Bright Beats Challenge - MATLAB Simulation Instructor Guide

Prerequisite Activities and Knowledge: Students should complete the micro:bit device activity before moving on to the MATLAB simulation.

Expected Completion Time: 60 minutes

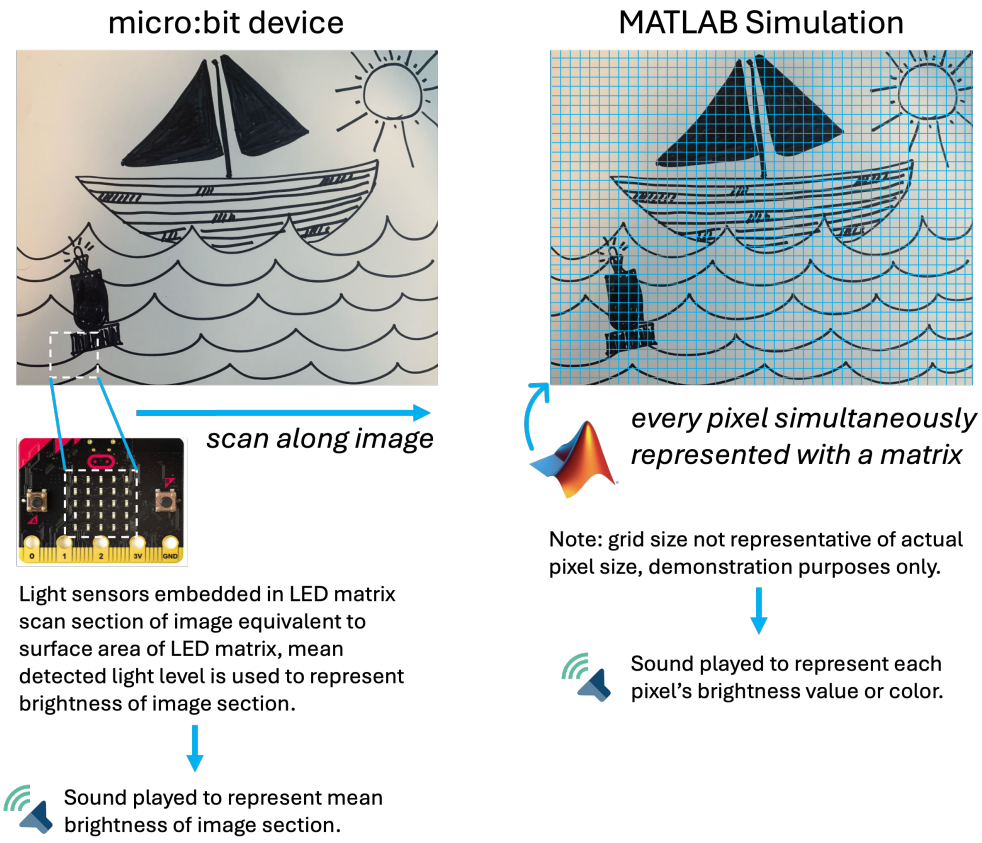
Table of Contents

micro:bit vs. MATLAB Overview

Key Conceptual Differences between micro:bit device function and MATLAB simulation

- Embedded in the micro:bit LED matrix is a grid of 9 light sensors (phototransistors). The mean amount of ambient light detected by the 9 sensors is used as the input to determine the sound that will be played. These light sensors are intended to detect big changes in ambient light, and cannot detect small differences or specific colors.
- MATLAB represents digital images as a matrix of pixels. Each pixel has a corresponding brightness value (for black and white or grayscale images) or corresponding RGB color code (for color images).
- micro:bit provides a single number to show how bright the surrounding environment is, while MATLAB can represent thousands of single pixels simultaneously.

The schematic below provides an overview of how the micro:bit device function and MATLAB simulation differ from each other.



Key Advantages of Using MATLAB to translate images to sounds

- MATLAB represents the brightness and color of each pixel by storing them as numeric values. To obtain those pixel values from a physical drawing, we can import a captured image or use a webcam connected to MATLAB. The micro:bit contains light sensors that can scan large parts of an image and combine their readings to obtain the average amount of ambient light.
- MATLAB has built-in signal processing tools that allow the user to easily customize smoothing applied to a signal and the sampling period.
- MATLAB is an industry standard tool in the field of signal processing and provides students with a more authentic experience of manipulating and interpreting signals.
- MATLAB code can be deployed for use on multiple types of hardware rather than being limited to micro:bit.
- Students have the option to explore and customize every part of MATLAB code without being limited to what's possible with micro:bit. This allows students with coding knowledge to fully customize how brightness values or colors are translated into sounds and customize the sounds themselves.

Part 1: Translate stripes to sounds

Expected duration: 20 minutes

Learning Objectives

By the end of this activity, students will be able to:

- Understand how digital black and white stripes are represented numerically by MATLAB
- Recognize and practice how information can be encoded as one signal (visual) and decoded as another (sound)
- Analyze how different systems (micro:bit vs MATLAB) interpret and convert different versions of the same visual signal, and discuss factors that affect accuracy and fidelity
- Engage in identifying and reconstructing visual patterns using auditory cues, reinforcing the connection to building a device that could help those who are visually impaired

Materials

- MATLAB®
- micro:bit device built previously
- Blank paper and black markers or pens

