Lesson One: Make Some Noise!

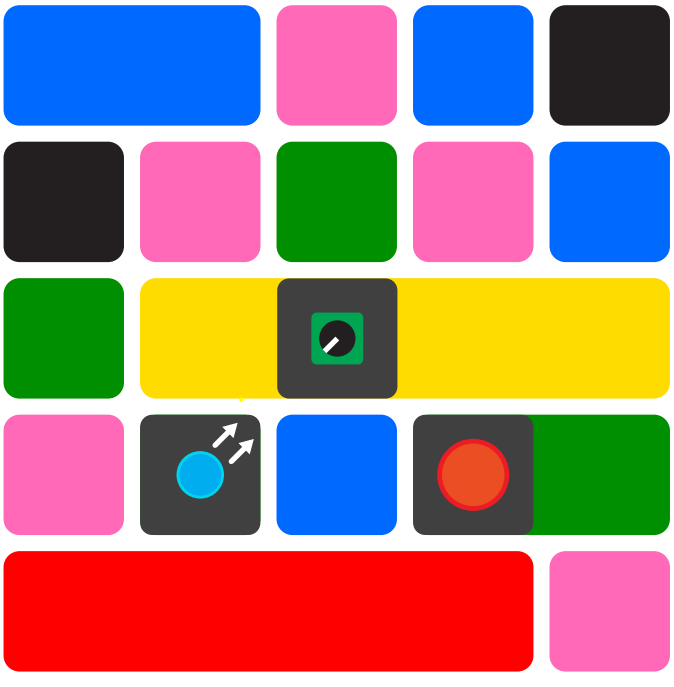
Setup

Materials


Knob


Button


LED



Welcome to your first experiment with sound! In this lesson, you'll use a knob to change the pitch of a tone and a button to play the sound. This simple setup lets you explore how sound works—turning the knob adjusts the frequency, making the pitch go higher or lower, and pressing the button triggers the note. By playing around, you'll start to understand how different pitches are created, just like on a musical instrument. Try turning the knob while holding the button to hear the smooth shift in pitch. What kinds of sounds can you create?

Press the Button – Does the sound always stay the same, or does it change?
Turn the Knob – Try turning it slowly and quickly. What happens to the sound?
Experiment – Try different positions on the knob and press the button.
Can you make high sounds? Low sounds?

- What happens if you turn the knob all the way left? What about all the way right?
- Can you make a melody by turning the knob and pressing the button in a pattern?
- What does this remind you of? Have you ever seen a device that works like this?

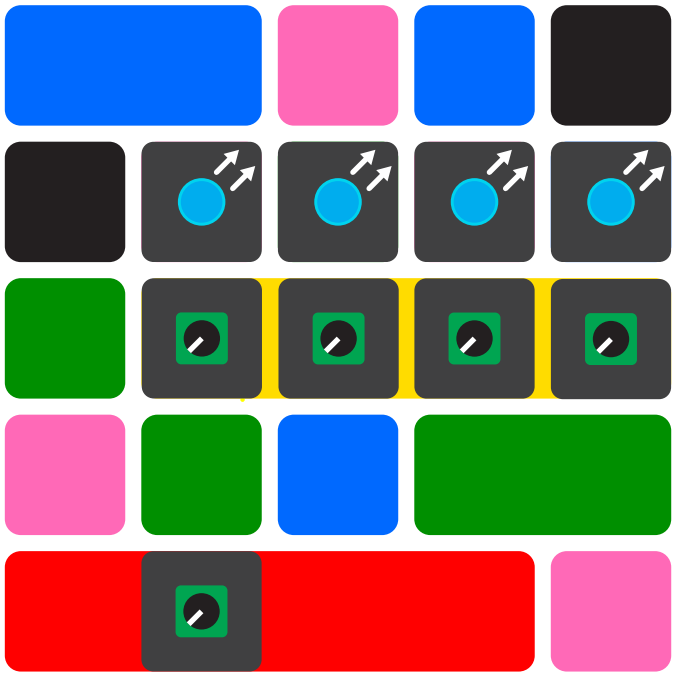
Lesson Two: Your First Song

Setup

Materials

Knob

LED



In this lesson, you'll make a four-step **sequencer**, which is a fun way to create repeating patterns in music! Each step has a knob that lets you change the pitch, kind of like making a musical recipe by adding different ingredients. As the sequence loops over and over, you can turn the knobs to hear how small changes make a big difference in the rhythm and melody. Once you've built it, try twisting the knobs in different ways to make your own unique pattern.

Turn the Knobs – How does turning a knob change the sound?
Speed it Up – What happens when you change the speed of the sequencer?

What makes a pattern sound interesing? What makes it sound happy or sad?

Does a slow pattern feel different from a fast one? What kind of music uses fast or slow patterns?

Challenge : Can you recreate a song or tune that you know?

