Coding Fundamentals

Micro:bit Python Programming
Binary and Visual Counter

Overview

In this lesson, students learn about binary to create a binary counter using Python.

Objectives

- Explain the necessity of binary numbers
- Be able to translate back and forth between base 2 and base 10 number systems
- Program the Micro:bit to create a visual binary counter

Materials

- micro:bit and micro-USB cord
- Computer with access to the internet

Approx. Time Required

1-2 hours

Cyber Connections

- **Programming** – Students will program in Python.
- **Hardware and Software** – Students will utilize small electronics and learn how a computer is programmed while using microcontrollers.

CYBER.ORG
THE ACADEMIC INITIATIVE OF THE CYBER INNOVATION CENTER

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

Binary and Visual Counter

Understanding Binary

- This lesson may need to span multiple sessions, as the binary lesson itself can be quite long.

- Binary (or "base 2") is a numeric system that uses only two digits — 0 and 1. Computers and other technological devices operate using binary to store data or perform new calculations. This is possible because at the very base level of ANY computer-controlled technology, an electrical current is either on or it is off. The item is either "true" or "false," which we have seen briefly with our conditional operations. Editors or compilers (like the Python online editor) are used to take the code written by programmers like you and transform it into "machine code" comprised of binary that the computer can read and use. For this reason, anything numbered that is used by a computer ALWAYS starts at 0, much like the arrays used in earlier lessons.

- So how does it work? A binary number is composed of any number of "bits." A bit is the smallest possible unit to store computational data, as it contains only a 0 or a 1. Each additional bit added to a binary number increases its storing capabilities exponentially. This concept will become clearer with the worksheets and tables below. As you can see, bits increase storage capacity from right to left, with each bit capable of storing one more than all the bits to the right of it combined.

- If your students are unfamiliar with exponents, this is a great time to introduce them and show that that each number is just the last multiplied by 2 again. Binary numbers can range to any number of bits, but since our Micro:bit only has rows of 5 LEDs, we will stick to 5 bits. Student worksheets contain the table below as well as an empty table to practice all possible binary values with 5 bits.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
1	0	0	0	0	1	00001
=					1	1
31	1	1	1	1	1	11111
=	16	8	4	2	1	31
0	0	0	0	0	0	00000
=						0
6	0	0	1	1	0	00110
=			4	2		6
23	1	0	1	1	1	10111
=	16		4	2	1	23

Base 2 to Base 10

ALWAYS work right to left when converting to binary (base-2) numbers from base-10, filling in the largest possible bit that will fit the number first.

An example conversion for the number 14 is given below:

14

Does 2^4 or 16 fit into 14? No, so mark a 0 on the binary bit and move on to the next one.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
14	0					0 _ _ _ _

14

Does 2^3 or 8 fit into 14? Yes, so subtract 8 and place a 1 in the binary bit before moving on.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
14	0	1				01 _ _ _

6

Does 2^2 or 4 fit in to 6? Yes, so subtract 4 and place a 1 in the binary bit before moving on.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
14	0	1	1			011 _ _

2

Does 2^1 or 2 fit in to it? Yes, so subtract 2, and place a 1 in the binary bit before moving on.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
14	0	1	1	1		0111 _

0

Now that you've reach 0, all remaining bits should be set to 0. This gives us our final number: 01110.

Base 10 Number	2^4	2^3	2^2	2^1	2^0	Total
14	0	1	1	1	0	01110

Base 10 to Base 2

Going from binary to base 10 is even easier. You simply need to add the value displayed by all the 1s in the binary number. This is displayed in the table on page 2, with the white lines showing this form of conversion.

Binary Visual Counter

Once students are comfortable with what binary is, why it's used, and how to convert to it, it is time to tie it in to a Micro:bit program. The good news is that there are no new concepts or commands needed to write this program, students already have all the tools they need to make their next project: a binary visual counter.

The program should keep a counter that is increased every time the A button is pressed. Students should then be able to convert this counter to a binary number using the exact method show above. Here is a sample code of a working binary counter:

```
from microbit import *

binCounter = 0  #Counter to hold the number to convert to binary

while True:
    if button_a.is_pressed():    #Add one to counter if button is pressed and wait 500 ms
        binCounter = binCounter + 1
        sleep(500)

    elif button_b.is_pressed():  #Subtract one from counter if button is pressed wait 500 ms
        binCounter = binCounter - 1
        sleep(500)

    elif binCounter > 31:  #Reset if the counter exceeds the about of bits we can show
        binCounter = 0

    BVNum = binCounter            #A variable to hold the number
    #while we convert it

    if BVNum/16 >= 1:                 #If 16 fits into it, show a 1 and subtract 16
        display.set_pixel(0,2,9)
        BVNum = BVNum - 16

    else:                                #else show a 0
        display.set_pixel(0,2,0)

    if BVNum/8 >= 1:                    #If 8 fits into it, show a 1 and subtract 8
        display.set_pixel(1,2,9)
        BVNum = BVNum - 8
```

```
else:                                  #else show a 0
    display.set_pixel(1,2,0)

if BVNum/4 >= 1:                      #If 4 fits into it, show a 1 and subtract 4
    display.set_pixel(2,2,9)
    BVNum = BVNum - 4

else:                                  #else show a 0
    display.set_pixel(2,2,0)

if BVNum/2 >= 1:                      #If 2 fits into it, show a 1 and subtract 2
    display.set_pixel(3,2,9)
    BVNum = BVNum - 2

else:                                  #else show a 0
    display.set_pixel(3,2,0)

if BVNum >= 1:                        #If 1 fits into it, show a 1 and subtract 1
    display.set_pixel(4,2,9)

else:                                  #else show a 0
    display.set_pixel(4,2,0)
```

Things to note:

- Students will need 2 variable for this, one to count and one to convert.
- The counter will need to be reset once it reaches 31, as that is the maximum that can be displayed with 5 LEDs.
- The else statements to set the light to 0 are necessary, otherwise the LEDs will stay on from previous numbers.