SCIENCE NORTH SCIENCE NORD
SUDBURY, ONTARIO, CANADA

Coding Circuits	Grade 6

Lesson Plan

Coding Tool	TinkerCAD Circuits
Time Required	Two periods

Math Curriculum Connections

Algebra

Overall Expectations

C3. Solve problems and create computational representations of mathematical situations using coding concepts and skills

Specific Expectations

C3.1 solve problems and create computational representations of mathematical situations by writing and executing efficient code, including code that involves conditional statements and other control structures

C3.2 read and alter existing code, including code that involves conditional statements and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code

Science Curriculum Connections

Grade 6

Electricity and Electrical Devices

- Electrical energy can be transformed into other forms of energy

Overall Expectations

2. Investigate the characteristics of current electricity and construct simple circuits

Specific Expectations

3.6 Explain the functions of the components of a simple electrical circuit

Description

Students will learn the basics of binary as the basis for computer information and perform a hands-on unplugged investigation of patterns in binary. Expanding from this knowledge, students will build simple circuits in TinkerCAD Circuits and investigate coding algorithms, including loops and conditional statements, that can be used to control a circuit's output.

Success Criteria

- 6th grade students will be able to identify the components of a simple circuit and how the components connect for the purpose of operating
- 6th grade students will be able to demonstrate how the output of a circuit can be controlled using conditional statements with TinkerCAD Circuits.

Materials and Media

- Devices that can support TinkerCAD Circuits (desktop app or browser version – see note below in Additional Resources)
- Binary Unplugged Activity
- Coding Guide

Computational Thinking Skills

This lesson takes two approaches to coding: abstract and practical. In the unplugged activity, students will take a step back from coding languages and logic that they may be familiar with to consider how computers interpret information at its most basic level: binary information, or bits. (See the "*Binary Unplugged Activity*" handout.)

In the online coding activity, students will be able to apply coding concepts to real-world examples. They will modify circuits and control outputs using loops and conditional statements. Conditional statements are used in coding to execute a condition if a statement is true (this can be considered analogous to helping a computer to make a programmed decision based on given factors or information).

Students will explore and modify conditional statements that control a circuit's output using TinkerCAD, a free software that uses simplified, block-based coding (similar to Scratch). More advanced (or curious) students can also use TinkerCAD Circuits to see how their block code translates to C, the programming language used for the Arduino microcontroller.

The "*Coding Handout*" for this lesson includes a more detailed step-by-step procedure for using TinkerCAD Circuits than is described below.

Introduction

Introduction to Binary

Computers use the binary system to represent information. We use binary in computers because, at their core, computers are made up of very simple circuits in which electrical current "flows" as electronic "switches" open and close. These circuits are so simple that everything that powers computer functions is represented by ON, or the presence of current (represented by the number 1) or OFF, the absence of current (represented by the number 0). Binary corresponds directly to how computers store information. Each zero or one is called a *bit* (or binary digit).

Computer data can be thought of as a series of "wires" connected to a series of switches that either pass or block the transmission of the current depending on whether a 1 or a 0 is to be transmitted.

Introduction to Circuits

Electrical circuits require a power supply (to provide electrical current to the circuit), conductors to transmit the current (e.g., wires), and a load (a device that transforms the energy from the electrical current to generate an output, like light, sound, or motion). We can control the output of a device by controlling the current that flows to the device. We generally do this with switches, which will open a circuit to stop the current's flow to the device and close to permit the current's flow to the device, stopping and starting the device, respectively.

During this lesson, we will be building simple circuits and we will be controlling them with switches. These switches will be further controlled with code so that we can program specific outputs based on given conditions, such as when a button is pushed (and a switch is closed) and vice-versa.