Lesson Three: Shape Your Sound

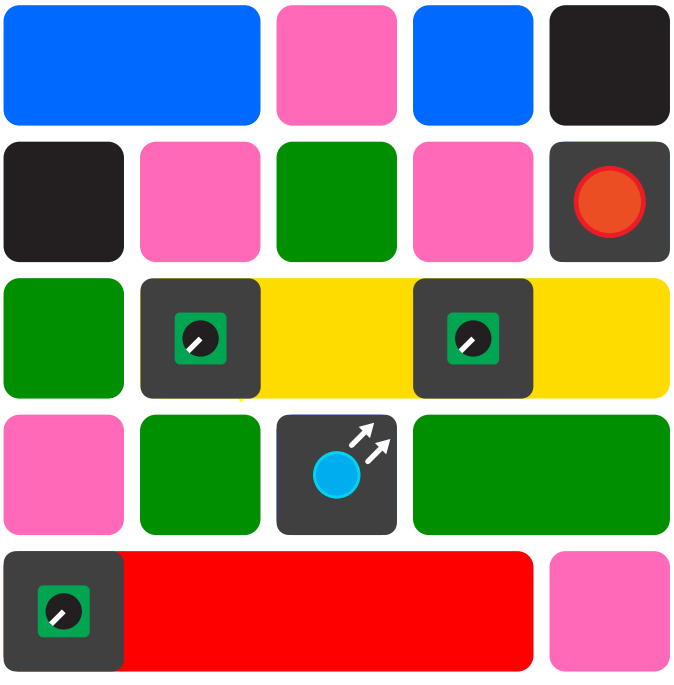
Setup

Materials

Knob

Button

LED



Have you ever noticed how some sounds start suddenly, like a clap, while others fade in gently, like a rising whistle? That's what the Attack knob controls–it changes how quickly a sound reaches full volume. The Release knob does the opposite–it controls how long the sound fades away after you let go of a note. In this lesson, you'll experiment with these knobs to see how they shape different sounds.

Make It Snappy – Turn the attack all the way down and the release all the way down. What happens? Try playing short, fast notes

Make It Smooth – Now turn the attack and decay all the way up. How does this ound different?

Experiment – Can you recreate the sound of different instru-ments? Think about what a piano sounds like. Now try a drum

- Which feels more important for your sound, how it begins or how it ends?
- How do different attack and release settings change the way a song feels?
- Can you think of an instrument that has a naturally slow attack or long release?

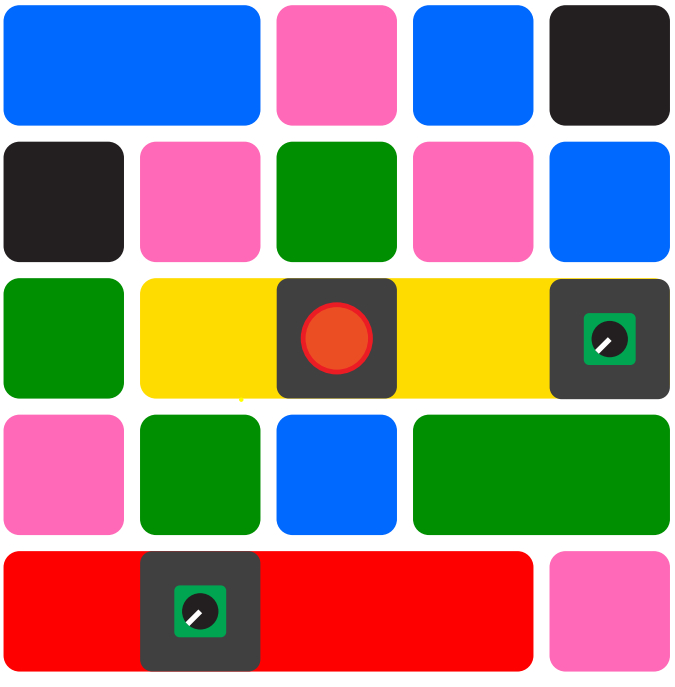
Lesson Four: New Tones

Setup

Materials

Knob

Button



Did you know you can change the way a sound feels, just like turning a flashlight dimmer or making a voice sound muffled? In this lesson, you'll use two special knobs to shape sound in fun ways! One knob makes the sound clearer or softer, like moving a blanket over and off a speaker. The other knob adds a little boost, making certain sounds pop out more, almost like a robot talking or a whistle getting sharper. Play around with both knobs and see what kind of cool sound changes you can make!

Make it Fuzzy or Clear – Turn the first knob slowly and listen to how the sound goes from muffled to bright. What does it remind you of?

Create a Wobbly Effect – Turn the second knob up. How does it sound when you turn the first knob now?

- Try to describe the difference in sound when you have each knob all the way up or down
- Can you think of any times when you heard a muffled sound, like speaking with your mouth covered?

