submit as they leave class.

Debug Your Hardware, Too

Discussion: Discuss with students the importance of checking their hardware alongside their programs.

Checking all of the electrical components and connections in their computing system is essential. You can relate this to The Digital Dive in Activity 1.1. If wired connections are poor or wireless-connected devices aren't paired, the data can't travel, and individual components can't talk with each other. You can compare this to the human body's transfer of signals through the nervous system. Communication between your brain and parts of your body is necessary for the body to respond and function properly.

A Helpful Comment

- Ask students how comments in the buggy LED program might have made it easier for them to isolate the anomaly or bug. How could a comment on the *while* code block have helped identify chunks of code when code tracing?
- Students should recognize that comments are a way for them to note specific chunks of code that do a specific task or procedure. They can also use comments to remind them of variables and operations when they create more complex code later in the unit. Instruct students to practice adding comments to their corrected LED program that would make it easier to identify outcomes in the code.

Debug Steps

Note: Students should consider these debugging steps as a suggested checklist while they debug their own programs and systems. Because this unit incorporates hardware designs, students will need to check their hardware as part of the debugging steps. This may not be necessary in other courses when only debugging software development. These steps are not a formal process, but simply a suggested procedure for identifying and resolving bugs.

Buggy Code

- Before you do this activity, ensure that the five buggy program files listed below are stored in a location students can access.
  - A1_4_termites.hex
  - A1_4_fire_ants.hex
  - A1_4_yellow_jackets.hex
  - A1_4_dragonflies.hex
  - A1_4_moths.hex
- Assign each student to one of the five teams listed in the APB. Try to divide students evenly among each team, but it's acceptable to have fewer students in one or two teams. Each student should complete a code tracing chart and test the program.
- For excelling students, three additional buggy program options are provided. After they have completed their assigned buggy program listed above, you may choose to assign students who are ready for a challenge one of the optional buggy programs.
  - A1_1_hornets_option.hex
  - A1_1_mantes_option.hex
  - A1_1_wasps_option.hex
- Encourage each team to communicate among themselves and share ideas about the bug and how to solve it. While students are debugging their team's program, observe their progress and facilitate as needed. You may ask them prompting questions such as:
  - What does the program do?
    Hint: Ask students whether they thoroughly read the first comment in their assigned program. This tells them what the program should do.
  - What outcomes have you tested?
  - Have you identified any outcomes that did not perform as expected?
  - Did you read all of the comments to better understand the program's purpose and logical sequence? Did you find any problems or inconsistencies?
  - Have you isolated the anomaly? How do you plan to fix or debug it?

Discussion: Some students may want to dive into the code and begin rearranging code blocks without completing a code trace. Discuss with students why it's important to code trace first.

If a computer scientist just starts rearranging code and the changes don't work, it's much more difficult to go back and see why the change didn't work and what was changed from the original buggy code. When programs get longer and more complex, it's important for computer scientists and programmers to document, comment, and take good notes to help make debugging quicker and more effective.

Import Your Buggy Program

Instruct the teams to share their debugging work and resolution. To help teams describe their work, you may choose to ask them some of the prompting questions listed above.

TEACHER EDITION

PLTW GATEWAY

Project 1.5

The Blinking Message

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.1B Demonstrate creativity and courage to take risks in proposing designs.

LO 1.2B Apply user-centered design principles when creating a solution.

LO 2.6A Analyze the implications of computing in society.

KNOWLEDGE AND SKILLS

KS

KS 1.1B1 Describe the importance of creativity and risk taking in engineering, scientific, or computational processes.

KS 1.2B1 Investigate the types of interactions between users and a proposed solution.

KS 2.6A1 Identify ways that computing has changed or is changing society.

KS 2.6A2 Give examples of technologies that assist or extend human capabilities.

ESSENTIAL QUESTIONS

EQ

OA 1 How do you express yourself and your creativity through computer science?

EQ 1.1 How can algorithmic thinking skills be used across multiple disciplines?

EQ 1.2 How can computer programs solve problems?

RESOURCES

1.5 The Blinking Message Rubric

Project 1.5 Answer Key

Preparation

Prepare micro:bit microcontrollers and USB cables for every two students. Students will program their projects and test them using the emulator before they download their .hex files to the micro:bit. Unless a microcontroller is available for each student, each micro:bit microcontroller will be shared between two students.

Project Time: 4 Days

Possible Student Misconceptions

- When creating a physical computing system, it will always function properly the first time tested.
- The program should know when to clear the LED grid before displaying another image. Clearing the screen doesn't need to be programmed.
- Errors in computer programs are the fault of the computer and not the human who programmed it.
- You cannot learn from failure; persistence is a waste of time.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Informative LEDs

- Instruct students to read about LED signs. You may have students read this section independently, in pairs, or as a class. If your schedule allows it, you may assign this reading to students as an out of class or flipped class activity before you begin the APB.
- Ask students to think about LED signs around their community. You may choose to have students respond to the following questions in pairs, as a Think-Pair-Share

exercise, or as a whole class. These are intended to activate students' prior knowledge about communicative uses for LEDs and animation in general.

- What are examples of stationary signs, scrolling signs, or animated signs in your community?
- What are the benefits of using an LED sign rather than a printed sign or a sign lit up with traditional light bulbs?
- Describe how an animator can make an image move on paper. Think about what that might look like within a computer program.

Emojis

- Instruct students to read about emojis independently or in groups.

> **Discussion**: As a class, discuss how emojis are used. Why is it easier to use an emoji to express a thought or emotion than words? Students might say an emoji is one character or small image rather than several characters in text. It's like the old saying, "A picture is worth a thousand words." Linguistic studies with emojis are currently being conducted to determine their value in communication.

The Blinking Message Project

> **Discussion**: You may choose to have a class discussion about ways people use technology to communicate and how this impacts our society. Discuss how technology improvements over time have sped up the transmission of messages and communication overall. To facilitate this discussion, you might show a progression from sending letters and telegraph messages to sending e-mails and text messages.
>
> If time allows, your students may be interested in researching technological communication achievements such as the invention of the telegraph, Morse code, teletypewriters (TTY), and text messages.

Students will share their planning assets, programs (.hex files), and their finished projects on the micro:bit. Sharing with others is a critical component of physical computing projects. Students' projects and work can be shared with peers, with you during teacher observations, or with members outside of the class, such as administrators. Consider your available time when choosing a method for student project sharing.

Blinking Message Requirements

Review The Blinking Message project requirements with students. Refer to the list of criteria and the rubric for the project. Students should use these as a guide and checklist for completing the project.

What to Do

- Facilitate as students work independently to plan their messages. This is the phase introduced in the What to Do activity where students plan the action their physical computing project should execute and flowchart the logic of their program.
- As you observe your students' planning, ask them prompts, such as:
  - How do you plan to make this image move to create an animation in your message?
  - It seems you want to create a pretty complex graphic. How might you be able to simplify it for a small 5 x 5 LED grid that you have available on the micro:bit? Have you planned out the LED grid for your graphic?
  - Will those letters be scrolling or flash on the grid?
  - How might you make that message clearer or easier to read with timed pauses?
- Your teacher observations during this stage of the project are a great way to assess your students' progress as they plan their blinking messages. If you notice students struggling, it may be beneficial to ask their classmates to provide ideas or share how they successfully planned their message with the class.
- Students can create additional LED planning grids as needed.

How to Do It

- Continue facilitating as students create their Blinking Message program in the Microsoft MakeCode programming environment. Instead of showing students where to find code blocks in the drawers, help students think about what type of code blocks are needed.
- For example, if students want to use a block that lights up an LED or set of LEDs, ask them where they could find code blocks for images. Students may respond with either the Basic or Images (under Advanced) drawers in the programming environment. Facilitate student learning by helping them think about where LED-related code blocks are found in the programming environment.
- Discourage students from using the *show icon* code block in the Basic drawer. This might be a good block for them to use to get ideas for images though. Students should be encouraged to use the LED planning grid to be creative and design their own, original images and emojis.

Debug Your Program

- Facilitate students as they debug their programs. Ask guiding questions that prompt students to think about their programs.
  - Did you follow the debug steps?
  - Did you find any hardware connection issues?
  - What outcomes are not working as expected?
  - When you code traced your program, what results did you find?
  - How might you break (or add) code blocks to test smaller (or larger) portions of code?
  - What could you change or alter in the code where you identified a bug that might fix the problem?
- When using one micro:bit per two students, allow each pair of students to share a micro:bit during the testing and debugging phases of their project. The Microsoft MakeCode emulator can also assist students with testing and debugging.
- Review the debug steps with students, as necessary, while they debug their programs.

Spread the Word

- Instruct students to share their projects with others (peers, teachers, administrators) depending on your implementation of the curriculum and given possible class time constraints.
- Students should download a final version of their program as a .hex file. Instruct students to submit their .hex files to you using a flash, network, or shared drive. Students will also need to save their project's .hex file for later use in the Get Connected activity.
- If enough micro:bits are available, allow students to carry their programmed micro:bits to show others. To run their programs when the micro:bit is disconnected from a computer, students need to attach the 3V battery pack to the microcontroller. This can be an opportunity for you to discuss the two power sources that the micro:bit uses: a computer using a USB connection or 3V battery power.
- To conclude the project, you may choose to have students reflect on the final questions in the APB individually or as a class discussion. The questions are for student reflection and consideration.

Challenge

- Students who have completed The Blinking Message and submitted their .hex files may be assigned to work on this additional challenge. This challenge option should be assigned to students who have demonstrated mastery, throughout the project, of skills learned in Lesson 1.
- If students need additional assistance on the challenge option after watching the video and reading the directions, you may direct them to the Help menu in the Microsoft MakeCode programming environment. The Hardware Help page provides additional information about using the compass and accelerometer sensors.
- Student responses to the challenge questions can help you further understand your students' prior knowledge of inputs and sensors, as you prepare for Lesson 2.

LESSON 1 INTERIM ASSESSMENT

The Computer Science for Innovators and Makers Lesson 1 Interim Assessment is located in the Unit Level Resources, Assessment Page. The assessment is a starting point for you to use with your students. Feel free to edit this assessment to best meet your needs and the needs of your students. Use the Lesson 1 Interim Assessment Answer Key, to evaluate student work.

TEACHER EDITION

Lesson 2

The Ins and Outs – Quick Guide

Preface

In this lesson, students explore a variety of sensors and actuators to use as inputs and outputs in physical computing projects. Using different materials to transfer electrical signals, such as conductive thread, alligator clips, and copper tape, students create their own input device—a sensor or switch—to interact with a program they develop on the microcontroller. They use these skills in the lesson's project to design, develop, and program a system to protect safes and secrets.

Essential Questions

1. How do you express yourself and your creativity through computer science?
2. What do programming best practices look like?
3. Why are interactions between machines, humans, and the environment beneficial, problematic or both?

Activity, Project, or Problem	Days
Activity 2.1 Need Input	3
Activity 2.2 Responding Output	3–4
Activity 2.3 Get Connected	3
Project 2.4 Secrets and Safes	6
Lesson 2 Assessment	1
Total	**16–17 Days**

TEACHER EDITION

PLTW GATEWAY

Activity 2.1

Need Input

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.2B Apply user-centered design principles when creating a solution.

LO 2.1A Collect, process, and analyze real or simulated data.

LO 2.2A Analyze and create algorithms.

LO 2.5A Describe the hardware components of an electronic device and how they interact with software and the environment.

KNOWLEDGE AND SKILLS

KS

KS 1.2B3 Recognize that understanding human-computer interaction can improve the design of devices and extend the abilities of humans.

KS 2.1A2 Understand that computers enable rapid processing of information.

KS 2.1A3 Collect, process, and interpret data to gain insight on a problem and draw conclusions.

KS 2.2A5 Create simple algorithms that involve variables, conditionals, operators, or logic.

KS 2.2A6 Create algorithms to perform repetitive or reusable tasks.

KS 2.5A3 Select and justify the hardware chosen to accomplish a task.

KS 2.5A4 Describe advantages and disadvantages of using different power sources for the mobility of a system.

ESSENTIAL QUESTIONS

EQ

OA 1 How do you express yourself and your creativity through computer science?

EQ 2.1 What do programming best practices look like?

EQ 2.2 Why are interactions between machines, humans, and the environment beneficial, problematic, or both?

RESOURCES

2.1 Input Output Device Images Sheet

2.1 AK Input Device Images Sheet

Activity 2.1 Answer Key

Preparation

- Determine teams of two students each for pair programming.
- If you have multiple classes or individuals using the same computers, previous student work will be visible in the programming environment. To limit this, consider instructing either the outgoing or incoming student to go to Home and select new project.
- Please note that banana plug connection wires are no longer included in the CSIM kit. Therefore specific information for using them has been deleted from the student course. However, for those teachers who do have the banana plugs we have left the information for proper use in the teacher resources.
- Print the I/O Device Images sheet in color on paper or mailing labels (1" x 2.63") as stickers for students to attach to their PLTW Gateway Notebooks in the Input Devices Chart.
- Create a micro:bit kit with the Input Devices:
  - Microcontroller (micro:bit) and USB cable
  - Flex sensor
  - Pressure sensor
  - Photocell resistor
  - Battery holder with 2 AAA batteries
  - Connection wires (pin to alligator clip)
  - Alligator clips
  - Copper tape - 2"
  - Aluminum foil - 2" x 1"

Safety Considerations

The pins labeled “3V” and “GND” relate to the power supply of the board, and they should never be connected directly together, as this could cause a short circuit and fry the processor or components on or attached to the micro:bit. Remind students often about this.

Activity Time: 3 Days

Possible Student Misconceptions

- Computers read and execute code within programs only in a linear manner, from top to bottom.
- Equations are used in computer programming the same way they are used in mathematics.
- All input devices have an on and off switch.
- Digital devices do not interact with their environment.
- Computer science equates to computer literacy—the understanding of using basic computer functions.
- Errors in computer programs are the fault of the computer and not the human who programmed it.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

What's an Input?

- Begin a conversation with your students helping them to see inputs in their environment. Examples they may come up with include a keyboard, cell phone, switch to turn on lights, key fob to unlock a car door, remote car starter, and garage door opener.
- Consider listing the examples on the board or typing to show them on a screen as students brainstorm and share. Call out the difference between inputs and outputs, but tell students that outputs will be discussed in the next activity.
- Students will work in pairs for the I/O Devices activity. They may work in the same teams as the previous activity or you can assign new partners.

- Students need their equipment for this activity. Either distribute the box of equipment for each pair of students, or if you are using a general storage area for each part, have students get the required input devices.
- Distribute the I/O Device Images sheet.
- Have students complete the drag-and-drop activity.
- Once correct, students will copy the chart into their PLTW Gateway Notebooks and include the images.
- Students should match the items from their kit to the images and then write the name and description of what they think it detects next to the image.

> **Reminder**: Remind students to keep the rest of the sheet with the output device images to use in Activity 2.2 Responding Output.

Pair Programming

Using pair programming collaboration here introduces the students to the approach and gets students acclimated with how it works. It will be applied again in future APBs.

Before you begin this activity with students, take the following steps to get ready:

1. Prepare the list of partners to pair up for pair programming.
2. Watch this video about pair programming facilitation.
3. Note: students will be watching a different video on Pair Programming Student's Perspective.



Pair Programming: Teacher's Perspective video

Programming Inputs

- Direct students to look at the front side of their micro:bit and find the A and B buttons.
- Guide students as they import their blinking message program from The Blinking Message Activity in Lesson 1. Students will modify their Blinking Message code to use the buttons as inputs to scroll the message and play the animation.
- If students have new partners for this activity, they should use the Driver's Blinking Message for the *buttonTest* program. Switch Drivers for the *lightTest*, and use the new Driver's Blinking Message for that program.

> **Reminder**: Remind students of the location and the naming convention you have established in your class. For example, are students saving to a USB drive, shared team folder, or computer desktop?

