Unit Plan

Inspire the Inventors of Tomorrow!

Table of Contents

Introducing the Piper Unit Plan v4.0 - updated June 2018 version

Welcome to the world of Piper! We created the Piper computer kit because we believe there is a great opportunity for students, teachers and parents alike to grow creative confidence around STEM, specifically engineering and computing. We started with the most prolific game of our time, Minecraft$_{tm}$, and put complex engineering theory into our custom worlds so that students could stay engaged while learning electronics. With the addition of PiperCode projects, we've built upon the excitement of constructing your own computer and learning with the help of favorite game, and evolved to students now programming the games themselves. Students must demonstrate their learning from the game by building electronics that will allow them to "power-up" to the next level, then further show deeper learning by programming their own games and constructing controllers.

"My favorite part is watching the engagement from our students."
- Ken Willers, Principal, School of the Madeleine, Berkeley, CA

Now updated in June 2018, the curriculum presented here is designed to be a flexible tool for teachers to make Piper teachable to any age group. Teachers will be able to use Piper starting as early as 2nd or 3rd grade (ages 7-8) all the way up to high schoolers with little to no prior hardware and software experience. For the public education system, we align educational standards, including direct connections to computational thinking, to meet your instructional program and share some brief ideas on curriculum integration that may be helpful as you implement our lesson plans.

Elementary and Middle School Models for Curriculum Integration

Implementing computer science instruction at the K–8 level is typically more flexible than in high school. Potential models include instructional units dedicated to computer science within general technology and media arts classes, dedicated weekly computing classes offered as electives and integration of computer science instruction into other content areas. These courses could be taught by a variety of teachers, such as elementary classroom teachers, subject area teachers in school (e.g. mathematics, science, technology, music, art, and library/media arts), or dedicated computer science teachers. Regardless of the instructional delivery model, attention should be given to aligning those instructional experiences with the framework's progressions (K-12 CS Framework, page 152; Margolis, Goode, & Chapman, 2015).

Please use the resources in this guide to support your classroom. We welcome collaboration and hope that you share your experiences with us and other educators to improve the experience for everyone by joining the Piper in the Classroom community on Google+.

We're also looking for educators who've successfully integrated Piper in their classrooms to share adaptations or new ideas. If you're interested in learning more, email us at educators@playpiper.com.

Thank you for choosing Piper!

The Piper team

Join the Piper EDU Community!

We can't wait to hear your stories about your Piper classroom experience. Was the experience fun and engaging for your students? What did you notice about your students using Piper that surprised you? Did you end up making your own lesson plans and resources to better fit your teaching methods? We want to hear it all!

Follow Piper on Social Media

Piper utilizes social media to communicate with users about their experiences and to create discussion surrounding innovation and computer science.

There are a lot of Piper enthusiasts just like you that are dying to hear about your experiences using Piper in the classroom. Share your story!

Piper in the Classroom

@StartWithPiper

@startwithpiper

@startwithpiper

Contact Information

Something going wrong with one of your Pipers? Have any questions you need answered about the Piper Computer Kit? Go to: http://support.playpiper.com ,email hi@withpiper.com or call 415.949.2083.

We appreciate and value your feedback! To share information about your experience with Piper in the classroom and discuss any needs email Melissa: melissa@withpiper.com.

Piper Unit Plan Goals

Piper develops creative confidence with electronics. Students learn engineering by starting with a fun and low-barrier experience. Then they can build off of this foundation towards a higher-level creative experience. Specifically, students will know and be able to do the following:

- Build electronics with a Raspberry Pi, a breadboard, and jumper wires
- Complete circuits and conductivity
- Construct inputs and outputs (buttons, switches, LED lights and buzzers)
- Understand binary states and parallel circuits

For educators, PiperCode is an opportunity to spread learning with the Piper kit with deeper engineering design and computational thinking challenges. Learners quickly progress from simple to complex computer science concepts and practices. Specifically, students will:

- Review key *electronics and programming understandings*, from the Piper Computer Kit
- Understand new *computational concepts*
- Practice *critical thinking*
- Explore *computational thinking practices* by completing prompts, creating code from scratch, or debugging and remixing base projects

Piper Scope and Sequence

Piper activities are sequenced through steps in physically building the computer, working through worlds (projects) in Story Mode, then completing projects in PiperCode. We like to describe the learning flow in 5 phases:

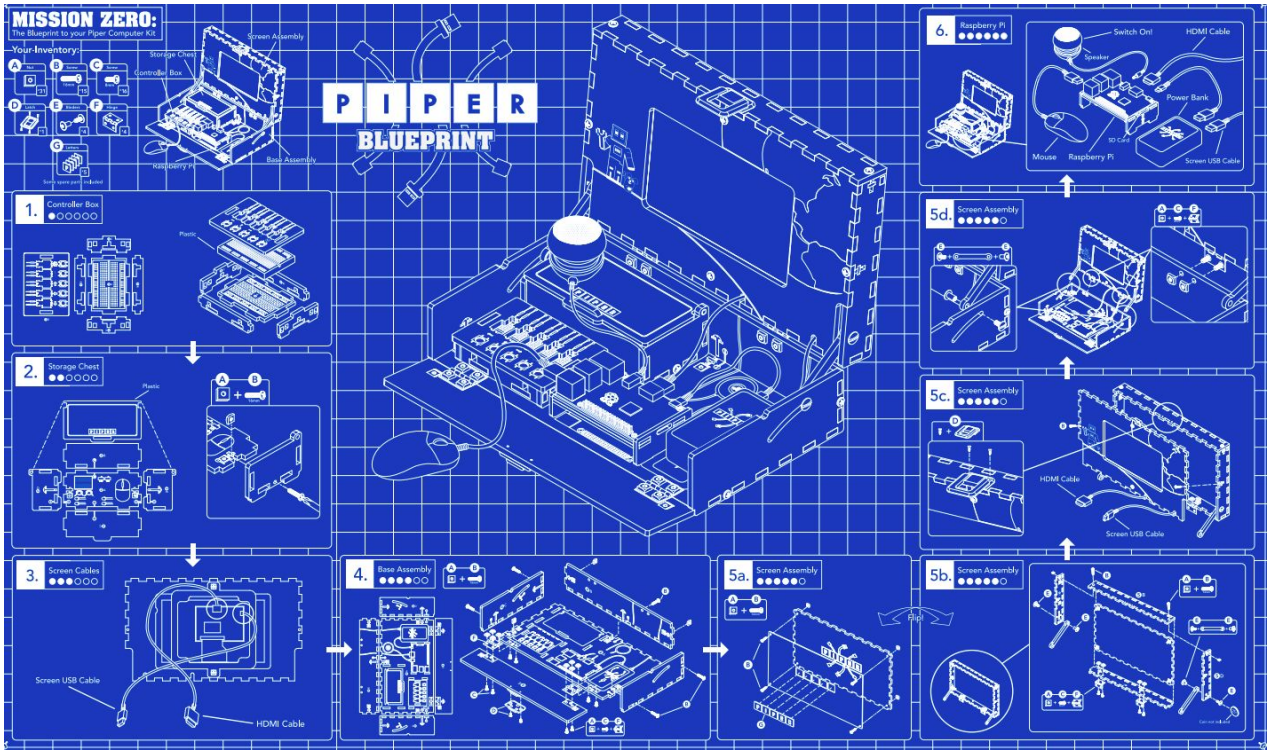
	Phase	*Tools*	*Students will...*
1	Build A Computer	Piper Computer Kit, Blueprint	Build the case and connect components of a working computer.
2	Build Electronics	Story Mode > Play	Build and understand input and output electronic components as part of computer engineering, while completing levels in a game.
3	Code	PiperCode > Make	Program basic actions including loops, sequences, and events, using visual programming blocks.
4	Extend	Story Mode MiniGames Creative Mode PiperCode > Build Games PiperCode My Projects	Design and program their own Creative Mode and PiperCode projects, including games, PiperCode visual programs, and Python code.
5	Deconstruct	Kit	Take apart the Piper Computer Kit and reflect on physical hardware and the learning process.

Phase 1: Build a Computer

In this first phase, students will build a fully functioning computer, including constructing the case and connecting components including a screen, Raspberry Pi, battery and more! The kit is complete with a large engineering blueprint that serves as step-by-step visual instructions to build the computer. Since a blueprint doesn't have much text, a first learning objective is for learners to understand engineering diagrams as instructions.

The blueprint is made of high quality paper. It is designed for kids to sit on top of it and work. They can also put it up on a wall or lay it out on a work bench area. Some facilitators may choose to show a hi-res version of the blueprint on a projector or monitor. Check out the full PDF of the blueprint.

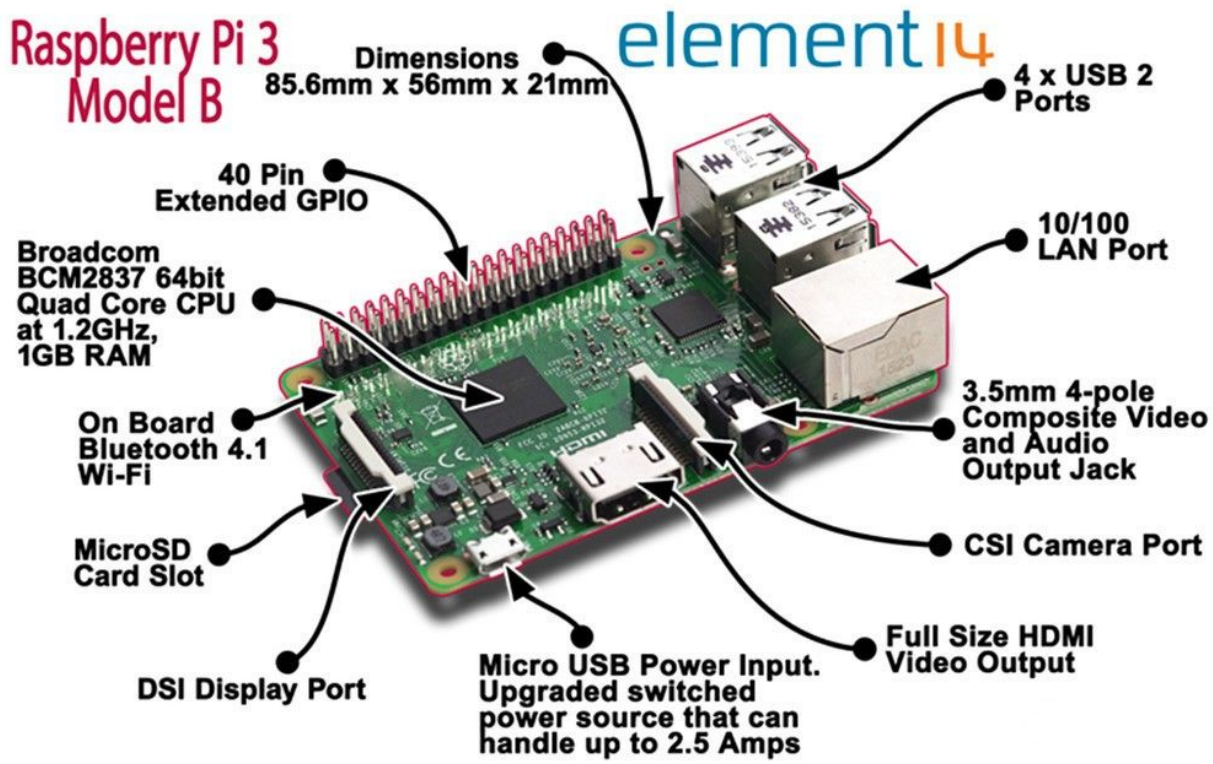
Want to see a video of the Piper Computer Kit being built? See video of Piper Computer Kit assembly.



The Raspberry Pi is the third most popular computer of all time! It is a capable little device that enables people of all ages to explore computing and learn how to program. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infrared cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work. More info. at: https://www.raspberrypi.org/

Image below highlights all the pieces that make up the Raspberry Pi. (image courtesy of Element14)



Want to build more with your learners with the Raspberry Pi in your Piper kit?
Check out the resources on https://www.raspberrypi.org/education/

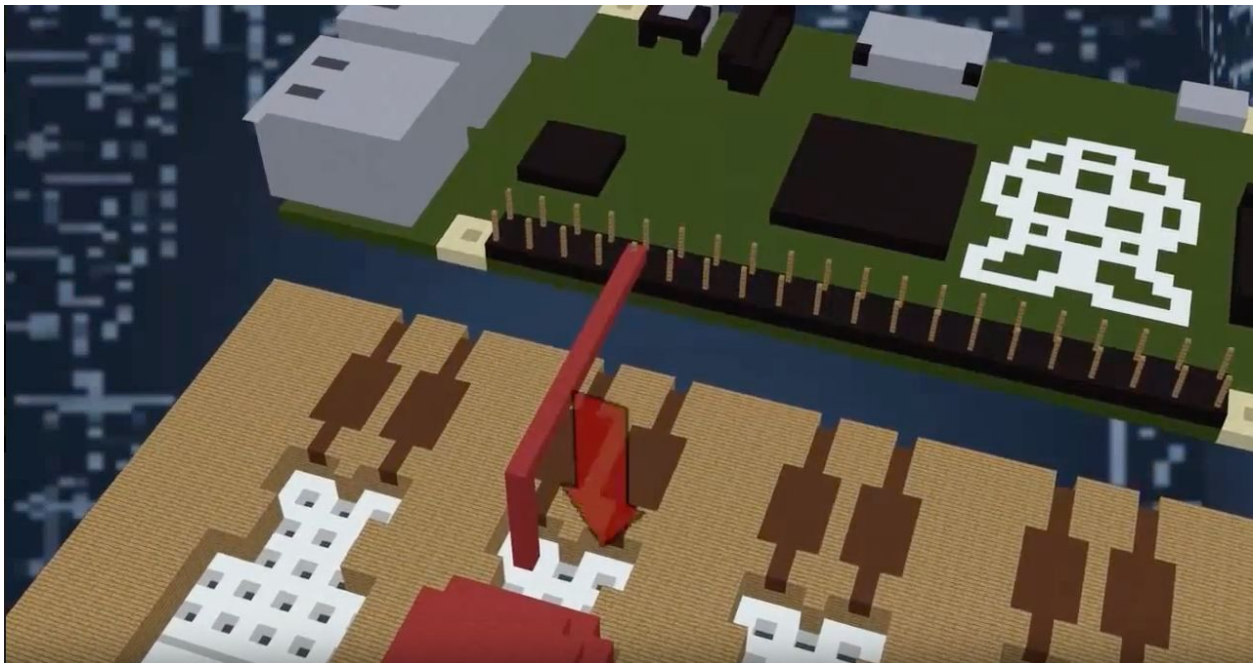
Phase 2: Inputs and Outputs

In Phase 2 we have the students play our custom Raspberry Pi Edition of Minecraft games where they must build electronics to solve challenges in the game.

The Raspberry Pi Edition of Minecraft is a "Creative Commons" licensed version of Minecraft. It is a bit different than the full version in that it does not have all the full features of the PC version. But, it also does not have creepers and zombies out to destroy the students' projects! The Raspberry Pi Edition of Minecraft also has an API (Application Programming Interface) that allows the Piper development team to build these educational games.

Note: Piper is not an official Minecraft product and is not approved by or associated with Mojang. The Raspberry Pi Edition of Minecraft by the Raspberry Pi Foundation is licensed under the Creative Commons Attribution 4.0 International License.

Within our custom games, we have replicated the Raspberry Pi, breadboards, wires, buttons, lights, buzzers, switches, etc. into the 3D Minecraft world. When students are presented with complex build instructions in Minecraft the kids get it! The game incentivizes students to learn electronics by tying these learning objectives directly to solving challenges in the game!



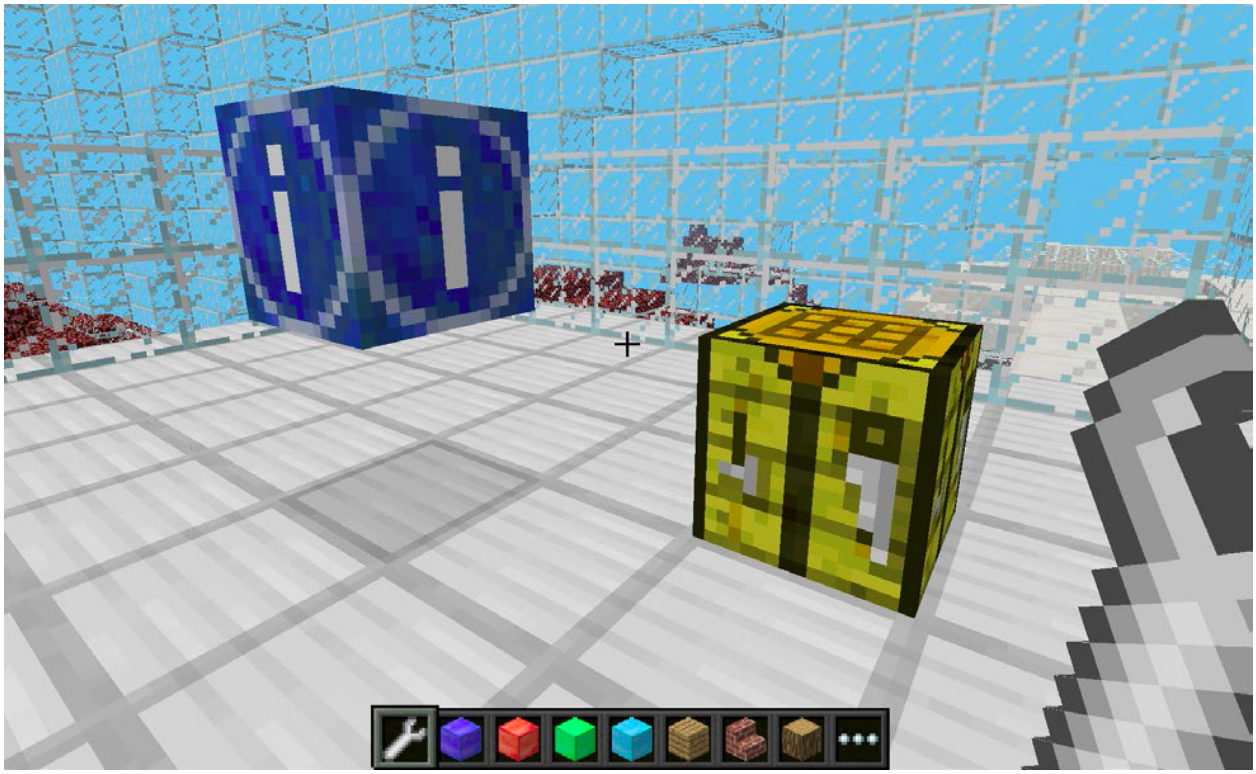
Raspberry Pi Edition of Minecraft

The Raspberry Pi Edition of Minecraft by the Raspberry Pi Foundation is different than the full version, but many of the key functions are the same.

New to Minecraft? 5 Functions to Know

1. Left click the mouse to "break" blocks.
2. Right click the mouse to "place" blocks.
3. Use the mouse scroll to go through inventory (found at the bottom of the game screen).
4. Right click on the "crafting table" to get instructions on what to build.
5. Click on the blue "info block" to get hints on the challenge.

Info. Block ↓ Crafting Table ↓



Inventory ↑

Still lost? Check out these tutorials on the Minecraft Wiki or trainings on Minecraft for Education.

