MultiWingSpan

BBC micro:bit
Connecting micro:bits Together

Introduction

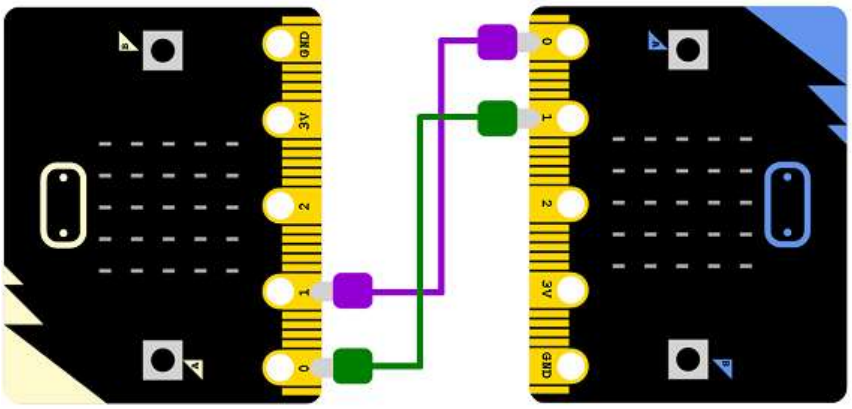
Using some alligator cables, you can connect two micro:bits together in order to send and receive messages.

The Circuit

We need to make 2 connections. On each micro:bit, we need two pins to work for us.

Name	Pin	Role
TX	pin 0	Transmit
RX	pin 1	Receive

We'll use these same pins on each micro:bit. You cross the cables so that the TX (transmit) pin on the left micro:bit attaches to the RX (receive) pin on the right hand one.



Programming - Version 1

This is Will's Python code converted from the block editor version of this project. The same program runs on both micro:bits.

```
from microbit import *
#variables
im = Image('99999:99999:99999:99999:99999:')

#done forever
while True:
    #detects when button A is pressed and sends pin 0 High
    if button_a.is_pressed():
        pin0.write_digital(1)
    #else pin 0 is low
    else:
        pin0.write_digital(0)
    #if pin 1 is read high turn the leds on
    mbit2 = pin1.read_digital()
    if mbit2==1:
       display.show(im)
    #else the display is off
    else:
        display.clear()
        sleep(10)
```

You can quite quickly put together some two player action once you have the connection up and running. If you put this version of the Shut The Matrix game on both micro:bits and are prepared to control taking turns yourself, you can have a two player game of sorts.

```
from microbit import *
import random

faces = [Image('00000:00000:00900:00000:00000:'),
        Image('00009:00000:00000:00000:90000:'),
        Image('00009:00000:00900:00000:90000:'),
        Image('90009:00000:00000:00000:90009:'),
        Image('90009:00000:00900:00000:90009:'),
        Image('90009:00000:90009:00000:90009:')]

def nleds(value):
    img = Image('00000:'*5)
    sp = img.set_pixel
    counter = 0
    for row in range(0,5):
        for col in range(0,5):
            if counter<value:
                sp(col,row,9)
            else:
                sp(col,row,0)
            counter += 1
    return img

def RandomImages(n, delay):
    for i in range(0,n):
            display.show(random.choice(faces))
            sleep(delay)
            display.clear()
            sleep(delay)

def PlayGame():
    counter = 0
    while counter!=25:
        if button_a.was_pressed():
            pin0.write_digital(1)
        else:
            pin0.write_digital(0)
        if pin1.read_digital()==1:
            display.clear()
            sleep(250)
            roll = random.randint(1,6)
            RandomImages(10,75)
            display.show(faces[roll-1])
            sleep(500)
            if counter+roll==25:
                # won
                counter = counter + roll
            elif counter+roll<25:
                # add on
                counter = counter + roll
            else:
            # go to end and come back
                counter = 50 - (counter + roll)
        display.show(nleds(counter))
        sleep(10)
    for i in range(0,10):
        display.show(nleds(25))
        sleep(200)
        display.clear()
        sleep(200)

# Start The Game
PlayGame()
```

Programming - Version 2

The problem with the approach taken so far is that we can only send and receive high and low signals. If we use the UART library, we can send and receive more complex messages. In this example, which can run on both micro:bits, text is sent and displayed on the screen.

```
from microbit import *

sleep(5000)
uart.init(baudrate=9600, bits=8, parity=None, stop=1, tx=pin0, rx=pin1)

def Send(m):
    b = []
    for c in m:
        b.append(ord(c))
    d = bytes(b)
    uart.write(d)

def Receive():
    if uart.any():
        m = uart.readall()
        s = ""
        for b in m:
            s = s + chr(b)
        display.scroll(s)

while True:
    Receive()
    if button_a.is_pressed():
        Send("HELLO")
        sleep(500)
    sleep(20)
```

The long delays here are simply for testing and being able to determine that things are happening as a result of expected inputs.

Challenges

1. With a bit of work, you could work out how to send image data. If you convert code from the Lights Out example game, you can make a method for allowing the user to draw an image on the matrix. Adapt the code from here and you can make it so that the image can be sent from the screen of one micro:bit to another.
2. You don't need to run exactly the same program on each micro:bit. Using the matrix of one micro:bit as an extension to the other would make for a much more interesting game. You could adapt one of the scrolling examples on this site to do that.
3. Connect two micro:bits together and two buzzers. Work carefully on the timing and see how close you can get to making some harmonic music.