Light Sensor

- Circulate and assist students with writing code to use the light sensor embedded in the micro:bit. Guide groups with leading questions and suggestions for debugging and pair-programming strategies.
- Remind students that the light sensor is built into the micro:bit, and it is an input device. It is different from the LEDs, which provide output.
- Equations are used in computer programming the same way they are used in mathematics. Review math symbols and how to write equations using the greater than and less than symbols.
- Prompt students to reflect on their flowcharting experience from Activity 1.2 What to Do, when students were first introduced to conditional statements. This will help to scaffold their learning on this concept and to draw meaningful connections across lessons.
- To get the light level to change, students may need to move close to a window or outside. Alternatively, they may want to change the light level value in the conditional statement.

Compass and Accelerometer

- If necessary, explain the compass and accelerometer in more detail. Give examples of instances where students may have used a compass (navigation system in car, giving directions) or an accelerometer (fitness tracker, gaming device, or pedometer).
- For more information about the micro:bit onboard sensors, access the Hardware topic from the **Microsoft MakeCode Help menu** .
- Demonstrate how to locate the Rock, Paper, Scissors and Compass projects.
- Demonstrate how the gear button in the Microsoft MakeCode programming environment will show additional code options. For example, the *else if* command is located in the *if then* block.
- If they need a refresher on creating comments, direct students to A Helpful Comment section in Activity 1.4 Crush the Bug.
- Direct students to exchange roles of Navigator and Driver at least once per class period.
- Encourage each team to communicate among themselves and share ideas about bugs and how to solve them. While students use a code tracing chart to debug their team's program, observe their progress and facilitate as needed. You may ask

them prompting questions such as:

- What does the program do?
- What outcomes have you tested?
- Have you identified any outcomes that did not perform as expected?
- Did you add comments to each code block to better understand the program's purpose and logical sequence? Did you find any problems or inconsistencies?
- Have you isolated the anomaly? How do you plan to fix or debug it?

- The goal is to help students find solutions to their problems as a team and become independent learners.
- Encourage students who complete the two programs quickly to modify the code on one or both programs to improve the output. For example, create LED labels that say NE, NW, SW, and SE for the compass program, or create a pause and clear screen between each shake of the accelerometer program.

Alligators

> **Watch Out!** The pins labeled "3V" and "GND" relate to the power supply of the board, and they should NEVER be connected together, because this could cause a short circuit and fry the processor or components on or attached to the micro:bit. Remind students often about this.

- If the micro:bit is powered by USB or battery, then you can use the 3V pin as a *power output* to power external components.
  - **3V**: *3-volt power output* or *power input*.
    (1) *power output*: If the micro:bit is powered by a computer (by USB) or a battery, then you can use the 3V pin as a power output to power peripherals .
    (2) *power input*: If the micro:bit is not being powered by computer or battery, you can use the 3V pin as a power input to power the micro:bit.
  - **GND**: attach to ground to complete a circuit
- If you hold the GND pin with one hand, you can program the micro:bit to detect yourself touching the 0, 1, or 2 pins with your other hand, giving you three more buttons to experiment with (you just used your body to complete an electrical circuit).

Banana Plugs

If your students are using the banana plug wires be sure they are only pushing the plug into the snug area, not all the way through the hole, for a good connection.

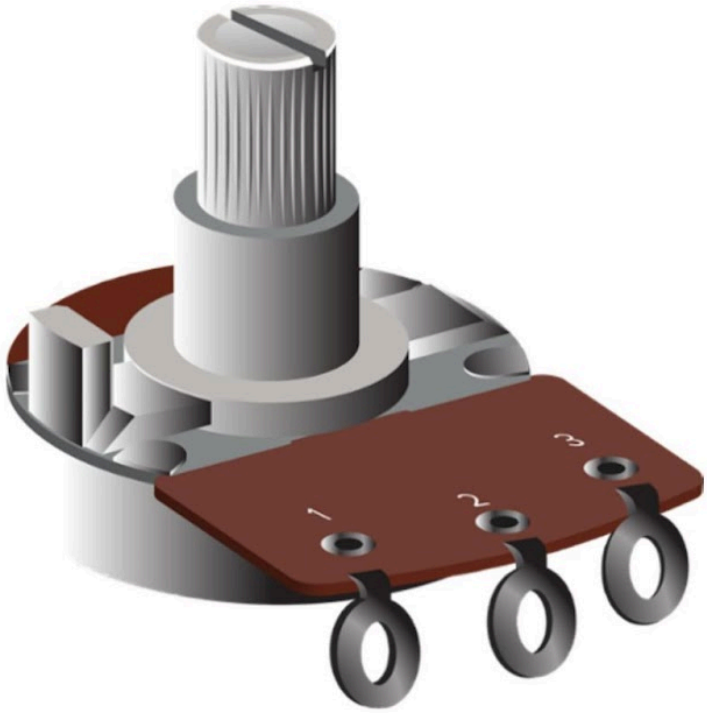


Lesson 2.1 - Figure 8

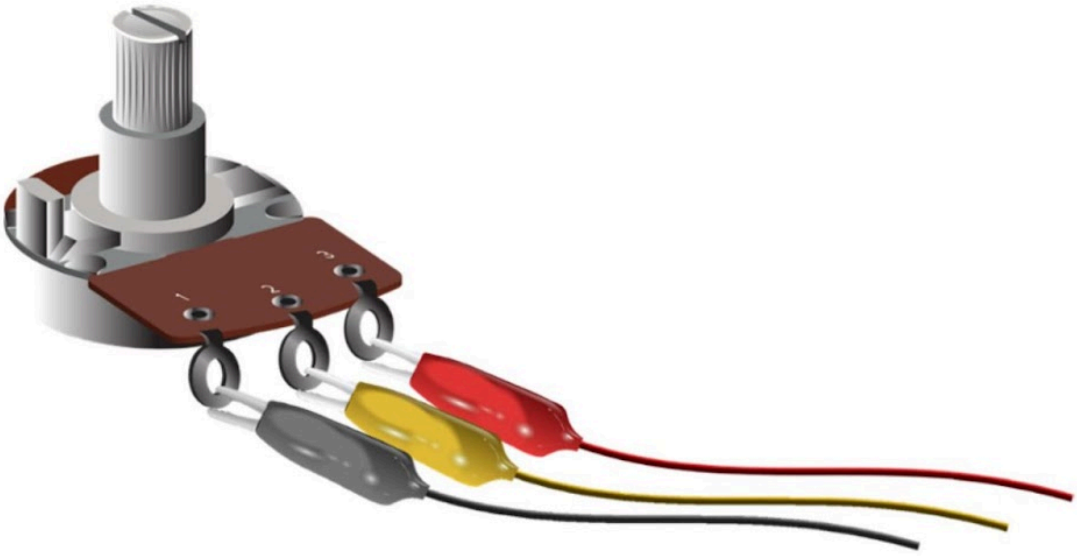
Analog and Digital Inputs

- Ensure students understand the difference between digital and analog sensors. Talk about devices they encounter in their everyday lives and which type of sensor/device they might be. For example:
  - Light switch – digital
  - Dimmer switch – analog
  - Garage door opener – digital
  - Thermostat – analog

Potentiometer (Optional)



- The potentiometer is another analog input device that can be used with the AnalogValue program.
- The potentiometer provides a variable voltage to an analog input via a twist knob. It can be used for any application where a user input across an analog range is required. For example, a knob to adjust the frequency of a radio station, a rotating knob to dim the lights in a room, or a control to adjust the speed of a motor all use a potentiometer.



- The black wire connects pin 1 on the potentiometer to GND on the micro:bit. The red wire connects pin 3 on the potentiometer to 3V pin on the micro:bit. The yellow wire connects pin 2 on the potentiometer to the corresponding GPIO on the micro:bit that's used in the program.
- The potentiometer is polarity sensitive. This means the red and black wires need to be connected correctly for a clockwise rotation of the potentiometer to provide an increasing value.
- The micro:bit analog input reads from 0 to 1023. Turn the potentiometer counter-clockwise all the way. The potentiometer value will read close to 0. Turn the potentiometer clockwise all the way, and the output value will increase as it turns up to a maximum value of 1023.

Note: To demonstrate analog values read by sensors, you may choose to have students connect an optional potentiometer to the micro:bit prior to using the pressure sensor.

1. Instruct students to create the *analogValue* program in the Pressure Sensor section.
2. Students should connect the potentiometer. Potentiometer pin 1 connects to micro:bit 3V pin. Potentiometer pin 2 connects to micro:bit pin 0. Potentiometer pin 3 connects to micro:bit GND pin.
3. After downloading the program to their micro:bit, have students test the potentiometer by slowly turning the knob.
4. Students should observe the values displayed on the LED grid as they turn the potentiometer knob to various positions.

Pressure Sensor

- You should circulate and assist students with writing code to use the external sensors attached to the micro:bit. Guide groups with leading questions (for example, Why do you think the sensor needs to be attached to the 3V pin?) and suggestions for debugging (for example, Does the pin number in your code match where your alligator clip is attached on the micro:bit?). Encourage pair-programming strategies.
- Explain to students that they will use two programs repeatedly during this section: the *inputTest* program they saved in Analog and Digital Inputs and the *analogValue* program they create in this Pressure Sensor section.
- After students open the *inputTest* program to modify it for use with each of the analog sensors, they need to save it with a new name as directed in the activity. The *analogValue* code is necessary to read the values that the analog sensors are reporting.

> **Note**: To test their code in the emulator, students will need to use their mouse to click the appropriate pin image in the software. For an analog input device, the higher they click on the pin, the higher the analog reading will be.

Photocell Resistor

Explain how a photocell works: A photocell includes a semiconductor, which works depending on the intensity of incident light rays. When the amount of light exceeds a certain frequency, the electrons of the semiconductor are delocalized or "freed", and a photocell starts conducting electricity.

Flex Sensor

Require students to create an algorithm in their PLTW Gateway Notebooks before they code this task. At some point they should realize they will need two conditional statements to create this program. Encourage them to problem solve and figure out how they could do this.

Create Your Own Input Device

- Provide each group with a small amount (2 inches) of copper tape and a small piece of aluminum foil to create their switches.
- When the copper tape ends touch or the aluminum foil ends touch each other, they will form a circuit, just like a switch or the push button. When the ends of either the aluminum foil or the copper tape touch, the program in the micro:bit should run.
- Consider only using one roll of copper tape at a time instead of each group having their own roll. To prevent the copper tape from unrolling, use a binder clip to hold the end in place.

TEACHER EDITION

PLTW GATEWAY

Activity 2.2

Responding Output

Teacher Resources

View Student Course

KNOWLEDGE AND SKILLS

KS

KS 1.3A1 Recognize that computational thinking can be applied in multiple disciplines.

KS 2.4B2 Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable).

ESSENTIAL QUESTIONS

EQ

EQ 2.1 What do programming best practices look like?

EQ 2.2 Why are interactions between machines, humans, and the environment beneficial, problematic, or both?

RESOURCES

- 2.2 AK Output Device Images Sheet
- A2_2_MultiOutputEX.hex
- Activity 2.2 Answer Key

Preparation

- Create bags that contain five output devices: LED, LED boards, Buzzer, standard servo, and continuous servo for each student pair.
- Prepare to distribute the other half of the I/O Device Images sheet if you collected it during Activity 2.1.
- Create pair-programming pairs (Driver and Navigator).
- Place the necessary files in a location that the students can access.
- If you have multiple classes or individuals using the same computers, previous student work will be visible in the programming environment. To limit this, consider instructing either the outgoing or incoming student to go to Home and select new project.

Safety Considerations

Watch out! The pins labelled 3V and GND relate to the power supply of the board, and they should NEVER be connected together.

Activity Time: 3–4 Days

Possible Student Misconceptions

- Microcontrollers do not interact with their surroundings.
- Coding is best done as an individual.
- Computer science is just programming or coding.
- Computer scientists or software engineers just make video games.
- Computers don't need input to make decisions.
- Errors in computer programs are the fault of the computer and not the human who programmed them.
- Pair programming is a "second-best" strategy for when there are not enough computers.
- Programs always work the first time.
- When creating a physical computing system, it will always function properly the first time it's tested.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

What's an Output?

- Begin a conversation to help your students see outputs in your school. Examples they may come up with include a school sign or message board, classroom bells, classroom lights, a classroom TV or projector, school furnace or air conditioning, fire alarm lights, and handicap access door movement.
  - Consider having two students at the front board recording inputs and outputs.
  - Have the class then focus outside their school environment to record a list of both inputs and outputs.
- Distribute a bag of outputs.
- Distribute the I/O Device Images sheet if you collected it or have the students take out their unused portion from Activity 2.1.
- Students should carefully pull the sensors out of the bag, complete the drag-and-drop activity in the course, and chart the sensors (with names and descriptions) in their PLTW Gateway Notebooks.

- While you walk around the room, make observations of student recognition of the various output devices. Consider posing questions to them such as:
  - Are those two servo motors the same?
  - Where do you interact with an output device like this in your daily life?

Exploring Outputs and Actuators

- Students need their micro:bit and related devices for this activity. Either distribute the box of equipment for each pair of students, or if you are using a general storage area, have students collect the equipment necessary for each day.

LED

> **Important**: As students observe the pins, remind them that the 3V pin and the GND pin should never be connected, and that they need to take care when they have loose wires on their desk.

- While the Driver imports and opens the file and the Navigator completes the wiring, notice whether students correctly attach the red wire (P0) to the long leg of the LED.

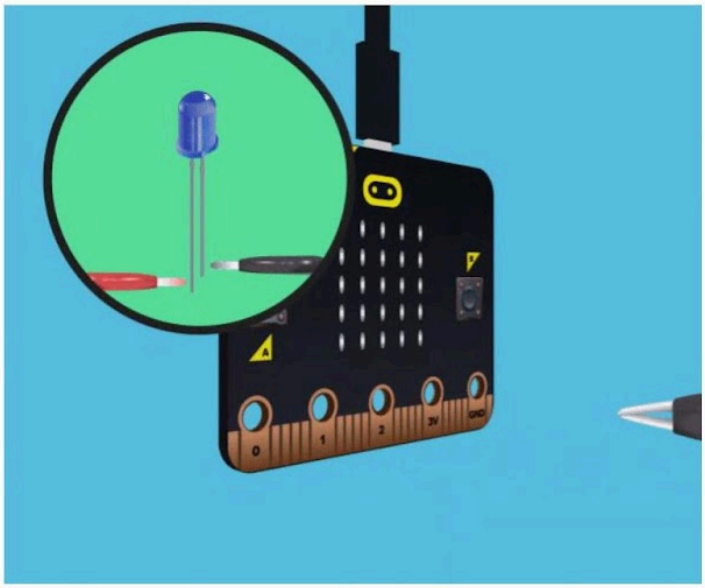


Figure 1. Alligator Clip Connections

Note: The LED connection is a "DC" or direct current system with electricity flowing in only one direction. LEDs only let current through and light up, when wired in the correct order.

- The long leg of the LED should be connected to the positive wire.
- If the legs have become bent (that happens over years of use!), you can identify the short or negative leg (connected to the GND, usually by a black wire) by looking closely at the LED and identifying the flat side of the "bulb". That side has the negative leg or GND.
- The micro:bit GPIO pins (0, 1, 2) have a limited output of both current and voltage; therefore, if your students plan to wire more than one single LED to be controlled by a single pin they will need to wire them in parallel and will be limited to two or three single LEDs or five or six from the LED board. It is not possible to light up more than one LED in series.