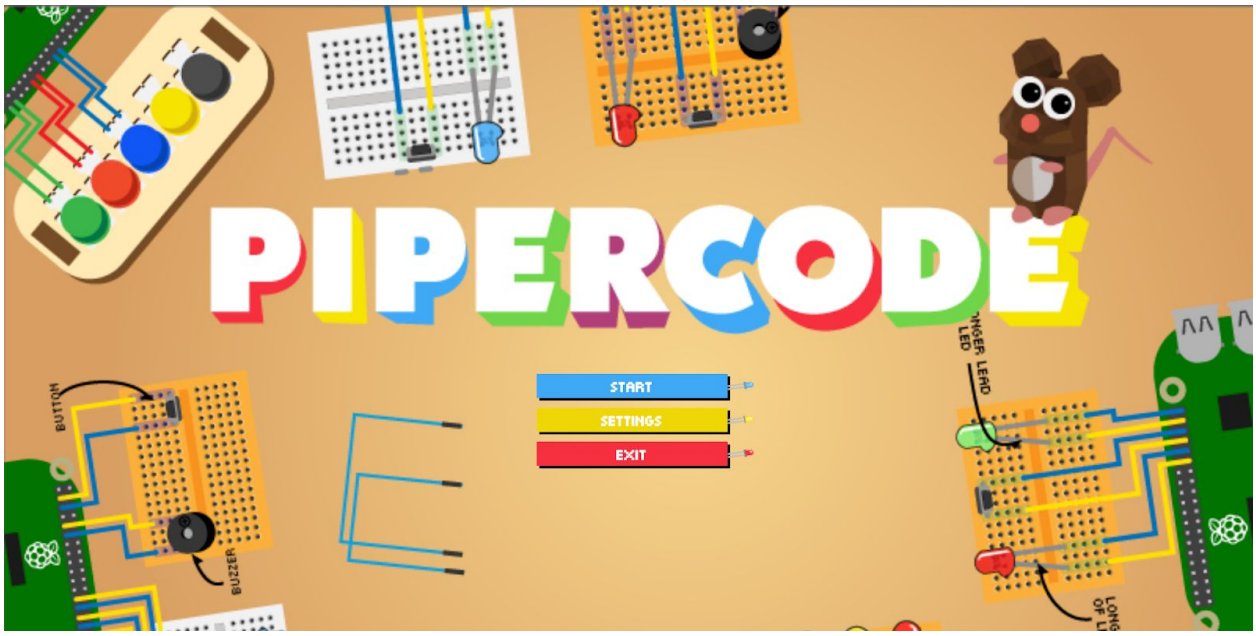
Why use Minecraft as a tool in your learning space? Watch this video.

Phase 3: Code

PiperCode is the newest extension of your Piper Computer, including seven activities diving deeper into coding and electronics to build video games, microcontroller outputs, and crafting. PiperCode was built using Google for Education's Blockly, a library for building visual programming editors. PiperCode is separate from the Raspberry Pi Edition of Minecraft projects. The idea is that your learners understand building inputs and outputs at this point and may now learn to program the inputs and outputs.

Through these first seven projects released in late 2017, you'll expand your world into coding interactive games, building controllers, and art projects with light and sound. PiperCode provides a self-paced interface for learners to complete coding challenges that unlock progressively, supported by step-by-step prompts and real-time video tutorials - all inside the computer they built in earlier lessons!

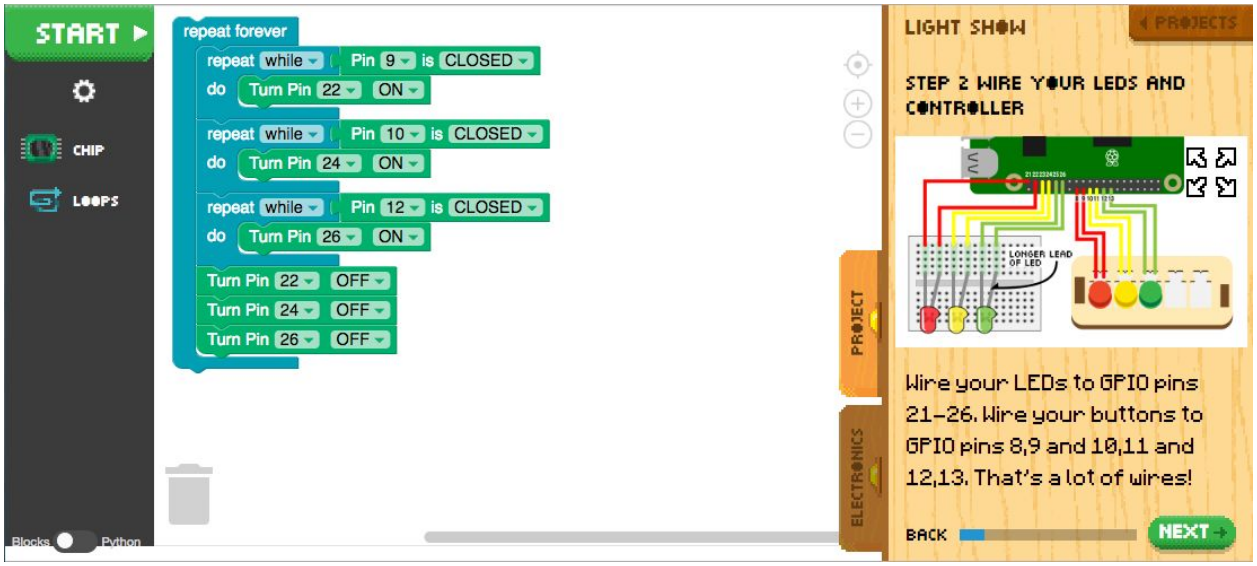
The main PiperCode user interface has three options in the main menu.



Once they click the START button, learners will see a menu of the individual projects. New projects are unlocked once they complete a previous project.

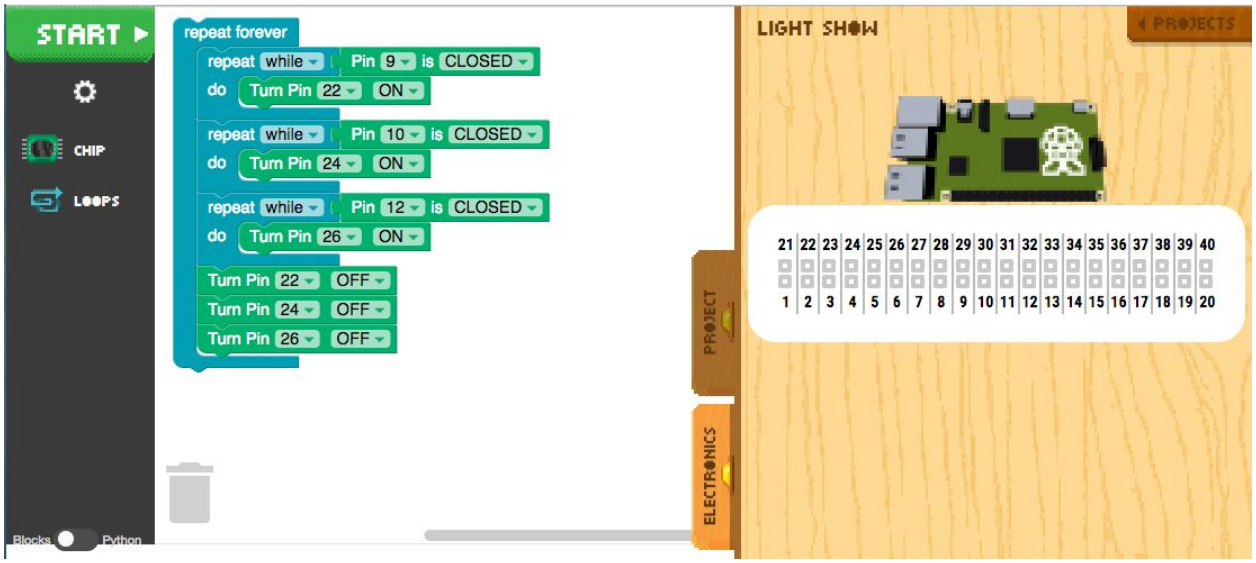
Inside each project, there is a PROJECTS tab in the upper right which will take you back to project menu. On the right side, you will see a tutorial that includes project instructions. Click NEXT to move to the next screen or BACK to review prior steps for troubleshooting.

Click this icon: to ZOOM the image making it easier to see. On the left side of the screen are coding blocks that may be dragged and dropped onto the working screen. Notice the TRASH CAN image where learners may drag and drop code blocks to remove them. Toggle between BLOCKLY and PYTHON in the bottom left corner of this screen, this is for your advanced learners. Most importantly, the START button is how to run code once it is written!



There is no keyboard required for PiperCode, everything is done with your mouse. When a variable is selected either a drop-down list will appear (to change a number) or a virtual keyboard will appear. In each case, use the mouse to select from the drop down or click on the appropriate letters or numbers to fill in the variable.

Learners may switch from the tutorial PROJECT tab on the right side of the screen to ELECTRONICS. Electronics will show the Raspberry Pi GPIO pin numbers. When running code, the GPIO pins that have current running through them will turn yellow. This is a great tool for troubleshooting wiring placement against the code!



Phase 4: Extensions

Extensions are a great way to expand your Piper Program! For example, if you want to have students work on Piper for 20 hours instead of 12 hours, extensions are going be a key part of your program. See the lessons in Phase 4 for more guidance on how to facilitate these extensions.

Here's some ideas from our Piper community blog:

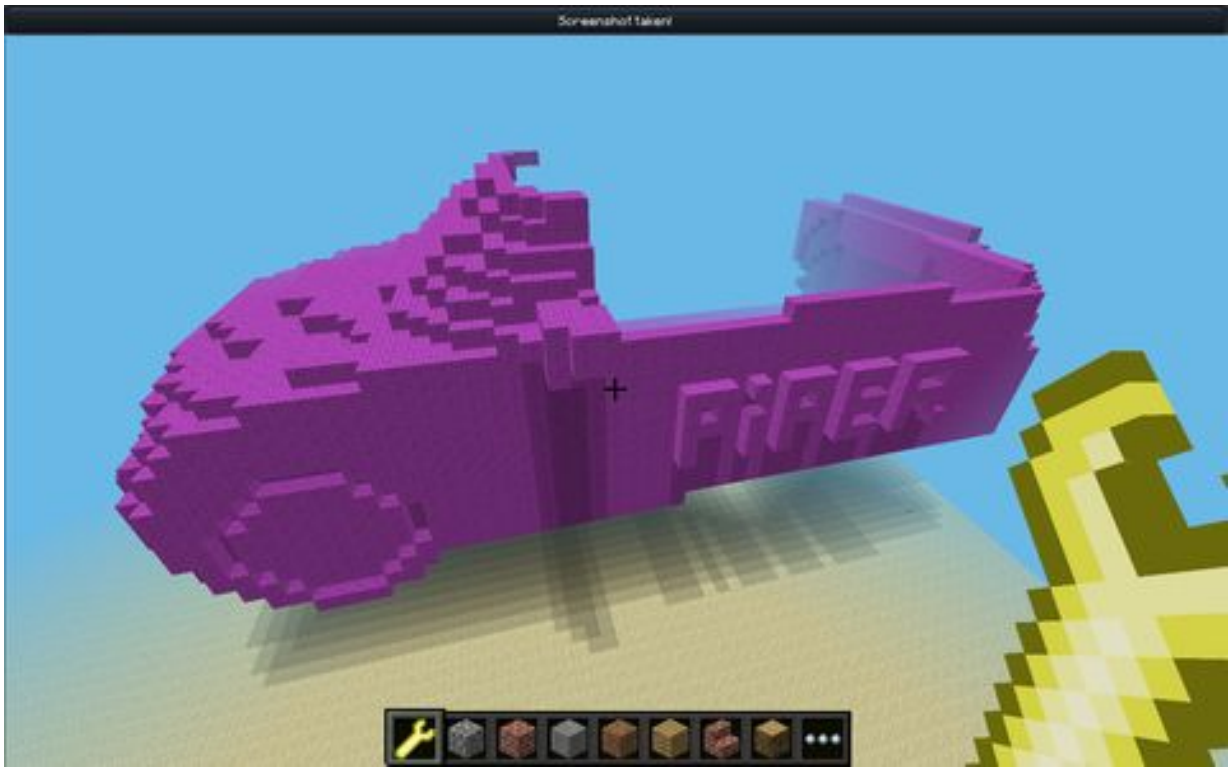
Piper Coding Project: Make music with the wave of a hand!

How to Build Your Own Car Robot - Part1: Build Chassis

How to Build Your Own Car Robot - Part2: Enable movement

How to Build Your Own Car Robot - Part 3: Control and programming

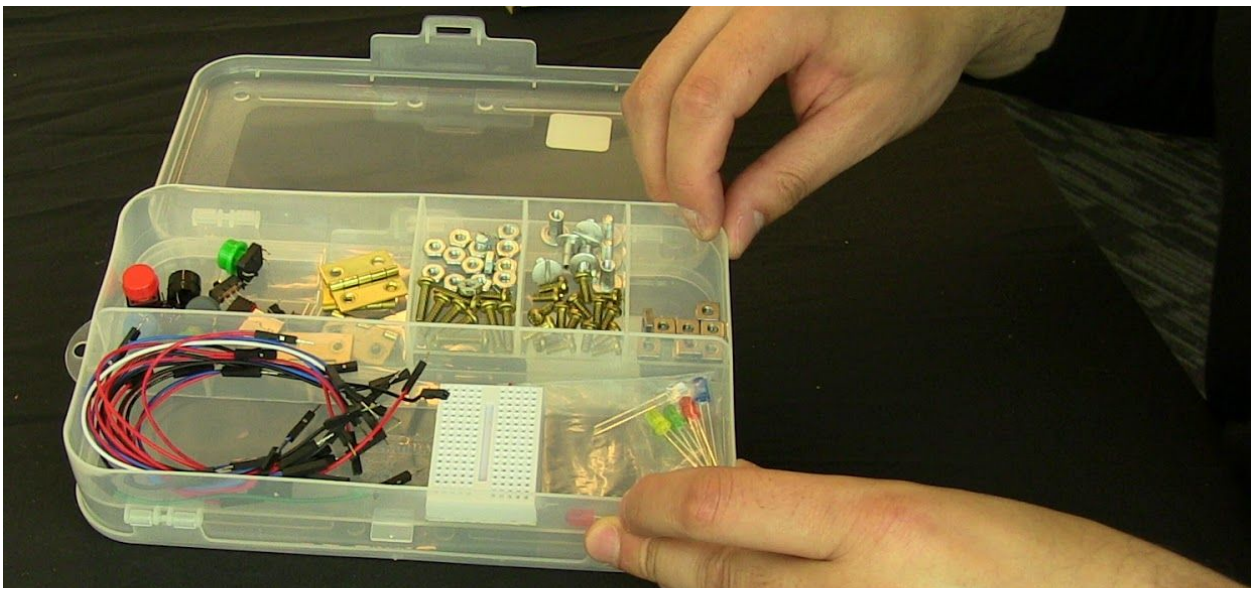
Turn your Piper into an Amazon Echo

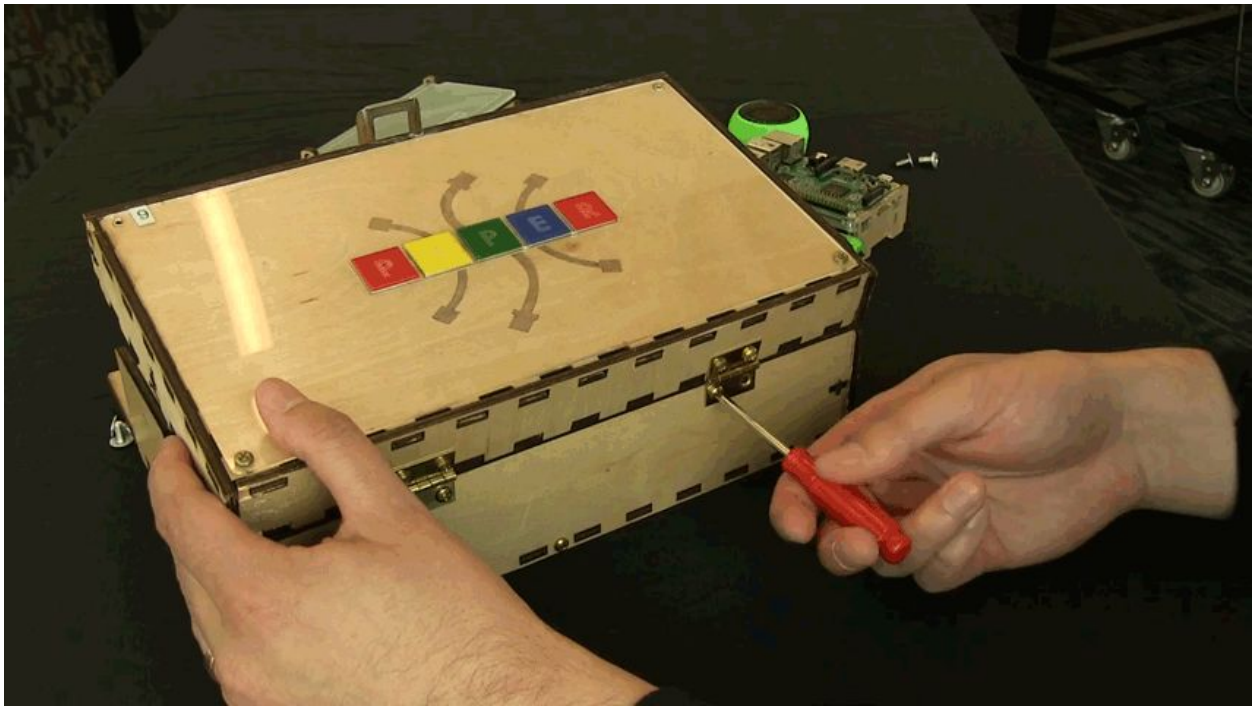


Phase 5: Deconstruct

Throughout the Piper learning experience, students have explored many basic concepts of electronics. To conclude their process, students work together to take apart their Piper Computers, evaluate their efforts, and reflect on their learning.

***Note: The Piper computer kit is designed to be reused roughly 4 times. Heavy reuse may result in decreased quality.*





Educator Preparation

Setting up Your Classroom

What steps should a teacher take when first starting to use the Piper Computer Kits?

1. When building Piper Computer Kits, students may work on tables or the floor. If there is not enough room for each team to spread out the poster, you can project the blueprint on a screen for the whole class to see or put them up on the walls.
2. Make sure to charge your Piper batteries! At the beginning of class hand out the batteries and at the end of class collect them and put them on the charger.
3. Plastic tubs make for great storage containers if you are REUSING Piper Computer Kits for multiple classes. It is key to keep the parts organized so that they don't get lost or broken.
4. If possible, connect one of the Piper Computers to your monitor or projector using the HDMI input on the Raspberry Pi. If not, connect your computer that has this Unit Plan and play the videos, slide presentations, etc. found in the Unit Plan.

PRO TIP: Once built, connect one Piper Computer to your large monitor or projector via HDMI. This allows the facilitator or select student can show the class what they are doing on their Piper!

Facilitation Approach

We want you to be successful using Piper in the classroom. To do that, we suggest using these practices to help ensure maximum learning and engagement for your students.

Collaboration

While Piper works well playing solo, it is even more effective in collaborative groups. Two or three students may all use one Piper kit collaboratively to play the digital game and build electronic gadgets. Additionally, in a classroom environment, we suggest to have students who are struggling to get help from students that are moving through the chapter quickly. This empowers the students to collaborate and solve problems, build community and agency without significant teacher guidance. We have found success in stopping the class every 10 minutes to have a student talk about a challenge in Piper that they overcame.

Facilitation vs Direct Instruction

With Piper, instructions are presented in the game. We recommend facilitating the class versus giving the students the answer. We understand that it is hard to turn down students when they get stuck, but we firmly believe this is part of the learning process, and what ultimately helps build creative confidence. When reading through our lessons, look for the "essential questions" to ask during a lesson. When students ask questions, try answering with leading questions!

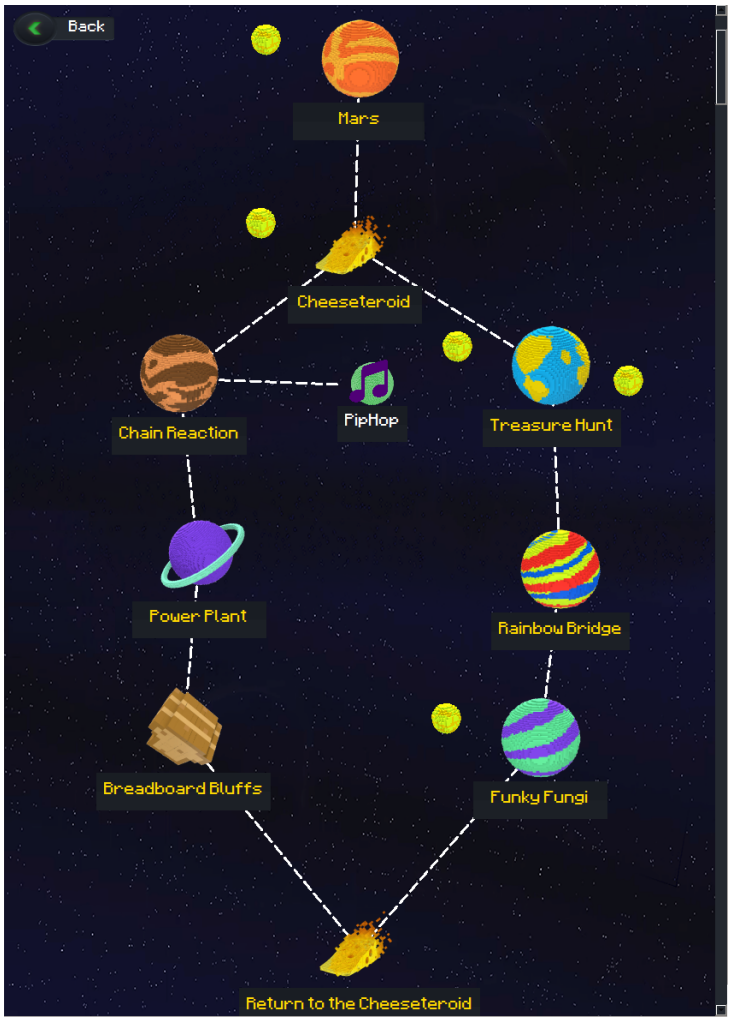
Set Expectations and Build Confidence

We have found success starting each session with a discussion and reinforcing expectations that Piper will be challenging, failure will happen (that is OK and GOOD!), and details are more important than speed, etc.. Have students talk about some things they have made themselves to give them confidence that they are now ready to create more!