Challenge:

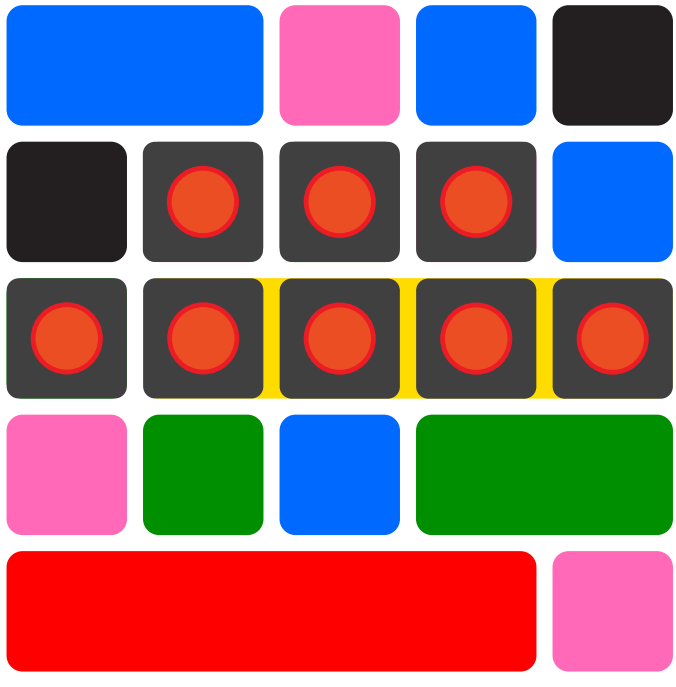
Can you use the two knobs to to make your sound feel like a robot? A laser? A whisper?

Lesson Five: Pocket Piano

Setup

Materials

Button



Imagine turning a row of buttons into your very own mini piano! In this lesson, each button plays a different note, and when you press multiple buttons together, you can create chords–just like on a real keyboard. Chords are what make music sound full and exciting, and they're used in almost every song you hear. By experimenting with different button combinations, you'll discover how some notes sound great together while others might clash. Try building happy-sounding chords, mysterious ones, or even ones that feel a little spooky! Get ready to explore how chords shape the feeling of music with just the press of a button.

Find a Happy Chord – Press different buttons together and listen for a combination that sounds bright and cheerful. What notes make it sound that way?
Find a Spooky Chord – Some chords feel mysterious or eerie
Create a Chord Progression – Try pressing different groups of buttons one after another. Can you make a short pattern that sounds like a song?

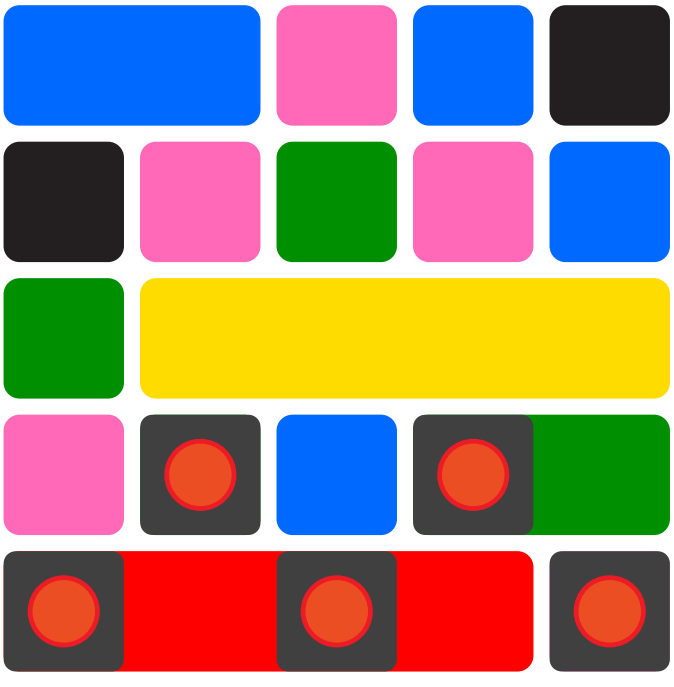
- What happens if you press buttons that are right next to each other? How does that sound?
- Why do you think some chords sound good and others sound bad? What makes a chord sound good?
- How can different chords express different feelings in music?

Lesson Six: Make a Beat

Setup

Materials

Button



Get ready to create your own beats! In this lesson, you'll turn five buttons into a simple drum machine, where each button plays a different drum sound. Just like a real drum set, you'll have different parts to mix and match—maybe a kick, a snare, and some hi-hats! By pressing the buttons in different patterns, you can create unique rhythms and explore how different drum sounds work together. Try playing along with a beat or making up your own! What kind of rhythm will you come up with?

Make a Beat: Press different buttons to hear the drum sounds. Try tapping out a steady beat or creating your own funky rhythm!
Layer Your Sounds: Experiment with pressing multiple buttons at once. Can you make a beat that sounds like a real drummer?
Speed It Up or Slow It Down: Try playing the buttons fast or slow. How does changing the speed affect the feel of the rhythm?

- What happens when you play the sounds in different orders?
- How do different drum sounds work together to create a groove?
- What kind of music could you make with just these five sounds?