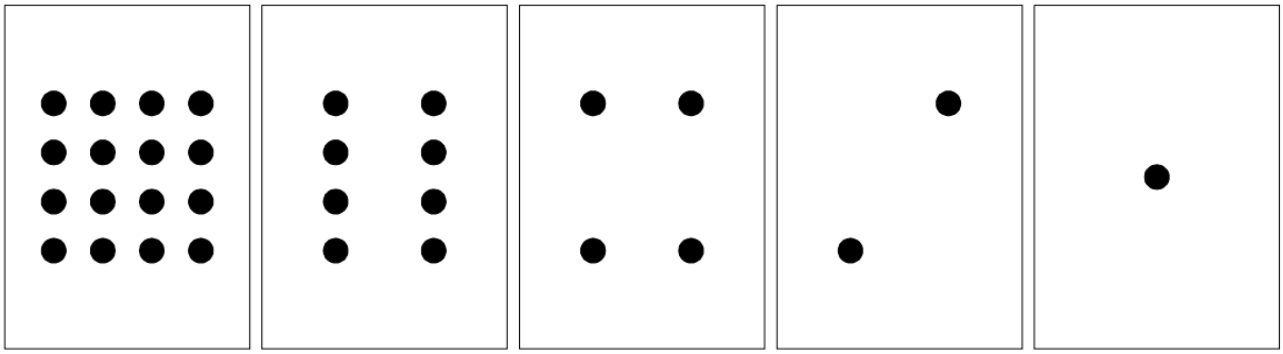
Action

Unplugged Activity

To understand how information can be conveyed using only 1s and 0s, have students practice translating binary into whole numbers.

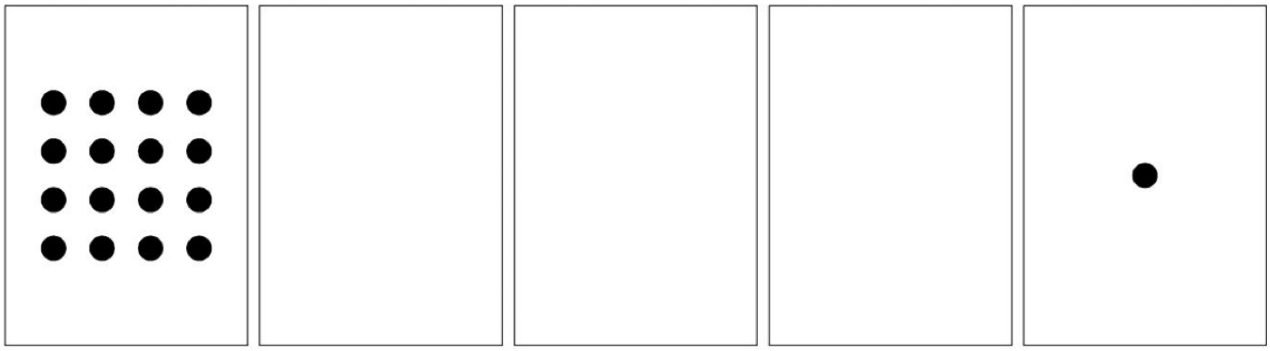
Binary Demonstration:

Create 5 cards with dots (pictured below). Each card has double the number of dots compared to the card to its right (ask students if they can identify this pattern). They should be arranged in the order pictured below (from left to right: 16 dots, 8 dots, 4 dots, 2 dots, 1 dot).



Choose five students as volunteers for this demonstration. Each student receives one of these cards and stands in a line in the same order as pictured. Explain that when a card is **not** showing (e.g., the card isn't flipped up to reveal its dots) it is represented by a zero. When the face of the card **is** showing its dots, it is represented by a one. To convert binary into a whole number, students add up the dots that are showing, that is to say only the dots represented by a one.

For example, ask students to represent 10001 with their cards. The ones are up, showing dots while the zeros are down, which gives us the sequence "up, down, down, down, up". By counting the number of dots shown, students can determine the whole number being represented:



(Answer = 17)

Ask the students to use their cards to make the numbers 1 – 20 in sequence (00001, 00010, 00011, …etc). The rest of the class should observe the sequence and try to identify a pattern in how the cards flip. Each card flips half as often as the card to its right.

*Note, this activity can also be done individually, with each student receiving their own set of 5 cards to manipulate at their own desk, rather than as a demonstration.

The "*Binary Unplugged Activity*" handout reinforces the idea of encoding information using a binary system by having students decode a binary message and then create their own.