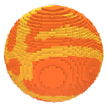
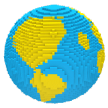
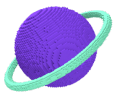
Guiding Your Learners through Story Mode & PiperCode Projects

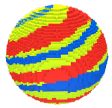
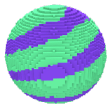
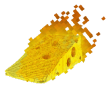
Once your students finish the Build phase and successfully power-up their Piper, Story Mode is the first full fun dive into the interactive worlds that marry gaming with computer engineering principles. Shown below is our Story Mode user interface, each planet is a main project and each moon represents a mini-game. When first opening Piper, students will see many of the planets and moons "locked." We do this so students will progress through the projects in order of complexity.

In the Guides and Slides for each corresponding lesson below, you'll find even more detailed introductions to each Story Mode level and PiperCode project, with more explanation around the concepts and completed circuit diagrams, as well as some troubleshooting hints for guiding your students through some of the hardest aspects of each level. Remember, each level builds upon itself and unlocks more complex ideas and explorations - a big part of your role as educator is to help them persist through frustrations, model how to test connections or find alternatives, and help them celebrate success!



Piper Story Mode MAP: Concepts, Hints, Tips and Build

Project Level	*Concept*	*Level Hints/Tips*	*What You'll Build*
 Mars	Electrical currents and circuits	Hardware hint : Be sure buttons are secure in the breadboard and not wobbly. But, do not push hard if the silver pins are not lined up. They will bend (fixable, but try not to bend them).	Left, Forward, and Right buttons on controller
 Cheeseteroid	Directional flow of circuit	Game hint : follow the red blocks to a crafting table. Open it, then look for an exit out of the room opposite of where you came in. This will take you to a set of doors that will be the completion of the chapter.	Jump button on controller
 Treasure Hunt	Electrical flow through a LED output	Game hint : find a series of clues from info boxes which will eventually lead to a crafting table. After you wire in the LED, use it as a "treasure detector". When Piperbot gets closer to the X, marking the hidden wrench, the LED will flash increasingly faster.	LED Output on small breadboard
 Chain Reaction	Digital binary bit states	Game hint : After finding the crafting table (use TNT to get to it), use TNT to blow through sand and castle walls to reach the next portal.	Switches on small breadboard
 Power Plant	Parallel circuits	Game hint :Turn the power ON to open gates, release water, pump water, raise elevators, and make your way to the next portal. There are two phases to complete this challenge.	Switches on small breadboard
 Breadboard Bluffs	Breadboards as circuits	Game hint : Knockout grey or glass blocks and plant colored blocks to complete circuits in a virtual (inside the game) breadboard and toaster. There are two toaster levels. Game hint: Watch video for tips to get past a bug in software.	Switches on small breadboard

 Rainbow Bridge	Outputs (buzzer) and component polarity	Game hint: The buzzer hints at which color will disappear next. The number of beeps indicates which color will disappear. Hardware hint : Be sure to line up the + on the piezo buzzer with the red wire and the left-most pin of the pin pair.	Add a sound output buzzer to small breadboard
 Funky Fungi	Parallel circuits; grounding; memory states	Game hint: Use each button to call a different structure from memory that will allow you to cross the chasm. Eventually, you'll plug in a switch which will let you "pave" a platform as you move or jump. Game tip: It can be easy to get lost in this level. After the second crafting table keep going up to reach the red platform with the portal.	Two buttons and a switch to small breadboard
 Cheeseteroid Return	On-demand vs. constant states, based on switch	Game hint: Use the switch to help Piperbot switch between constructing and destroying the cheese world to get through. Game tip: Having reached the part where you have to see what both power blocks are doing, there is a crack in the geometry that is easy to miss. Otherwise, keep building, destroying, and alternating from constant and on demand mode until you get through. Game Note: This is the longest and hardest level in the game. Once completed, Piperbot saves earth and creative mode is unlocked.	Add one switch

There are images of MOONS near some of the planets which are fun and educational MINI-GAMES. Mini-games may be played over and over as learners are trying to improve their score or time. Use these as extensions to reinforce learning concepts around building inputs and outputs.

See more on Story Mode Project Levels and Mini-games embedded in corresponding lessons.

Top Ten Educator Tips

Before You Begin:

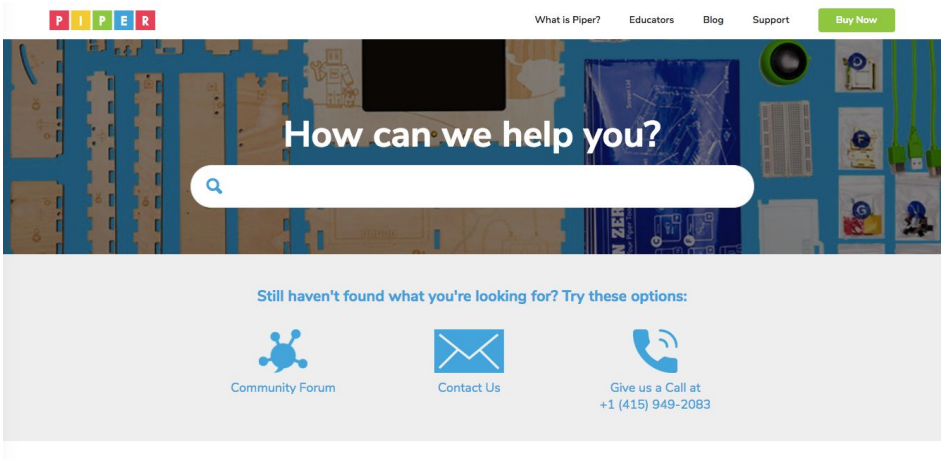
1. Build and play yourself first! The best way to support your students is understand the process. Try building the Piper computer ahead of time, working through Minecraft: Raspberry Pi Edition, and projects in PiperCode!
2. Do your updates! If your kits have been sitting around for a while, make sure you have Updated the Game Software for the most updated StoryMode and PiperCode software.
3. Before turning on any Piper computer, you need to charge the Piper batteries . You should also make it a habit to charge batteries between every lesson! Check out this Piper Support Charging the Battery page.

Support As You Go:

4. If you don't hear sound, you will need to charge the Piper speakers! Check out this Piper Support Charging the Speaker page.
5. During Phase 1: Build a Computer lessons, have a built Piper available for reference. Watch the Piper Support Piper Assembly Video or have this playing in the background.
6. If your learners build the kit and turn on the battery, but the **screen is black**, there's some common missed steps - check out the Piper Support My Screen is not Working page.
7. If your screen is built correctly, but the game still isn't booting up, it's usually that the SD card is not installed correctly - check out the Piper Support The Game is not Booting Up page for more info.
8. We lost some screws! or...Ooops...a part broke! First, if you bought a Piper Classroom bundle, you have extra parts in your spare parts kit! If not, contact Piper Customer Service by emailing hi@playpiper.com.
9. Serve your English Language Learners! - Piper Story Mode comes in English, Chinese (Mandarin), Spanish, French, and Japanese. To choose the language, go into the game's settings menu and select the language option. Read more on Piper Support What Languages does Piper support? Page.
10. Confused about moving around inside Minecraft or Piper Story Mode levels? - Ask one of your learners who's a Minecraft enthusiast to give you a tour! Or check out the New to Minecraft? 5 Functions to Know section of this document.

.

Need more help? Contact Piper Customer Service by emailing hi@playpiper.com or visit https://support.playpiper.com/hc/en-us


What is Piper?
Educators
Blog
Support
Buy Now
How can we help you?
Still haven't found what you're looking for? Try these options:
Community Forum
Contact Us
Give us a Call at
+1 (415) 949-2083

Evaluate the Learning - Measure Creative Confidence Growth

At Piper, our mission is to improve the confidence of students using Piper around CREATING technology! We measure our success around growth in confidence.

Dr. Joel Sadler, Piper co-founder, received his PhD from Stanford where he wrote his thesis on "THE ANATOMY OF CREATIVE COMPUTING: ENABLING NOVICES TO PROTOTYPE SMART DEVICES." In the thesis, Joel came up with a pre/post survey to measure confidence change around novices creating technology. Each survey has 7 questions, 5 of which are the same between the pre and post surveys. (Want to see Joel's thesis? Click here.)

Piper gives you access to these surveys so that you can see the growth in your students' for yourself!

Pre/Post Student Engagement Survey

How it works:

1. Print out the Piperbot Pre-Survey and have students complete BEFORE starting the Piper program. Link to Piperbot Pre-Survey
2. File the completed Pre-Surveys away until the Post-Surveys are complete
3. Print out the Piperbot Post-Survey and have the students complete AFTER the Piper program is completed. Link to Piperbot Post-Survey
4. Compile both the completed pre and post surveys and use the similar questions to see growth in your students' confidence.

*Note: If your school or district is using Piper, we will provide you with electronic (and print) versions of surveys to fill out before and after your students' experience with the Piper computer kit. These will be distributed during Implementation and analyzed for you by Piper. The above surveys are solely for personal use.

Unit Plan Overview

Duration: Each day is equivalent to a 45 - 65 minute period for a total of 12 core lessons.

Phase		Lesson	Piper Tool	Learning
1	Build A Computer-Engage			
Days 1-4	1.1	What is a Piper Computer?	Blueprint	Computers connect to components to form a system
	1.2	Teamwork & Roles	Kit	Perform different roles when collaborating
	1.3	Troubleshooting	Kit	Determine solutions to solve hardware and software problems
	1.4	Safety, IT maintenance, and organization; Reflection	Kit	Break down problems into smaller, manageable tasks
2	Build Electronics-Exploration			
Days 5-8	2.1	Buttons and Breadboard	PiperCraft: Mars and Cheeseteroid	Electric currents
	2.2	Basic Inputs & Outputs (LEDs, buttons and switches)	Treasure Hunt and Chain Reaction	Energy Transfer
	2.3	Polarity & Audio Outputs (Buttons, switches, and power generation)	Power Plant and Rainbow Bridge	Devices that converts energy
	2.4	Parallel Circuits (Buttons and switches in parallel configurations)	Funky Fungi & Return to Cheeseteroid	Fair tests, failure points, and prototyping
3	Code-Explanation			
Days 9-11	3.1	Intro to computational thinking & programming (algorithms & loops)	PiperCode: Blink	Develop and use abstractions

	3.2	Sequences and loops (with inputs)	PiperCode: Stoplight	Program loops and sequences through algorithm design and simulation
	3.3	Events (with inputs/outputs)	PiperCode: Lightshow	Program events and test & debug
	3.4	OPTIONAL: Light and sound inventions	PiperCode: Frog Frenzy	Practice programming loops; Sound as an output
4	Extend-Elaboration			
	4.1	Power-Up - Play	Story Mode MiniGames	Remix, reuse, iterate
	4.2	Power-Up - Design	Story Mode MiniGames	Fostering inclusive computing culture
	4. 3	Create	Creative Mode PiperCode > Build Games PiperCode My Projects	Create, test and refine computational artifacts
5	Deconstruct-Evaluation			
Day 12	5.1	Take units apart and reflection	Kit	Teamwork roles during collaboration and all phases of project development

Educational Standards & Frameworks

California Computer Science Standards

As you begin your Piper Journey, going from build, through story mode, to disassembly, we at Piper want to let you know that we have taken a careful look in 2018 at aligning to both California Computer Science Standards and the Next Generation Science Standards. For NGSS, a particular focus is on Engineering Design and Crosscutting concepts. We have aligned these standards so that you can integrate Piper seamlessly into your day as well as use Piper to extend lessons in Science, Math, and ELA.

The California Computer Science Standards (hereafter referred to as "the standards") are based on computer science core concepts and core practices, aligned to the K12 Computer Science Framework at https://k12cs.org/. The standards were developed by educators (members of the State Board of Education-appointed Computer Science Standards Advisory Committee), utilizing work done by the Computer Science Teachers Association. The standards are designed to be accessible to each and every student in California. Computer science core concepts and practices in the standards are vertically aligned, coherent across grades, and designed in developmentally appropriate grade spans K–2, 3–5, 6–8, and 9–12. The K–12 standards are referred to as core. As students progress through the standards from grades K–12, they build conceptual knowledge through active engagement in creative problem solving activities with an awareness of cultural and societal contexts. The standards are framed around the 5 core concepts including: the composition of computing systems (CS), the connective power of networks and information systems (NI), the informational potential of data and analysis processing (DA), the development of algorithms and programming (AP), and the impacts of computing on culture and society (IC).

Finally, the computer science core concepts are enacted via the c*omputer science core practices*. As students engage in the computer science core practices, they learn to persevere in solving authentic, community-based problems, grounded in computer science core concepts. These are key takeaways when your students engage with Piper. The computer science core practices, covered in greater detail in the *What is Computer Science?* section, include K-12CS P1-7 Practices:

1) P1: Fostering an Inclusive Computing Culture
2) P2: Collaborating Around Computing,
3) P3: Recognizing and Defining Computational Problems,
4) P4: Developing and Using Abstractions,
5) P5: Creating Computational Artifacts,
6) P6: Testing and Refining Computational Artifacts,

7) P7: Communicating About Computing. Standards integrate computer science practices with concept statements.

The Concepts and Practices of the K–12 Computer Science Framework

Core Concepts	Core Practices
1. Computing Systems	1. Fostering an Inclusive Computing Culture
2. Networks and the Internet	2. Collaborating Around Computing
3. Data and Analysis	3. Recognizing and Defining Computational Problems
4. Algorithms and Programming	4. Developing and Using Abstractions
5. Impacts of Computing	5. Creating Computational Artifacts
	6. Testing and Refining Computational Artifacts
	7. Communicating About Computing

Next Generation Science Standards (NGSS)

The key point of the new standards is to use performance expectations (PE) to show what students should be able to know and do by the end of instruction. Each performance expectation integrates the NGSS three dimensions introduced in the National Research Council's *A Framework for K-12 Science Education* enabling students to *learn science* by *doing science.*

The three dimensions or 3-Ds are;

1. Disciplinary Core Ideas (DCI); Life Science, Earth and Space Science, Physical Science
2. Crosscutting Concepts (CCC): Patterns, Cause and effect: Mechanism and explanation, Using mathematics and computational thinking, Systems and system models, Structure and function, Stability and change
3. Science and Engineering Practices (SEP): Developing and using (electronic) models, Asking questions

The NGSS look completely different than previous science standards and implementing them requires a major shift in classroom instruction and learning. Piper helps with this new shift by addressing the 3 dimensions of science instruction.

Beginning in elementary, students in kindergarten through fifth grade begin to develop an understanding of the four disciplinary core ideas: physical sciences; life sciences; Earth and space sciences; and engineering, technology, and applications of science. In the earlier grades, students begin by recognizing patterns and formulating answers to questions about the world around them. By the end of fifth grade, students are able to demonstrate grade-appropriate proficiency in gathering, describing, and using information about the natural and designed world(s). The performance expectations in elementary school grade bands develop ideas and skills that will allow students to explain more complex phenomena in the four disciplines as they progress to middle school and high school. While the performance expectations shown in kindergarten through fifth grade couple particular practices with specific disciplinary core ideas, instructional decisions should include use of many practices that lead to the performance expectations. The standards spiral through secondary education with performance expectations building on previous conceptual development and progressing to more age appropriate and grade level rigor.

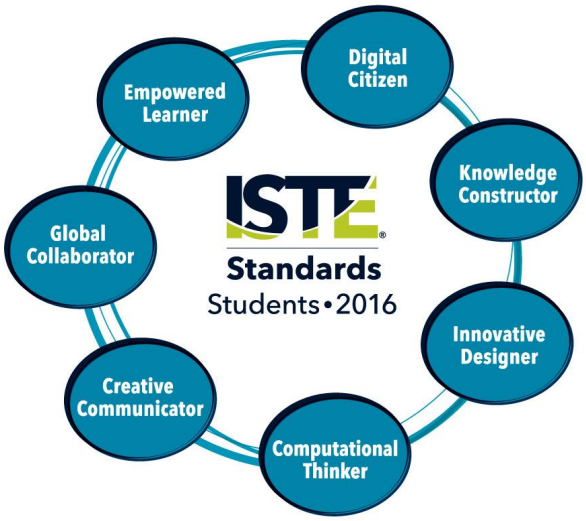
Common Core State Standards

The Common Core State Standards for English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjects (hereafter referred to as "the Standards") are the culmination of an extended, broad-based effort to fulfill the charge issued by the states to create the next generation of standards in kindergarten to grade 12 to help ensure that all students are literate and college and career ready no later than the end of high school. Piper has included CCSS alignments and references in this unit plan to help teachers use Piper lessons to augment and support core instructional practices.

Other K-12 Standards Addressed In This Unit

ISTE

The International Society for Technology in Education (ISTE) Standards for Students were designed in 2016.



ISTE 2016 - Global Collaborator

Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.

7c - Students contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.

ISTE 2016 - Computational Thinker

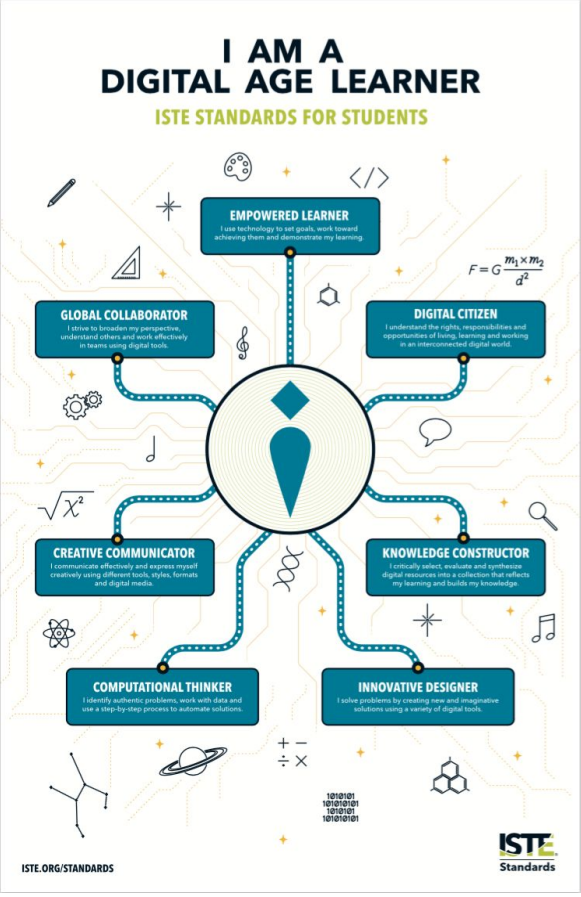
Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.

5a - Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.

5b - Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.

