Mix:Tape Challenge

Last Revised: February 4, 2023

Table of Contents

Mix:Tape Challenge 2
Micro:Bit Simulator 2
Coding Languages 3
Challenge Rules 4
Judging & Scoring 5
Submission 5

Disclaimer

It is your responsibility to read and understand this document on a regular basis because we may update it from time to time.

If you have questions, please contact our Micro:Bit Team at orcinfo@ieeeottawa.ca

Mix:Tape Challenge

In the 1970s, people liked to create collections of music on cassettes and share them with each other for their friends to appreciate. Your task is to use a Micro:bit simulator to make a robotic mixtape: a music player that can play a collection of songs along with LED visuals. You can even share it with your friends!

Micro:Bit Simulator

For this challenge, you will be using a Micro:Bit simulator that is freely available to everyone. You can access it through this link https://makecode.microbit.org/ or download it through the Microsoft Store if you have a windows computer. You can also download the iOS or Android app through their respective app store but we would only recommend this option if you have a large tablet: https://www.microsoft.com/en-ca/p/makecode-for-micro-bit/9pjc7sv48lcx?activetab=pivot:overviewtab

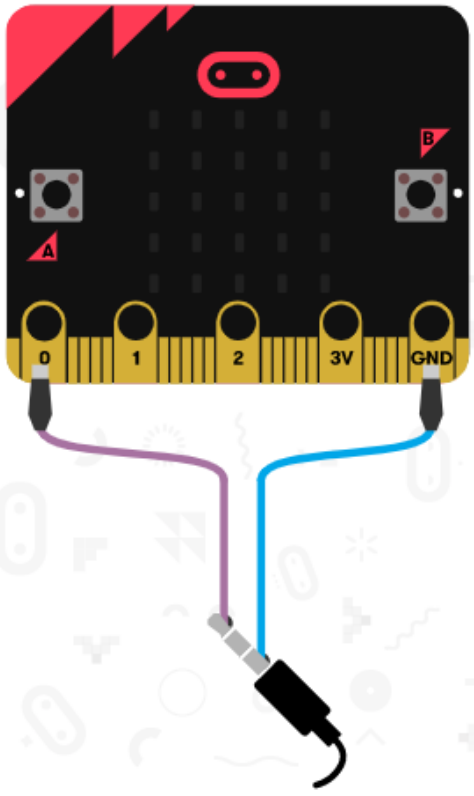


Image 1: Micro:Bit Simulator

However, if you have previously purchased the kit, you may use your physical Micro:Bit to test your code. This is optional, as you may find it

slower than just using the simulator in the software. If you decide to do this, the only relevant component will be the Micro:Bit chip. Regardless whether you decide to use the simulator or physical kit, you will only be submitting your code for judging and scoring and we will be testing your code on the simulator. You should however keep in mind that when in the context of the simulator, it is impossible to press two buttons at once. You can also use the pins as buttons when using the simulator.

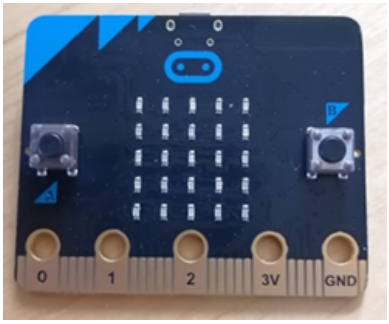


Image 2: Physical Micro:Bit

Coding Languages

The MakeCode editor allows you to program in code blocks, Javascript or Python. You may use any of the three of these to actually complete the work, but as our judging will be done in the coding blocks environment, it will be your responsibility to make sure that your code both functions in this environment, as well as being well organized.

There are some inconsistencies when moving from one coding environment to another. When moving from either Javascript or Python to the coding blocks environment, it will attempt to transform your melodies into the blocks representation of the melody, which is limited to a single octave, has no sharps and flats, and is only 8 notes long. However, it will only attempt to transform it to that representation if you directly put the music string into the playMelody function. If you save it as a string variable separately, it will correctly transform it into a codeblocks string and everything will work. Using proper variables for the melodies is the easiest way to make sure the melodies will not be transformed when going from Javascript or Python to codeblocks.

If you do not separately save it as a string variable, it will attempt to transform it. If you do not include flats and sharps, and your melody does not use notes that are outside of the code blocks' octave, it will succeed in transforming it and will also cut your melody off at 8 blocks long.

The code blocks environment can also take advantage of this behavior. You can play more complex melodies by saving the melody to a string variable and feeding that into the playMelody function.

The documentation is available here:
https://makecode.microbit.org/reference

Challenge Rules

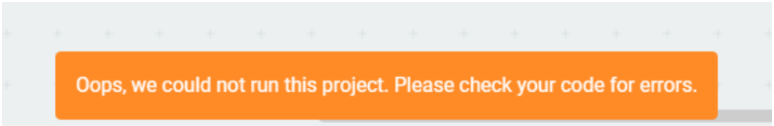
1. Your music player must include at least 3 melodies with a different icon displayed on the LED display for each melody.

2. Melodies should be at most 2 minutes long.

3. Your music player must play each melody in order, and restart the first melody after the last one.

4. Your music player must have at least two features using buttons (e.g. volume up/volume down, mute/unmute, reset to first melody, next/previous).

5. If your code produces an error message on program start, you will automatically get a score of 0. If it produces an error message for only certain inputs, points will be deducted.



6. You must submit an instruction manual that details how your music player works, showing a diagram of what inputs on the Micro:Bit are linked with which features, along with any additional information you deem necessary to understand how to operate your music player.

7. The instruction manual should not be more than 1 page. It can either be typed up, or handwritten and scanned, as long as it is legible. If it cannot be read, you will get a score of 0 for the section.

8. Any functions that work in the simulator are allowed, as this will be the way we test it. Radio functionality to have more than one simulated Micro:Bit is also explicitly allowed.

Judging & Scoring

1. Your code will be judged based on the concepts of good quality code. This includes visual organization of the code blocks, no code reuse, proper use of function/loops, proper naming of variables/functions, proper code comments. Judging will be performed in the coding blocks format.

2. Your music player will be judged on its creativity and originality. It should include melodies and LED icons programmed by you. You may use real songs and don't need to compose everything from scratch, but will be penalized if you use the predefined melodies.

3. All scoring will take your grade level into consideration.

4. The winner of the Challenge will be determined by the combined score of your instruction manual, your one page report, required code features, code quality, creativity, originality, and degree of difficulty. The team with the highest combined score will be the winner of the Challenge.

5. In addition, you can earn bonus points if you include more than 2 button features and/or if the songs are based on a theme that is described in the report.

6. Decisions of the judges are final.

Submission

To submit your work, you are required to fill out a form provided in the link provided below. This form requires your team name, your code as a .hex file, your instruction manual that can be scanned or typed and the one page report that must be typed. Both the instruction manual and report must be submitted as a .pdf file. The link to the submission form may be found here: http://www.orc.ieeeottawa.ca/submission/