Wearable Tech
A tour of some Micro:bit possibilities using MicroPython.

Introduction
The Micro:bit has been conceived so that there are endless possibilities with it. Just as Alan Turing postulated; the computer is a 'Universal Machine', it can change its use depending on the instructions that are given to it. The following activities are designed to allow your pupils to think that they could create anything from a computer. It might take some, or a lot, of imagination to see that the item created is what you want it to be, but that isn't the point. The point is to make some use out of the equipment given just by adding in some code.

Using Python on the Micro:bit website:
The Python section of the Micro:bit website is just a simple to use as the rest of the site. Once you have created your program you download it to your computer and then upload it to the Micro:bit using the file explorer. (Full, detailed explanations on the website itself.)

Using Mu
Mu is the offline IDE for creating programs in MicroPython that will work on the BBC Micro:bit. It works in much the same way as the Micro:bit website, but it has a powerful little tool called REPL that allows you to run one line at a time on the Micro:bit.

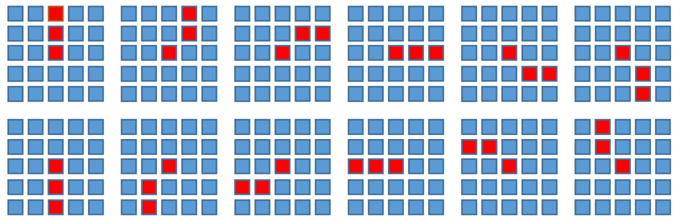
NB. Mu doesn't have a 'Save As…' feature. One way around this is to create a new program and then copy and paste the code you want to save as a new name into that file.

Things to bear in mind:
- It is assumed that you have a basic understanding of Python.
- The specific commands for MicroPython are available in this document. (https://microbit-micropython.readthedocs.io/en/latest/microbit_micropython_api.html)
- The code examples should work, but they are intended as a talking point. There might be many 'correct' answers. You can compare your solution to the one given and compare and contrast them.
- Don't forget to comment on your code. (That's just as important for you as it is for your students.)
- These resources should be used as a starting point. Feel free to expand, enhance and extend them.

The activities –
- Making a simple watch.
- Making an accurate watch.
- Make a Pedometer.
- Making an impact measurer.
- Using the Micro:bit for direction.
- Making a game.
- Combining all of the above apps into one multifunctioning, spectacular Apple watch rival at 1/200th of the price!

Activity 1 – Making a simple watch.
Aim –
To make a 'useable' watch using a Micro:bit.

How –
Using the built in images of clock faces, the Micro:bit can be used as a watch. Every 5 minutes the minute hand will shift to the next position.



Hints –
1. Use the following command to show a particular clockface:
   display.show(Image.CLOCK1)
2. Use the following command to wait for a number of milliseconds:
   sleep(5000) #This will cause the program to stop for 5 seconds before it starts again.
3. When testing this you might not want to wait for a full 5 minutes (300,000ms) to see if the display will change.

Activity 2 – Making an accurate watch.
Aim –
To make a watch that will be accurate to within a minute (or even better one second)

How -
With only 25 LEDs to work with, there isn't really enough space on the display to be able to show a full display of the time without coding the time in some way. That is your first problem to solve. After that it is just a case of displaying the time using the LEDs that are available using the code that you have devised.

Hints -
- Using binary numbers will allow a number higher than 5 to be displayed with 5 bits. E.g. Off, Off, On, Off, On could represent 5 and Off, On, Off, Off, On could represent 9 therefore representing 59 over two lines of the Micro:bit display.
- The display could flash in time with each passing second?
- Could the brightness of an LED be used to show the time?
- Can all of the sections of the display be linked to one variable? E.g. timePastMidnight

One Possible Design –
Row 1 represents the hours in binary 0 - 23; all 5 bits needed. 1 on far right, 16 on far left.

Row 2 represent the tens of minutes in denary; all 5 bits needed.

Rows 3 and 4 represent the units of minutes in denary. Row 3 shows 0-5, row 4 shows 6-9.

Row 5 (Optional) animation of the seconds

Activity 3 – Make a pedometer App.

Aim –

To make an app for the Micro:bit that will show how many steps have been taken.

How -

The accelerometer will detect the amount of force created on any of three directions (x, y or z). When you walk there will be change in the forces and therefore you can get the Micro:bit to detect when it has taken a step. If this detection can be linked to a variable, the Micro:bit can count up the number of steps that have been taken. It should be a doddle to display that on the screen; shouldn't it?

Hints -

1. Start off only using one of the directions on the accelerometer.
2. Test the accelerometer using REPL to see what a 'step' should feel like. NB. The 'step' could be a small tap on the body of the Micro:bit.
3. The display of the steps could be a number displayed on the screen. Will this work well with large numbers? I.e. thousands of steps.
4. Can the number of steps taken be displayed as a code on the screen? What will the code be?