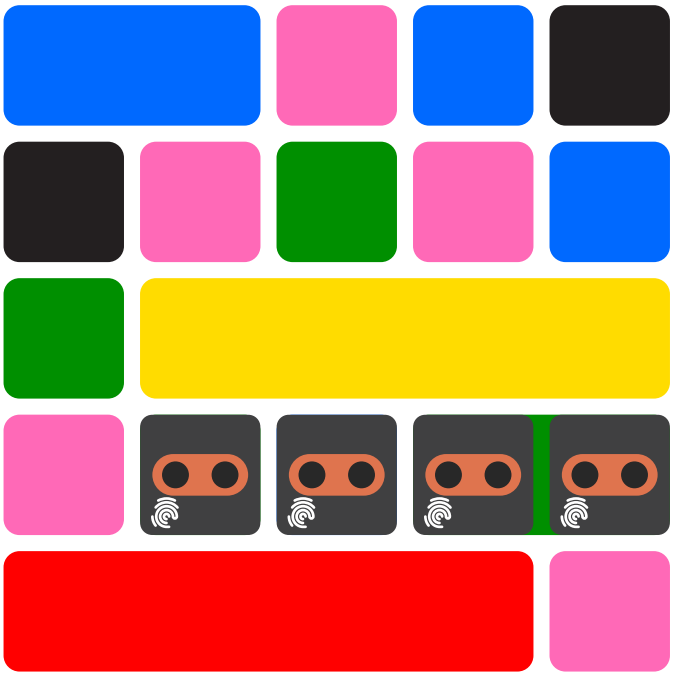
Challenge: Try to recreate the beat of your favorite song using only the five buttons. Can you match the rhythm just by listening?

Lesson Seven: Music with Touch

Setup

Materials

Touch Sensor



Lets get creative. In this lesson, you'll use the touch sensors to turn everyday objects into drums! Have you ever touched a tablet or phone screen and seen it react to your finger? That's because your body holds a tiny bit of electricity–and so do the wires and parts in this lesson! When you touch one of these special pads, the sensor feels a change, kind of like getting a tiny electric 'tickle.' It uses that feeling to know you're touching it. Use the alligator clips to attach four different objects and see what happens when you touch them.

Make Drums with Fruit – Connect four different fruits with the alligator clips and use them to make some noise

Try Using Plastic or Wood – Find some plastic or wood objects and connect those with the clips. Does this work as well?

Find Some New Objects – Try even more objects, see what works and what doesn't work

- Why do you think some objects work better than others? What is the difference between these objects?
- Where have you seen touch sensors before in real life? What kind of devices use them?
- Why do you think your phone screen doesn't work when you are wearing gloves?

Lesson Eight: Sonic Sensors

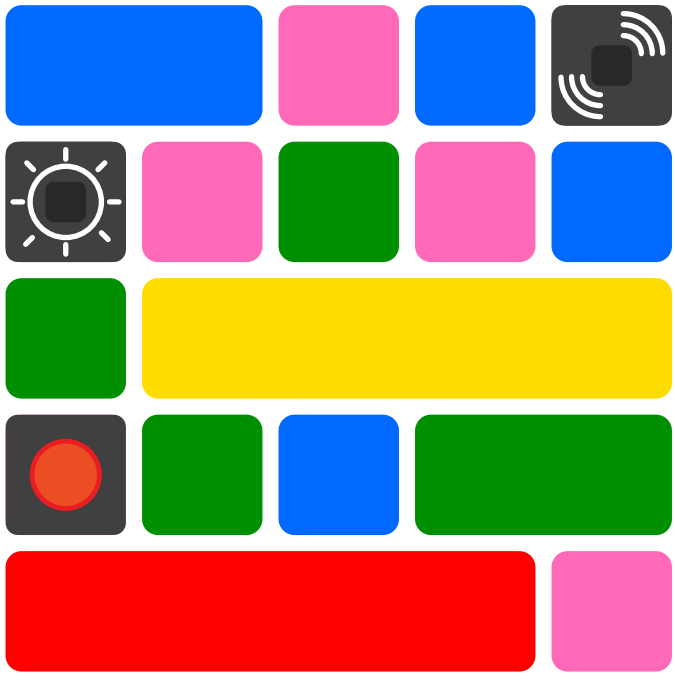
Setup

Materials

Button

Light Sensor

Movement Sensor



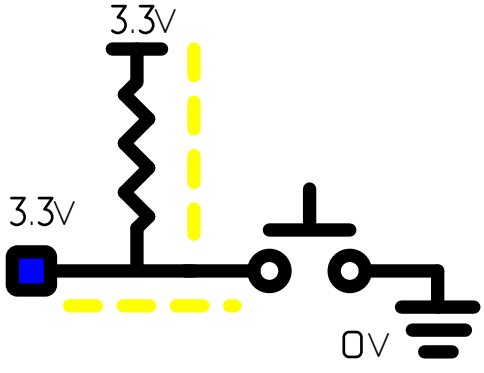
Have you ever played a game where you tilt the screen to move a character? Or noticed how your phone screen dims when it's too bright outside? That's because there are sensors that help machines understand the world around them–just like we use our eyes and ears! In this lesson, you'll explore how a motion and a light sensor help machines react to the world. You'll build your own experiment where lights and sounds change when you pick up and tilt your board or cover the sensor from light. You'll also combine both sensors to create something really smart–like a robot eye that opens in the dark, or a secret device that responds only when picked up.