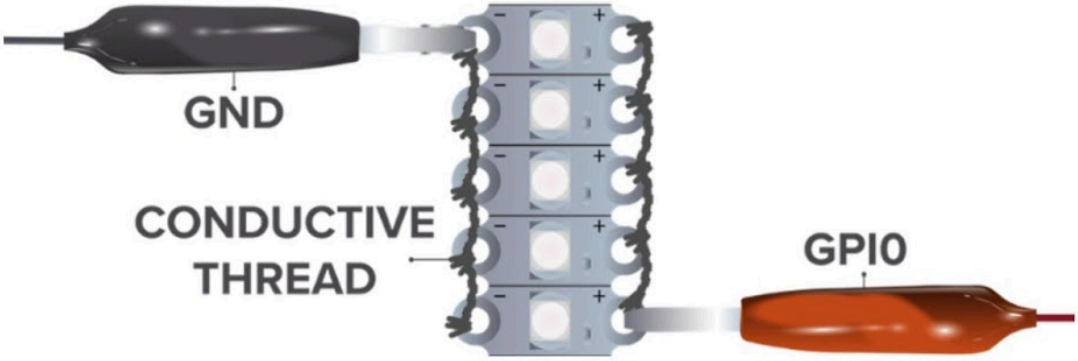


Figure 2. Parallel-wired LED Board

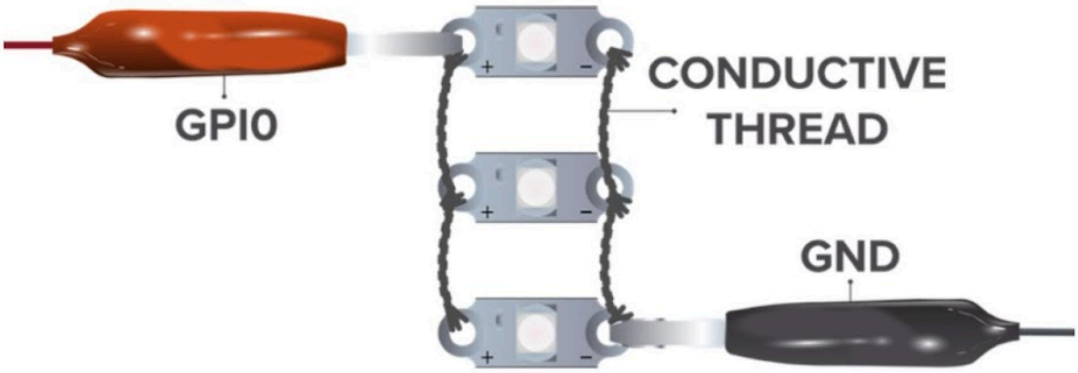


Figure 3. Parallel-wired LED Board Broken Apart

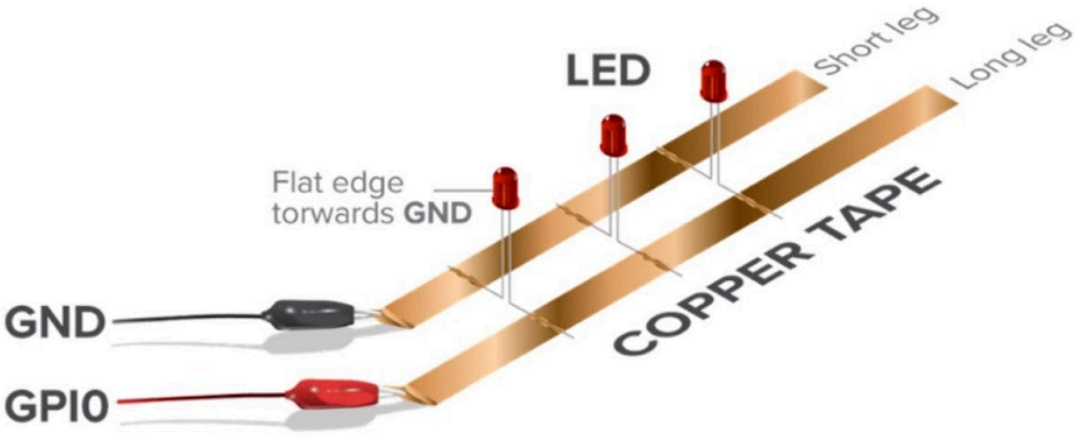


Figure 4. Parallel -wired with Standard LED

- When using the *LED_1* program, the single LED should blink on and off continuously. When students wire the GND to the long leg and the P0 to the short leg, there will be no light, because current will not be flowing through the LED.

When students observe the emulator, they should see the value next to P0 alternating between 1 and 0.

- This is a great time to remind students that LEDs are a digital output and are either on or off. Each time it is on (and wired correctly), an LED lights up "1", and when it is off, "0". There is no in-between here, just on or off, unlike analog devices, which can have a range of values!
- Before you move on to the LED board, students should identify the forever block, which makes the program loop continuously. With the LED Board, they will learn to use a Repeat loop.

LED Board

- Consider having your students switch the Driver/Navigator roles with each new part of the activity or when you feel enough time has gone by (perhaps 10 minutes).
- While the Driver imports and opens the file and the Navigator completes the wiring, observe whether the Navigator is successfully connecting all the ground wires to GND.
- LEDs only let current through and light up if wired in the correct order. Students need to look closely on the top of the LED board to determine the positive and negative sides.

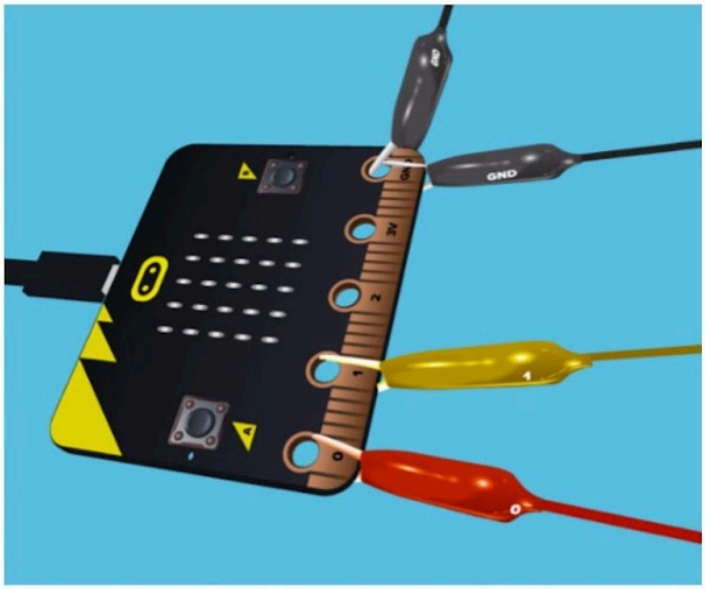


Figure 5. micro:bit Connections

What's in a Pause?

- Facilitate as students work in pairs. Make sure to rotate the positions for Driver and Navigator. Listen to their discussions and encourage them to ask questions of each other. When one suggests a change, ask, "What do you think that will do?"
- Make sure students save their modified program in a naming convention that works for your class (such as A2_2_LED_Board_LastNameLastName) and in a place you specify.

> **Discussion**: Facilitate a class discussion about where outside of school students have seen lights and light displays, similar to what they have been working with on their micro:bits. Or perhaps group students and have each group come up with a list of lights and light displays and then share it with the whole class.

Buzzer

- Consider having your students switch the Driver/Navigator roles with each new part of the activity or when you feel enough time has gone by (perhaps 10 minutes).
- While the Driver imports and opens the file and the Navigator completes the wiring, they may question which wire goes where on the buzzer. LEDs have a required connection order, but for this device it does not matter.
- Depending on the time it took the Driver and Navigator to accomplish their tasks, the buzzer may play a single note, or the program may have run to completion before the buzzer connection was complete.
- Students will likely want to try several modifications. While they are doing a revision, it might be a great time to switch the Driver/Navigator roles.

PlayMe Code

- This code is a simple version of Mary Had a Little Lamb.

Optional Extension: Based on the time you have in class, consider having the students create their own song and then play it on the emulator.

If your students have the opportunity to continue coding outside of class, you may want to have a concert of their songs during the next class or even later in the lesson. If they will be working on this outside of class, remind them of how and where they should save their code file.

Standard Servo

- Consider having your students switch the Driver/Navigator roles with each new part of the activity or when you feel enough time has gone by (perhaps 10 minutes).
- Students will need to connect three wires to the servo connector plug.

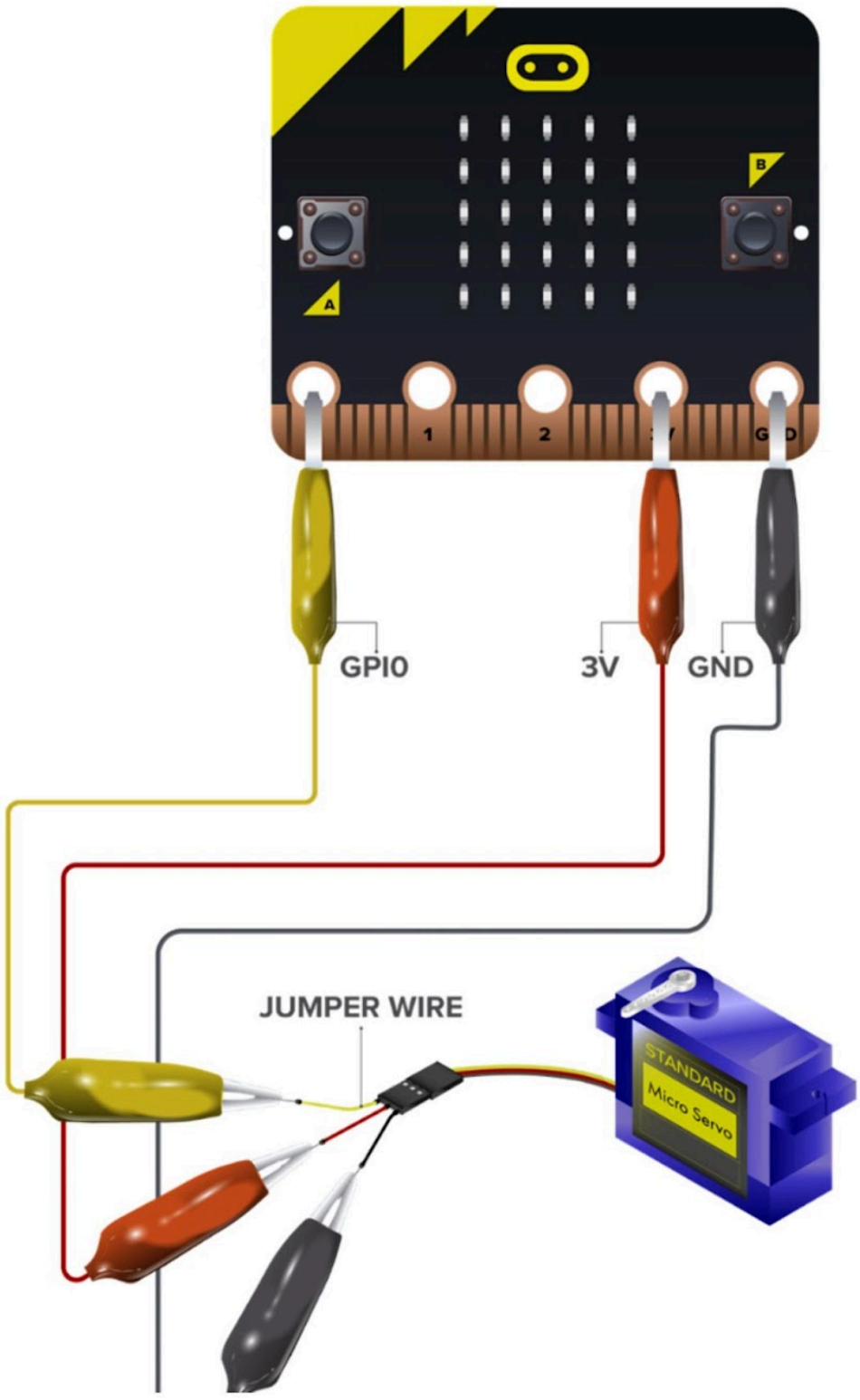

B
A
1
2
GPIO
3V
GND
JUMPER WIRE
STANDARD
Micro Servo



Figure 6. Standard Servo Connections

- Pin 0 connects to the yellow wire
- Pin 3V connects to the Red wire
- GND connects to the brown wire

Note: Standard servos rotate only a certain range, in this case 180 degrees.



Figure 7. Standard Servo Range

When programming the servo, keep in mind that 0 is at one end, 90 degrees is in the middle, and 180 is at the other end. The servo only moves in this range; therefore, when working on the project in this lesson or the problem in Lesson 3, students need to orient this type of servo correctly.

- Make sure students insert banana plugs just far enough to get a snug fit. If they push them in all the way, the fit is loose and will provide an inconsistent connection.
- Depending on where you or your students place the horn (and which horn is used), the A button will make the horn move 180 degrees in one direction, and the B button will make it move 180 degrees in the other direction.

- Servos draw a lot of power from the microbit and function best at the top range of micro:bit's operating voltage. If your students design a solution that involves multiple inputs and/or outputs consider using the separate battery pack to power the servo. The wiring diagram below can help you guide your students in adding this external power source. Note a method of piggybacking the alligator clips to insure good contact on only the GND.

> **Reminder**: Remind students of the file storage location and the naming convention you have established in your class.

Continuous Servo

- Consider having your students switch the Driver/Navigator roles with each new part of the activity, or when you feel enough time has gone by (perhaps 10 minutes).
- Servos draw a lot of power from the microbit and function best at the top range of micro:bit's operating voltage. If your students design a solution that involves multiple inputs and/or outputs consider using the separate battery holder to power the servo. The wiring diagram below can help you guide your students in adding this external power source. Note a method of piggybacking the alligator clips to insure good contact on only the GND.
- Students need to connect three pin wires to the servo motor. If you have a black wire, use that to the ground (brown wire). Connect the red pin to the red wire, and another color (yellow, white, etc.) pin to the orange wire. If you do not have the pin colors shown, don't worry; you just need to remember what is connected where.

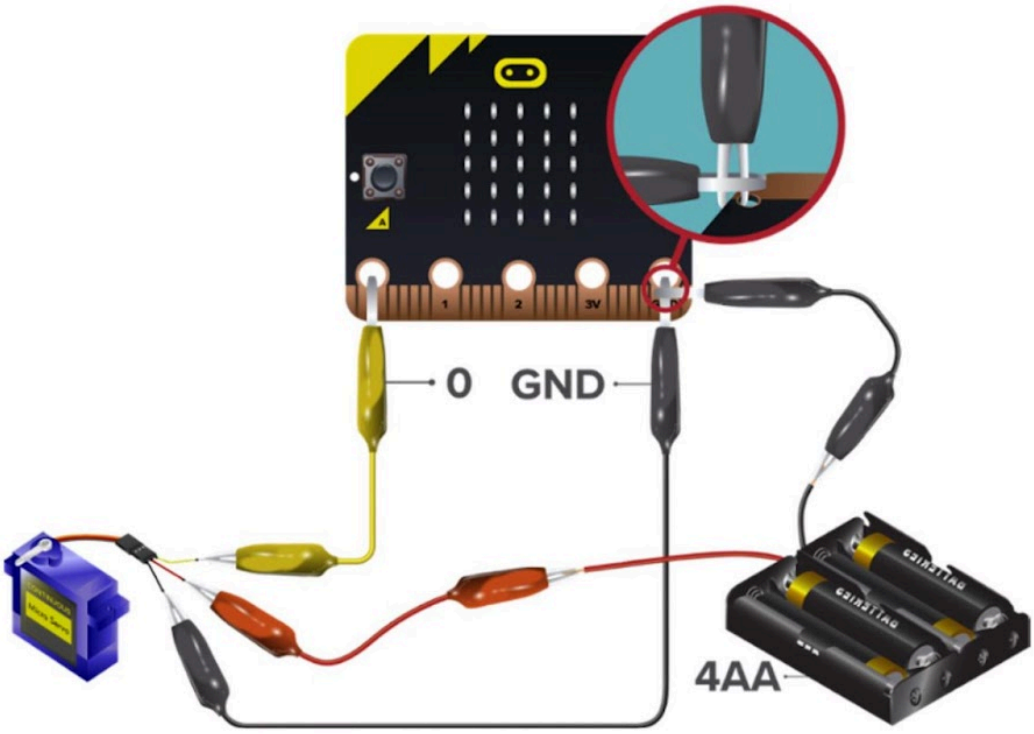


Figure 8. Battery Pack to Servo Motor Connection

- Pin 0 connects to the yellow wire
- GND connects to the brown wire
- 4 AA battery holder red wire connects to servo red pin
- 4 AA battery holder black wire connects to GND

- Make sure the students only insert banana plugs far enough to get a snug fit. If they push them in all the way, the fit is loose and will provide an inconsistent connection.
- Determine whether your students know the rotational directions of clockwise (CW) and counterclockwise (CCW).
- The emulator does not demonstrate a continuous servo motion, so the students need to make all these observations and conduct their modified code on their micro:bit.
- Modifying the values closer to 90 in will cause the servo to slow down its rotation. Possible results (depending on voltage supply):
  - 180 is CCW » 10 s/10 rotations

- 110 is CCW » 13 s/10 rotations
- 70 is CW » 260 s/10 rotations
- 0 is CW » 10 s/10 rotations

Note: When set to full, 180 CCW and 0 CW, the continuous servo rotates about once a second. Students will find when adjusting the speed, that the closer it is set to 90, the slower the servo goes. There will be a point approaching 90 where the servo will just stall and not move.