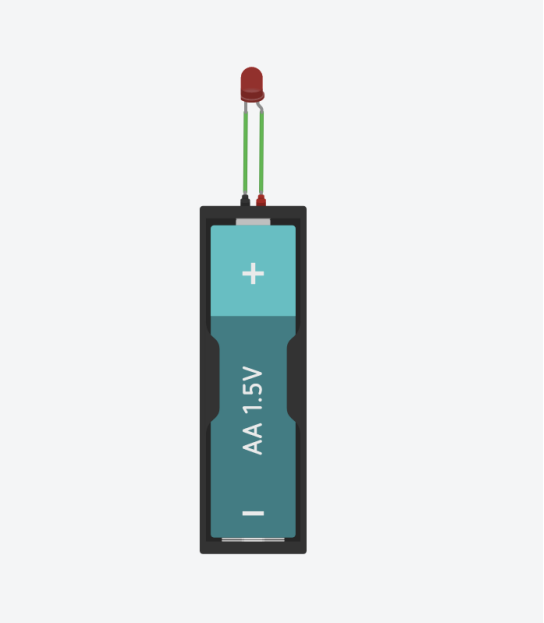
Coding Activity

Create a Simple Circuit in TinkerCAD Circuits

To create the most basic electric circuit, we will need (1) an energy source (in this case, a battery); (2) a conductor (wires), and a load (an LED). Ask students to describe everyday devices that are powered with circuits. How are these devices controlled (i.e., do they use a switch)?

Note that students can follow along using the '*Coding Guide'* handout for reference.

- Find the LED and the 1.5V battery in the components menu and drag them into the programming canvas.
- Draw wires by clicking on the startpoint and endpoint with your mouse or trackpad. A wire will automatically be drawn between these points.
- Test your circuit by pressing the Start Simulation button. What happens to your LED?

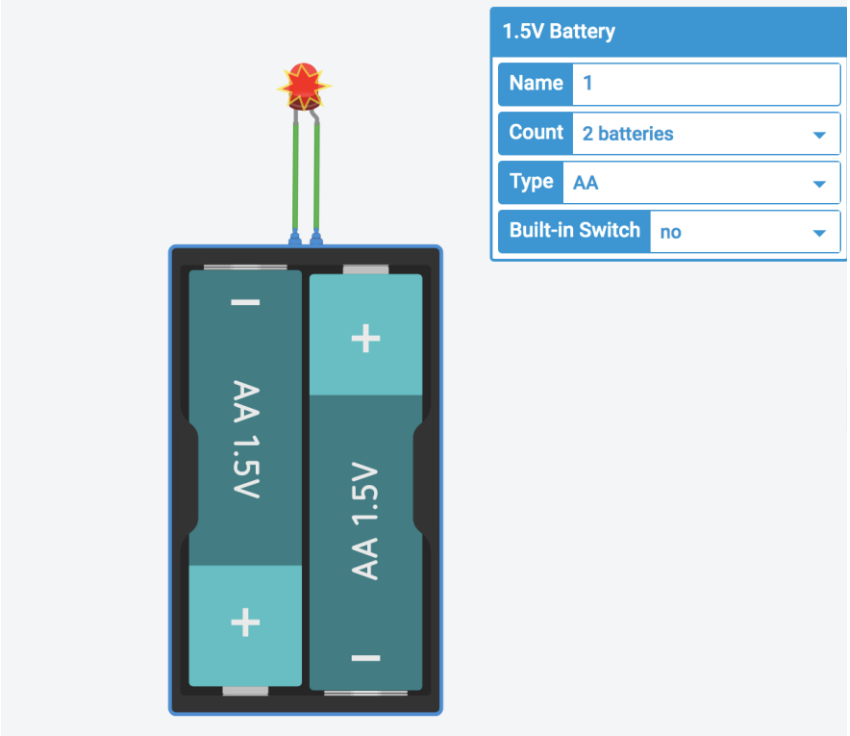


Descriptive text: Image showing a red LED connected to a AA battery with two wires to form a simple circuit.

*Note: Just like with real circuits, how you connect your wires matters! The LED's cathode (or positive lead) needs to connect to the negative terminal of the battery and the LED's anode (or negative lead) needs to connect to the positive terminal of the battery.

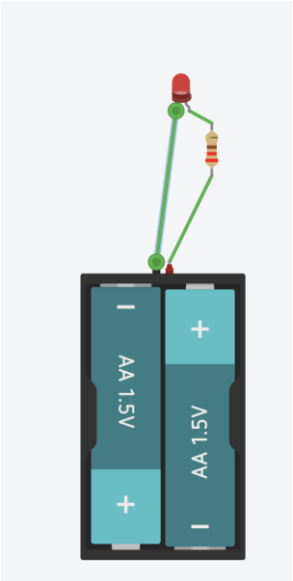
Adding Resistors

- We are going to increase the amount of current coming from our power source by adding a second battery. Do this by left-clicking on the power source and updating the 1.5V battery count to **2 batteries**. What happens to your LED?