5c - Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

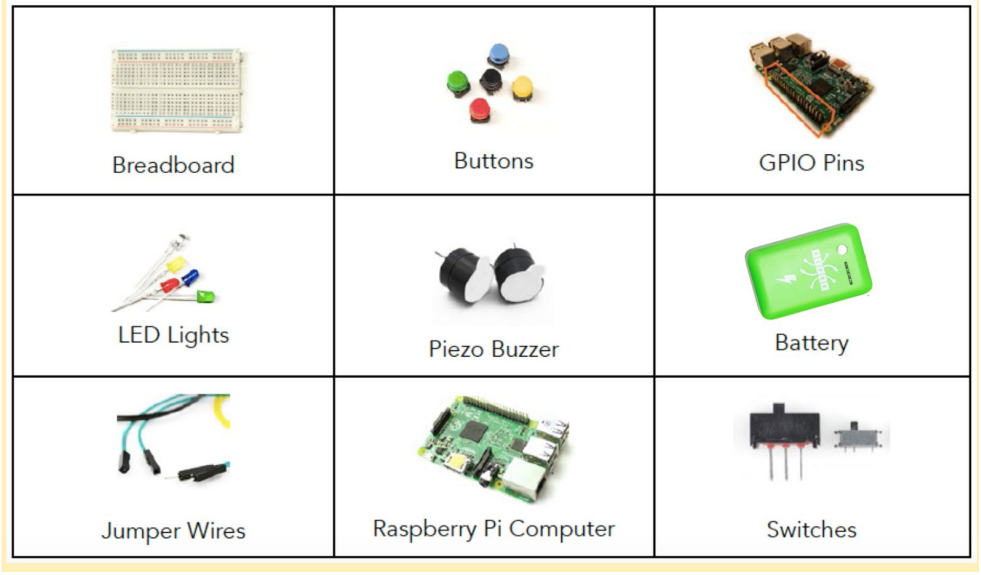
5d - Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.



Vocabulary

Electronics

- **BreadBoard**: A flexible wiring board that lets us connect components to the Raspberry Pi.
- **Buttons**: Basic electronic components to momentarily close circuits.
- **GPIO Pins (General Purpose Input Output)**: A programmable interface that connects almost anything (such as buttons and LEDs) to the Raspberry Pi.
- **Jumper Wires**: Wires used to connect from the GPIO to the breadboard. Ours are 'socket' to 'pin', but they can have any combination of those two.
- **LED (Light Emitting Diodes)**: A diode is a gate that allows current to flow in only one direction. LEDs provide light.
- **Piezo Buzzer**: Basic audio resistive component. The higher voltage it gets, the higher the pitch it emits.
- **Raspberry Pi**: A low cost microcomputer. Together with Raspbian, a special operating system, Raspberry Pi is a full blown computer.
- **Switches**: Basic electronic components that close circuits permanently.



Programming & Coding

- **Sequence**: identifying a series of steps for a task
- **Loops**: running the same sequence multiple times
- **Parallelism**: making things happen at the same time
- **Events**: one thing causing another thing to happen
- **Conditionals**: making decisions based on conditions
- **Operators**: support for mathematical and logical expressions
- **Data**: storing, retrieving, and updating values

Need more?
Check out https://k12cs.org/glossary/

K12 COMPUTER SCIENCE

LESSON PLANS WITH STANDARDS ALIGNED BY PHASE

Phase 1 - Build a Computer

	Phase	*Tools*	*Students will...*
1	Build A Computer	Piper Computer Kit, Blueprint	Build the case and connect components of a working computer.

California Standards Alignment for Phase 1 Build a Computer

Beginning with phase 1, all phases will align with standards that apply to all the lessons in the phase. Use them with daily or weekly agendas and planning. Phase 1 is where the learners first build a Piper kit.

Concept and Sub-Concept	Standard
Computing Systems: Devices	CA 3-5.CS.1 Describe how computing devices connect to other components to form a system. (P7.2)
Computing Systems: Troubleshooting	3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)
Algorithms & Programming	3-5.AP.13 Decompose problems into smaller, manageable tasks which may themselves be decomposed. (P3.2) 3-5.AP.18 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development (ie. following the Piper Blueprint)
Impacts of Computing	CA CS 3-5.IC.20 Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.
Practices	P1. Fostering an Inclusive Computing Culture P2. Collaborating Around Computing P4. Developing and Using Abstractions P5. Creating Computational Artifacts P6. Testing and Refining Computational Artifacts

CA NGSS Connections:

(4-PS4-3) Generate and compare multiple solutions that use patterns to transfer information.

3–5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem (Performance Expectation).
3–5-ETS1-3. Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (P.E.3.4.7)

Common Core State Standards Connections:

ELA/Literacy -

(CA CCSS for ELA/Literacy SL.3.1, SL.4.1, SL.5.1):(CA CCSS for ELA/Literacy W.3.2, W.4.2, W.5.2)

- RI.4.1 - Refer to details and examples in a text when explaining what the text says explicitly and when drawing inferences from the text. (4-PS4-3)
- RI.4.9 - Integrate information from two texts on the same topic in order to write or speak about the subject knowledgeably. (4-PS4-3)

Mathematics -

MP1 Make sense of problems and persevere in solving them.

MP6 Attend to precision

MP5 Use appropriate tools strategically. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)

Lessons 1.1 - 1.4 - Build the Piper Computer

Lessons 1.1 What is a Piper Computer?

Established Goals

The goal of this phase of is that students become literate the Piper materials, tools, and hardware components in the kit. Instead of front loading a sheet of terms, students can experience the Parts, Purposes, Complexities thinking routine (described below) developed by Agency by Design to explore computer and electronics concepts and vocabulary. The goal of the activity is for students to engage in close looking, close reading of text, inspire question making and natural curiosity, and have student observations and questions recorded on a graphic artifact (poster with labeled drawings of the parts).

Learning Objectives:
Students will

- Define a computer and its essential parts.
- Explore the idea that every component of the Piper computer interface has purposes.
- Define a blueprint and practice how to use it as a set of instructions.
- Understand safety and good IT maintenance practices.
- Practice forming and acting as a collaborative team.

Suggested Sequence:
• Lesson 1.1 : Parts, Purposes, Complexities of the basic microcomputer components
• Lesson 1.2 - 1.4: Introduce case wooden parts,brass hardware, kit build, and SD card

---

Preparation

- Suggested student to kit ratio is 2:1 up to 3:1
- Prepare enough medium or large format drawing paper (or other surfaces such as dry erase tables, poster and panels) and colored pencils or markers.
- Make sure each kit contains the following pieces: Raspberry Pi, SD card*, mouse, display, cables (power, USB, HDMI), speaker, wires, breadboard, colored buttons, battery*.
- *Note: If you don't want students to go into the software yet hold on to the power pack or SD card.
- Charge the batteries and speakers before every session
- Hold on to the blueprint, laser cut wood pieces and hardware until time to build

Introduction (5 minutes):

•Assess prior knowledge-Have students draw on a sheet of paper what they think the components of a computer are. Pair share with a partner. Then call on a few teams to share out.
• Announce the new unit to the students:

> "By the end of this unit, you'll know how to build a computer, its parts and how to create basic electronics while playing a Minecraft mod. In the first few sessions, you'll be building the microcomputer from scratch, including constructing the case, connecting the components, and troubleshooting any problems as a team."

• Hook: show full how-to video (or have playing in background).
• Announce expectation that before we can play, we need to learn to build a computer.

Main Activity (80% of class time):
(50% of class time):
• Distribute kits in pairs or teams of 3, without the blueprint or prompt letter.
• Set norms, roles and responsibilities of setup, organization, safety, and cleanup.
• Prompt students to address the prompts below through a labeled illustration of the system (aka the blueprint or your own hand drawing) on butcher block pieces of white paper under the kits. Students should be aware they will be sharing their work with the class later.

In different colored pens or on colored sticky notes, students sketch 3 levels of reflection on the pieces of the kit:

- What are its parts? What are its various pieces or components?
- What are its purposes? What are the purposes for each of these parts?
- What are its complexities? How is it complicated in its parts and purposes, the relationship between the two, or in other ways? Who is it built for and why?*

* This Parts, Purposes, & Complexities discussion can be critical in facilitating understanding of the computer science concept that computing technologies that have changed the world, and learners' ability to express how those technologies influence, and are influenced by, cultural practices (CA CS Standard 3-5.IC.20)

• During this time, the teacher should be roaming around the room, asking provocative questions such as:

- What do you think that is?
- Where do you think it connects to?
- What would you call that?
- See that label or number on it? Look it up on the internet and tell me what it is.
- How does it work?
- Are you following the blueprint?

(30% of class time):

• Define blueprint as….

> Sample definition of blueprint (https://www.merriam-webster.com/dictionary/blueprint)
> : a photographic print in white on a bright blue ground or blue on a white ground used especially for copying maps, mechanical drawings, and architects' plans
> 2: something resembling a blueprint (as in serving as a model or providing guidance);especially : a detailed plan or program of action

• Distribute blueprints and encourage students to think on own or in pairs on how it visually instructs you

Closing Activity (15-20% of class time):
• Group discussion: Have one group volunteer to share their work with the class. Visibility for whole class will be key.
• Address misconceptions as students share.
• Collect work, allow time for students to clean up after themselves.

Lesson 1.2-1.3 - Teamwork and Troubleshooting

Established Goals

Students will also experiment with the components to learn how they fit and work together.

During the three build sessions, facilitators guide the learners through constructing the case, connecting the components, and successfully powering up the Piper mini computer.

It usually takes 2-4 hours to complete a kit build, so lessons 1.2 and 1.3 are the same, but give your students more time if needed, then 1.4 incorporates final reflection before moving on to Phase 2.

Learning Objectives:
Students will:

- Decompose the challenge of building a computer by breaking down the tasks into steps.
- Build teamwork skills by combining understanding and assigning roles to solve problems.
- Perform different roles when collaborating with peers
- Explore how physical connections to components build a mini computer system, including both input and output devices.
- Construct a model that illustrates how hardware and software work as a system.
- Troubleshoot and problem solve to ensure Piper computer case and components power up correctly.

Lesson 1.2 & 1.3: Build a Computer: Reflection

Suggested Sequence:
• Lesson 1.2 - 1.4: Introduce case wooden parts and brass hardware, kit build, SD card, then reflect before diving into software when kit turns on.

Introduction (5-15 minutes):
• Distribute kits in pairs or teams of 3. Reinforce norms, roles and responsibilities of setup, organization, safety, and cleanup.
• Kit should be built by end of day 4. Students should work together and use the blueprint as a building guide.*
• At start of each day, have a different group share the current status of their build and share a lesson learned or discovery moment.* Facilitate a collaborative discussion in which they identify and list computing system problems and then describe common successful fixes.

* These activities will help the learning be led by the learners themselves and will also help you reinforce the computer science learning around teamwork and problem solving (Reference CA

CS Standards 3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies; also see 3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. - For example, students could prepare for and participate in a collaborative discussion in which they identify and list computing system problems and then describe common successful fixes. (CA CCSS for ELA/Literacy SL.3.1, SL.4.1, SL.5.1)

Main Activity (80% of class time):

• Have students open the kit, review the types of materials, tools, and connect groupings to the flow of visual instructions in the blueprint.
• Have students organize parts by build step and differentiate between types of screws, fasteners, etc.
• Have students put case and electronic components together.

Closing Activity (10-15% of class time):

• Have students record any new findings on their graphic artifact (drawings) from Lesson 1.1 and share out to the whole group (post on walls).
• Make sure groups organize, label, and store their parts and builds still in progress.
• Collect notes and kits.

Lesson 1.4: Build a Computer: Reflection

Introduction (2-5 minutes):
• Announce expectation: Today is the last session to build your Piper Computer. If a team near you has fallen behind, it's your responsibility to help them catch up or share your lesson learned.

Main Activity (50-75% of class time):
• Students finish building kit and graphic artifact. Try not to micromanage here... students will make mistakes and find ways to solve the problem themselves.

Closing Activity (25% of class time):
• Teacher led discussion: Phase 1 Component Review slides. Students should take notes from the slides on their graphic artifacts.
• Clarify misconceptions and answer questions as needed.

EXTEND: Have a group discussion comparing their Piper Computers to other computers the students interact with daily (desktops, laptops, tablets, smartphones, etc.) Share earlier versions of computers, including Steve Job's early prototypes.*

* This extension discussion could help you reinforce the computer science learning around the role of technology in society and culture (reference CA CS standard CA CS 3-5.IC.20 Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.)

Phase 2 - Build Electronics

	Phase	*Tools*	*Students will...*
2	Build Electronics	Story Mode > Play	Build & understand input and output electronic components, as part of computer engineering, while completing levels in a game

California Standards Alignment for Phase 2 Build Electronics

These standards apply to all the lessons in this phase, when the learners dive into Story Mode and when the Piper powers up! Use them with daily or weekly agendas and planning.

Concept and Sub-Concept	Standard
Computing Systems: Devices	CA 3-5.CS.1 Describe how computing devices connect to other components to form a system. (P7.2)
Computing Systems: Hardware & Software	CA 3-5.CS.2 Demonstrate how computer hardware and software work together as a system to accomplish tasks. (P4.4)
Computing Systems: Troubleshooting	3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)
Algorithms & Programming	3-5.AP.13 Decompose problems into smaller, manageable tasks which may themselves be decomposed. (P3.2) 3-5.AP.18 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development.
Practices	P1. Fostering an Inclusive Computing Culture P2. Collaborating Around Computing P4. Developing and Using Abstractions P5. Creating Computational Artifacts P6. Testing and Refining Computational Artifacts

CA NGSS Connections:

(4-PS3-2) & (4-PS4-2) Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat, and electric currents.

(4-PS3-4) Apply scientific ideas to design, test, and refine a device that converts energy from one form to another.

(4-PS4-3) Generate and compare multiple solutions that use patterns to transfer information.

3–5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem (Performance Expectation).

3–5-ETS1-3. Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (P.E.3.4.7)

Common Core State Standards Connections:

ELA/Literacy -

(CA CCSS for ELA/Literacy SL.3.1, SL.4.1, SL.5.1):(CA CCSS for ELA/Literacy W.3.2, W.4.2, W.5.2)

W.4.7	Conduct short research projects that build knowledge through investigation of different aspects of a topic. (4-PS3-4)
W.4.8	Recall relevant information from experiences or gather relevant information from print and digital sources, take notes and categorize information, and provide a list of sources. (4-PS3-4)

Mathematics -

MP1	Make sense of problems and persevere in solving them.
MP2	Reason abstractly and quantitatively. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)
MP6	Attend to precision
MP5	Use appropriate tools strategically. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)

Lesson 2.1 - Buttons & Breadboards

Established Goals:
Having built the kit and learned the physical structure and functions/purposes of the all parts, students will now go through the fundamental concept of wiring a circuit and understanding how circuits on the breadboard work.

Learning Objectives:
• Practice breadboarding and wiring.
• Understand wire and pin positions for specific buttons.
• Understand that circuits flow from positive to negative, through a component.
• Understand the difference between closed and open circuits.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):
• Introduce game and tell students that they will be learning how to make Piperbot move using wires, buttons and breadboards.

Main Activity (30-40 minutes):
• Allow students to go through **Mars** and **Cheeseteroid** worlds.
  - Review the Project Level Guides for Mars & Cheeseteroid
• During this time, the teacher should be roaming around the room, asking the essential questions of this lesson*:
  - What happens when you press the button (both in the physical and electrical worlds and inside the computer)?
  - Why do we need two wires per component?

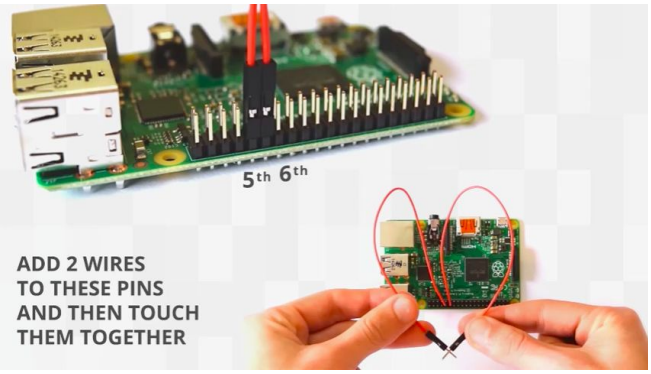
*These checks for understanding help reinforce the science skill of making observations and providing evidence that energy can be transferred from place to place, through sound, light, heat, and electric currents (NGSS (4-PS3-2) &(4-PS4-2))

• Note: Students may have trouble getting the buttons to work. Remember that good contact is the key to good electric flow. Buttons and wires need to be properly aligned and seated. Students may need assistance getting the wiring just right.
• Students put kit away to avoid distractions during teacher led discussion.

Closing Activity (10-15 minutes):
• Teacher led discussion: 2.1 Slides - How do breadboards work?

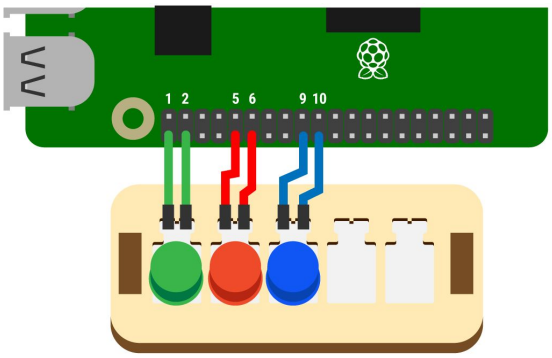
Introduction:

Piperbot is immobile because his interior wiring has decayed over 50 years of inactivity and is no longer working.. To allow Piperbot to move, the student must troubleshoot by first completing a circuit in the game then physically complete circuits with jumper wires connected to GPIO (Global Placement Input Output) pins on the Raspberry Pi.



Concept:

The concept at work in this lesson is electrical currents, conductivity, and circuits. The student should understand that after connecting the red wires, there is an electric current flowing through both pins on the Pi that sends a message (INPUT) for Piperbot to move forward or strafe (slide) sideways. Only when the circuit is completed can the current flow properly. When the wires are placed into the breadboard, pushing the button, rather than touching the metal ends, allows the current to flow.



GPIO setup: Green:1,2; Red: 5,6; Blue: 9,10

Troubleshooting

The first questions you will recieve are "Why can't I move?" The answer is that "you haven't built your controller yet!" They must build their "move" button first in order to move.

A common point of failure is the specific placement of the wires on the GPIO pins on the Raspberry Pi. The necessary attention to detail will take some students by surprise but will set the precedent for the level of precision expected through the game. If there are two or more students working on a single Piper, moving Piperbot with the wires while directing him with the mouse is a good way to incorporate both players. Additionally, be sure that students are firmly placing the buttons on the breadboard. If they are loose or wobbly, the connection won't be stable enough to complete the circuit!

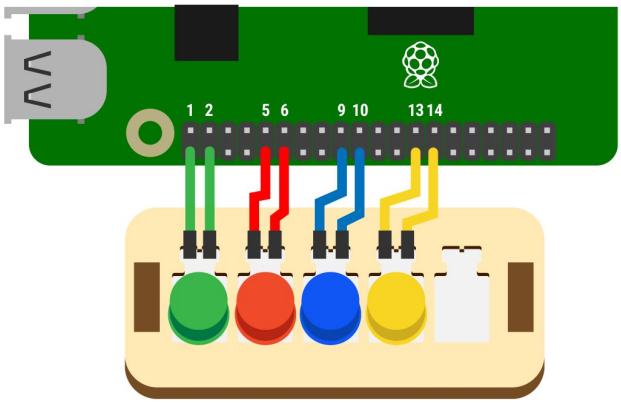
Videos:

Video #1	Introduction video played when opening the Mars project
Video #2	Instruction video about wiring to Raspberry Pi played when all circuits have been completed in the game
Video #3a	Instruction video about crafting tables played after player exits telescope room, goes through the door and up the stairs
Video #3b	Instruction video about repairing Piperbot's controls plays after Video #3a

Project Level Guide - Cheeseteroid

Introduction:

Piperbot is inside the Cheeseteroid maze and needs a new jumping function to get through. Students can accomplish this by continuing with the wiring scheme presented earlier. The main difference is that the yellow wires will go into pins 13 and 14. Once able to jump, it's just a matter of getting through the maze following the red blocks.



Concept:

Building upon the learnings from Mars: electrical currents, circuits, and buttons. This is a good time to introduce the additional concept of directional flow of circuits. Draw upon an analogy between the Raspberry Pi and a battery to teach about how current flows from positive to negative. Ask students to observe the animation to determine which are the positive and negative poles. Students should understand that one pin is positive and one is negative.

GPIO setup: Yellow, 13/14

Troubleshooting:

To solve the challenge follow the red blocks, they will lead you to a crafting table. The crafting table hints at a Golden Wrench, but the actual Golden Wrench is not available until the final chapter Cheeseteroid Return. A tip is to hold the forward and jump buttons down and put pressure forward on the buttons to get good electrical contact from all components. This helps students move and jump continuously over the rugged terrain.