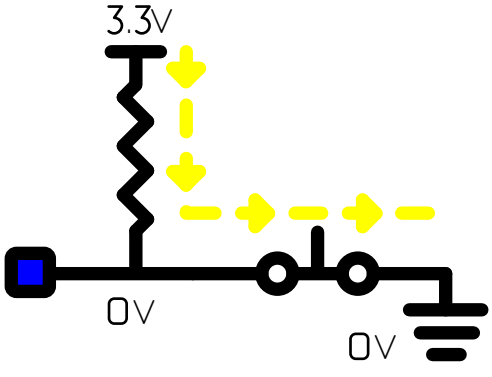
Cover the Light Sensor – How does the sound change when you cover and uncover it?
Move the Board – Tilt the board forward and back, left and right. How does this change the sound?
Combine Them – Move the board and cover the light sensor at the same time. See how the two sensors work together to create new sounds

- What is the difference between how the motion and light sensor react to you?
- Where have you seen lights that turn on when it gets dark?
- How can these sensors be used to make devices smarter?

Lesson One: Buttons

Buttons are one of the most basic inputs in electronics – they let you interact with your circuit by pressing something physical. But just connecting a wire from a button to a **microcontroller** pin isn't enough. You need to think about what the pin sees when the button is not pressed.

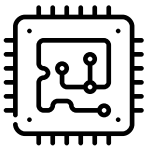




Button

A **button** (or switch) is an electrical component that connects or disconnects two points in a circuit

Microcontroller



A **microcontroller** is a device that runs programs and can read sensors and control other devices

The button is open (like a broken bridge).
No current flows through the button to GND.
The only path connected to the input pin is through the pull-up resistor to **3.3** V
So the pin sees HIGH voltage (close to **3.3** V), because there's nothing pulling it down.

The button closes the circuit between the input pin and GND.
Now, there's a low-resistance path from the input pin directly to ground.
The voltage at the pin drops to **0** V (GND).
A small current now flows from **3.3** V through the pull-up resistor into the grounded pin.

```
Button myButton(Green, D);
myButton.Pressed();
myButton.onPress(myFunction);
```

Creates a Button object
Returns true when pressed
Assigns a function to press

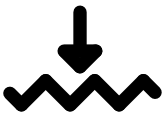
Challenge: Try moving the button to a different square and changing the code to match it
Challenge: Add a second button and change the code to make an "On" and "Off" button

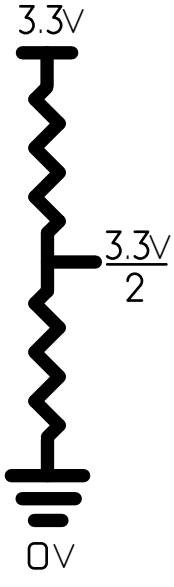
Lesson Two: Knobs

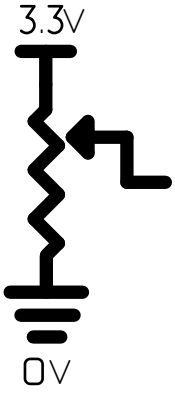
A **knob,** often called a **potentiometer,** is an electrical component that lets you adjust resistance by turning it, which changes the voltage at its output. When used with a microcontroller, it acts like a volume dial—providing a smooth range of values instead of just on or off. This makes it perfect for controlling things like brightness, pitch, speed, or anything that can vary gradually.

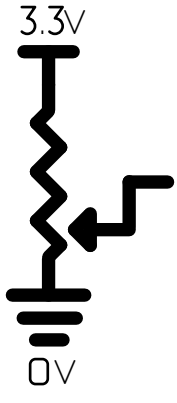
Potentiometer

A **potentiometer** is a variable resistor that changes its output voltage based on the position of its knob, making it useful for adjusting levels like brightness, pitch, or volume.









A voltage divider is a simple but powerful circuit that takes an input voltage and splits it into a smaller output voltage. It works using two resistors connected in series: one end goes to a high voltage (like **3.3**V), the other to ground (**0** V), and the output is taken from the point between the two resistors. The voltage at that point depends on the ratio of the two resistors – this is a basic principle used all over electronics to scale voltages.