- Pressing the A button will cause the servo to rotate CCW.
- Pressing the B button will cause the servo to rotate CW.
- Pressing the A and B button will cause the servo to stop.

Discussion: Consider having a class discussion on where students have seen the result of either a standard servo or a continuous servo. Possibilities include:

- Standard servo - a train warning gate or windshield wipers
- Continuous servo - model boats, planes, drones

Using Inputs to Control Outputs

- Consider having your students switch the Driver/Navigator roles with each new part of the activity, or when you feel enough time has gone by (perhaps 10 minutes).
- Students will use one LED of the LED board (no need to break it off) and conductive thread. They may need instruction in tying knots. Consider suggesting for that they loop the conductive thread through the eyelet twice, then pull tight and tie a double overhand knot.
- Most likely the best knot for students to use will be bringing the conductive thread through the eyelet twice and then tying a slip knot.

Connecting the LED

- One wire to the GND is connected to the negative side of the LED.
- One wire to P0 is connected to the positive side of the LED.
- Students will need to remove the forever loop and add the on button pressed command from the Input drawer. (Repeat loops are optional.)

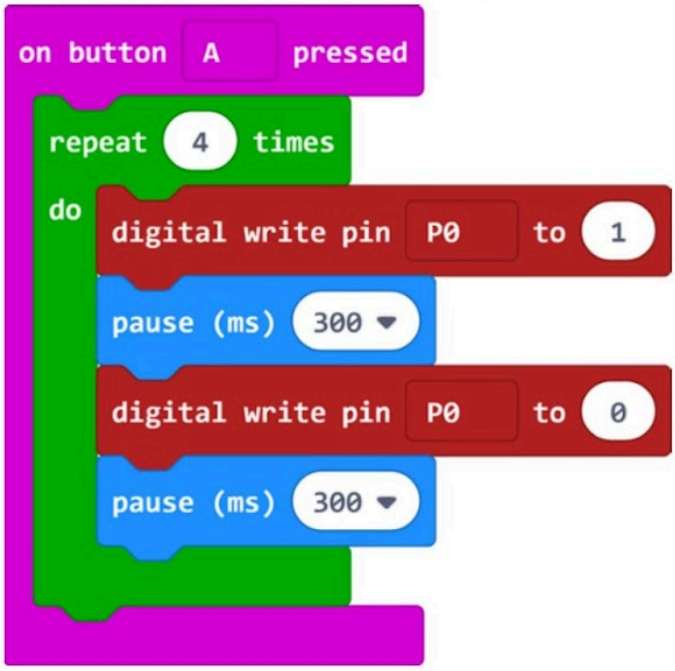


Figure 9. Program for Connecting the LED

Connecting the Buzzer

- One wire to the GND or the grounded clip of LED
- One wire to P1
- To control the buzzer from pin 1, students need to direct the button B control by using the analog set pitch pin to pin 1 command. To find this command in the Advanced drawer, select **Pins** > **...More**.

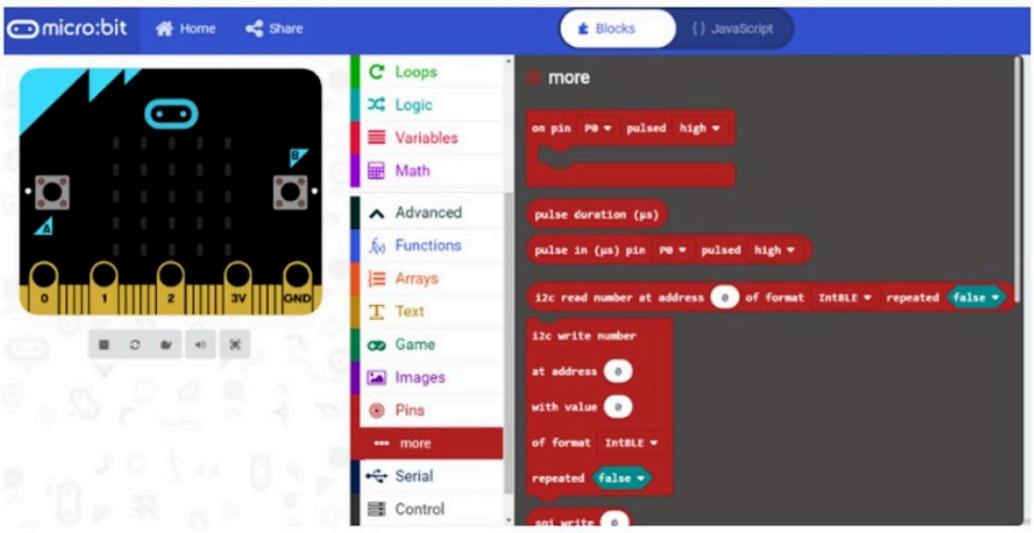


Figure 10. More Drawer

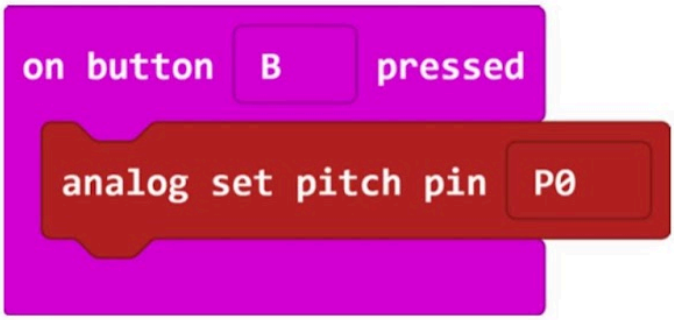


Figure 11. on button B pressed Code Block



Figure 12. Buzzer Tones

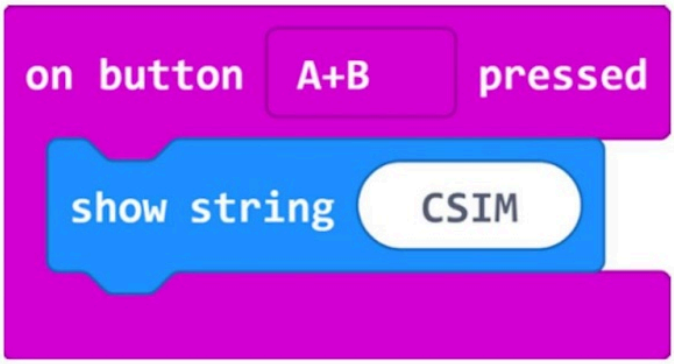


Figure 13. CSIM display

TEACHER EDITION

PLTW GATEWAY

Activity 2.3

Get Connected

Teacher Resources

View Student Course

LEARNING OBJECTIVE

LO

LO 3.4A Abide by professional, legal, and ethical standards when using digital resources.

KNOWLEDGE AND SKILLS

KS

KS 1.2B1 Investigate the types of interactions between users and a proposed solution.

KS 2.4A4 Trace a program and deduce the values that variables will have after the code is executed.

KS 2.6A3 Describe possible positive and negative social impacts of computing.

KS 3.4A2 Describe ethical and legal implications for accessing and copying digital information.

ESSENTIAL QUESTIONS

EQ

EQ 2.1 What do programming best practices look like?

EQ 2.2 Why are interactions between machines, humans, and the environment beneficial, problematic or both?

RESOURCES

Activity 2.3 Answer Key

Preparation

- Ensure students can download and access the activity's .hex files. You may choose to download them and place them in an accessible location for students, such as a shared network drive or student flash drives.
- Prepare and store micro:bits and any input/output components in an easily accessible location for students.

Ethical Considerations

As students program and test the transfer of data packets from one micro:bit to another, make sure they are sharing micro:bits and collaborating between teams appropriately. Downloading micro:bit programs to another team's microcontroller without permission is unethical and can be considered inappropriate hacking of the device. When appropriate, discuss with students the responsible use of micro:bits and the transfer of digital information including code.

Activity Time: 3 Days

Possible Student Misconceptions

- Algorithms are all linear, meaning they follow only one set, sequential path.
- Computer/computing variables are the same as mathematical variables.
- Computer systems send and receive wireless information magically.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Bluetooth Wireless Technology

- Instruct students to read the introduction independently or as a class. Remind students of The Digital Dive activity they completed during Activity 1.1 The Brain. You might ask students to recall the wireless connections part of the activity and describe their experience. Let students know they will be diving deeper into wireless technology in this activity.
- Instruct students to read about the term Bluetooth®. Students might find it interesting that there are several theories around how King Harald Bluetooth got the name "Bluetooth". One theory is that the king had a visible bad tooth that was extremely dark and appeared to be blue.
- Students who have difficulty reading the brief history of Bluetooth® technology may need additional support, such as reading this text out loud as a class and discussing it.
- You might make connections to history by encouraging students to research King Harald Bluetooth and learn more about his reign. Students could do this outside of class or as a collaborative project with a social studies or history class.
- Remind students of The Brain activity and how they transferred data packets using the Connection card during The Digital Dive. The Connection cards ensured that the devices were paired, but were also passed along with the data packet that told the next team what to do.

Micro:bit BLE Antenna

- Ensure students follow the APB instructions and identify the BLE antenna on their micro:bits.

Class Discussion: *Why do you think the code blocks you'll use to program the micro:bit BLE antenna are listed in a Microsoft MakeCode drawer called Radio?*

Student responses to this formative assessment question should include a description of Bluetooth technology. BLE is a low-energy form of Bluetooth wireless technology that uses short-wave radio signals to communicate between electronic devices.

Bluetooth technology often exchanges data packets using short-range radio frequencies between 2402 and 2480 MHz. For this reason, one of the drawers with code blocks that use the BLE antenna is called "Radio".

- Throughout this activity, you might refer students to the reference pages that describe the code blocks found in the following drawers. Students can access these pages from the Help menu in the programming environment.
  - Radio **https://makecode.microbit.org/reference/radio**
  - Variables **https://makecode.microbit.org/blocks/variables**
- The Radio drawer code blocks only connect micro:bits to other micro:bits. Students who have mastered the Radio code blocks in this activity may later explore the use of additional code blocks in the Bluetooth and Devices drawers to connect their micro:bit to a smartphone in the optional Get Connected challenge.

What's the Answer?

- Ensure students understand that variables in math are typically unknown or numerical values. Variables in computer programs are used to store three data types, and that data can change during the runtime of the program. Students will experience this in the Code Tracing a Variable section of this activity.

Data Type	Examples
Number	–2.5, 0, 3000
Text (String)	"I love apples!", "Hello world!"
Logic (Boolean)	True, False

- In this activity, students will only use variables to store numeric values in computer programs. To alleviate potential student misconceptions in future computer science courses, make sure you explain that variables in computer programming can store more than just numeric values.

> **Note**: Your students may be interested to know that the example code used to show the string "hello world" in this APB is written in the *Python*® computer programming language.
>
> ```
> x = str('hello world')
> ```

- When students try out the *length of string* code block, they should see the number 5 on the LED grid in the emulator. Five is the number of characters in the string *howdy*.
- Before students proceed to the procedure for trying out the math program, briefly discuss what they think will appear on the LED grid of the receiving micro:bit. You might do a Think-Pair-Share or whole class discussion.

Calculation Program

- Facilitate as students work independently to complete the steps to try out the math program. Give them five minutes or less to try it out before asking the class, What value of *x* appeared on the receiving micro:bit's LED grid in the emulator?
- Allow students a minute or so to figure out the final step of this section to see how data packets can be sent both directions between micro:bits.

Code Tracing a Variable

- As students complete the Code Tracing a Variable section of the activity, facilitate their learning by observing students and assisting as needed. You may need to remind students how to create a code trace chart. Refer them to the code trace chart image in the APB or the code tracing information in Crush the Bug.
- As suggested in the APB, encourage students to show their math computation work in the Outcome column. Then students should clearly state the current value of *x* below their work. Highlight how the variable's data changed during the runtime of this program and how that is different from variables in math.

Smart Devices

Discussion: Discuss the use of smart devices and the Internet of Things (IoT) as a class. Ask students to think about and share devices they've seen used as a connected device to send videos, messages, and other data.

Ask students how these connected devices make our lives easier.

- What are the benefits to controlling other devices with a smart device?
- Are there situations where a smart device is probably not necessary?
- You may share the example of a refrigerator that can send a photo or a tweet on Twitter.
  - Is this a useful feature for a refrigerator?
  - Would it be helpful to have a photo of what items are currently inside your refrigerator? Why or why not?
- Do you think vehicles should be connected to each other and share data between cars? Is there a security risk with connecting cars?

These are current ethical questions in the tech industry. There are no correct answers to these discussion prompts. Allow students to form their own conclusions about whether or not certain devices need to be "smart". Ask students whether they think their answers will change in the future as technology changes.

Remote Control

- Facilitate the final exercise in this activity. Assist students by prompting them with questions when necessary. As students unplug their micro:bits from USB cables and hook up battery packs, ask them to think about the advantages and disadvantages of using another power source with the micro:bits. Can you send programs more easily to the micro:bit when it's using battery power?
- Observe students as they complete the Remote Control section of the activity. Make note of teams' progress as they test their programs. You may ask yourself the following questions during your observations:
  - Are students successfully connecting input and output components?
  - What issues are students having with programming?
    - Unsure of what code block to use
    - Unsure of how to program radio signals using new code blocks in the Radio drawer
    - Difficulty programming code for two micro:bits and understanding the logic behind the program
  - Are students using debug strategies to successfully fix any issues with their programs?
- If you notice students having difficulty in the areas described above, you may decide to ask questions that promote the use of previously learned skills, such as debugging strategies. As necessary, you can remind students of programming techniques and strategies learned so far in the unit to trigger their thinking.

> **Reminder**: As students complete this exercise, remind them of pair programming roles. Ensure the Driver and Navigator in each student team are adhering to their defined roles. If not, review the pair programming strategy with students. You may choose to have students watch the pair programming video again outside of class.

Debug Steps

Refer students to the debug steps and facilitate as necessary. While students are debugging their team's program, observe their progress and facilitate as needed. You may ask them prompting questions such as:

- What does the program do?
- What outcomes have you tested?
- Have you identified any unexpected outcomes?
- Did you read all your comments to check the program's logical sequence? Did you find any problems or inconsistencies?
- Have you isolated the anomaly? How do you plan to fix or debug it?

TEACHER EDITION

PLTW GATEWAY

Project 2.4

Secrets and Safes

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 1.2A Apply an iterative process to solve a problem or create an opportunity that can be justified.

LO 3.3B Justify decisions and provide rationales when making trade-offs between resources.

KNOWLEDGE AND SKILLS

KS

KS 1.2A1 Identify a problem and justify the pursuit of a solution to the problem.

KS 1.2A2 Explain the goal of any design process.