Videos:

Video #4	Storytelling video plays after completing Mars game
Video #5	Storytelling video about the golden wrench

Lesson 2.2 - Basic Inputs & Outputs

Established Goals
Having learned the basics of circuits, breadboards and the different electrical properties of different materials, students will understand the difference between an output and input by comparing the LED behavior to button and switch behavior.

Learning Objectives
• Understand there are many different kinds of inputs and outputs.

• Understand electricity can be controlled to behave like binary (most basic computer language) and either have a high or low voltage: high = 1 = ON, low = 0 = OFF

• Understand that electric flow is sensed by the computer hardware and programmed to have an effect in software.

• Understand that the computer is programmed to detect the state of the electricity going into the pin.

• Understand that the computer is programmed to send a high voltage to the pin when light is desired.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):
• Tell students they will be working through 2 more levels of the game today, which will introduce using LED lights and switches.

Main Activity (30-40 minutes):
• Allow students to go through **Treasure Hunt** and **Chain Reaction** worlds.
  - Review Project Level Guides for Treasure Hunt and Chain Reaction.
• During this time, the teacher should be roaming around the room, asking the **essential questions*** of this lesson:
  - What happens when you press the button in the electrical and software worlds? What happens when the LED is lighting up?
  - Why do we need two wires per component?
  - What is the difference between a switch and a button?

*These checks for understanding help reinforce the science skills of generating and comparing multiple solutions that use patterns to transfer information (NGSS (4-PS4-3))

• Note: Students may have trouble getting the buttons to work. Remember that good contact is the key to good electric flow. Buttons and wires need to be properly aligned and seated. Students may need assistance getting the wiring just right.
• Students put kit away to avoid distractions during teacher led discussion.

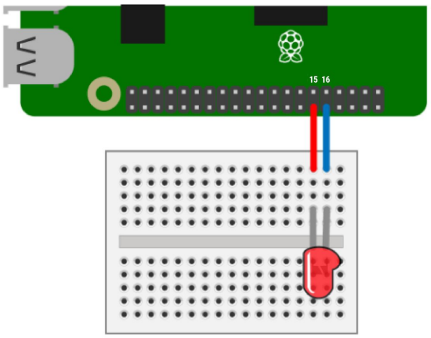
Closing Activity (10-15 minutes):
• Teacher led discussion: 2.2 SLIDES - Basic GPIO (LEDs, buttons and switches)
    - Students participate in discussion, take notes on graphic artifact.

Project Level Guide - Treasure Hunt

Introduction:

Piperbot's task is to find the golden wrench. To do so, Piperbot will need to find a series of clues from info boxes which will eventually lead to a crafting table. Once there, Piperbot will wire an LED to a small breadboard and Pi. The LED has been coded as a "treasure detector". When Piperbot gets closer to the X, marking the hidden wrench, the LED will flash increasingly faster.

GPIO setup: Red 15, Blue 16



Concept:

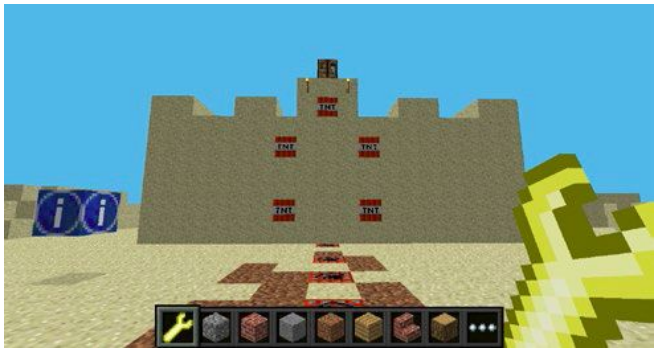
Building on the concepts of current and circuits practiced on the previous functions, the student should understand that after connecting the wires to the LED, there is an electric current flowing from the Raspberry Pi through the Light Emitting Diode (LED), which sends a visual message (OUTPUT) to the player.

Troubleshooting:

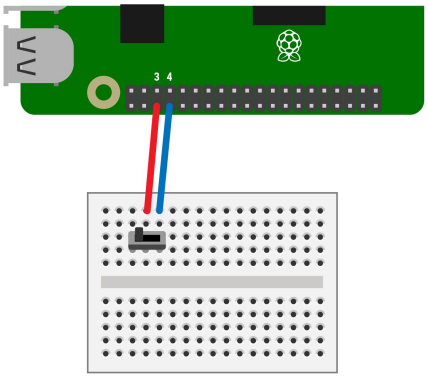
The instructions note to break the crystal which is located above the teal blue blocks. Note that when building your electronic, the LED output has been coded to work without care of polarity. The LED can be put in either way and should work. Bent LEDs may be an issue.

Project Level Guide - Chain Reaction

Introduction:
Piperbot will need to blow its way through sand and castle walls to reach the next portal. Along the way, momentary switches (buttons) and switches are compared and contrasted. Make sure to follow the first instructions, to left click the TNT block in order to find the crafting table.



GPIO setup: Red 3, Blue 4



Concept:
This is mostly review, however instead of enabling motion, holding the switch down enables a different wrench function. This is the basic concept behind digital binary bit states (ON/OFF is equivalent to 1/0).
Note: Unlike a button which has to be held down to hold a bit state, switches hold the state depending on the position of the switch.

Troubleshooting:
For young people, the biggest pain point is having the manual dexterity and hand strength to properly seat the switches on the mini breadboards. This is a good opportunity to revise attention to precision when wiring components.

Lesson 2.3 - Polarity & Audio Outputs

Established Goals

Having understood the difference between an output and input, students will review using buttons and switches and learn the new concept of polarity.

Learning Objectives

- Understand buttons and switches are used in most complex electric and electronic devices, no matter the size.
- Understand switches are different to buttons in that they allow a sustained effect. Buttons are momentary.
- Understand components with polarity have to be placed a certain way in the circuit.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):

- Tell students they will be playing through 2 more levels of the game today, exploring the many ways to use buttons and switches in Piper, and learning about polarity.

Main Activity (30-40 minutes):

- Allow students to go through the **Power Plant** and **Rainbow Bridge** worlds.
  - Refer to Project Level Guide
- During this time, the teacher should roam around the room asking the **essential questions*** for this lesson:
  - In Power Plant: Where have you seen buttons and switches used before?
  - In Rainbow Bridge: Look closely at the buzzer, what special markers are used on it to indicate polarity? What happens if you put in an LED instead of the buzzer?

*These checks for understanding help reinforce the learning of the science skill of applying scientific ideas to design, test, and refine a device that converts energy from one form to another. (NGSS (4-PS3-4) A)

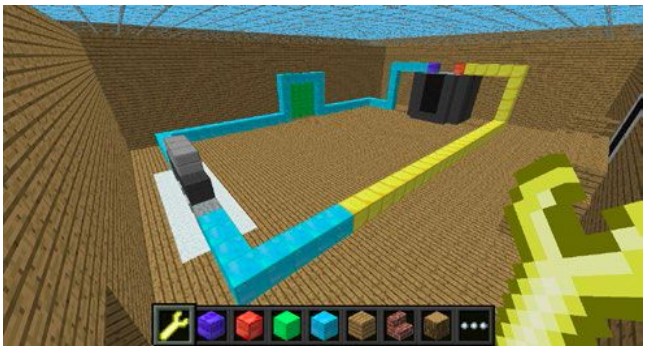
- Students put kit away to avoid distractions during teacher led discussion.

Closing Activity (10-15 minutes):

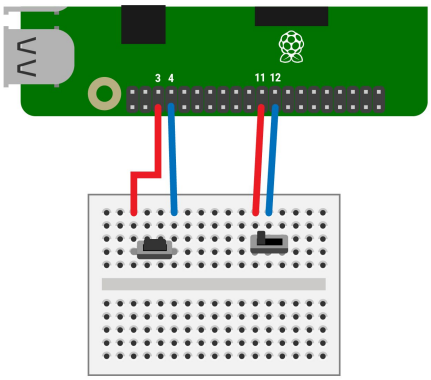
- Teacher led discussion: 2.3 SLIDES - Buttons and Switches. Polarity and Audio output
  - Use these slides to unpack uses of buttons and switches in the outside world and polarity and audio output.

Project Level Guide - Power Plant

Introduction:
Piperbot is now inside a new game world and its a power plant! Turn the power ON to open gates, release water, pump water, raise elevators, and make your way to the next portal. There are two phases to complete this challenge.

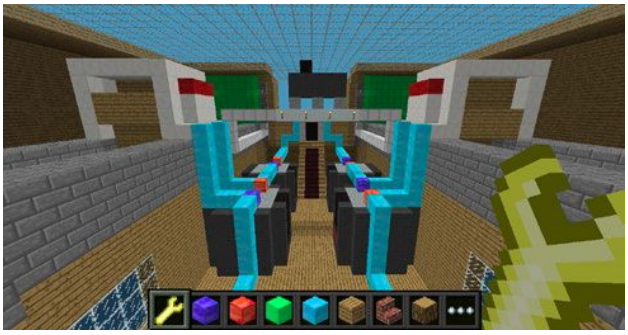


GPIO Setup: Red 3, Blue 4, Red 11, Blue 12



Concept:
This level is a review of buttons and switches. Another concept: hydroelectric power generation can also be introduced (NGSS Secondary school concept). As water falls down, it releases kinetic energy which is transformed by the generator and stored in the battery. The energy can then released to activate gates and elevators.

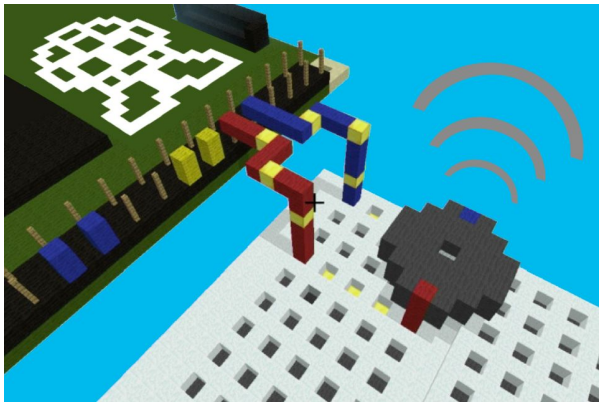
Troubleshooting:
For young people, the biggest pain point is having the manual dexterity to properly seat the switches on the mini breadboards. Bent components may start to be an issue. This is a good opportunity to check for attention to precision when wiring components.



Project Level Guide - Rainbow Bridge

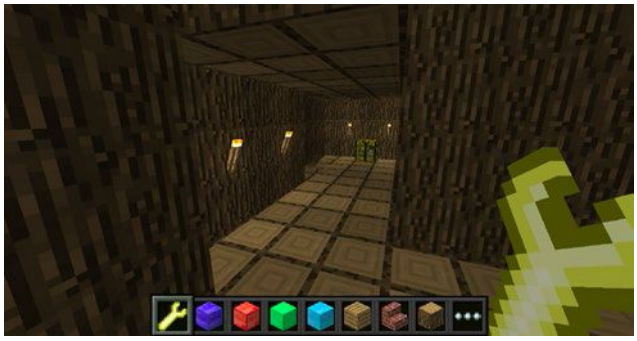
Introduction:
To reach the next portal, Piperbot has to cross a constantly changing rainbow bridge. One random color of the rainbow bridge disappears periodically. To be able to cross, Piperbot will have to enable its buzzer, which hints at which color will disappear next. The number of beeps indicates which color will disappear.

GPIO setup: Red 15, Blue 16



Concept:
This is a review of outputs. A new type of output is introduced with a clicker/buzzer. Another concept to touch upon is component polarity. The positive (red wire) current must go into the long leg of the piezo buzzer and exit through the short leg, through the blue wire and back to the R-Pi. You can draw an analogy to a one way gateway: the piezo buzzer only lets current flow one way.

Troubleshooting:
Polarity is most likely the one issue that may come up. Point to the plus sign on the piezo buzzer; it needs to line up with the red wire and the left-most pin of the pin pair.



Lesson 2.4 - Parallel Circuits

Established Goals

Having reviewed buttons and switches and learned about polarity, students will now look at parallel circuits with buttons and switches.

Learning Objectives

• Understand that parallel circuits, together with buttons and switches can be used to simplify the button setup. Many buttons can share a ground pin.
• Understand that by using switches in combo with buttons in parallel setups, effects can be stacked. Different memory states can be called, or different functions can be activated.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):
• Address misconceptions from previous lesson and tell students they will be playing through 2 more levels of the game today, exploring the many ways to use buttons and switches in Piper, and learning about parallel circuits.

Main Activity (30-40 minutes):
• Allow students to go through the Breadboard Bluffs, Funky Fungi, and Return to Cheeseteroid worlds.
  - Refer to the Project Level Guides for Funky Fungi, Cheeseteroid Return, and Breadboard Bluffs.
• During this time, the teacher should be roaming around the room, asking the essential questions* of this lesson:
  - In Funky Fungi: What is the benefit of the two buttons sharing a pin? How does adding the switch affect the buttons and behavior in the game?
  - In Return to Cheesteroid: How does adding the switch affect the buttons and behaviour in the game? How is this setup different than in Funky Fungi?

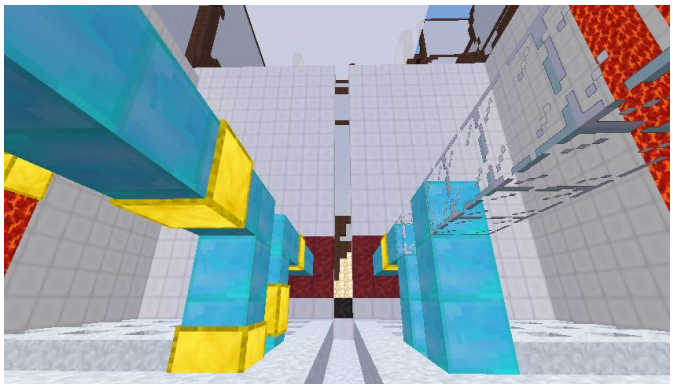
*These checks for understanding help reinforce the learning of the science practices of planning and carrying out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (NGSS 3–5-ETS1-3. (P.E.3.4.7))

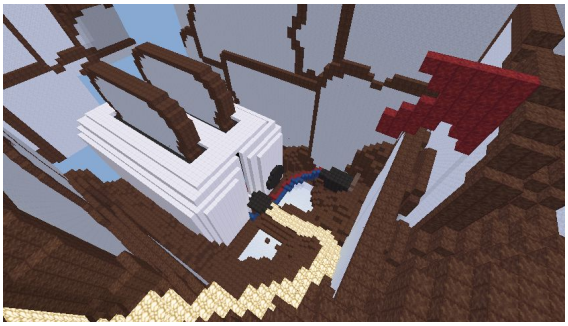
• Students put kit away to avoid distractions during teacher led discussion.

Closing Activity (10-15 minutes):
• Teacher led discussion: 2.4 SLIDES - Parallel buttons and switches
  - Use these slides to unpack Parallel circuits. Allow time for students to ask questions.

Project Level Guide - Breadboard Bluffs

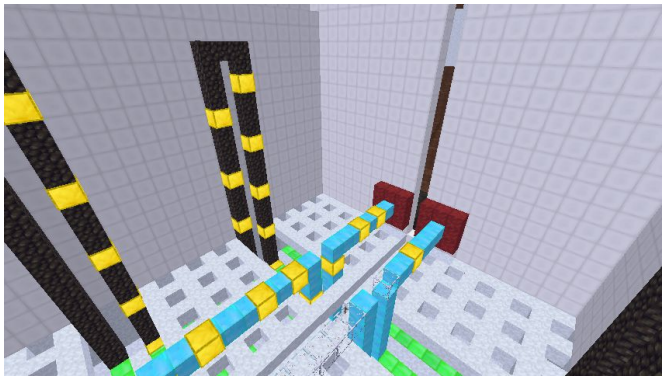
Introduction:
Piperbot now goes inside an actual breadboard! Students will make repairs first to a breadboard, then go inside a toaster to make sure the current is flowing properly. When the toaster is fixed, watch the toast pop!





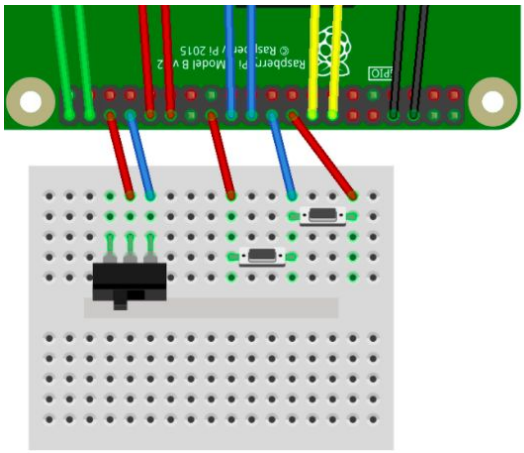
Concept:
This is a deep dive into breadboards and how they work. The toaster provides insight into circuits in a real world environment. By placing students inside a virtual breadboard (and a toaster) and having them make repairs, they will come out understanding how the current flows inside.

Troubleshooting:
There is a glitch in our current game that is being fixed for our next update. There is a video that should play when you complete each of the two levels, but the video is not working and does not allow you to get to the next level. Please watch this video for instructions on how to make it through!



Introduction:

Piperbot is in a magical world of funky fungi and chasms searching for the next portal. This is review of buttons and switches however they are combined now in a new wiring scheme. Two buttons allow you to call a different structure from memory that will allow you to cross the chasm. Eventually, you'll plug in a switch as well, which will let you 'pave' a platform as you move or jump.



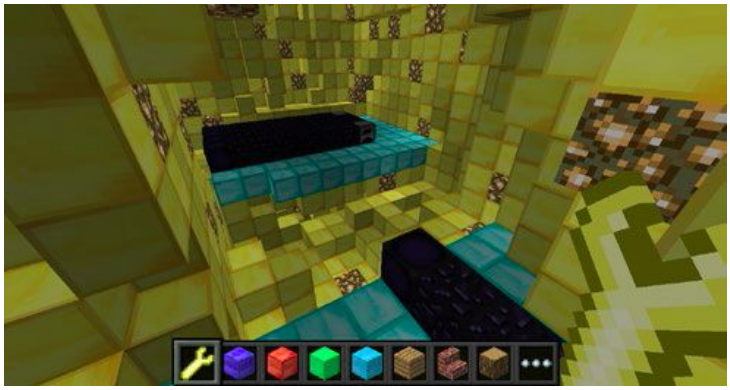
Concept:

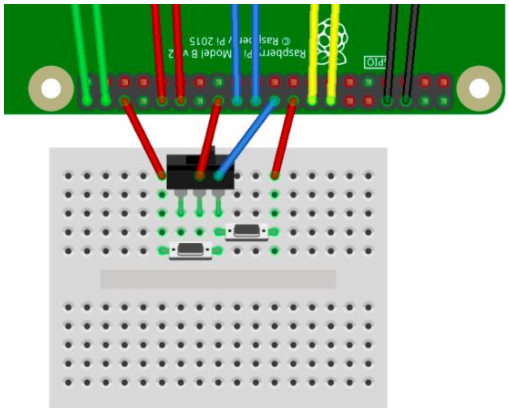
In a parallel setup, the buttons allow you to call different memory states and the buttons to share what is called a ground. A third state, set by a switch, allows you to call from memory continuously instead of on demand.

Troubleshooting:

It can be easy to get lost in this level. After the second crafting table, keep going up to reach the red platform with the portal.

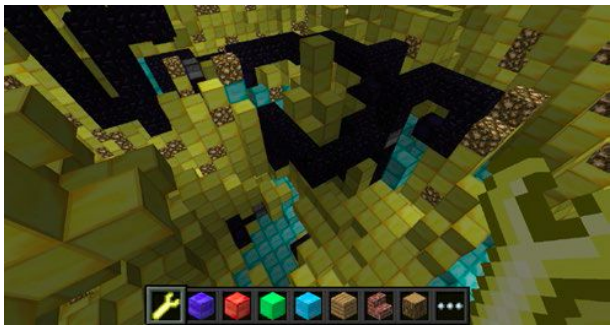
Introduction:
This is the longest and hardest level in the game. Once completed, Piperbot saves earth and the student unlocks creative mode.





Concept:
A similar button and switch setup as the one in Funky Fungi are used in this level. Piperbot will have to alternate between constructing and destroying the cheese world to get through. Instead of memory states, the switch allows the player to switch between on demand construction/destruction and constant construction/destruction.