A potentiometer is like a voltage divider built into a single component. It has a resistive strip inside, and a movable contact called the wiper. As you turn the knob, the wiper slides along the strip, changing the resistance on either side of it. One end of the pot is connected to **3.3**V, the other to GND, and the wiper gives you a voltage somewhere in between – depending on its position. This lets you create a smooth, adjustable voltage that can be read by a microcontroller's analog input.

```
Knob myKnob(Yellow, C);
myKnob.read();
```

Creates a knob object
Returns knob's position, **0 - 255**

Challenge: See if you can reverse the direction of the knobs so high is to the left and low is to the right

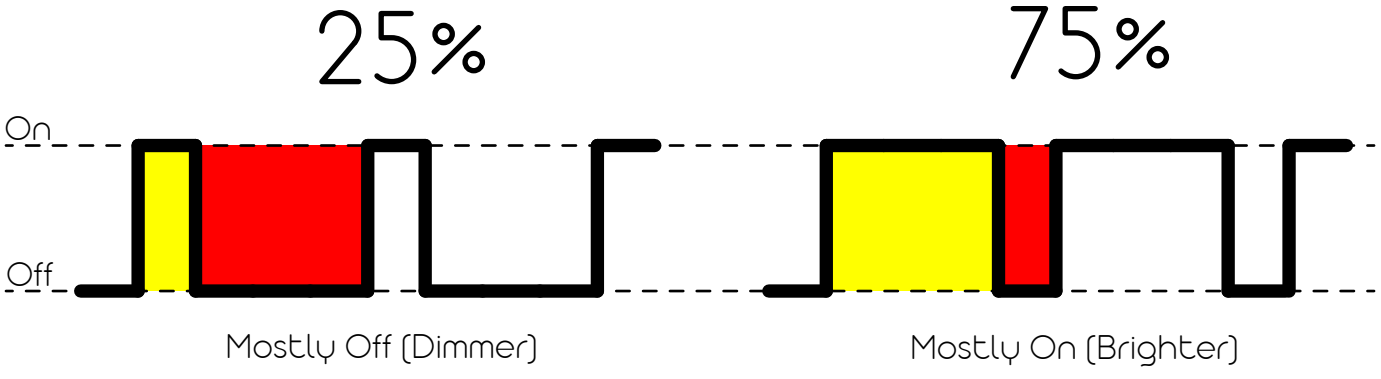
Challenge: Add another knob and use it to control the speed of the notes

Lesson Three: LEDs and PWM

An **LED** (Light Emitting Diode) is a small, energy-efficient light that turns on when current flows through it in the correct direction. While a microcontroller can only turn pins fully on or off, we often want to control how bright an LED is. This is where **PWM**, or Pulse Width Modulation, comes in. By turning the LED on and off very quickly – faster than our eyes can see – we can create the illusion of different brightness levels. This technique is used in everything from LED dimmers to audio synthesis to motor speed control.

LED

An LED emits light when electrical current passes through it



When we use PWM (Pulse Width Modulation) to control an LED, we're not actually changing how much voltage the LED gets – we're turning it on and off very quickly, thousands of times per second. This rapid switching is so fast that our eyes can't see the flicker. Instead, we perceive it as the LED getting dimmer or brighter, depending on how long it stays on during each cycle.

This idea is called the duty cycle – the percentage of time the signal is on during one complete on/off cycle. A **100**% duty cycle means the LED is on the whole time (fully bright), while a **0**% duty cycle means it's always off. A **50**% duty cycle means the LED is on for half the time and off for half the time – so it looks dimmer, but still visibly on. The lower the duty cycle, the less light (or power) the LED outputs on average.

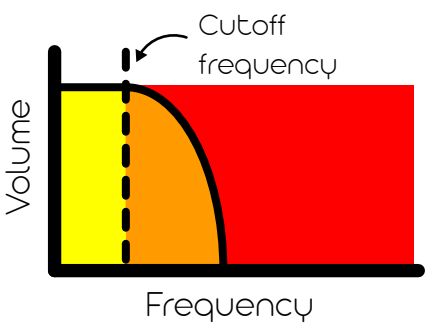
```
LED myLed(Blue, A);
myLED.on();
myLED.off();
myLED.toggle();
myLED.dim();
```

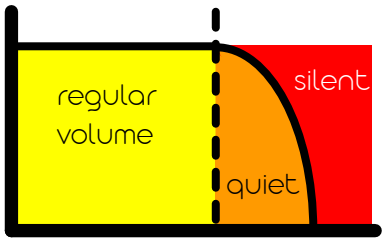
Creates an LED object
Turns on the LED
Turns off the LED
Toggles the LED on or off
Sets the LED to any brightness level

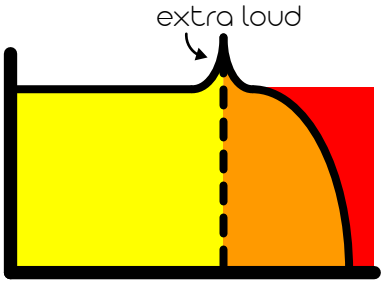
Challenge: Create a musical instrument that combines all three modules you've explored so far – the button, knob, and LED.

Lesson Four: Filters

An audio filter is a tool that changes how a sound feels by letting some frequencies through and reducing others. Filters are used to make sounds brighter, warmer, thinner, darker, or more focused. Filters don't change the volume of a sound – they change its tone or color. Think of a low-pass filter like turning down the treble on a stereo. A high-pass filter does the opposite – it cuts the bass.