KS 1.2A3 Compose appropriate design requirements that include specifications, constraints, desired features, and testable parameters.

KS 1.2A4 Generate ideas or build upon existing ideas to innovate.

KS 1.2A5 Create and evaluate multiple solution options based on criteria, and select the best option.

KS 1.2A6 Iteratively test throughout the design process to ensure that a solution meets the requirements.

KS 1.2B2 Involve prospective users early and often during the design process.

KS 1.2B3 Recognize that understanding human interaction with a solution can improve the design and extend the abilities of humans.

KS 3.3A1 Identify resources managed in a project to include time, capital, energy, information, materials, people, tools, or machines.

KS 3.3B1 Recognize that processes involve trade-offs when weighing the need against effort and resources available.

KS 3.3B2 Demonstrate sound judgement when making decisions regarding resources and their potential trade-offs.

KS 3.4A1 Describe property rights of digital media and software to facilitate sharing and avoid infringement.

KS 3.4A1 Describe the importance of protecting personal information.

ESSENTIAL QUESTIONS

EQ

OA 1 How do you express yourself and your creativity through computer science?

EQ 2.1 What do programming best practices look like?

EQ 2.2 Why are interactions between machines, humans, and the environment beneficial, problematic, or both?

EQ 2.3 How is a design process used to develop physical computing systems?

MATERIALS

- Optional: Wire cutters/strippers
- Optional: Additional wire

RESOURCES

- 2.4 Secrets and Safes Rubric
- Wiring Diagram
- Project 2.4 Answer Key

Preparation

- Ensure all materials listed and various construction tools, such as scissors and duct tape, are available for students as they begin designing their solution for the project.
- Inputs, outputs, micro:bits, and conductive materials should be easily accessible to the students.
- As necessary, collect cardboard, foil, and other miscellaneous materials students might use to conduct electricity or construct their device. You might also ask students and others at your school to contribute or supply a variety of materials.

Ethical Considerations

During this project, discuss with students the importance of cybersecurity and protection of personal data. Relate this to the protection of physical objects as described in the project's design requirements. Ask students to consider how protecting personal physical objects is different from or similar to protecting personal digital files and information. Is it the responsibility of the designers of computer systems, including physical computing systems, to protect those who use it? Why or why not?

Project Time: 6 Days

Possible Student Misconceptions

- When creating a physical computing system, the system and code will function properly the first time tested.
- Coding is best done as an individual.
- Pair programming is a "second-best" strategy for when there aren't enough computers.
- There is only one algorithm to solve a problem.
- You cannot learn from failure; persistence is a waste of time.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Keep It Secret... Keep It Safe

- Ensure students understand the three security device options that they can create. To refresh their memories, you might sum up the following skills and techniques they learned in Lesson 2 about programming with inputs and outputs:
  - Ways to connect hardware (inputs and outputs)
  - Programming with conditionals and loops
  - Using comments to document your algorithms and program logic
  - Debugging steps
  - Programming inputs and outputs
  - Using wireless technology
- Review the pair programming strategy and the Driver and Navigator roles. If necessary, you may instruct students to review the video on pair programming in Need Input activity.

Design Requirements

- Review the design requirements and project rubric with students.
- To assist students' thinking, encourage them to think of different ways people can be alerted or notified. Using the output components available, students might

decide to use sound only, light only, or both sound and light outputs in their design. When using the BLE antenna, students might send signals to other micro:bits or devices to trigger an alert.

> **Note**:
> - If two teams of students (four students total) decide to create a physical computing system that relies on BLE antennas and Bluetooth® technology, allow them to pair up.

Design Process

- Refer students to the Gateway Design Process shown in the APB. Students will follow this process throughout the project. Each stage of the process is outlined in the APB headings.
- Students will be guided through a design process during this project. However, it isn't the learning or instructional intent to focus on understanding and using a formalized design process. Encourage students to plan, make mistakes, and learn from those mistakes to make improvements as the primary goal for using a design process throughout the project.
- You might have students write their responses for each stage of the design process in their PLTW Gateway Notebooks. These responses can then be presented as part of each team's final design solution to demonstrate their work.

Define the Problem

- Each student should individually write their team's response and show their work in their own PLTW Gateway Notebook.

Generate Concepts

- Facilitate the design process with students as they discuss the questions for each of their design solution ideas. Ensure students are discussing all possible ideas from each team member.

- If they have concerns or questions while considering each design idea, refer students to the design requirements and review available resources as needed.
- For formative assessment, observe the students. Note their collaboration in this early stage of the design process. These observations can be compared later to their teamwork abilities toward the end of the design process to measure their growth in collaboration skills.

Design a Solution

Facilitate as students design their hardware and flowcharts. Ask students questions about their designs to help them think through how their devices and programs will function.

- Keep in mind the limitations of LEDs presented in **Activity 2.2**.
- Keep in mind the power limitations of the micro:bit and the alternative independent battery pack option presented in **Activity 2.2**.

Make sure each student is documenting their plans (hardware sketch and flowchart) in their own PLTW Gateway Notebook. These learning artifacts may be used later as evidence for assessment purposes.

> **Reminder**: Refer students to the flowchart graphic if they have difficulty remembering the symbols for creating a flowchart.

Build and Test Your Device

> **Note**: If two student teams (four students) are developing a physical computing system together that uses Bluetooth® wireless technology, then you may recommend that one student team work on the program while the other builds the hardware. Have the two teams switch between working on the software and hardware about every 20 minutes.

- As you facilitate, observe students' progress while they construct their hardware and create their programs in the Microsoft MakeCode programming environment. Make note of students' teamwork, collaboration, and ability to resolve any potential design issues. Record your observations on the Interactions Rubric as appropriate.
- Instruct students to save their team's .hex file in a location of your choosing. This might be a flash drive, local drive, or a shared network drive. You might establish a folder for each class you teach and instruct each class to save their .hex files in their class folder. Alternatively, instead of using folders to distinguish between multiple classes, you might instruct students to include a class number at the beginning of their file names.
- Refer to the debug steps as necessary to help students during their device testing. You may ask them prompting questions such as:
  - What does the program do?
  - What outcomes have you tested?
  - Have you identified any outcomes that were unexpected?
  - Did you read all the comments to better understand the program's purpose and logical sequence? Did you find any problems or inconsistencies?
  - Have you isolated the anomaly? How do you plan to fix or debug it?

Evaluate Your Solution

- Continue to facilitate as teams test their design solutions and practice their presentation skills. Ensure students have individuals in class and outside of class to try out their devices to get feedback from various people.
- Encourage students to take their physical computing systems home to test with family and friends.
- Use your observations to assess each team's progress, their ability to debug any issues that arise, and their practice of presentation skills. Record your observations on the project rubric.

Import Note: Remind students that like the debugging steps for fixing code, the design process is iterative. Encourage student teams to persevere and look for opportunities to improve their physical computing system rather than dwelling on mistakes or issues that arise. This is important during both hardware and software development.

- Refer students to the steps for sharing their code in the programming environment. This is a great way for students to share their project and have others test their program. You might also promote social debugging—sharing code for others to assist with debugging. Talk with students about appropriate ways to give credit when sharing others' code. This might be as simple as including the original creator's name or handle, a fictional username.

Present Your Solution

Presentation Options

- You might choose to have each student team present to the whole class while you complete a rubric for each team or student.
- Additionally, you might have each team present to a school administrator or member of the community while you or the audience completes a rubric for each team or student. You might decide to hold an exhibition of student projects and invite the school or community to participate in reviewing each team's physical computing system.
  - Keep in mind that these presentation methods will take additional time to complete and assess.
- Option: Have each team peer assess another team as they present. Each student will use the Interactions Rubric to evaluate the presenting team's presentation. You can use these rubrics, along with your observations and additional learning artifacts, to formatively assess each student.

- To formatively assess student understanding the importance of teamwork and perseverance throughout designing and creating physical computing systems, you

might have students respond to the final three APB questions in their PLTW Gateway Notebooks or on an exit ticket. Even if you choose not to formally assess student responses for these questions, they are important to discuss as a class or reflect on in small groups as students prepare for the unit problem in Lesson 3.

LESSON 2 INTERIM ASSESSMENT

The Computer Science for Innovators and Makers Lesson 2 Interim Assessment is located in the Unit Level Resources, Assessment Page. The assessment is a starting point for you to use with your students. Feel free to edit this assessment to best meet your needs and the needs of your students. Use the Lesson 2 Interim Assessment Answer Key, to evaluate student work.

TEACHER EDITION

Lesson 3

Program the Physical World – Quick Guide

Preface

Within teams, students become innovators and makers. Teams apply their physical computing knowledge and skills as they design and create one of three problem options:

- A wearable safety device someone might use when completing a physical activity outside at night
- An engaging art installation to help improve a community space
- A useful mechanical dispenser for a person or animal who needs assistance to retrieve an object

Teams collaborate and learn that solving authentic problems involves the unit content knowledge, as well as skills from other disciplines, such as communications, mathematics, and science.

Essential Questions

1. How do you express yourself and your creativity through computer science?
2. Do all professions need computer science?
3. Does society need computer science and physical computing projects?
4. How are ethics, safety, and computing projects related?

Activity, Project, or Problem	Days
Activity 3.1 Clean Up Your Code	2
Problem 3.2 Interactions	10
Unit Assessment	1
Total	**13 Days**

TEACHER EDITION

PLTW GATEWAY

Activity 3.1

Clean Up Your Code

Teacher Resources

View Student Course

KNOWLEDGE AND SKILLS

KS

KS 2.3A1 Explain the purpose of an abstraction.

KS 2.3A2 Identify what has been made more general by an abstraction and identify what details have been hidden or removed.

ESSENTIAL QUESTION

EQ

EQ 3.2 Does society need computer science and physical computing projects?

RESOURCES

Activity 3.1 Answer Key

Preparation

Ensure all materials are available for students.

Activity Time: 2 Days

Possible Student Misconceptions

- When creating a program, the code will function properly the first time tested.
- There is only one algorithm to solve a problem.
- Programs that are shorter, in length of code, run faster when executed by a computer system.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Elegant Code

- Instruct students to read the Elegant Code section as a team or class. Discuss with students how the code is simplified in the Blinking2 program. Ask students which program is easier to read and has a logical sequence that is easier to understand. What should happen when either Blinking1 or Blinking2 program is executed on a micro:bit?
- To consider how elegant code can decrease coding errors, students should recall the debugging skills they've learned so far. Creating a simplified and clean version of their code, complete with comments, can help students avoid coding errors and more quickly identify bugs when they arise.

Programming Languages

- Instruct students to read the Programming Languages section to learn a little about text-based programming languages.
- Encourage students who are interested in various computer programming languages to do research on the side. Some interesting languages include CoffeeScript, Logo, *Python*®, Go, and Swift. Students might choose to research the history of a programming language and its popular uses. Ask students to consider what types of programming languages might exist in the future. Will more visual-based languages, like the one used in Microsoft's MakeCode be developed?

Lines of Code

- Instruct students to complete the steps in Lines of Code independently to further explore the Blinking1 and Blinking2 programs. Ensure that students have access to the two .hex files before they start this activity.

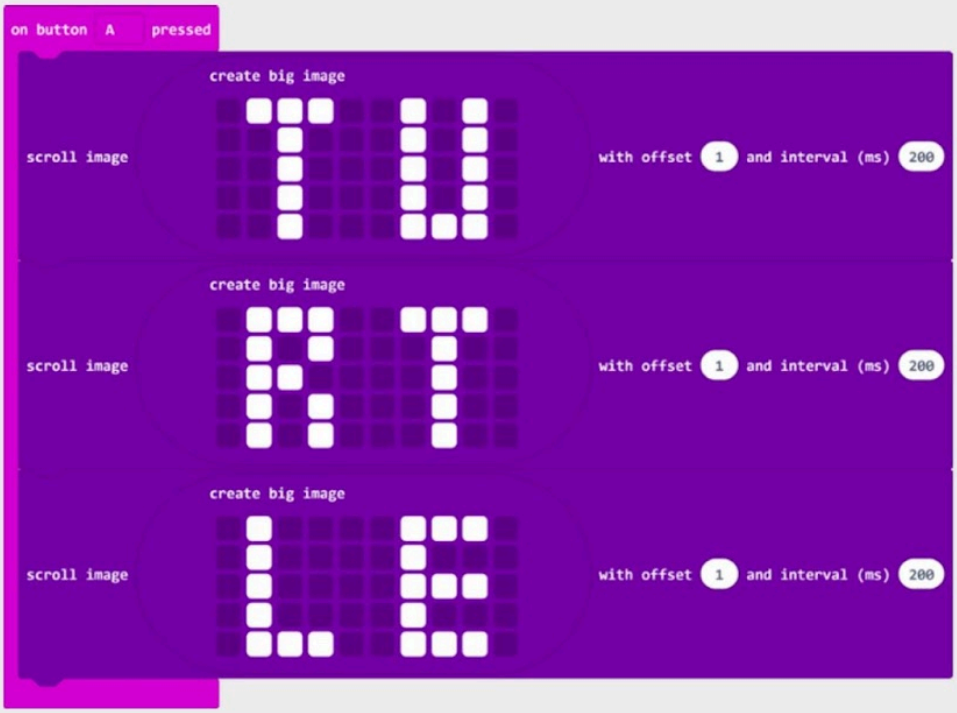
> **Important**: It's important for students to understand that the length of a program's code doesn't impact the speed a computer system can execute or run that program. To determine how fast a computer program will run, one must consider the complexity of the program's algorithms and other factors. Address this misconception at the end of the Lines of Code section.

Abstraction

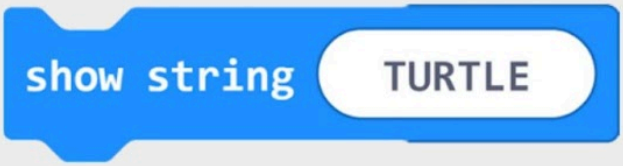
- You may have students complete the Abstraction section independently or as a class. Completing this exercise as a class will contain classroom noise, if that's a concern. You may also choose to have students use earbuds or headphones when running the programming environment emulator.
- When students have finished testing both programs, have them reflect on their observations by answering the questions in this section of the APB. Discuss what they observed. Ask them how abstractions might make code easier to create, read, and simplify.

Student Accomodation: Deaf students and learners with hearing difficulties can use the following steps for the Abstraction section. You may decide to complete these as a class.

1. In the programming environment, create the following program.



2. Run the program in the emulator or on a micro:bit.
3. Identify the word spelled on the LED grid.
4. In your program, delete all of the *scroll image* code blocks.
5. Add the following code block from the Basic drawer inside the *on button A pressed* code block.



6. Again, run the program in the emulator or on a micro:bit.

Code Cleanup

Instruct students to complete the Code Cleanup exercise independently. Ensure that students have access to the .hex file before they begin this section. Facilitate student learning as they complete the exercise. Encourage students to discuss their solutions with classmates after attempting to solve the problem independently. This isn't considered cheating, but rather a social coding opportunity for students to discuss alternative solutions.

TEACHER EDITION

PLTW GATEWAY

Problem 3.2

Interactions

Teacher Resources

View Student Course

LEARNING OBJECTIVES

LO

LO 3.4B Identify ethical considerations that must be considered when creating solutions or opportunities.

LO 3.4C Consider accessibility and equity when designing products, creating solutions, and collaborating with others.

LO 3.5A Identify the variety of careers related to engineering, biomedical sciences, and/or computer science.

KNOWLEDGE AND SKILLS

KS

KS 3.1A1 Apply team norms to encourage productivity and define how a team will function and measure its success.

KS 3.3A2 Devise, prioritize, and execute a plan to solve a problem, including short-term and long-term objectives when appropriate.

KS 3.4B1 Recognize that ethical considerations include but are not limited to safety, impact on future generations, or recognizing the work of others (intellectual property).

KS 3.4B2 Explain how universal design considers the broadest possible spectrum of human ability in the design of products, environments, and information systems.