(Descriptive text: Image showing a red LED connected to two AA batteries with two wires to form a simple circuit. The LED is covered by a red explosion symbol.)

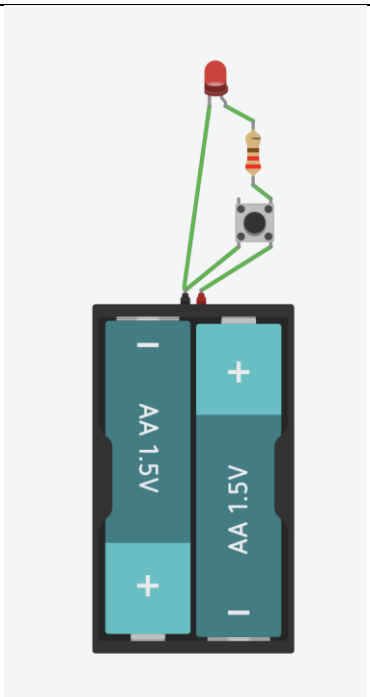
- The current running through the LED is beyond the maximum that our load can support. Overloading a circuit can cause the wiring to overheat, ruin the device that you're trying to power, or start a fire. To reduce the current from our load, we can add a resistor. Grab a resistor from the components panel and drag it into the programming canvas.
- Reconnect the circuit by redrawing the wires so that a wire connects the negative terminal of the battery pack to terminal 1 of the resistor, and a wire connects terminal 2 of the resistor to the LED.
- Start the simulation. What happens to your circuit?



(Descriptive text: Image showing a simple circuit with a red LED, two green wires, a battery pack, and a resistor. The LED is lit.)

Adding a Switch

- Let's add a switch that will allow us to control our LED. From the components menu, choose the pushbutton and drag it into the programming canvas. We will want to integrate our button into the circuit by connecting it to both terminals of our power source, as well as to the anode of our LED (via a resistor). See the image below, which illustrates one possible way to connect the switch.



(Descriptive text: Image showing a simple circuit with a red LED, two green wires, a battery pack, and a resistor and a switch. The LED is lit.)

In the way that this circuit is set up, the LED is lit while the button is NOT pressed and turns off when the button is pressed. This is set up as what is known as a normally-on switch, meaning that the circuit is interrupted when you press the button. You may be more familiar with normally-off switches, where there is no connection until the button is pushed (e.g., doorbells, electronic car key buttons).

Adding Code

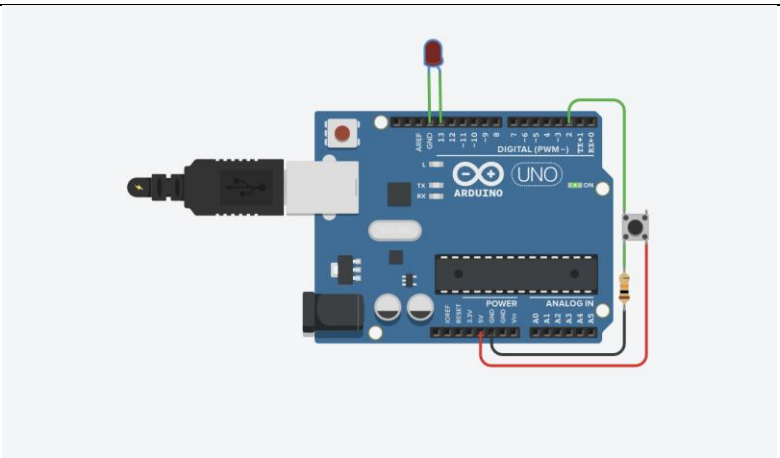
Next to the Start/Stop simulation button, you will notice that there is a **Code** button. Clicking this button will reveal code to control our circuit output (in this case, how our LED lights up). With our current circuit, there is no code because our circuit is not currently programmable (the only control we have is whether our circuit is open (off, or 0), or closed (on, or 1).