Low Cutoff Frequency

High Cutoff Frequency

High Resonance

Imagine sound as a mix of low, medium, and high tones – like bass, midrange, and treble. A filter can focus on a certain part of that mix. For example, a low-pass filter lets the deep, bassy sounds through but cuts out the brighter, higher tones. It makes a sound feel softer, rounder, or more distant. The point where the filter starts to cut things out is called the cutoff frequency, and you can adjust it with a knob to shape the sound in real time.

Resonance is like adding a spotlight right at the cutoff point – it makes the frequencies near that edge stand out more. Turning up the resonance makes the sound feel sharper or more "peaky," almost like it's whistling or ringing at that point. When you combine cutoff and resonance, you can sweep through a sound and make it feel alive, animated, or dramatic – this is how filter sweeps and cool sound effects are made in

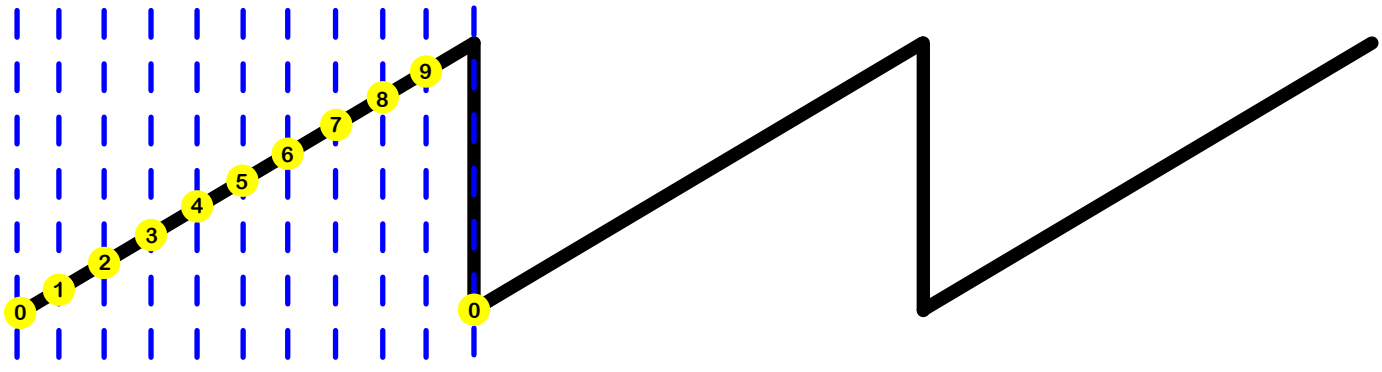
```
Filter myFilter();
myFilter.setCutoff();
myFilter.setResonance();
myFilter.setType();
```

Creates a filter object
Sets the cutoff frequency in Hz
Sets the resonance amount **0.0 - 1.0**
Sets the filter type (lowpass, highpass)

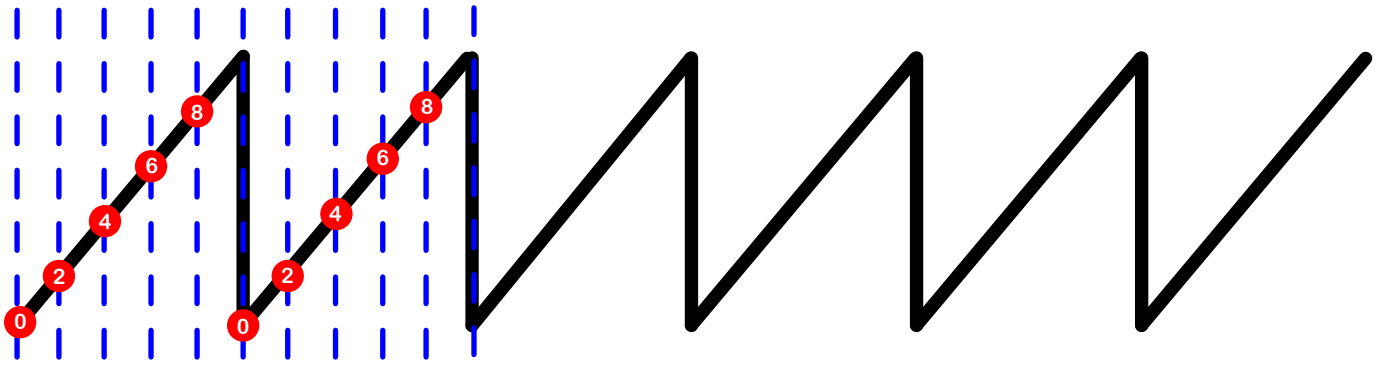
Challenge: Change the filter from a low-pass filter to a high-pass filter and observe how this changes the sound

Lesson Five: Digital Synthesis

In digital synthesis, sound is created not by vibrating physical objects or analog circuits, but by generating a stream of numbers that represent a waveform – and then playing those numbers fast enough to become sound. One of the simplest and most common ways to do this is with a lookup table: a pre-filled array that stores the shape of a waveform, like a sine wave, square wave, or triangle wave. Think of it like drawing one full cycle of a wave and saving it in memory. Instead of recalculating each point every time, the microcontroller just steps through the table over and over again, sending those values to a speaker.