KS 3.4C1 Describe and give examples of universal design strategies that increase accessibility to appropriate resources.

KS 3.5A1 Recognize the education and credentialing requirements for careers in engineering, biomedical sciences, or computer science.

KS 3.5A2 Explore a variety of careers related to engineering, biomedical sciences, or computer science.

KS 3.5B1 Demonstrate personal responsibility and intitative for independent learning, keeping in mind that technology and processes will always evolve.

ESSENTIAL QUESTIONS

EQ

EQ 3.1 Do all professions need computer science?

EQ 3.2 Does society need computer science and physical computing projects?

EQ 3.3 How are ethics, safety, and computing projects related?

MATERIALS

- Optional: Wire cutters/strippers
- Optional: Additional wire

RESOURCES

- 3.2 Interactions Rubric
- Wiring Diagram

Preparation

- Ensure all materials listed and various construction tools, such as scissors and duct tape, are available for students as they begin designing their solution for the project.
- Inputs, outputs, micro:bits, and conductive materials should be easily accessible to the students.
- As necessary, collect cardboard, foil, and other miscellaneous materials students might use to conduct electricity or construct their device. You might also ask students and others at your school to contribute or supply a variety of materials.

Problem Time: 10 Days

Caution: When working with conductive thread, keep the thread strands separated and don't cross them. Conductive thread is like working with open or exposed wires. Crossing thread strands may short circuit and cause intense heat.

Possible Student Misconceptions

- When creating code and a physical computing system, the system and code will function properly the first time tested.
- Coding is best done as an individual.
- Pair programming is a "second-best" strategy for when there aren't enough computers.
- There is only one possible algorithm to solve a problem.
- You cannot learn from failure; persistence is a waste of time.

The following guidance relates to content in the student course. The headings align with sections in the corresponding student content page.

Physical Computing Interactions

- Instruct students to read the unit problem descriptions and design requirements. Students should also review the Physical Computing (Unit Problem Options) video, first presented in The Brain activity, to get an overview of the three unit problem options.
- To expand the unit problem option where students create a device that dispenses objects, you may allow students to design, code, and develop a physical computing system that provides a service to pets. This can include any programmed device that assists or provides comfort to an animal.
- Facilitate student learning throughout their iterative design process during this APB. However, the purpose of this unit problem is to provide students with an open-ended problem where they can demonstrate the knowledge and skills they've gained throughout this unit. While you may provide students with some prompting questions during their work, you might refrain from giving the same level of facilitation and support you did during the Secrets and Safes project. Use this unit problem to comprehensively assess your students' learning.
- Ensure students understand the three unit problem options. You might sum up the skills and techniques they've learned about programming with inputs and outputs in Lessons 1 and 2. Assign students to teams of two or allow students to choose their partners.

User Interactions

- You may have students read this section independently, in pairs, or as a class. If your schedule allows it, you may assign this for students to read as an out-of-class or flipped-class activity before they begin the APB.

> **Discussion**: Discuss with students the importance of designing computing systems that are safe for users and can secure their data. Universal design is not the same as an accommodation. Interactive systems that are universally designed meet the accessibility needs of all users without having to make special accommodations for a specific group of users.

Go Team!

- Review the pair programming strategy and the roles of a Driver and a Navigator. If necessary, instruct students to review the video on pair programming in Activity 2.1 Need Input.
- You may have students read this section independently, in pairs, or as a class. If your schedule allows it, you may assign this for students to read as an out-of-class or flipped-class activity before they begin the activity.
- Facilitate the students' creation of team norms. Assist them with developing a brief and concise list of rules or team expectations.

Design Process

- Refer students to the Gateway Design Process shown in the APB. Students follow this process throughout the project. Each stage of the process is outlined in the headings that follow.

Define the Problem

- You might have students write their responses for each stage of the design process in their PLTW Gateway Notebooks. Students can present these responses to demonstrate their work as part of their team's final design solution. Students should individually write their team's response and show their work in their own PLTW Gateway Notebook.
- To ensure students clearly understand what's expected, review the design requirements and the **problem rubric** with students.

> **Note**:
> - If two teams of students (four students total) decide to create a physical computing system that relies on BLE antennas and Bluetooth® technology, allow them to pair up.

Generate Concepts

- Facilitate the design process with students as they discuss the questions for each of their design solution ideas. Ensure students are discussing all possible ideas from each team member.
- If they have concerns or questions while considering each design idea, refer students to the design requirements and review available resources as needed.
- As they generate concepts and discuss their design solution ideas, observe the students and answer questions about their designs. Facilitate their discussions as needed and note their collaboration in this early stage of the design process. You can compare these observations later to their teamwork abilities toward the end of the process.

Design a Solution

- Facilitate students as they design their hardware and flowcharts. Ask students questions about their designs to help them think through how their hardware and program will function. Keep in mind the limitations of LEDs and Servos presented in **Activity 2.2**. Make sure each student is documenting their plans (hardware sketch and flowchart). You may use these later for assessment purposes.

> **Reminder**: If students have difficulty remembering the symbols for creating a flowchart, refer them to the flowchart graphic.

Build and Test Your Device

- As you facilitate, observe students' progress while they construct their hardware and create their programs. Make note of students' teamwork, collaboration, and ability to resolve any potential design issues. Record your observations on the Interactions Rubric as appropriate.
- Instruct students to save their teams' .hex file in a location of your choosing. This might be a flash drive, local drive, or a shared network drive. You might establish a folder for each class you teach and instruct each class to save their team's .hex file in their class folder. Alternatively, instead of using folders to distinguish between multiple classes, you might instruct students to include a class number at the beginning of their file names.

> **Note**: If two student teams (four students) are developing a physical computing system together that uses Bluetooth® wireless technology, then you may recommend that one student team work on the program while the other builds the hardware. Have the two teams switch between working on the software and hardware about every 20 minutes.

- Refer to the debug steps as needed to help students during their device testing. You may ask them prompting questions such as:
  - What does the program do?
  - What outcomes have you tested?
  - Have you identified any unexpected outcomes?
  - Did you read all the comments to better understand the program's purpose and logical sequence? Did you find any problems or inconsistencies?
  - Have you isolated the anomaly? How do you plan to fix or debug it?

Reminders:

- If team issues arise, remind students to review their team norms. Facilitate the resolution of any team issues by discussing potential solutions and team norms with the students.
- Refer students to the Documentation Reminder in the APB for instructions to capture screen images of their code blocks and save them to share their coding progress later.

Evaluate Your Solution

- Continue to facilitate as teams test their design solutions and practice their presentation skills. Ensure students have individuals in class and outside of class to try out their devices to get feedback from various people.
- Use your observations to assess each team's progress, their ability to debug any issues that arise, and practice their presentation skills.
- You might instruct students to use the problem rubric as a peer feedback tool during this stage of the design process.

Important: Remind students that like the debugging steps for fixing code, the design process is iterative. Encourage student teams to persevere and look for opportunities to improve their physical computing system, rather than dwell on mistakes or issues that arise. This is important during hardware and software development.

- Refer students to the Presentation Practice Tips in the APB for a checklist of items they should consider while preparing for their presentation.
- Refer students to the steps for sharing their code in the Microsoft MakeCode programming environment. This is a great way for students to share their project and have others test their program. You might also promote social debugging—sharing code for others to assist with debugging.

Present Your Solution

Presentation Options

- You might choose to have each student team present to the whole class, while you complete a rubric for each team or student.
- Additionally, you might have each team present to a school administrator or member of the community, while you or the audience completes a rubric for each team or student. You might decide to hold an exhibition of student projects and invite the school or community to participate in reviewing each team's physical computing system.
  - Keep in mind that these presentation methods will take additional time to complete and assess.
- Option: Have each team peer assess another team as they present. Each student will use the Interactions Rubric to evaluate the presenting team's presentation. Along with your observations and additional learning artifacts, you can use these rubrics to formatively assess each student.

Conclusion

After students reflect on the two final questions, discuss them as a class. Ask students to share their thoughts and explain what they learned throughout the unit. You may also ask that students write their answers to the final two questions.

UNIT SUMMATIVE ASSESSMENT

The Computer Science for Innovators and Makers Unit Summative Assessment is located in the Unit Level Resources, Assessment Page. Students will take the End-of-Unit Assessment. The assessment is a starting point for you to use with your students. Feel free to edit the assessment to best meet your needs and the needs of your students. Use the CS for Innovators and Makers Summative Assessment Answer Key, to evaluate student work.

PLTW | Gateway

Gateway Computer Science for Innovators and Makers (CSIM)
Curriculum Revisions May 2019

The following revisions were implemented in the **2018-2019 Computer Science for Innovators and Makers (CSIM)** curricula and released May 2019.

Course Resources		
Title of page/asset	**Description of update**	**Description of update**
Unit-wide	Update	Instructions, media elements, and assessments have been updated to match the latest Microsoft MakeCode programming environment.
Teacher Resources: General Prep and Considerations page	Update	Information regarding both versions of the micro:bit has been added in Tech Tips.

Lesson 1 Blink!		
Title of page/asset	**Description of update**	**Description of update**
Activity 1.1 The Brain	New	Note and image added about another version of the micro:bit.
Activity 1.1 Teacher Resources	New	Note about the new version of the micro:bit has been added.
Activity 1.3 How to Do It	Update	Clarified directions for downloading .hex file and sending to the micro:bit.
Activity 1.3 Teacher Resources	Update	Information added for Chrome settings for sending .hex files to a micro:bit using Chromebooks.
Activity 1.3 LED Grid Practice Exercises (and Teacher Resource)	Update	Use of the unplot block has been removed. Students can edit their pyramid solutions by deleting the unnecessary lines of code from their algorithms.

PLTW | Gateway

Lesson 2 The Ins and Outs		
Title of page/asset	**Description of update**	**Detailed description of update**
Activity 2.3 Get Connected	New	Note added about the possibility of the compass and accelerometer sensors to be combined as a motion sensor.
Activity 2.3 Challenge	Update	The nested challenge has been removed, because the app no longer supports the new hardware.

Lesson 3		
Title of page/asset	**Description of update**	**Description of update**
Activity 3.1 Clean Up Your Code	Update	Information that the code blocks are based on Blockly has been removed, as this is no longer accurate.

The following revisions were implemented in the **2017-2018 Computer Science for Innovators and Makers (CSIM)** curricula and released Sept 2018.

Course Resources		
Title of page/asset	**Description of update**	**Description of update**
Teacher Guide – General Prep and Considerations – Course Resources – Input Output and Input Output Poster	Update	All references to speaker changed to buzzer

Lesson 1 Blink!		
Title of page/asset	**Description of update**	**Description of update**
Activity 1.1	Update	Step 4 updated so each component is labelled as a number, versus a letter, so that the component list matches the microcontroller images.

PLTW | Gateway

Lesson 2 The Ins and Outs		
Title of page/asset	**Description of update**	**Detailed description of update**
Activity 2.1	Update	Flex Sensor .gif file and text updated to show the bend in the proper direction
Activity 2.2	Update	All references to speaker changed to buzzer
Activity 2.2	Update	Updated materials list to accurately reflect equipment changes to the wires. Removed reference to a sensor guide.

The following revisions were implemented in the **2018-2019 Computer Science for Innovators and Makers (CSIM)** curricula and released May 2018.

Course Resources		
Title of page/asset	**Description of update**	**Description of update**
General Student Resources	Content Updated	Resources updated and streamlined for enhanced alignment through Gateway units.
General Prep and Considerations	New	Information added to Teacher Guide to support course implementation and instruction.
Hyperlinks	Update	Links updated throughout Student Edition and Teacher Guide.
All APB's – student and teacher version	New	Figure # and Captions added, references to images or media updated.
APB's without existing Conclusion Questions	New	Add conclusion questions to APBs that do not currently include any.
Lesson 1 and 2 Assessments	Update	Now referred to as Interim Assessments
Rubrics	Update	Downloadable rubrics available in Student version
Answer Keys	Update	Nest the answer keys under the associated teacher resource page to help clean up the table of contents.
Reference page	Update	Updated reference page content in Inkling - pull all references from quick guide.
Teacher Guide	Add	Add Professional Development Opportunities and Community Forums information into the General Prep and Considerations page.

PLTW | Gateway

Lesson 2 The Ins and Outs		
Title of page/asset	**Description of update**	**Detailed description of update**
Activity 2.1 Need Input	Update	Update Graphics to show gator-to-pin connection cables, female connectors for pins on flex, and pressure sensors
Lesson 2 the Ins and Outs	New	Added potentiometer to the IM kit. Added teacher support for using potentiometers to help students understand the concept of analog input devices and reading analog values on the micro:bit.

SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



PLTW Curriculum Framework – Computer Science Innovators and Makers
Lesson 1: Blink

Desired Results *(stage 1)*		
Meaning		
ESSENTIAL QUESTIONS: *Students will keep considering …* • OA – How do you express yourself and your creativity through computer science? • Q1 – How can algorithmic thinking skills be used across multiple disciplines • Q2 - How can computer programs solve problems?		
Meaning	*Acquisition*	
Domains/Understandings *Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.* *Domains are areas of expertise that an employer in a specific field may seek.* *Students will develop understandings about…* ***"I will be able to address real-world challenges because I understand…"*** *Students will understand that …*	Learning Objectives *Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)* *Objectives are functions that directly relate to the workplace or in an applied academic setting.* *Students will use knowledge and skills to…* ***"In the workplace or academic setting, I will need to know and be able to…"***	Knowledge and Skills *Knowledge and skills include the essential facts and basic concepts that a student should know and be able to recall in order to perform the competency.* *Knowledge and skill statements are foundational to the performance of a skill.* *Student will know and be able to…* ***"After I learn the information, I will be able to use my knowledge and skills to…"***
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the	**LO1.1A**: Describe and/or analyze moments within a problem solving process where persistence, iteration, and the positive aspect of failure played an important	**KS1.1A1**: Recognize that identifying complex problems, defining them clearly, and proposing solutions can be

© 2017 Project Lead The Way, Inc.

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria. VA: ASCD

positive role of failure are important mindsets and habits of action. They are developed over time in problem solving processes, inquiry, and computational thinking.	role in gaining understanding about a problem or unexpected observation.	difficult and requires persistence and iteration. (A1.3, A1.4, P1.5) **KS1.1A2**: Describe how failure can produce positive outcomes by improving understanding. (A1.3, P1.5)
	LO1.1B: Demonstrate creativity and courage to take risks in proposing designs.	**KS1.1B1**: Describe the importance of creativity and risk taking in engineering, scientific, or computational processes. (P1.5) **KS1.1B2**: Develop solutions employing non-traditional techniques; novel combinations of artifacts, tools, techniques; and exploration of personal curiosities throughout a creative process. (P1.5)
	LO1.1C: Describe how models are used to make predictions and/or learn about a phenomenon, situation, or design.	**KS1.1C1**: Identify various models that may be used, which include but are not limited to physical models (prototypes), mathematical models, simulations, schematics, code, or 2D and 3D representations. (A1.2, P1.5)
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	**LO1.3A**: Apply computational thinking to solve problems.	**KS1.3A2**: Choose appropriate computational practices when solving a problem. (A1.4) **KS31.3A3**: Demonstrate the ability to decompose a problem into smaller parts. (A1.4)
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	**LO2.1A**: Collect, process, and analyze real or simulated data.	**KS2.1A1** Understand that computers enable rapid processing of information. (A1.1)
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	**LO2.2A**: Analyze and create algorithms.	**KS2.2A1**: Create an algorithm or sequence of steps to accomplish a task. (A1.2, P1.5) **KS2.2A2**: Identify different algorithms that can be used to solve the same problem. (A1.2, A1.4, P1.5) **KS2.2A3**: Analyze, break down, and explain the logic of an algorithm.(A1.2, A1.4, P1.5)



© 2017 Project Lead The Way, Inc.