Activity 4 – Making an impact measurer.

Aim –

This activity was inspired by the boxing wearable tech seen on the Gadget Show. There are devices available that can be used to track how well you exercise and therefore allow you to see how well you are progressing towards your exercise goals. The Gadget Show showed a small piece of tech that was clipped to a boxing glove so that it could measure the force of an impact on the training pad. This data was sent to a smartphone so that the data could be analysed by an app, ready for display on the screen. The Micro:bit can do all of this too...

Problems -

1. Which direction of the accelerometer will you use? Or two, or all three?
2. How will you combine the data from the accelerometer directions if you use more than one?
3. How will you display the information to be output on the LEDs?
4. How long will the display show the maximum force before it resets? Or will it manually reset?
5. How will you generalise this solution so that it is useful to the people that do not do boxing? I.e. most of us

Hints –

1. Create a variable to hold the current value from the accelerometer.
2. Create a variable to hold the maximum value that has been registered by the accelerometer.
3. Create a loop to persistently ask for the value held by the accelerometer.
4. Use the LEDs to display the values that are held in the variables.

Activity 5 – Using the Micro:bit to know where you are going.

Aim –

To create an app that will allow the user to know which direction they are going in.

How –

The Micro:bit has a magnet sensor on it so it can be used as a compass. The challenge is how to display this compass data on the screen so that it is useful to the person that will use it.

Problems -

1. Will you display the name of the compass point (N, E, S, W) on the screen depending on the direction that the Micro:bit is pointing in?
2. Will you create a display that always points North no matter which direction the Micro:bit points in?

Hints -

1. The compass needs to be calibrated first.
2. The direction of the compass can be accessed using ***compass.heading().***
3. Each direction of the compass can be represented using either the Clock face images e.g. ***display.show(Image.CLOCK1)*** or the direction arrows e.g. ***display.show(Image.ARROW_N)***

Activity 6 - Making a Game

Aim –

To create a game that will allow players to test their knowledge of binary numbers up to 31 (5 bits).

How –

- A binary number will be displayed on the top line of the screen taking up 5 LEDs.
- It'll be displayed for three seconds.
- In that time the player needs to work out which number is being displayed and call it out (or commit it to memory).
- The number will then be displayed in denary on the screen.

Hints –

Once you've had a go at this game, you could make up your own. And possibly more fun than working with binary numbers!

Activity 7 - Combining all of the above apps into one multifunctioning, spectacular Apple watch rival at 1/200th of the price!

Aim

To be able to have all of the Apps created on one large program that can be controlled using one (or more) of the buttons, A/B.

How

- Each of the previous apps could be placed into a function.
- If a button (A or B, you decide) is pressed then a variable can be incremented.
- Depending on the value of the variable, an if, elif, else statement can be used to call the appropriate function.

Hints

Sorry – over to you, if you've got this far, you'll have no problems.. There are hints on the readthedocs.org documentation.

Example Code –

Please feel free to adapt and amend as required. The code may inefficient, or might not work as you want it to work and it has not really been acceptance tested. However, it is a starting point…

Simple Watch code example –

```
from microbit import *
rest = 300000 #change this to a smaller value for
testing e.g. 1000
while True:
        display.show(Image.CLOCK1)
        sleep(rest)
        display.show(Image.CLOCK2)
        sleep(rest)
        display.show(Image.CLOCK3)
        sleep(rest)
        display.show(Image.CLOCK4)
        sleep(rest)
        display.show(Image.CLOCK5)
        sleep(rest)
        display.show(Image.CLOCK6)
        sleep(rest)
        display.show(Image.CLOCK7)
        sleep(rest)
        display.show(Image.CLOCK8)
        sleep(rest)
        display.show(Image.CLOCK9)
        sleep(rest)
        display.show(Image.CLOCK10)
        sleep(rest)
        display.show(Image.CLOCK11)
        sleep(rest)
        display.show(Image.CLOCK12)
        sleep(rest)
```