[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, ...]



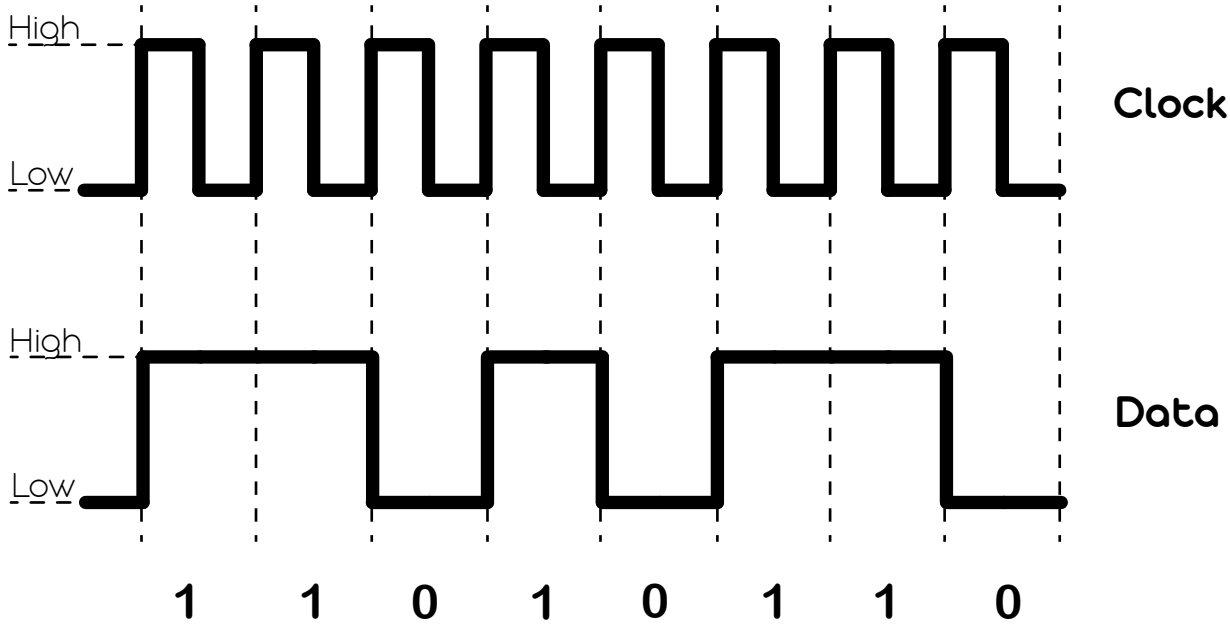
The speed at which it goes through the table determines the pitch – faster stepping creates higher frequencies, and slower stepping creates lower ones. This stepping is done using a phase accumulator, which keeps track of where we are in the table and allows us to jump forward by small amounts to change the frequency. By using different tables, changing the step size, or blending between waveforms, we can create a wide variety of sounds – all digitally. This technique, called wavetable synthesis, is used in everything from vintage game sounds to modern software synthesizers.

```
Tone myTone();
myTone.setFrequency();
myTone.on();
myTone.off();
```

Creates a tone object for sound
Sets the note frequency in Hz
Turns the note on (audible)
Turns the note off (silent)

Lesson Six: Communication

In electronics, digital components often need to communicate with each other – sending data, commands, or sensor readings between chips, modules, or microcontrollers. This is where digital communication protocols come in. One of the most widely used is called I²C (Inter-Integrated Circuit). I²C is a system that lets multiple devices talk over just two wires: one for the clock (SCL) and one for the data (SDA). Each device on the I²C bus has a unique address, like a name, so the microcontroller can send data to a specific part – like a sensor or display – without confusion. I²C is perfect for connecting small modules because it's simple, flexible, and supports multiple devices on the same lines. In this lesson, you'll learn how digital communication works, and how to read data from something like a motion sensor or send text to a screen – all using just two pins.



Digital communication happens when two devices share information using just **0**s and **1**s, sent one bit at a time. To do this, they often use a clock signal and a data signal. The clock acts like a metronome – it keeps a steady beat and tells the receiving device exactly when to read the data. The data line carries the actual information, changing between high (**1**) and low (**0**) voltages. Every time the clock ticks, the receiving side checks the data line to see whether the bit is a **1** or a **0**. These bits are grouped together into **8**-bit chunks, called bytes. A byte might represent a number, a letter, or a command – depending on how the devices are programmed to interpret it. By carefully syncing the clock and data lines, digital devices can share complex information quickly and reliably, even though it's all built from simple on-off signals.

```
Sensor mySensor();
mySensor.read();
```

Creates a sensor object for reading data
Returns the decimal value of the sensor