Troubleshooting:
Having reached the part where you have to see what both power blocks are doing, there is a crack in the geometry that is easy to miss. Otherwise, keep building destroying and alternating from constant and on demand mode until you get through.



Video: Click here to watch the video that appears at the end of Return to Cheeseteroid.

Phase 3 - Code

	Phase	*Tools*	*Students will...*
3	Code	PiperCode > Make	Program basic actions including loops, sequences, and events, using visual programming blocks

California Standards Alignment for Phase 3 Code

These standards apply to all the lessons in this phase, when the learners move into PiperCode projects. Use them with daily or weekly agendas and planning.

Concept and Sub-Concept (s)	Standard
Computing Systems: Devices	CA 3-5.CS.1 Describe how computing devices connect to other components to form a system. (P7.2)
Computing Systems: Hardware & Software	CA 3-5.CS.2 Demonstrate how computer hardware and software work together as a system to accomplish tasks. (P4.4)
Computing Systems: Troubleshooting	3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)
Algorithms & Programming: Algorithms Variables Control Modularity Program Development	3-5.AP.11 Create programs that use variables to store and modify data. (P5.2) 3-5.AP.12 Create programs that include events, loops, and conditionals. 3-5.AP.13 Decompose problems into smaller, manageable tasks which may themselves be decomposed. (P3.2) 3-5.AP.14 Create programs by incorporating smaller portions of existing programs, to develop something new or add more advanced features. (P4.2, P5.3) 3-5.AP.17 Test and debug a program or algorithm to ensure it accomplishes the intended task. (P6.2) 3-5.AP.18 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development.

Practices	P1. Fostering an Inclusive Computing Culture P2. Collaborating Around Computing P4. Developing and Using Abstractions P5. Creating Computational Artifacts P6. Testing and Refining Computational Artifacts

CA NGSS Connections:

(4-PS3-2) &(4-PS4-2) Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat, and electric currents.

(4-PS3-4) Apply scientific ideas to design, test, and refine a device that converts energy from one form to another.

(4-PS4-3) Generate and compare multiple solutions that use patterns to transfer information.

3–5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem (Performance Expectation).;

3–5-ETS1-3. Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (P.E.3.4.7)

Common Core State Standards (CCSS) Connections:

ELA/Literacy -

(CA CCSS for ELA/Literacy SL.3.1, SL.4.1, SL.5.1):(CA CCSS for ELA/Literacy W.3.2, W.4.2, W.5.2)

W.4.7	Conduct short research projects that build knowledge through investigation of different aspects of a topic. (4-PS3-4)
W.4.8	Recall relevant information from experiences or gather relevant information from print and digital sources; take notes and categorize information, and provide a list of sources. (4-PS3-4)

Mathematics -

MP1	Make sense of problems and persevere in solving them.
MP2	Reason abstractly and quantitatively. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)
MP5	Use appropriate tools strategically. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)
MP6	Attend to precision

Lesson 3.1 - Intro to Computational Thinking & Programming

Established Goals

A basic challenge to get students thinking about simple instructions and sequences, or in coding terms, creating algorithms and loops.

Learning Objectives

- Create quick basic commands for real world problems then link to coding concepts.
- Understand computational thinking concepts, including algorithms and loops.
- Review these key *electronics and programming understandings*:
    - Wire and pin positions for specific **inputs and outputs**.
    - Electric flow is **sensed** by the computer hardware and programmed to have an **effect** in software, and thus on the screen.
    - The computer is programmed to **detect** the state of the electricity going into the pin and to **send** a high voltage to the pin when light is desired.

The Lesson (1-Day Plan)

Preparation

- Open a side space in the room for students to move around for first part of the lesson.
- Create 5 Sample everyday activities for the pairs to complete such as peeling a banana, tying a shoe, opening a door, brushing your teeth, or making a peanut butter and jelly sandwich.
    - OPTIONAL: Simplify this activity for younger learners by just having them coach each other on drawing a square on a piece of paper.
- Suggested student to kit ratio is 2:1 up to 3:1.
- Make sure Piper kits are built, connected, functioning, and batteries are charged for the Raspberry Pi and the speaker.
- If previous students have completed projects and you want a new student start from the beginning, click the Reset All Projects button on the top left of the Settings screen (yellow button linked off main menu).

Main Activity - Part 1 (15 minutes)

1. In groups of two, assign one learner to be Ruby “The Programmer” and one to be “The Robot”.
2. Assign each pair an activity (may be written on a piece of paper pulled from a hat)
3. Ask the Programmer to explain to their partner (Robot) how to perform the steps needed to complete their activity using words only (no non-verbal motions link hand movements)!
4. Switch pairs.

5. After the activity, use this as an opportunity to talk about the importance of simple, clear instructions and sequences (if the room is loud, say: "'Head Programmer' says, all Rubies stop!").
6. Congratulate learners for creating their first algorithms and pseudo code!

(Adapted from Teachers Learning Code Getting Started Guide)

Main Activity - Part 2 (20 minutes)

Tour PiperCode (5 minutes):

- Introduce the PiperCode main menu platform and tell students that they will be learning to code the basic functionality of video games and building simple lights and controllers using wires, buttons, LEDs, and breadboards
- Tour of the PiperCode interface: 3.1 SLIDES - Intro Comp Thinking

EXTENSION: Review core terms and "stuff" (aka components) from first Piper lesson Glossary, especially BreadBoard, GPIO Pins, LED, and Buttons by playing Piper Pictionary/Charades - put the glossary terms on slips of paper and have team pull from a hat, giving them 1 minute to draw it or mime the function and the rest of their team has to guess!

Complete Blink project (15 minutes):

- Allow students to go through Blink project steps in their pairs or groups.
- During this time, the teacher should be roaming around the room, asking the *essential questions** of this lesson:

    - Why are the GPIO pins important to interface the chip with the lights as outputs?
    - How do you start and stop the code?
    - Why code a loop to make the light blink? (delve more into why the Repeat Forever code versus just a one-time sequence)
    - Why does writing a code to make the flow of electricity to the LED turn on then off cause a blink?
    - How do you make the light blink faster, slower, longer or shorter? OR How does changing the number of milliseconds (ms) wait change the behavior of your blink?

*These checks for understanding help reinforce learning of the computer science practices of how computer hardware and software work together as a system to accomplish tasks. (CA 3-5.CS.2 (P4.4))

Closing (10 minutes)

Depending on the age of your students and time left, choose one of the Computational Thinking frameworks and introduce the main concepts using slides; Ask students to reflect which of these ideas and practices they acted out while being Ruby or the Robot.

EXTENSION:

- Have students write down their code in their design journals and share with other

groups to execute.
- Choose some sample code to review as a group - do students recognize any patterns? Any ways they could simplify their algorithms?

PROJECT GUIDE: PiperCode Blink

Software Hints:
- All but the first Blink project are locked until the student clicks Next and successfully follows the prompts to build the code of each project.
- The right click menu on the blank Canvas has some handy actions:
    - Undo/Redo actions
    - When the Code Canvas gets crowded, tuck away blocks (Clean up, Collapse/Expand)
    - Add comment to leave yourself a note
- The right click menu on a code block also has shortcuts like:
    - Duplicate - make an exact copy of the selected block or group of blocks to save time
- Click the four arrowed icon on the top right of diagrams and videos to expand to full screen - this is helpful when trying to exactly line up lights and wires to the right pins on the breadboard.
- Under Electronics tab in the Project Panel, you can always reference the GPIO pin map, but you might also want to coach the students to create their own physical "cheat sheet" diagram in their design journal or on sticky labels under the Pi board.
- You need to click the green Start block on the top left of the Block Library to run your code. You should also see the lights on the Raspberry Pi board react.
- To save time, you can right-click on a block in the Code Canvas and duplicate a single or group of blocks as a sort of copy and paste of your code.
- Stop running the code when you're adjusting blocks or variables!
- If you adjust the variables of the code blocks while its running, you have to start/stop to make sure you communicate the new code to the board.
- If your code blocks string gets long:
    - click on the white of the Code Canvas and drag up to move the whole canvas.
    - Use the mouse scroll to zoom in and out.

Troubleshooting:
- A quick refresher and energizer is to have a speed contest of who can set up and power-up the basic components of the Piper kit, and also add a breadboard.
- You also might have to coach some students on how to use a scroll-wheel mouse (as opposed to the track pad or touch screen), including how to left click to select and right click to see an extra menu.

If you have time, have the students stop at the end of every project and draw a diagram of their hardware setup or note any roadblocks and how they troubleshot solutions, or how they might build it differently the next iteration.
- If the LEDs aren't lighting up, make sure the left negative leg matches the negative lead off the GPIO board.
- The colors of the wires don't matter to function but for later projects with more buttons

and lights, it's much better to set the habit of using consistent colors to keep track.

Lesson 3.2 - Loops & Sequences

Established Goals

The goal of this lesson, as the second dive into coding, challenges the students to extend understanding to solve problems by coding behaviors in the language of PiperCode. It includes practice changing the sequence of the lights, based on pivoting essential questions about a real life situation, a local intersection stoplight.

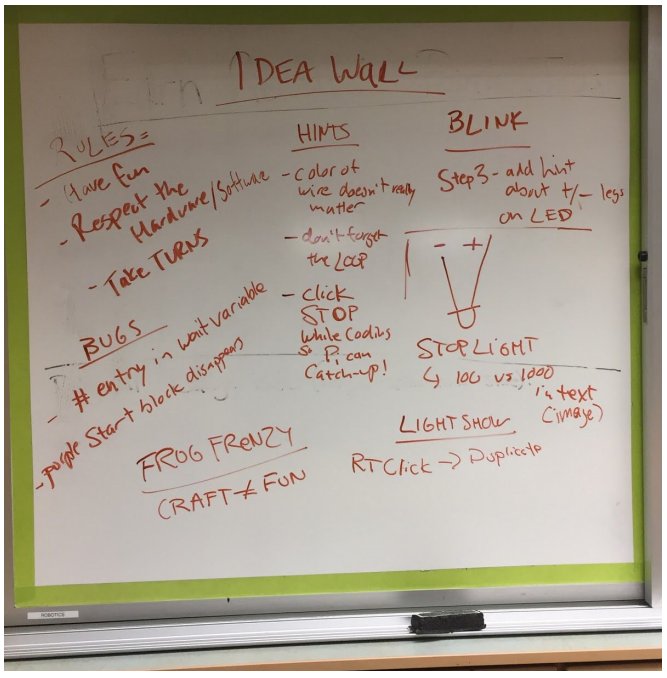
Learning Objectives

- Practice breadboarding and wiring
- Review and understand *computational concepts* of:
  - Loops: running the same sequence multiple times.
  - Sequence: identifying a series of steps for a task
- Demonstrate computational thinking core concepts, including:
  - *Algorithm Design* by creating an ordered series of instructions for solving similar problems or for doing a task, such as turning a light off and on in the right order.
  - *Simulation* by developing a program to imitate real-world process of a stoplight.

Preparation

- Make sure Piper kits are built, connected, functioning, and batteries are charged for the Raspberry Pi and the speaker.

The Lesson (1-Day Plan)



Introduction (5 minutes):

- Start an Idea Wall, either on a white/chalk/Smart board or a shared document, with sections for students to document and share their ideas, key discoveries, questions, hints, etc. with their classmates.
- HINT: Seed each sections with some reflections from the last session's closing or your own notes, but be deliberate to use the language and terms of your local students.

Main Activity (30-40 minutes):

- Allow students to go through Stoplight project steps in their pairs or groups.

- During this time, the teacher should be roaming around the room, asking the *essential questions** of this lesson:

  In Stoplight project:
  - How is the sequence more complicated in this project than Blink?
  - How can you simulate a real-life stoplight (start with green, go to yellow, rest on red)? (answer = wait variable time!)
  - How might we adjust the wait times to make your stoplight safer for pedestrians or cyclists?
  - EXTENSION: How might we create a two-way stoplight? (answer = add a breadboard and run parallel block code sequences.)

* These checks for understanding help reinforce learning of computer science skills such as creating programs that use include loops, sequences; decompose problems into smaller tasks; or incorporate smaller portions of existing programs, to develop something new or add more advanced features. (CA CS 3-5.AP.12 through 14)

Closing Activity (10-15 minutes):

- Students put kit away to avoid distractions during teacher led discussion.
- Teacher led discussion: 3.2 SLIDES - Loops and Sequences
- Review major concepts and link to when they learned them while building PiperCode, such as: How do Loops and Sequences build further on the foundations of computation?
- [OPTIONAL]: Have students add more to the Idea Wall, or perform a task, such as formative assessment using Google Draw or the whiteboard or a personal journal to draw a map or write in Pseudocode, the basic logic of each of the project's codes.
- Quick reflection exit slip survey (optional). Use this to get feedback from students of their understanding.

Lesson 3.3 - Events

Established Goals

The goal of this lesson is for students to refresh their learning of Piper hardware and electronic components and dive into computational tinkering! This third project in the initial Learn PiperCode cluster challenges the students to practice key electronic and programming skills by orchestrating cause and effect.

Learning Objectives

- Practice breadboarding and wiring
- Understand *computational concept* of:
    - events: one thing causing another thing to happen.
- Explore *computational thinking design practices* including:
    - *experimenting and iterating*: developing a little bit, then trying it out, then developing more.
    - *testing and debugging*: making sure things work, and finding and solving problems when they arise.

Introduction (5 minutes):

- Revisit the Idea Wall, either on a white/chalk/Smart board or a shared document, with sections students to document and share their ideas, key discoveries, questions, hints, etc. with their classmates.
- Tease out the main concepts and design practices or ask students to explain more to the whole group if just fragments.
- Remind students of the rules around troubleshooting on their own or asking a partner, before asking for help from an adult (SAY: ask three before me!)

Main Activity (30-40 minutes):

- Allow students to go through the Light Show project steps in their pairs or groups.

During this time, the teacher should be roaming around the room, asking the ***essential questions**** of this lesson:

- How does the block you attach to "If", effect the block attached to "do"? (answer = you're coding an event, where one circumstance triggers a light to act differently)
- OR talk me through your code step-by-step in regular English.
- If your code is not working or your lights are not behaving the way you intended, how do you fix it?

* These checks for understanding help reinforce learning of computer science skills such as creating programs that use include events, while testing and debugging a program or algorithm to ensure it accomplishes the intended task. (CA CS 3-5.AP.12 & 17)

Closing Activity (10-15 minutes):

- Students put kit away to avoid distractions during teacher led discussion.
- Teacher led discussion: 3.3 SLIDES - Events
- Review major concepts and link to when they learned them while building PiperCode, such as: How do Loops, Sequences, and Events build the foundations of computation?

- [OPTIONAL]: Have students perform a task, such as add more to the Idea Wall or a formative assessment using Google Draw or the whiteboard or a personal journal to draw a map or write in Pseudocode the basic logic of each of the project's codes.
- Quick reflection exit slip survey (optional). Use this to get feedback from students on their understanding.

Lesson 3.4 OPTIONAL: Light and Sound Inventions

Established Goals

Build a random button controller for a custom made game with 4 different colored LED lights and a buzzer.

Learning Objectives

- Practice coding loops.
- Review these key electronics understandings:
  - Binary state of a button as an input.
  - Sounds as auditory outputs.
- Practice computational concepts of
  - loops: running the same sequence multiple times.
  - events: while a pin's condition is on or off, another action happens.

Preparation

- Have the craft materials ready, but hold them until the students have watched the videos, then build the code, and the basic breadboard.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):

- Students watch video in project panel (2 minutes) for Frog Frenzy
- Tour of the PiperCode interface

Main Activity (30-40 minutes):

1. Allow students to go through Frog Frenzy project steps 1-8 to build the input/outputs and the base code.
2. As students hit step 9, hand out craft materials and encourage them to either copy but preferably iterate designs for a character.

During this time, the teacher should be roaming around the room, asking the *essential questions** of this lesson:

- ★ What if we moved the lights or button to other spots on the breadboard?
- ★ How is the code for Frog Frenzy different than the first three projects? (answer= one input (button) and 2 outputs (2 LEDs)?
- ★ If the pin (37) that the button is on/set to is off, what happens?
- ★ How do you win or beat the game? How is this commanded steps laid out in the

blocks?

★ How is the game play defined by the code and timing of the button?
★ What is the significance of the "while" part of the second type of forever block?
★ Name 2 sequences you see in this code? How do they command actions in the LEDs?

* These checks for understanding help reinforce learning of computer science skills such as creating loops, but also incorporating smaller portions of existing programs to develop something new or add more advanced features. (CA CS 3-5.AP.12 & 14). You can also tie back to science concepts of sound, energy transfer, patterns for transferring information, and testing prototypes (NGSS (4-PS4-2 through 4).

Software Hints:

- When you click Start and run the code, you'll see when PiperCode is running each line; and if you open the Electronics tab, you'll also see the pins light up too!

Facilitation NOTES:

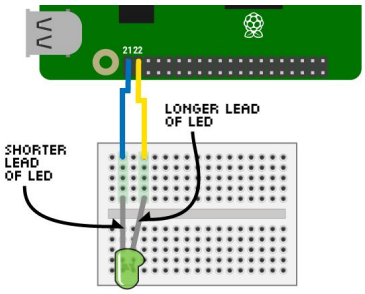
- This project flows differently than the first 3 projects - it starts with video tutorials and an existing code and prompts students to hack backward to understand the concepts behind each line of code.
- Coach the students to be creative with designing their character, but don't let them speed through it or get stuck in finer details - you want to make sure to scaffold their learning pathway toward both learning the coding but also getting creative connecting it to their personal interests.

Closing Activity (10-15 minutes):

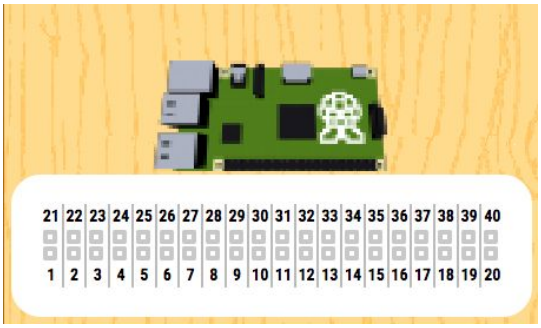
- Teacher led discussion: Slides - How do loops and events work?
- Have students perform a task or formative assessment to document Frog Frenzy as a game in their design journal, and brainstorm ideas to remix either the code to have the game play differently or the narrative interface of a different story involving two different colored light.
- Quick reflection exit slip survey (optional). Use this to get feedback from students of their understanding.

PiperCode is series of projects that spark coding and continue computer engineering into basic micro-controlling. Listed below are quick descriptions of all of the projects.

Blink

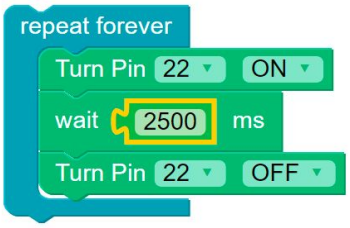


Introduction:
This is the introductory project to learn programming where the students will build and program a LED light to blink.

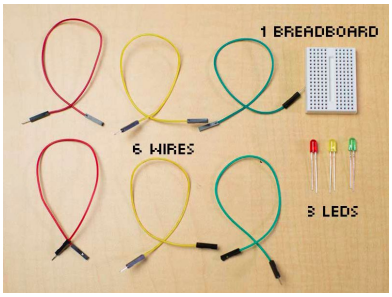


Concepts:
Students will recall from Story Mode (Phase 2), that GPIO pins on the Raspberry Pi have a number. They will need to program the GPIO pins on their breadboard with the LED light output so that the LED's respond to the code. Then they will program the light to turn on, wait, then turn off, in a loop that repeats forever.

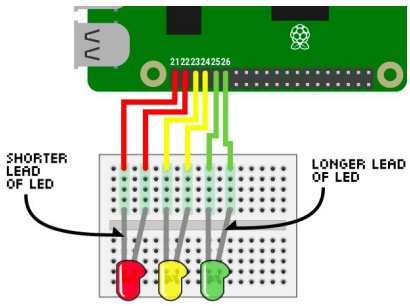
Troubleshooting:
The most common mistakes students are to have the incorrect pin number in the code. Also make sure that they are first turning ON the pin, then turning it OFF.