- In the **Components** drop-down menu, choose **Arduino.** Drag the example **Button** into the programming canvas.
- This circuit looks very similar to the circuit that we made previously. Our Arduino is a programmable microcontroller powered by USB. The current flows through pin 13, is controlled via a resistor to the load (the LED), which is, in this case, a tiny built-in light on the Arduino.
- If you press the Code button, you will see block code to program the circuit (If you have used Scratch or Blockly before, you may note that this coding format is very similar).

The existing code is a loop (although no loop block exists in this case — if you choose to change your view to Blocks + Text, you will be able to see the loop in C, the coding language used for Arduino) that checks the pin connected to the pushbutton (pin 2) to see if the button is being pressed (HIGH, or 1) or not (LOW, or 0). We then see a conditional statement for both pushbutton states. If the buttonState = HIGH, then the controller sets the pin controlling the LED (pin 13) to HIGH and the LED turns on; else, the pin controlling the LED is set to LOW and the LED is turned off.

(Descriptive Text: Image showing block code that describes the program that uses Arduino and a pushbutton to control an on-board LED light)

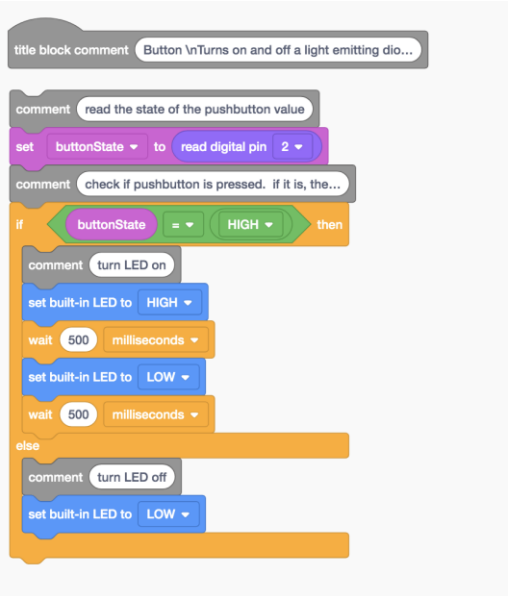
- Let's add an external LED to see the effects more clearly. We can drag an LED to the programming canvas and connect the cathode to the Ground (GND) pin and the anode to pin 13 (the LED pin on the Arduino board). Now, when you push the button, the red LED should light up.



(Descriptive text: Image showing an Arduino board with wires connecting to a pushbutton and a red LED.)

Take it Further

- Let's change the output of our conditional statement so that the LED blinks on an off on its own while the pushbutton is pushed. We will do that by adding a loop within the if statement of our conditional. The loop will set the LED to HIGH, wait for 500 milliseconds (or 0.5 seconds), set the LED to LOW, wait for 500 milliseconds, and repeat as long as the button is pushed.
-



(Descriptive text: Image showing block code that describes the program that uses Arduino and a pushbutton to control an LED light to make it blink.)

- Try experimenting with the parameters of your code. Can you create your own blinking pattern by adjusting the wait blocks?

Closure and Assessment

By the end of this lesson, students should be able to recognize the components required to create a simple circuit, and be able to describe the similarity between an electrical circuit and the flow of information within a computer as being controlled by ON/OFF switches or binary digits. Students should be able to modify simple conditional statements to change the output of a digitally-created circuit, test their code, and iterate.

For assessment, collect the "binary unplugged activity" handouts from the students. Review their work to ensure that they understood the concept of the binary system by appropriately decoding the binary message and created a message of their own that followed the system's rules of logic.

Adaptations

- The binary demonstration may be done individually, with each student receiving their own set of 5 cards to manipulate at their own desk, rather than as a demonstration.
- The binary demonstration can also be done with students sitting down in a row of five.
- The colours of the wires and LED components in this program can have their colours changed easily by left clicking the component and selecting a different colour in cases where colour distinction may be challenging.

Extensions

- Students who finish early can attempt more creative circuit designs. Can they succeed in programming an RGB LED? Note: the cathode must be connected to the Ground (GND) pin, and each of R, G, and B, leads must be assigned their own pin and resistor.

Additional Resources

- Tinkercad.com (Free account required) *Note, as of November 2020, Circuits is only available in desktop/internet browser version of TinkerCAD. The TinkerCAD app does not allow you to create or simulate circuit designs at this time. The online account is free and only one account should be required (all students can login under same account); however, you may choose to create individual student accounts if desired.