		KS2.2A4: Locate and debug errors within an algorithm. (A1.4, P1.5)
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	**LO2.4A:** Analyze the structure and functionality of a program.	**KS2.4A1**: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures.(A1.3) **KS2.4A2**: Describe the appropriate code blocks and convention used in the programming language. (A1.3, A1.4, P1.5) **KS2.4A3**: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events. (A1.3, A1.4, P1.5)
	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	**KS2.4B1**: Plan a program using appropriate strategies such as natural language or flowcharting. (A1.3, P1.5) **KS2.4B3**: Test code frequently as it is being developed. (A1.3, P1.5) **KS2.4B4**: Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line-by-line, generating output, or commenting out code. (A1.4, P1.5)
	LO2.4C: Adapt or improve existing code.	**KS2.4C1**: Find code relevant to a problem and extend or apply it to a new purpose. (A1.4, P1.5) **KS2.4C2**: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments. (A1.4, P1.5)
D2.5 Computer Systems: Effectively harnessing computing power requires an understanding of how computers work	**LO2.5A**: Describe the hardware components of an electronic device and how they interact with software and the environment.	**KS2.5A1**: Describe the modular components of computer hardware, particularly input devices, processors, memory, and output devices. (A1.1, P1.5) **KS2.5A2**: Describe the interaction between computing devices and the surrounding environment. (A1.1, P1.5)
	LO2.5B: Identify methods in which electronic devices communicate with each other.	**KS2.5B1**: Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth ®.(A1.1, A1.3) **KS2.5B2**: Use a systematic process to identify the source of a problem within individual and connected devices (e.g. restart device, check connections, and swap in working components). (A1.3, A1.4, P1.5)



© 2017 Project Lead The Way, Inc.

D2.6 Social Impacts of Computing: Computer science solutions have global impacts in economic, environmental, and societal contexts.	**LO2.6A**: Analyze the implications of computing in society.	**KS2.6A1**: Identify ways that computing has changed or is changing society. (P1.5) **KS2.6A2**: Give examples of technologies that assist or extend human capabilities. (P1.5)
D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	**LO3.1A**: Collaborate effectively on a diverse team.	**KS3.1A1**: Describe how diverse perspectives in collaboration typically produce the best results in a process. (A1.2) **KS3.1A3**: Recognize individual strengths when defining roles and responsibilities. (A1.2)
D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	**LO3.2A**: Communicate effectively for specific purposes and settings.	**KS3.2A1**: Communicate to meet the needs of the audience and be appropriate to the situation. (A1.2, A1.3, P1.5) **KS3.2A2**: Use accurate and appropriate terminology. (A1.2, A1.3, P1.5)
	LO3.2B: Document a process appropriately.	**KS3.2B1**: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations. (A1.2, A1.3, P1.5)
D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	**LO3.3A**: Demonstrate the ability to manage multiple resources throughout a project.	**KS3.3A3**: Manage digital files appropriately. (A1.3, P1.5)
D3.5 Career Awareness: It is important to prepare a flexible education plan that matches your interests, knowing that you can change or modify that plan as you discover more about career opportunities.	**LO3.5B**: Describe the role, connections between disciplines, and impact of engineering, biomedical science, and/or computer science on society.	**KS3.5B2**: Describe the impact of engineering, biomedical science, or computer science on invention and innovation. (A1.1)



© 2017 Project Lead The Way, Inc.

PLTW Curriculum Framework – Computer Science Innovators and Makers
Lesson 2: The Ins and Outs

Desired Results *(stage 1)*		
Meaning		
ESSENTIAL QUESTIONS: *Students will keep considering …* • OA – How do you express yourself and your creativity through computer science? • Q1 – What do programming best practices look like? • Q2 – Why are interactions between machines, humans, and the environment beneficial, problematic or both? • Q3 – How is a design process used to develop physical computing systems?		
Meaning	*Acquisition*	
Domains/Understandings *Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.* *Domains are areas of expertise that an employer in a specific field may seek.* *Students will develop understandings about…* ***"I will be able to address real-world challenges because I understand…"*** *Students will understand that …*	Learning Objectives *Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)* *Objectives are functions that directly relate to the workplace or in an applied academic setting.* *Students will use knowledge and skills to…* ***"In the workplace or academic setting, I will need to know and be able to…"***	Knowledge and Skills *Knowledge and skills include the essential facts and basic concepts that a student should know and be able to recall in order to perform the competency.* *Knowledge and skill statements are foundational to the performance of a skill.* *Student will know and be able to…* ***"After I learn the information, I will be able to use my knowledge and skills to…"***
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the positive role of failure are important mindsets and habits of action. They are developed	**LO1.1A**: Describe and/or analyze moments within a problem solving process where persistence, iteration, and the positive aspect of failure played an important role in gaining	**KS1.1A1**: Recognize that identifying complex problems, defining them clearly, and proposing solutions can be difficult and



© 2017 Project Lead The Way, Inc.

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria. VA: ASCD

over time in problem solving processes, inquiry, and computational thinking.	understanding about a problem or unexpected observation.	requires persistence and iteration. (A2.1, A2.2, A2.3, P2.4) **KS1.1A2**: Describe how failure can produce positive outcomes by improving understanding. (A2.1, A2.2, A2.3, P2.4)
	LO1.1B: Demonstrate creativity and courage to take risks in proposing designs.	**KS1.1B1**: Describe the importance of creativity and risk taking in engineering, scientific, or computational processes. (A2.2, P2.4) **KS1.1B2**: Develop solutions employing non-traditional techniques; novel combinations of artifacts, tools, techniques; and exploration of personal curiosities throughout a creative process. (A2.2, P2.4)
	LO1.1C: Describe how models are used to make predictions and/or learn about a phenomenon, situation, or design.	**KS1.1C1**: Identify various models that may be used, which include but are not limited to physical models (prototypes), mathematical models, simulations, schematics, code, or 2D and 3D representations. (A2.1, A2.3, P2.4)
D1.2 Problem Solving Process and/or Design Process: Many disciplines, including engineering, computer science, and biomedical science, use an iterative, systematic process to solve problems.	**LO1.2A**: Apply an iterative process to solve a problem or create an opportunity that can be justified.	**KS1.2A1**: Identify a problem and justify the pursuit of a solution to the problem. **KS1.2A2:** Explain the goal of any design process. (P2.4) **KS1.2A3**: Compose appropriate design requirements that include specifications, constraints, desired features, and testable parameters. (P2.4) **KS1.2A4**: Generate ideas or build upon existing ideas to innovate. (P2.4) **KS2.1A5**: Create and evaluate multiple solution options based on criteria, and select the best option. (P2.4) **KS1.2A6**: Iteratively test throughout the design process to ensure that a solution meets the requirements. (P2.4)



© 2017 Project Lead The Way, Inc.

	LO1.2B: Apply user-centered design principles when creating a solution.	**KS1.2B1**: Investigate the types of interactions between users and a proposed solution. (A2.3, P2.4) **KS1.2B2**: Involve prospective users early and often during the design process. (P2.4) **KS1.2B3**: Recognize that understanding human interaction with a solution can improve the design and extend the abilities of humans. (A2.1, P2.4)
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	**LO1.3A** Apply computational thinking to solve problems.	**KS1.3A1**: Recognize that computational thinking can be applied in multiple disciplines. (A2.2, P2.4) **KS1.3A2**: Choose appropriate computational practices when solving a problem. (A2.2, A2.3, P2.4) **KS1.3A3**: Demonstrate the ability to decompose a problem into smaller parts. (A2.1, A2.2, A2.3, P2.4)
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	**LO2.1A**: Collect, process, and analyze real or simulated data.	**KS2.1A1** Understand that computers enable rapid processing of information. (A2.3, P2.4) **KS2.1A2**: Collect, process and interpret data to gain insight on a problem and draw conclusions. (A2.1, P2.4) **KS2.1A3**: Compare methods of data representation. (A2.1, A2.3)
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	**LO2.2A**: Analyze and create algorithms.	**KS2.2A1**: Create an algorithm or sequence of steps to accomplish a task. (A2.1, A2.3, P2.4) **KS2.2A2**: Identify different algorithms that can be used to solve the same problem. (A2.2, P2.4) **KS2.2A3**: Analyze, break down, and explain the logic of an algorithm. (A2.3, P2.4) **KS2.2A4**: Locate and debug errors within an algorithm. (A2.2, A2.3, P2.4)



© 2017 Project Lead The Way, Inc.



		KS2.2A5: Create simple algorithms that involve variables, conditionals, operators, or logic. (A2.1, A2.3, P2.4) **KS2.2A6**: Create algorithms to perform repetitive or reusable tasks. (A2.1, A2.2, A2.3, P2.4)
D2.3 Abstraction: An abstraction generalizes a concept, allowing people to ignore some of the details. Abstractions make it easier to handle complexity by focusing on the relevant pieces of the big picture and hiding the details.	**LO2.3A**: Develop and apply abstractions.	**KS2.3A3**: Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. (P2.4)
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	**LO2.4A**: Analyze the structure and functionality of a program.	**KS2.4A1**: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures. (A2.3, P2.4) **KS2.4A2**: Describe the appropriate code blocks and convention used in the programming language. (A2.1, A2.2, A2.3, P2.4) **KS2.4A3**: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events. (A2.3, P2.4) **KS2.4A4**: Trace a program and deduce the values that variables will have after the code is executed. (A2.3, P2.4)
	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	**KS2.4B1**: Plan a program using appropriate strategies such as natural language or flowcharting. (A2.1, A2.3, P2.4) **KS2.4B2**: Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable). (A2.2, A2.3, P2.4) **KS2.4B3**: Test code frequently as it is being developed. (A2.1, A2.2, A2.3, P2.4) **KS2.4B4**: Debug programs using a variety of strategies to isolate and identify problems,

© 2017 Project Lead The Way, Inc.

		including analyzing error messages, analyzing variable values line-by-line, generating output, or commenting out code. (A2.1, A2.2, A2.3, P2.4)
	LO2.4C: Adapt or improve existing code.	**KS2.4C1**: Find code relevant to a problem and extend or apply it to a new purpose. (A2.1, A2.2, A2.3) **KS2.4C2**: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments. (A2.2, A2.3, P2.4)
D2.5 Computer Systems: Effectively harnessing computing power requires an understanding of how computers work	**LO2.5A**: Describe the hardware components of an electronic device and how they interact with software and the environment.	**KS2.5A1**: Describe the modular components of computer hardware, particularly input devices, processors, memory, and output devices. (A2.1, A2.2, A2.3, P2.4) **KS2.5A2**: Describe the interaction between computing devices and the surrounding environment. (A2.1, A2.2, A2.3, P2.4) **KS2.5A3**: Select and justify the hardware chosen to accomplish a task. (A2.1, A2.2, A2.3, P2.4) **KS2.5A4**: Describe advantages and disadvantages of using different power sources for the mobility of a system. (A2.1, A2.3, P2.4)
	LO2.5B: Identify methods in which electronic devices communicate with each other.	**KS2.5B1**: Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth ®.(A2.3, P2.4) **KS2.5B2**: Use a systematic process to identify the source of a problem within individual and connected devices (e.g. restart device, check connections, and swap in working components). (A2.1, A2.3, P2.4)



© 2017 Project Lead The Way, Inc.

D2.6 Social Impacts of Computing: Computer science solutions have global impacts in economic, environmental, and societal contexts	**LO2.6A**: Analyze the implications of computing in society.	**KS2.6A1**: Identify ways that computing has changed or is changing society. (A2.3, P2.4) **KS2.6A2**: Give examples of technologies that assist or extend human capabilities. (A2.1, A2.3, P2.4) **KS2.6A3**: Describe possible positive and negative social impacts of computing. (A2.3, P2.4)
D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	**LO3.1A**: Collaborate effectively on a diverse team.	**KS3.1A1**: Describe how diverse perspectives in collaboration typically produce the best results in a process. (P2.4) **KS3.1A3**: Recognize individual strengths when defining roles and responsibilities. (A2.1, A2.3, P2.4)
D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	**LO3.2A**: Communicate effectively for specific purposes and settings.	**KS3.2A1**: Communicate to meet the needs of the audience and be appropriate to the situation. (A2.2, P2.4) **KS3.2A2:** Use accurate and appropriate terminology. (A2.1, A2.2, P2.4)
	LO3.2B: Document a process appropriately.	**KS3.2B1**: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations. (A2.1, P2.4)
D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	**LO3.3A**: Demonstrate the ability to manage multiple resources throughout a project.	**KS3.3A1**: Identify resources managed in a project to include time, capital, energy, information, materials, people, tools, or machines. (P2.4) **KS3.3A3**: Manage digital files appropriately. (A2.1, A2.3, P2.4)
	LO3.3B: Justify decisions and provide rationales when making tradeoffs between resources.	**KS3.3B1**: Recognize that processes involve tradeoffs when weighing the need against effort and resources available. (P2.4) **KS3.3B2**: Demonstrate sound judgement when making decisions regarding resources and their potential tradeoffs. (P2.4)



© 2017 Project Lead The Way, Inc.

D3.4 Professionalism and Ethics: Successful computer scientists typically exhibit specific personal and professional characteristics that lend themselves to the creative, collaborative, and solution driven nature of the profession.	**LO3.4A:** Abide by professional, legal and ethical standards when using digital resources.	**KS3.4A1**: Describe property rights of digital media and software to facilitate sharing and avoid infringement. (P2.4) **KS3.4A2**: Describe ethical and legal implications for accessing and copying digital information. (A2.3, P2.4) **KS3.4A3**: Describe the importance of protecting personal information. (P2.4)
D3.5 Career Awareness: It is important to prepare a flexible education plan that matches your interests, knowing that you can change or modify that plan as you discover more about career opportunities.	**LO3.5A:** Describe the role, connections between disciplines, and impact of engineering, biomedical science, and/or computer science on society.	**KS3.5A2**: Describe the impact of engineering, biomedical science, or computer science on invention and innovation. (A2.3, P2.4)



© 2017 Project Lead The Way, Inc.

SPRINGFIELD PUBLIC SCHOOLS
ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS



PLTW Curriculum Framework – Computer Science Innovators and Makers
Lesson 3: Program the Physical World

Desired Results *(stage 1)*		
Meaning		
ESSENTIAL QUESTIONS: *Students will keep considering …* • OA – How do you express yourself and your creativity through computer science? • Q1 - Do all professions need computer science? • Q2 - Does society need computer science and physical computing projects? • Q3 - How are ethics, safety, and computing projects related?		
Meaning	*Acquisition*	
Domains/Understandings *Domains are key understandings and long-term takeaways that go beyond factual knowledge into broader and more conceptual comprehensions.* *Domains are areas of expertise that an employer in a specific field may seek.* *Students will develop understandings about…* ***"I will be able to address real-world challenges because I understand..."*** *Students will understand that …*	Learning Objectives *Objectives articulate what skills students need to be able to do. (The learning objectives will become targets of assessment.)* *Objectives are functions that directly relate to the workplace or in an applied academic setting.* *Students will use knowledge and skills to…* ***"In the workplace or academic setting, I will need to know and be able to..."***	Knowledge and Skills *Knowledge and skills include the essential facts and basic concepts that a student should know and be able to recall in order to perform the competency.* *Knowledge and skill statements are foundational to the performance of a skill.* *Student will know and be able to…* ***"After I learn the information, I will be able to use my knowledge and skills to..."***
D1.1 Mindset: Ethics, analytical thinking, creativity, persistence, iteration, and the positive role of failure are	**LO1.1A**: Describe and/or analyze moments within a problem solving process where persistence, iteration, and the positive aspect	**KS1.1A1**: Recognize that identifying complex problems, defining them clearly, and

© 2017 Project Lead The Way, Inc.