Steps

- Ask students to open MATLAB Online
- Ask students to copy the folder containing 'BrightBeatsChallenge_Student.mlx' and the 'ExampleImages' folder to their MATLAB Drive.
- Have students check that they are in the appropriate file directory. In the 'Files' pane of the MATLAB environment, students should see the file 'BrightBeatsChallenge_Student.mlx' and the folder 'ExampleImages'.
- Ask students to open the following script and follow instructions from top to bottom

```
open 'BrightBeatsChallenge_Student.mlx'
```

- Ensure students either select an octave of piano notes to use *or* click 'Use default notes'.
- Have students try generating several random striped images and the corresponding brightness value plots. Note that students must click 'Plot image brightness' every time they generate a random striped image to visualize the corresponding brightness value plot.
- Have students practice listening to the sounds that are played to represent the black and white stripes and attempt the listening challenge. If you'd like to include a hands-on activity, you could ask students to try to draw what they think the pattern of black and white stripes is based on listening to the sounds alone.
- Have students attempt the micro:bit device challenge. Ask students to discuss the differences they perceive between the micro:bit device's function and the MATLAB simulation. Reinforce the connection and differences between how their micro:bit device and the MATLAB simulation translate the stripes into sounds.

Part 2: Translate a grayscale image to sounds

Expected Duration: 20 minutes

Learning Objectives

By the end of this activity, students will be able to:

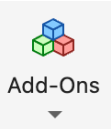
- Explain how digital images are represented by rows of pixels in MATLAB
- Understand how brightness values vary between 0 (black) and (1) in a grayscale image
- Define the concept and value of smoothing as a signal processing technique
- Define the concept and value of choosing an appropriate sampling period as an important step in signal processing

Materials

- MATLAB®
- *Optional*: webcam to take a picture or personal pictures uploaded to device

Steps

- Ask students to follow the instructions in Part 2 of the Student Activity Live Script sequentially.
- Ensure students *either* upload a picture from their device, load an example image, *or* take a picture with their webcam. If students would like to try all three, make sure they understand that they can only do the activity with one image at a time. Note that in order to take a picture with a webcam, the 'MATLAB Support Package for USB Webcams' must first be installed. To install the support package, click on 'Explore Add-Ons' from the 'Add-Ons' dropdown menu in the MATLAB 'Home' toolstrip (see image below for reference). Search for 'Webcam' in the Add-Ons library and install the support package.



- If you are able to take a picture of a student's drawing created for the micro:bit device activity, consider uploading the image for students to use, reinforcing the connection and distinction between the micro:bit device and MATLAB simulation.
- Ensure students first convert their image to grayscale.
- Reinforce the similarity between cutting a physical image into strips and selecting one row of a pixel matrix representing a digital image.
- After students plot the brightness values of the pixels in the grayscale image, give them time to try to map the brightness values to what they see in the image. Ask them if the brightness values make sense and match the shades of gray they see in the image.
- Ask students to try applying different amounts of smoothing to the brightness signal and discuss the pros and cons of smoothing signals. Have them try to pick an 'ideal' amount of smoothing that accurately represents the image. Discuss how smoothing changes the fidelity of the signal.
- Ask students to first try using a sampling period of 1 (play every pixel as a sound). Discuss why a sampling period longer than 1 would be more appropriate for the task. Conversely, discuss why a large sampling period (like every 100th pixel) is also not appropriate for the task. Have the students debate

what an appropriate sampling period would be - one that balancing saving time and resources with preserving the fidelity of the signal.
- To reinforce the concept of sampling period, have students engage in the listening challenge while trying different sampling periods. Ask students how different sampling periods affect their visualization of the image.
- Reinforce the connection between scanning words on a page with your eyes or your fingers and what MATLAB is doing by scanning the image from right to left, top to bottom. Ask students if they think the default sampling period chosen for the whole image demo is appropriate.

Part 3: Translate colors to sounds

Expected Duration: 20 minutes

Learning Objectives

By the end of this activity, students will be able to:

- Analyze how different systems (micro:bit vs MATLAB) interpret and convert different versions of the same visual signal, and discuss factors that affect accuracy and fidelity
- Appreciate the power of MATLAB to not only detect brightness, but also colors in an image
- Engage in identifying and reconstructing visual patterns using auditory cues, reinforcing the connection to building a device that could help those who are visually impaired
- Apply the simulated activity to a real-world example of how a device that plays different sounds for different colors could be used to help those who are visually impaired

Materials

- MATLAB®
- micro:bit device built previously
- Blank paper and colored markers or pens

Steps

- Ask students to follow the instructions in Part 3 of the Student Activity Live Script sequentially.
- Ensure students either select notes for each color to use *or* click 'Use default notes'. If selecting notes, ensure students select *different* notes for each color.
- Have students try generating several random colored stripe images and the corresponding sound translation. Note that students must click 'Play sounds' every time they generate a new image to hear corresponding the sound translation.
- Have students practice listening to the sounds that are played to represent each color and attempt the listening challenge. Students can use paper and pen to write down their guesses for colors as the sounds play.
- Have students attempt the micro:bit device challenge. Ask students to discuss the differences they perceive between the micro:bit device's function and the MATLAB simulation. Reinforce that MATLAB is able to detect the color of pixels in the image, while the micro:bit light sensor can only detect changes in ambient light. The amount of light that can pass through paper may differ depending on the color,

and that is what will drive the micro:bit to play different sounds for different colors. But the micro:bit light sensor itself cannot detect colors.

- Reinforce the concept that code in MATLAB can be deployed on hardware. Real engineers and scientists often simulate their device designs and functions in MATLAB first before launching them on hardware, to avoid wasting time and resources.