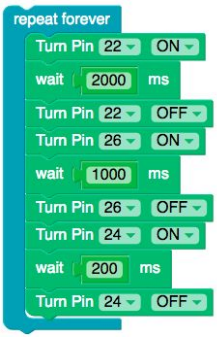
Stoplight



Introduction:
In this project, students will build three LED lights and program them to light up in a sequential order similar to a traffic stoplight.

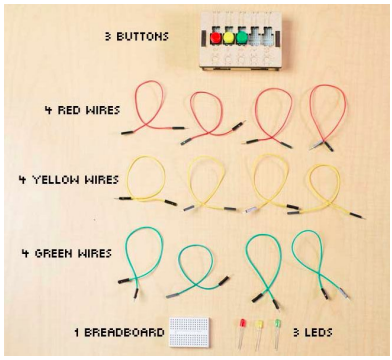


Concept:
In this project, students will use their understanding of the concept of sequences in loops to form algorithms and automation, while practicing testing and debugging.

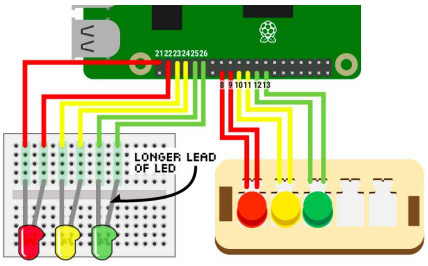


Troubleshooting:
Similar to Blink, most students will make mistakes with the pin numbers. Note that a stop light goes from red to green to yellow pins (22, 26, 24), students will instead program red to yellow to green since those are sequential pin numbers (22, 24, 26).

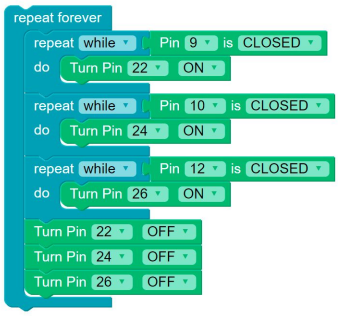
Light Show



Introduction:
Students have learned how to build buttons and LED lights, now they will learn how to program buttons to turn an LED light on and off.



Concept:
In Blink and Stoplight students learned how to program an event output in the form of an LED. With Light Show, they will now program inputs (buttons) to outputs (LEDs).



Troubleshooting:
Similar to Blink, most students will make mistakes on the pin numbers. Note that a stop light goes from red to green to yellow pins (22, 26, 24), students will instead program red to yellow to green since those are sequential pin numbers (22, 24, 26).

Phase 4 - Extend

	Phase	*Tools*	*Students will...*
4	Extend	4.1 Power-Up - Play 4.2 Power-Up - Design 4. 3 Create Creative Mode PiperCode > Build Games PiperCode My Projects	Use Story Mode MiniGames to extend their play and design by creating music, designing their own avatar, and designing 3D models. Design and program their own Creative Mode and PiperCode projects, including games, PiperCode visual programs, and Python code.

California Standards Alignment for Phase 4 Extend

These standards apply to all the lessons in this phase, when the learners move into Piper Extend projects. Use them with daily or weekly agendas and planning.

Concept and Sub-Concept (s)	Standard
Computing Systems: Devices	CA 3-5.CS.1 Describe how computing devices connect to other components to form a system. (P7.2)
Computing Systems: Hardware & Software	CA 3-5.CS.2 Demonstrate how computer hardware and software work together as a system to accomplish tasks. (P4.4)
Computing Systems: Troubleshooting	3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)
Algorithms & Programming: Algorithms Variables Control Modularity Program Development	3-5.AP.11 Create programs that use variables to store and modify data. (P5.2) 3-5.AP.12 Create programs that include events, loops, and conditionals. 3-5.AP.13 Decompose problems into smaller, manageable tasks which may themselves be decomposed. (P3.2)

	3-5.AP.14 Create programs by incorporating smaller portions of existing programs, to develop something new or add more advanced features. (P4.2, P5.3) 3-5.AP.17 Test and debug a program or algorithm to ensure it accomplishes the intended task. (P6.2) 3-5.AP.18 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development.
Practices	P1. Fostering an Inclusive Computing Culture P2. Collaborating Around Computing P4. Developing and Using Abstractions P5. Creating Computational Artifacts P6. Testing and Refining Computational Artifacts
CA NGSS Connections: (4-PS3-2) &(4-PS4-2) Make observations to provide evidence that energy can be transferred from place to place by sound, light, heat, and electric currents. (4-PS3-4) Apply scientific ideas to design, test, and refine a device that converts energy from one form to another. (4-PS4-3) Generate and compare multiple solutions that use patterns to transfer information. 3–5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem (Performance Expectation). 3–5-ETS1-3. Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (P.E.3.4.7)	

Common Core State Standards (CCSS) Connections:

ELA/Literacy -

(CA CCSS for ELA/Literacy SL.3.1, SL.4.1, SL.5.1):(CA CCSS for ELA/Literacy W.3.2, W.4.2, W.5.2)

W.4.7	Conduct short research projects that build knowledge through investigation of different aspects of a topic. (4-PS3-4)
W.4.8	Recall relevant information from experiences or gather relevant information from print and digital sources; take notes and categorize information, and provide a list of sources. (4-PS3-4)

Mathematics -

MP1	Make sense of problems and persevere in solving them.
MP2	Reason abstractly and quantitatively. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)
MP5	Use appropriate tools strategically. (3-5-ETS1-1),(3-5-ETS1-2),(3-5-ETS1-3)
MP6	Attend to precision

Lesson 4.1 - Power-Up: Play

Established Goals

The goal of this phase is to empower students to begin elaborating on their first stages of learning in engineering, computer literacy, design, coding, and programming. In this lesson, when they have finished other Story Modes early, students can play further with the STEM concepts first learned by responding to challenges, by extending to create their own media and designs.

Learning Objectives

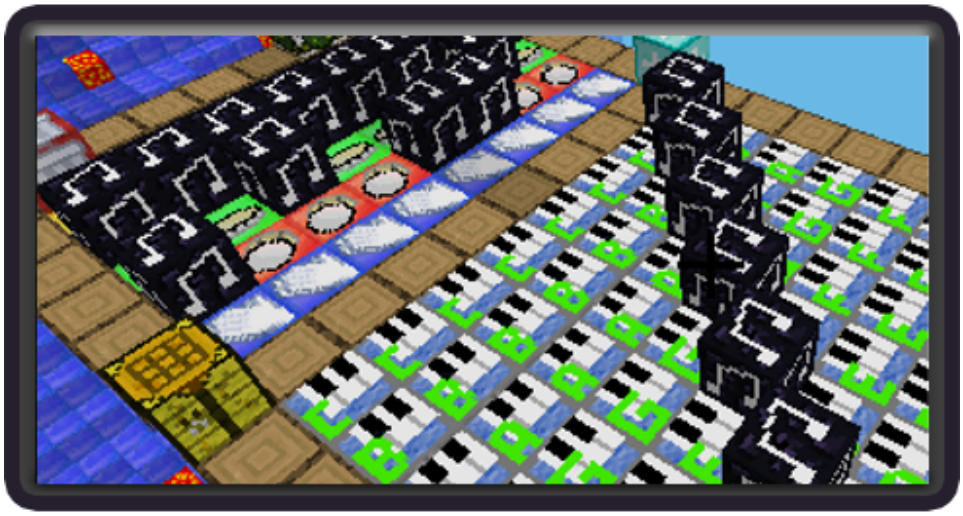
- Use Story Mode MiniGames to extend their play.
- Apply science and computer science concepts from the preceding phases.
- Create programs by incorporating smaller portions of existing programs, to develop something new or add more advanced features.
- Foster inclusive computing culture, by engaging learners around their personal interest in music.

NOTE: MiniGames may be utilized for your faster students who go through the main projects quicker than the other students or teams. Have them play a Mini Game and further practice building electronics.

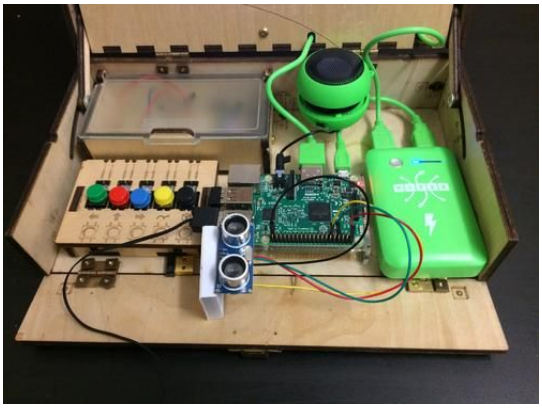
Introduction (5 minutes):

Explain:

> As you might have noticed as you played through the Story Mode levels, MiniGames unlock. MiniGames are represented by moons next to the planets and may be unlocked after completing the main project.
>
> Each mini game consists of building a "power-up" or electronic component using the mini breadboard in your Piper Computer Kit. Instead of solving a challenge and moving to the next level, in MiniGames you may play over and over and try to beat your last score or time.



For example, If you want to try PiperCoding Project: Make music with the wave of a hand!, students can build a theremin - if you have wifi access and comfort level using the terminal interface on a Raspberry Pi and purchase an additional HC-SR04 Ultrasonic Sensor.



Lesson 4.2 - Power-Up: Design

Established Goals

The goal of this phase is to empower students to begin elaborating on their first stages of learning in engineering, computer literacy, design, coding, and programming. In this lesson, if they have finished other Story Modes early, students can play further with the STEM concepts first learned by responding to challenges, by extending to create their own media and designs.

Learning Objectives

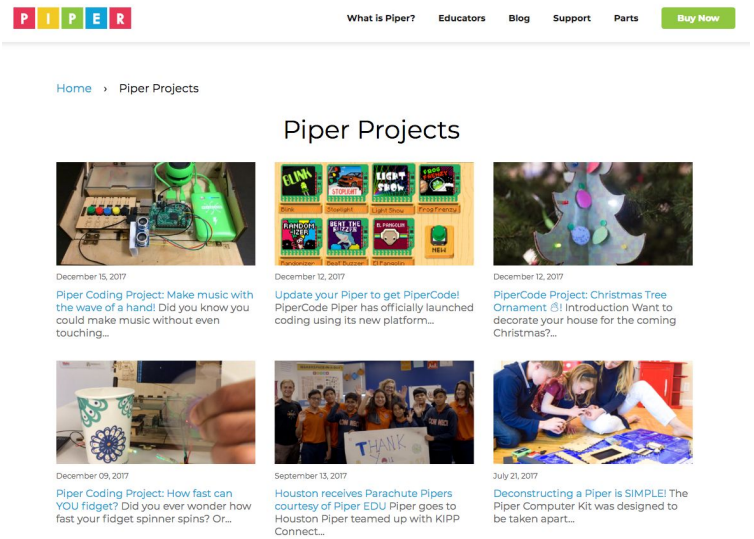
- Use Story Mode MiniGames to extend their design skills by creating music, designing their own avatar, and designing musical instruments, interactive art, and 3D prints.
- Foster inclusive computing culture, by engaging learners around their personal interest in art and design.
- Create, test and refine computational artifacts through physical computing.

NOTE: Need help getting started facilitating open ended project based learning through making? Check out the resources through Maker Ed http://makered.org/resources/projects-learning/ or http://makered.org/resources/professional-development/.

Also, many of these projects will necessitate you purchasing arts & crafts or other Raspberry Pi additions or using 3D printers or other digital fabrication tools.

Activity (usually 2-4 instructional hours): Student led project based learning!

Student follow Piper blog post instructions to try designing their hacks, such as a holiday tree, a 3D printed car, or an internet of things gadget!



See links to Piper Projects blog:

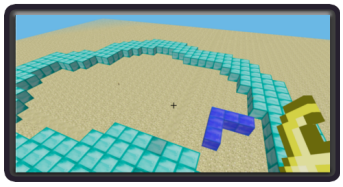
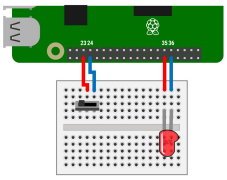
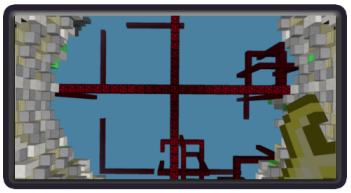
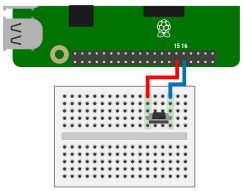
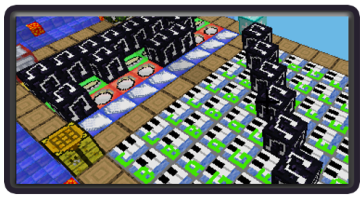
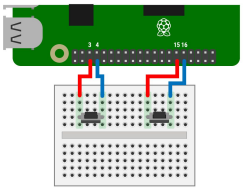
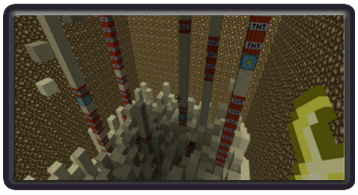
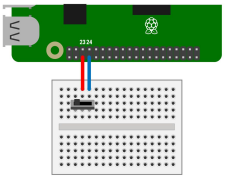
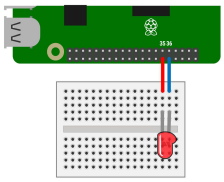
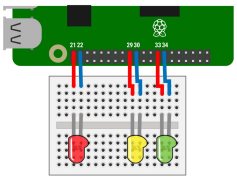
How to Build Your Own Car Robot - Part1: Build Chassis

How to Build Your Own Car Robot - Part2: Enable movement

How to Build Your Own Car Robot - Part 3: Control and programming

Turn your Piper into an Amazon Echo

Reflection (15-30 minutes): Have the students create project portfolio blog posts or provide a maker journal project template.

Mini Game	*To Unlock*	*Game Image*	*GPIO Setup*
Snake Trap	Mars		
Ring Race	Cheeseteroid		
Pip Hop	Chain Reaction		
Explosive Escape	Treasure Hunt		
Dark Maze	Treasure Hunt		
Chest Quest	Funky Fungi		

Lesson 4.3 - Create

Established Goals

The goal of this phase is to empower students to begin elaborating on their first stages of learning in engineering, computer literacy, design, coding, and programming.

Learning Objectives

- Use Creative Mode to design and test games and controllers.
- Build games and controllers in PiperCode to…
    - Create games and other physical computing projects.
    - Move from one coding language to another (visual to Python).
    - Learn the basics of event handling in coding.
    - Practice computational and design thinking.

NOTE: Need help going deeper with facilitating open-ended project-based learning through design thinking and challenge-based learning? Check out the resources through Digital Promise's Maker Promise Leadership initiative:
http://digitalpromise.org/maker-leadership/teaching-and-learning/

Activity (usually 4-6 instructional hours):

Some personal student-led projects* ideas may include:

- Create your own interactive games and controllers from scratch
- Design and test their own new PiperCode projects from scratch, using either PiperCode visual programming blocks or Python code

*Facilitating and guiding learners through documentation and reflection of these personal projects helps reinforce computer science learning around determining potential solutions to solve simple hardware and software problems using common troubleshooting strategies, as well as the practices of creating, testing, and refining computational artifacts, after developing and using abstractions (CA CS 3-5.CS.3 P6.2; P4 through 6)

By extending into programming PiperCode games, students can complete projects that explore deeper computational thinking concepts and skills.

		Concept	*Students will...*
	Frog Frenzy	Loops, Sound Outputs	Design a character with LED eyes and sounds
	Randomizer	Conditionals	Build a board game
	Beat' Buzzer	Complexity	Code a multiplayer game
	El Pangolin	Data	Collect data based on a new button input
	My Projects > NEW	Computational thinking design practices	Design and test their own new PiperCode projects from scratch, using either PiperCode visual programming blocks or Python code

Phase 5 - Deconstruct

	Phase	*Tools*	*Students will...*
5	Deconstruct	Kit	Take apart the Piper kit and reflect on parts and processes.

California Standards Alignment for Phase 5 Deconstruct

These standards apply to all the lessons in this phase, when the learners move into PiperCode projects. Use them with daily or weekly agendas and planning.

Concept and Sub-Concept (s)	Standard
Computing Systems: Devices	CA 3-5.CS.1 Describe how computing devices connect to other components to form a system. (P7.2)
Computing Systems: Hardware & Software	CA 3-5.CS.2 Demonstrate how computer hardware and software work together as a system to accomplish tasks. (P4.4)
Computing Systems: Troubleshooting	3-5.CS.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)
Algorithms & Programming: Program Development	3-5.AP.18 Perform different roles when collaborating with peers during the design, implementation, and review stages of program development.
Practices	P1. Fostering an Inclusive Computing Culture P2. Collaborating Around Computing P4. Developing and Using Abstractions P5. Creating Computational Artifacts P6. Testing and Refining Computational Artifacts
CA NGSS Connections: ((4-PS3-4) Apply scientific ideas to design, test, and refine a device that converts energy from one form to another. (4-PS4-3) Generate and compare multiple solutions that use patterns to transfer information.	

3–5-ETS1-2. Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem (Performance Expectation). 3–5-ETS1-3. Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved. (P.E.3.4.7)
Common Core State Standards Connections: *ELA/Literacy -* SL.4.5 Add audio recordings and visual displays to presentations when appropriate to enhance the development of main ideas or themes. *(4-PS4-2)*

Lesson 5.1 - Take Apart and Reflection

Established Goals

Throughout this unit, students have explored and learned many basic concepts of electronics. To tie the unit up, students learn to take apart their Piper kits, evaluate their efforts,collaborate, and reflect on their learning. These activities serve as a summative assessment.

Learning Objectives

- Reflect on a good maker mindset, including recycling, reusing, and conserving resources
- Practice and reflect upon techniques for IT troubleshooting.
- Design and implement organizational systems.
- Classify parts by shape and function for reuse.
- Learn to think like an engineer who prototypes ideas, tests, then reflects on challenges and opportunities.
- Practice communication and teamwork skills in constructive critique, iterative feedback loops for better designs, and personal reflection.

The Lesson (1-Day Plan)

Introduction (2-5 minutes):

• Address misconceptions from previous lesson and announce that today they will be taking apart the kits and evaluating their work in the unit.

Main Activity (30-40 minutes):

• To understand the best practices to take apart the kit, take students through basic steps to disassemble the kit including:

1. *Do not unscrew the latch. This piece is not designed for reuse in the building process.*
2. Take all the electronic components out of the case.
3. Unscrew hinge stops on both sides.
4. Unscrew hinges from the bottom case.
5. Unscrew display screws from the case, but not the display from the flat board.
6. Except for the latch and four acrylic plate screws, unscrew all hardware on display case.
7. Gently take wood off the top of the case apart.
8. Unscrew bottom case and gently take wood apart.
9. Neatly sort and store all the pieces.

Go through 5.2 SLIDES - Piper Kit Take Apart or let students go through them on their own, reference the Inventory later in this document.

Closing/Reflection Activity (10-15 minutes):

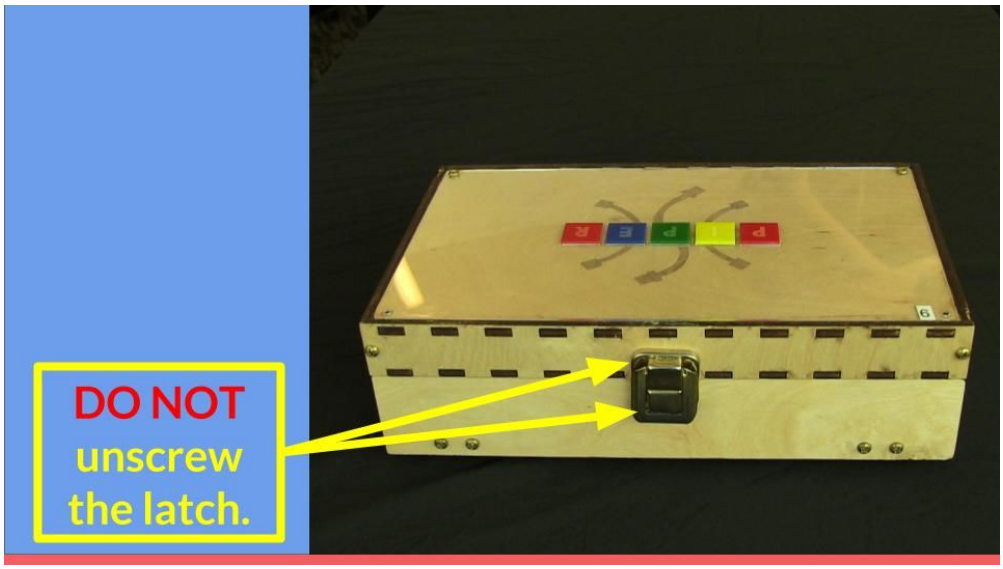
- Have students complete end of unit reflection individually through the Post Survey or for their team through the Team Evaluation - Effort & Participation survey.
- [OPTIONAL Reflection Activity] Assess conceptual development (new knowledge) - Have students draw on a sheet of paper what they think the components of a computer are. Pair share with a partner. Then call on a few teams to share out. *
- Unless there is something else planned for your class, give students free time or have students share some of their reflection points.