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria. VA: ASCD

important mindsets and habits of action. They are developed over time in problem solving processes, inquiry, and computational thinking.	of failure played an important role in gaining understanding about a problem or unexpected observation.	proposing solutions can be difficult and requires persistence and iteration. (B3.2) **KS1.1A2**: Describe how failure can produce positive outcomes by improving understanding. (B3.2)
	LO1.1B: Demonstrate creativity and courage to take risks in proposing designs.	**KS1.1B1**: Describe the importance of creativity and risk taking in engineering, scientific, or computational processes. (B3.2) **KS1.1B2**: Develop solutions employing non-traditional techniques; novel combinations of artifacts, tools, techniques; and exploration of personal curiosities throughout a creative process. (B3.2)
	LO1.1C: Describe how models are used to make predictions and/or learn about a phenomenon, situation, or design.	**KS1.1C1**: Identify various models that may be used, which include but are not limited to physical models (prototypes), mathematical models, simulations, schematics, code, or 2D and 3D representations. (B3.2)
D1.2 Problem Solving Process and/or Design Process: Many disciplines, including engineering, computer science, and biomedical science, use an iterative, systematic process to solve problems.	**LO1.2A**: Apply an iterative process to solve a problem or create an opportunity that can be justified.	**KS1.2A1**: Identify a problem and justify the pursuit of a solution to the problem. (B3.1, B3.2) **KS1.2A2**: Explain the goal of any design process. (B3.2) **KS1.2A3**: Compose appropriate design requirements that include specifications, constraints, desired features, and testable parameters. (B3.2) **KS1.2A4**: Generate ideas or build upon existing ideas to innovate. (A3.1, B3.2) **KS1.2A5**: Create and evaluate multiple solution options based on criteria, and select the best option. (B3.2) **KS1.2A6**: Iteratively test throughout the design process to ensure that a solution meets the requirements. (B3.2)



© 2017 Project Lead The Way, Inc.

	LO1.2B: Apply user-centered design principles when creating a solution.	**KS1.2B1**: Investigate the types of interactions between users and a proposed solution. (B3.2) **KS1.2B2**: Involve prospective users early and often during the design process. (B3.2) **KS1.2B3**: Recognize that understanding human interaction with a solution can improve the design and extend the abilities of humans. (B3.2)
D1.3 Computational Thinking: Computational thinking is the thought processes involved in formulating problems and solutions that can be effectively carried out by a computer or electronic device. Common concepts of computational thinking include: the use of algorithms, abstraction, problem decomposition, and data analysis and processing.	**LO1.3A**: Apply computational thinking to solve problems.	**KS1.3A1**: Recognize that computational thinking can be applied in multiple disciplines. (B3.2) **KS1.3A2**: Choose appropriate computational practices when solving a problem. (A3.1, B3.2) **KS1.3A3**: Demonstrate the ability to decompose a problem into smaller parts. (A3.1, B3.2)
D2.1 Data: With the aid of computational power, a tremendous quantity of data can quickly and efficiently processed and analyzed to help solve a problem.	**LO2.1A**: Collect, process, and analyze real or simulated data.	**KS2.1A1** Understand that computers enable rapid processing of information. (A3.1, B3.2) **KS2.1A2**: Collect, process and interpret data to gain insight on a problem and draw conclusions. (B3.2) **KS2.1A3**: Compare methods of data representation. (B3.2)
D2.2 Algorithms: A wide range of professionals use algorithms, a sequence of steps used to solve a problem. This can be accomplished with or without coding.	**LO2.2A**: Analyze and create algorithms.	**KS2.2A1**: Create an algorithm or sequence of steps to accomplish a task. (A3.1, B3.2) **KS2.2A2**: Identify different algorithms that can be used to solve the same problem. (A3.1, B3.2) **KS2.2A3**: Analyze, break down, and explain the logic of an algorithm. (A3.1, B3.2) **KS2.2A4**: Locate and debug errors within an algorithm. (B3.2)



© 2017 Project Lead The Way, Inc.

		KS2.2A5: Create simple algorithms that involve variables, conditionals, operators, or logic. (A3.1, B3.2) **KS2.2A6**: Create algorithms to perform repetitive or reusable tasks. (A3.1, B3.2)
D2.3 Abstraction: An abstraction generalizes a concept, allowing people to ignore some of the details. Abstractions make it easier to handle complexity by focusing on the relevant pieces of the big picture and hiding the details.	**LO2.3A**: Develop and apply abstractions.	**KS2.3A1**: Explain the purpose of an abstraction. (A3.1, B3.2) **KS2.3A2**: Identify what has been made more general by an abstraction and identify what details have been hidden or removed. (A3.1, B3.2) **KS2.3A3**: Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. (A3.1, B3.2)
D2.4 Programming: Creating a set of statements processed by an electronic device to perform a task.	**LO2.4A**: Analyze the structure and functionality of a program.	**KS2.4A1**: Identify and describe the high-level structures of a program, such as user interface components, data components, event handlers, and procedures. (A3.1, B3.2) **KS2.4A2**: Describe the appropriate code blocks and convention used in the programming language. (A3.1, B3.2) **KS2.4A3**: Identify possible events that can occur during runtime and select the appropriate event-handler blocks to respond to these events. (A3.1, B3.2) **KS2.4A4**: Trace a program and deduce the values that variables will have after the code is executed. (B3.2)
	LO2.4B: Create programs by developing and testing code in a modular, incremental approach.	**KS2.4B1**: Plan a program using appropriate strategies such as natural language or flowcharting. (B3.2) **KS2.4B2**: Design and develop a program by breaking a large plan into smaller modules using procedures and event handlers (if applicable). (B3.2)



© 2017 Project Lead The Way, Inc.

		KS2.4B3: Test code frequently as it is being developed. (A3.1, B3.2) **KS2.4B4**: Debug programs using a variety of strategies to isolate and identify problems, including analyzing error messages, analyzing variable values line-by-line, generating output, or commenting out code. (A3.1, B3.2)
	LO2.4C: Adapt or improve existing code.	**KS2.4C1**: Find code relevant to a problem and extend or apply it to a new purpose. (A3.1, B3.2) **KS2.4C2**: Improve readability of code by creating or improving documentation, using descriptive variables and procedure names, and using comments. (A3.1, B3.2)
D2.5 Computer Systems: Effectively harnessing computing power requires an understanding of how computers work.	**LO2.5A**: Describe the hardware components of an electronic device and how they interact with software and the environment.	**KS2.5A1**: Describe the modular components of computer hardware, particularly input devices, processors, memory, and output devices. (B3.2) **KS2.5A2**: Describe the interaction between computing devices and the surrounding environment. (B3.2) **KS2.5A3**: Select and justify the hardware chosen to accomplish a task. (B3.2) **KS2.5A4**: Describe advantages and disadvantages of using different power sources for the mobility of a system. (B3.2)
	LO2.5B: Identify methods in which electronic devices communicate with each other.	**KS2.5B1**: Recognize various ways of transferring data from one electronic device to another, such as USB connection, Wi-Fi, or Bluetooth ®.(B3.2) **KS2.5B2**: Use a systematic process to identify the source of a problem within individual and connected devices (e.g. restart device, check connections, swap in working components). (B3.2)



© 2017 Project Lead The Way, Inc.

D2.6 Social Impacts of Computing: Computer science solutions have global impacts in economic, environmental, and societal contexts.	**LO2.6A**: Analyze the implications of computing in society.	**KS2.6A1**: Identify ways that computing has changed or is changing society. (A3.1, B3.2) **KS2.6A2**: Give examples of technologies that assist or extend human capabilities. (B3.2) **KS2.6A3**: Describe possible positive and negative social impacts of computing. (B3.2)
D3.1 Collaboration: Effective collaboration requires an ability to function within teams and is often necessary for successful problem solving, experimentation or design work.	**LO3.1A**: Collaborate effectively on a diverse team.	**KS3.1A1:** Describe how diverse perspectives in collaboration typically produce the best results in a process. (B3.2) **KS3.1A2**: Apply team norms to encourage productivity and define how a team will function and measure its success. (B3.2) **KS3.1A3**: Recognize individual strengths when defining roles and responsibilities. (B3.2)
D3.2 Communication: Effectively communicating orally and in writing for a given audience is an essential skill for success in all fields.	**LO3.2A**: Communicate effectively for specific purposes and settings.	**KS3.2A1**: Communicate to meet the needs of the audience and be appropriate to the situation. (A3.1, B3.2) **KS3.2A2**: Use accurate and appropriate terminology. (A3.1, B3.2)
	LO3.2B: Document a process appropriately.	**KS3.2B1**: Present data and information through a variety of accepted means such as: graphs, charts, images, video, schematics, code, 3D models, or simulations. (B3.2)
D3.3 Project Management: The discipline of carefully projecting or planning, organizing, motivating and controlling resources to achieve specific goals and meet specific success criteria.	**LO3.3A**: Demonstrate the ability to manage multiple resources throughout a project.	**KS3.3A1**: Identify resources managed in a project to include time, capital, energy, information, materials, people, tools, or machines. (B3.2) **KS3.3A2**: Devise, prioritize, and execute a plan to solve a problem, including short-term and long-term objectives when appropriate. (B3.2) **KS3.3A3**: Manage digital files appropriately. (A3.1, B3.2)



© 2017 Project Lead The Way, Inc.

	LO3.3B: Justify decisions and provide rationales when making tradeoffs between resources.	**KS3.3B1**: Recognize that processes involve tradeoffs when weighing the need against effort and resources available. (B3.2) **KS3.3B2**: Demonstrate sound judgment when making decisions regarding resources and their potential tradeoffs. (B3.2)
D3.4 Professionalism and Ethics: Successful computer scientists typically exhibit specific personal and professional characteristics that lend themselves to the creative, collaborative, and solution driven nature of the profession.	**LO3.4A**: Abide by professional, legal and ethical standards when using digital resources.	**KS3.4A1**: Describe property rights of digital media and software to facilitate sharing and avoid infringement. (B3.2) **KS3.4A2**: Describe ethical and legal implications for accessing and copying digital information. (B3.2) **KS3.4A3**: Describe the importance of protecting personal information. (B3.2)
	LO3.4B: Identify ethical considerations that must be considered when creating solutions or opportunities.	**KS3.4B1**: Recognize that ethical considerations include but are not limited to safety, impact on future generations, or recognizing the work of others (intellectual property). (B3.2) **KS3.4B2**: Explain how universal design considers the broadest possible spectrum of human ability in the design of products, environments and information systems. (B3.2)
	LO3.4C: Consider accessibility and equity when designing products, creating solutions, and collaborating with others.	**KS3.4C1**: Describe and give examples of universal design strategies that increase accessibility to appropriate resources. (B3.2)
D3.5 Career Awareness: It is important to prepare a flexible education plan that matches your interests, knowing that you can change or modify that plan as you discover more about career opportunities.	**LO3.5A**: Identify the variety of careers related to engineering, biomedical sciences, and/or computer science.	**KS3.5A1**: Recognize the education and credentialing requirements for careers in engineering, biomedical science or computer science. (B3.2) **KS3.5A2**: Explore a variety of careers related to engineering, biomedical sciences, or computer science. (B3.2)
	LO3.5B: Describe the role, connections between disciplines, and impact of	**KS3.5B1**: Demonstrate personal responsibility and initiative for independent learning,



© 2017 Project Lead The Way, Inc.

	engineering, biomedical science, and/or computer science on society.	keeping in mind that technology and processes will always evolve. (B3.2) **KS3.5B2**: Describe the impact of engineering, biomedical science, or computer science on invention and innovation. (B3.2)

© 2017 Project Lead The Way, Inc.

Format guided by: Wiggins, G. & McTighe, J. (2011). The understanding by design guide to creating high quality units. Alexandria. VA: ASCD



IEP AND 504 ACCOMMODATIONS AND MODIFICATIONS FOR EARLY CHILDHOOD AND ELEMENTARY SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
Review of directions
Use of mnemonics
Have student restate information
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Use of a study carrel
Assistance in maintaining uncluttered space
Weekly home-school communication tools (notebook, daily log, phone calls or email messages)
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Use of manipulatives
No penalty for spelling errors or sloppy handwriting
Follow a routine/schedule
Alternate quiet and active time
Teach time management skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Untimed tests
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book tests
Read test and directions to student
Provide study outline prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator and/or number line for math tests
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS MODIFICATIONS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Have parent sign behavior chart
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

IEP AND 504 ACCOMODATIONS AND MODIFICATIONS FOR MIDDLE AND HIGH SCHOOL STUDENTS:

Large print textbooks
Textbooks for at-home use
Additional time for assignments
A locker with adapted lock
Review of directions
Review sessions
Use of mnemonics
Have student restate information
Provision of notes or outlines
Concrete examples
Adaptive writing utensils
Support auditory presentations with visuals
Weekly or bi-weekly home-school communication
Space for movement or breaks
Study sheets and teacher outlines
Extra visual and verbal cues and prompts
Lab and math sheets with highlighted instructions
Graph paper to assist in organizing or lining up math problems
Use of tape recorder for lectures
Use of Interactive notebook
Use of a scribe
Use of computers and calculators
Books on tape
Graphic organizers
Quiet corner or room to calm down and relax when anxious
Preferential seating
Alteration of the classroom arrangement
Reduction of distractions
Answers to be dictated
Hands-on activities
Ability to type all assignments
Follow a routine/schedule
Teach time management and organizational skills
Rest breaks
Verbal and visual cues regarding directions and staying on task
Agenda book and checklists
Daily check-in with case manager or special education teacher
Adjusted assignment timelines
Visual daily schedule

Varied reinforcement procedures
Immediate feedback
Work-in-progress check
Personalized examples

TESTING AND ASSESSMENT ACCOMMODATIONS AND MODIFICATIONS:

Answers to be dictated
Frequent rest breaks
Additional time
Oral testing
Choice of test format (multiple-choice, essay, true-false)
Alternate ways to evaluate (projects or oral presentations instead of written tests)
Accept short answers
Open-book or open-note tests
Read test and directions to student
Provide study guides prior to tests
Highlight key directions
Test in alternative site
Use of calculator
Extra credit option
Pace long-term projects
Preview test procedures
Simplified test wording; rephrased test questions and/or directions

GENERAL ACCOMMODATIONS AND MODIFICATIONS:

Allow outlining, instead of writing for an essay or major project
Use of alternative books or materials on the topic being studied
Computerized spell-check support
Word bank of choices for answers to test questions
Provision of calculator for math tests
Film or video supplements in place of reading text
Reworded questions in simpler language
Projects instead of written reports
Highlighting important words or phrases in reading assignments
Modified workload or length of assignments/tests
Modified time demands

Pass/no pass option
Modified grades based on IEP

BEHAVIOR ACCOMMODATIONS FOR STUDENTS:

Breaks between tasks
Cue expected behavior
Daily feedback to student
Have contingency plans
Use de-escalating strategies
Use positive reinforcement
Use proximity/touch control
Use peer supports and mentoring
Model expected behavior by adults
Have parent sign homework
Set and post class rules
Chart progress and maintain data



SPRINGFIELD PUBLIC SCHOOLS

ENGAGING STUDENTS • FOSTERING ACHIEVEMENT • CULTIVATING 21ST CENTURY GLOBAL SKILLS

Accommodations and Modifications for ESL Students

Give tests orally
Give instructions/directions in writing and orally
Assign a peer tutor, same language or English speaking
Allow use of bilingual/translation dictionary
Provide summaries of text or study guides of particular chapters
Allow errors in speaking
Allow errors in writing
Accept writing in first language
Highlight key vocabulary
Reduce amount of work required
Assess comprehension through demonstration or other alternative means (gestures, drawings)
Rephrase questions, directions, and explanations
Reduce multiple choices on assessments
Provide study guides and/ or outlines
Shorten reading assignments
Allow extended time for test/project completion
Use of audio books
Use of graphic organizers
Use reduced text, so that print is not dense
Use visuals/pictures throughout lesson/unit/assignemt
Adapt homework to reflect language proficiency/home support
Adapt homework to reflect home support
Provide hands-on activities and explanations
Allow extended time to answer questions, and permit drawing, as an explanation
Use of translation dictionaries to locate words in the native language
Use anecdotal records and portfolio of work as forms of assessment
Use story retellings to assess comprehension