* These checks for understanding help reinforce AGAIN all the learning of computer science skills from the earlier phases, but adds a new layer of teamwork and troubleshooting, especially if you give the learners time constraints and make them work in teams to put the deconstructed kits away using a plan and organizational scheme they devise themselves!

During the final reflections, make sure to have them articulate how:

- computing devices connect to other components to form a system.
- computer hardware and software work together as a system to accomplish tasks.
- they determined potential solutions to solve simple hardware and software problems using common troubleshooting strategies.
- performed different roles when collaborating with peers during the design, implementation, and review stages of program development.

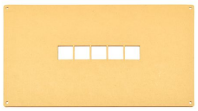
(CA 3-5.CS.1-3 (P7.2)(P4.4)(P6.2) then 3-5.AP.18)

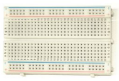


REFERENCES

Piper Computer Kit - Inventory Checklist

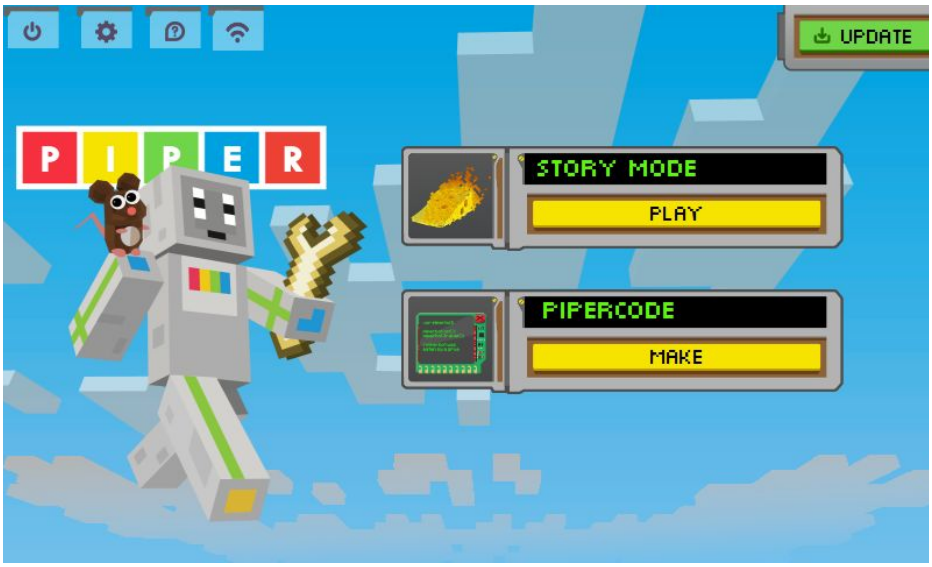
Use this inventory checklist to make sure your Piper Computer Kit is complete after disassembly!

ITEM	QUANTITY	IMAGE
Wood parts: 1A,1B (x2), 1C (x2), 1D, 2A, 2B, 2C, 2D, 2E, 2F, 4A, 4B, 4C, 4D, 4E, 5A, 5B, 5C,5D, 5E	24	
Top - Clear Acrylic	1	
PIPER Colored Letters	1 set	
A - Nuts	31	
B - Medium Screws	15	
C - Small Screws	16	
D - Latch Top, Bottom & Screws	1	

E - Swing Arm screw and nut	2	
F - Hinges	4	
Blueprint	1	
Welcome card	1	
Cables (Screen USB power Raspberry Pi USB power, screen HDMI)	3	
Large Breadboard	1	
Mini Breadboards (White/Yellow)	2	
Buttons (red, yellow, green, blue, black)	1 set	
LED Lights	5	
Electronic accessories (buttons, switches, buzzers)	9	

Wires (red, blue, yellow, green, black)	25	
Raspberry Pi 3	1	
Battery	1	
Speaker	1	
Screen	1	
Mouse	1	
Screwdriver	1	

Piper Computer Kit Settings



The main screen is our Piper User Interface.

When connected to the internet, click the update button to download the latest Piper software

Power icon

- Exit to desktop
- Reboot
- Shutdown

Options icon

- Sound Volume: on/off and adjust volume
- Subtitles: on/off
- Language: English, Spanish, Japanese, Chinese (Mandarin)
- Reset Play Data: Resetting play data will lock levels

Wireless Internet Icon

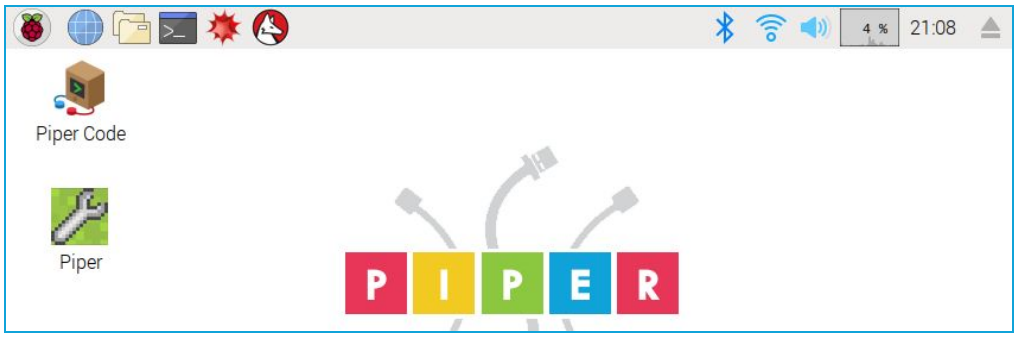
- Tool to select a wireless network to connect to via Wifi

Support Icon

- When connected to Internet, link goes to Piper support page

Using the Piper Computer Desktop

By clicking on the power icon, then selecting "Exit to desktop" you will now open the Piper Desktop. Piper is a fully functioning computer! The Piper Desktop will look similar to the desktop of a PC. Piper is pre-loaded with lots of great content!



Programming - BlueJ Java IDE - Greenfoot Java IDE - Python (2 and 3) - Scratch - Sonic Pi - Wolfram	Libre Office - Base - Calc - Draw - Impress - Math - Writer
Internet - Claws Mail - Raspberry Pi Resources - Web Browser	Games - Piper - Minecraft Pi - Python Games
Other - Debian Reference - Piper Support - Raspberry Pi Help - The Magpi	Accessories - Archiver - Calculator - File Manager - Image Viewer - Keyboard - PDF Viewer - Task Manager - Terminal - Text Editor
Preferences - Appearance Settings - Audio Device Settings - Keyboard and Mouse - Main Menu Editor - Raspberry Pi Configuration	

How to Lock Story Mode Levels and Reset Play Data

If you would like to reuse a Piper Computer for a new team of students, we recommend resetting the play data so the levels are locked. This will allow the student to progress from simple to more complex projects. Here are step-by-step instructions to Lock Levels and Reset Play Data:

1. From the main Piper screen, click on the SETTINGS icon (2nd icon from left)
2. Click the icon to RESET PLAY DATA
3. Click SAVE

How to Unlock All Story Mode Levels

If you have multiple classes using the same Piper kits, it may be advantageous to unlock all levels. The levels are locked initially so the student completes a simple level before moving to a more complex one. Here are step-by-step instructions on how to unlock all Story Mode levels:

1. On the main Piper screen, click the POWER icon in the top left corner
2. Select "Exit to desktop"
3. Click the Raspberry Pi logo in the upper left corner and select ACCESSORIES and then FILE MANAGER
4. Double click the PIPER folder
5. Double click the GAME folder
6. Double click the USER DATA folder
7. Double click the PLAYDATA folder
8. Double click the LEVELSCOMPLETED.TXT file
9. With your mouse copy the number "1"
10. With your mouse, highlight each number "0" and then PASTE the number "1" in its place
11. Do that for all zeroes so they are now all ones
12. From the top menu, select FILE then SAVE
13. Exit using the X at the top right corner
14. All levels in Story Mode will now be unlocked!

How to Reset PiperCode Projects or Unlock all Projects

Resetting PiperCode projects or Unlocking PiperCode Projects is simple. From the main PiperCode menu, select SETTINGS then you will see options to RESET ALL PROJECTS or UNLOCK ALL PROJECTS.

More on Standards & Frameworks
From Computer Science to Computational Tinkering

Computational thinking is a relatively new term, first defined by Jeannette Wing in 2006, that is different than computer science. Defined plainly in a 2016 European Commission report,

> "Computational Thinking is a specific type of problem-solving that entails distinct abilities, e.g. being able to design solutions that can be executed by a computer, a human, or a combination of both."

Google's Exploring Computational Thinking core concepts are defined as:

- *Abstraction* is identifying and extracting relevant information to define main idea(s).
- *Algorithm Design* is creating an ordered series of instructions for solving similar problems or for doing a task.
- *Automation* is having computers or machines do repetitive tasks:
    - *Data Collection* is gathering information.
    - *Analysis* is making sense of data by finding patterns or developing insights.
    - *Representation* is depicting and organizing data in appropriate graphs, charts, words, or images.
- *Decomposition* is breaking down data, processes, or problems into smaller, manageable parts.
- *Parallelization* is simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal.
- *Pattern*
    - *Generalization* is creating models, rules, principles, or theories of observed patterns to test predicted outcomes.
    - *Recognition* is observing patterns, trends, and regularities in data.
- *Simulation* is developing a model to imitate real-world processes.

Computational Thinking for Creative Learning

The MIT Media Lab / Harvard Graduate School of Education materials for Scratch and Google for Education Exploring Computational Thinking parse out computational concepts and mindsets necessary to succeed in computational thinking through creative design practices:

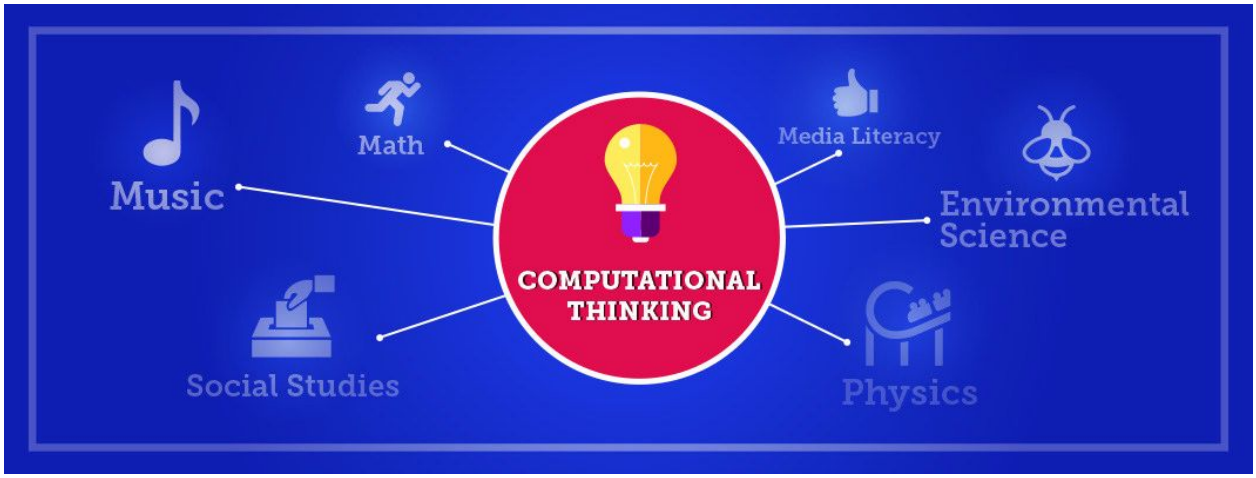
Design practices:

- **experimenting and iterating**: developing a little bit, then trying it out, then developing more
- **testing and debugging**: making sure things work, and finding and solving problems when they arise
- **reusing and remixing**: making something by building on existing projects or ideas
- **abstracting and modularizing**: exploring connections between the whole and the parts

Computational Thinking

Computational thinking (CT) is a problem solving process that includes a number of characteristics, such as logically ordering and analyzing data and creating solutions using a series of ordered steps (or algorithms), and dispositions, such as the ability to confidently deal with complexity and open-ended problems. CT is essential to the development of computer applications, but it can also be used to support problem solving across all disciplines, including math, science, and the humanities. Students who learn CT across the curriculum can begin to see a relationship between subjects as well as between school and life outside of the classroom. (Google for Education).

- Variable: stores a piece of information i.e. score of a game that increases by 1 value for each goal
- Array: allows you to store more than just one piece of information
- Function: a type of procedure or routine that performs a distinct operation. There are often 'canned' functions that exist already like the 'jump' block
- Sequence: identifying a series of steps for a task. Computers and visual programming languages like PiperCode and Scratch read and perform commands in order from top to bottom.
- Events: one thing causing another thing to happen i.e. 'when clicked' block
- Conditionals: making decisions based on conditions i.e. if some condition is met do something, else do nothing or something else
- Loops: running the same sequence multiple times, i.e. repeat or forever blocks
- Boolean Logic: and, or, not are examples of boolean logic; they are values that can be either true or false
- Parallelism: making things happen at the same time
- Operators: mathematical and logical expressions i.e. X+X block
- Debugging: finding problems in code and solving them
- Remixing: taking an existing project or idea and making it new by changing or adding to it
- Modularizing: exploring connections between the whole and the parts
- Syntax: the spelling or grammar of a programming language. PiperCode and Scratch's blockly structure removes the need for syntax
- Algorithm: a step-by-step set of operations to be performed to help solve a problem



Digital Promise
Accelerating Innovation in Education

NEXT GENERATION SCIENCE STANDARDS

CSforALL

Computational Tinkering

Recently, a group of informal educators and researchers led by San Francisco's The Exploratorium Tinkering Studio further adapted the computational thinking frameworks to incorporate the qualities of "tinkering", through physical computing and hands-on engineering projects in makerspaces and digital fabrication labs (more in Association for Science Technology Centers' Making & Tinkering Spaces in Museums Community of Practice work in 2016/17).

Computer Science *skills*	+	Tinkering *Qualities*
Sequencing Pattern Recognition Parallelism Conditionals Operators Decomposition Data Manipulation Algorithms Abstraction Recursion Loops		(Personal) Are learners generating & expressing their own ideas? (Constructive) Are they designing and creating projects? (Exploratory) Are they experimenting /playing with ideas and materials (physical and/or coding)? (Social) Are they collaborating and sharing with others? (Playful) Are they making things that spark joy and surprise? (Reflective) Are they revising their ideas based on what they are noticing?

Facilitation Resources - The Tinkering Studio

In this next iteration of the Unit Plan, we now challenge both the learners and educators to go further, to spark, sustain, and deepen. Here's some moves suggested by The Tinkering Studio at The Exploratorium:

Facilitation Move	*Descriptions of facilitator's interactions*
Spark initial interest and participation through demonstrations, modeling, parallel play, or questions	• Uses demonstrations, modeling, parallel play and/or questions to help learners take up materials and begin • Establishes safe physical and psychological (non-judgmental and reciprocal) context for participation
Sustain participation through frustration, distraction, or boredom through introduction of new tools, approaches, or analyses	• Provides new tools or ideas to scaffold learners through frustrating moments • Welcomes and works with learners' ideas and progress, including "mistakes" or wrong directions • Works with learners' ideas and creations to re-engage them as their interests and attention wane • "Re-voices" learners' ideas and choices in ways that others can understand and that might connect to the everyday world, prior interests and experiences
Deepen understanding and commitment through complexification of concepts and making connections to learners' interests	• Helps to complexify the work through new ideas or tools, including making connections or analogs with prior experiences, everyday life and scientific practices • Uses questions, actions, or comments to encourage learners to talk about, share, reflect on, and develop their ideas

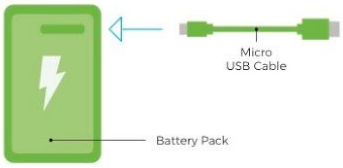
More at https://tinkering.exploratorium.edu/learning-and-facilitation-frameworks

ATTENTION, BEFORE FIRST USE!

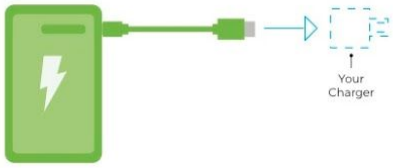
Please review the **Piper Unit Plan**, an Educator's Guide with Lesson Plans, Best Practices, Troubleshooting and many more resources to make your Piper experience successful! You can find the Piper Unit Plan here: **http://bit.ly/pipereducation**.

Before you start building your Piper Computer Kit, charge the **Battery Pack** and then the **Speaker** using the **Micro USB Cable** provided and **your own tablet/phone charger.** This process may take several hours.

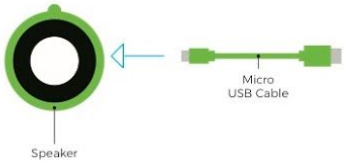
1. Insert the Micro USB Cable to the side of the Battery Pack.



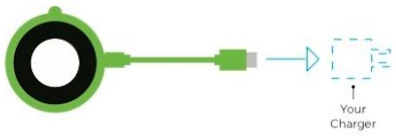
2. Connect the Micro USB Cable to your charger.
It may take approximately 6-10 hours for first full charge.



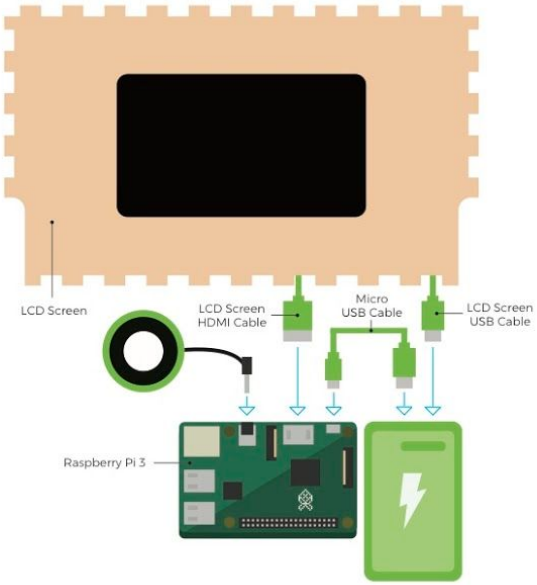
3. Insert the Micro USB Cable to the side of the Speaker.



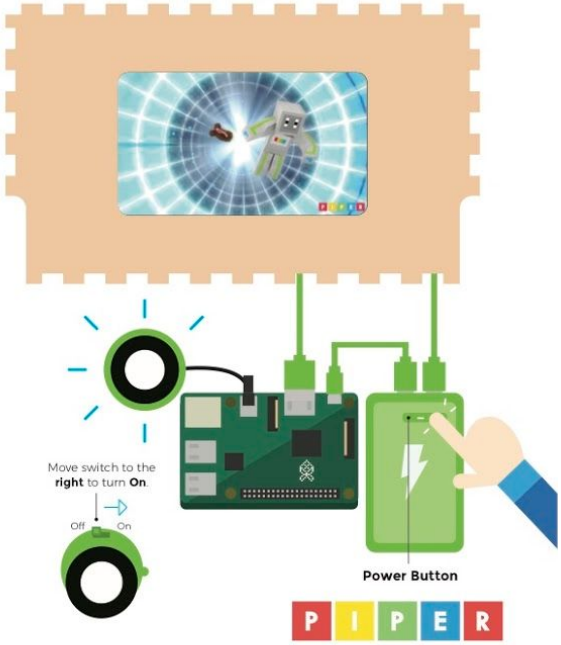
4. Connect the Micro USB Cable to your charger.
It may take approximately 2-3 hours for first full charge.



5. The Battery Pack and Speaker should be **off** before you connect them to your fully assembled Piper Computer Kit.



6. **Before turning the Battery Pack on**, make sure all cables are connected and the speaker has been turned on. Then, press the **Power Button** on the Battery Pack.



For additional support please visit: **support.playpiper.com**
For Educator Resources please visit: **playpiper.com/educators**