Accurate Watch Code Example –

```
from microbit import *
second = 1000
timePastMidnight = 0

def displayTime(tPM):
   displaySeconds(tPM)
   displayMinutes(tPM)
   displayHours(tPM)

def displaySeconds(tPM): #display the correct bits for
the seconds on row 5.
   noOfSecs = tPM // 1000 #the number of seconds
past midnight.
   while noOfSecs >= 60: #This will repeatedly remove
60 seconds until the value is less than 60.
      noOfSecs -= 60
   brightness = noOfSecs % 10 #the remainder when
the number of seconds is divided by 10
   if noOfSecs >= 50:
      display.set_pixel(0,4,brightness)
   if noOfSecs >= 40:
      display.set_pixel(1,4,brightness)
   if noOfSecs >= 30:
      display.set_pixel(2,4,brightness)
   if noOfSecs >= 20:
      display.set_pixel(3,4,brightness)
   if noOfSecs >= 10:
      display.set_pixel(4,4,brightness)
   if noOfSecs >= 0:
      display.set_pixel(0,2,brightness)
   else:
      display.set_pixel(0,0,0)

def displayMinutes(tPM):
   noOfMins = tPM // 60000
   while noOfMins >= 60: #This will repeatedly remove
60 minutes until the value is less than 60.
      noOfMins -= 60
#display the correct bits for the minutes on rows 2.
   if noOfMins // 10 == 5:
      display.set_pixel(4,1,9)
      display.set_pixel(3,1,9)
      display.set_pixel(2,1,9)
      display.set_pixel(1,1,9)
      display.set_pixel(0,1,9)
   elif noOfMins // 10 == 4:
      display.set_pixel(3,1,9)
      display.set_pixel(2,1,9)
```

```
        display.set_pixel(1,1,9)
        display.set_pixel(0,1,9)
    elif noOfMins // 10 == 3:
        display.set_pixel(2,1,9)
        display.set_pixel(1,1,9)
        display.set_pixel(0,1,9)
    elif noOfMins // 10 == 2:
        display.set_pixel(1,1,9)
        display.set_pixel(0,1,9)
    elif noOfMins // 10 == 1:
        display.set_pixel(0,1,9)
    else:
        display.set_pixel(0,0,0)
#display the correct bits for the minutes on rows 3
and 4.
    if noOfMins % 10 == 9:
        display.set_pixel(4,2,9)
        display.set_pixel(3,2,9)
        display.set_pixel(2,2,9)
        display.set_pixel(1,2,9)
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
        display.set_pixel(0,3,9)
    elif noOfMins % 10 == 8:
        display.set_pixel(4,2,9)
        display.set_pixel(3,2,9)
        display.set_pixel(2,2,9)
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
        display.set_pixel(0,3,9)
    elif noOfMins % 10 == 7:
        display.set_pixel(4,2,9)
        display.set_pixel(3,2,9)
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
        display.set_pixel(0,3,9)
    elif noOfMins % 10 == 6:
        display.set_pixel(4,2,9)
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
        display.set_pixel(0,3,9)
    elif noOfMins % 10 == 5:
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
        display.set_pixel(0,3,9)
    elif noOfMins % 10 == 4:
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
        display.set_pixel(1,3,9)
    elif noOfMins % 10 == 3:
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
        display.set_pixel(2,3,9)
    elif noOfMins % 10 == 2:
        display.set_pixel(4,3,9)
        display.set_pixel(3,3,9)
    elif noOfMins % 10 == 1:
        display.set_pixel(4,3,9)
    else:
        display.set_pixel(0,0,0)

def displayHours(tPM):
    noOfHrs = tPM // 3600000
    #display the correct bits for the hours on row 1 in
binary.
    if noOfHrs >= 8:
        display.set_pixel(1,0,9)
        noOfHrs -= 8
    if noOfHrs >= 4:
        display.set_pixel(2,0,9)
        noOfHrs -= 4
    if noOfHrs >= 2:
        display.set_pixel(3,0,9)
        noOfHrs -= 2
    if noOfHrs >= 1:
        display.set_pixel(4,0,9)
        noOfHrs -= 1
    else:
        display.set_pixel(0,0,0)

while True:
    display.clear()
    displayTime(timePastMidnight)
    timePastMidnight += 1000
    sleep(second)
```

NB. - This example code is incredibly inefficient and certainly not elegant in the way that is has been programmed. It therefore lends itself to looking at potential improvements (Evaluation) and spotting repetitions in the code (Pattern Recognition). The use of loops and arrays/lists can be used in a more developed solution.

Pedometer Code Example –

```
from microbit import *
steps = 0

while True:
    #display.clear()
    agx = accelerometer.get_x()
    if agx > 300:
        steps += 1
        sleep(100)
    if steps > 0:
        display.set_pixel(0,0,9)
    if steps > 10:
        display.set_pixel(1,0,9)
    if steps > 20:
        display.set_pixel(2,0,9)
    if steps > 30:
        display.set_pixel(3,0,9)
    if steps > 40:
        display.set_pixel(4,0,9)
    if steps > 50:
        display.set_pixel(0,1,9)
    if steps > 60:
        display.set_pixel(1,1,9)
    if steps > 70:
        display.set_pixel(2,1,9)
    if steps > 80:
        display.set_pixel(3,1,9)
    if steps > 90:
        display.set_pixel(4,1,9)
```

Impact Measurer Code Example –

```
from microbit import *

while True:
    #display.clear()
    agx = accelerometer.get_x()
    if agx > 0:
        display.set_pixel(0,0,9)
    if agx > 100:
        display.set_pixel(1,0,9)
    if agx > 200:
        display.set_pixel(2,0,9)
    if agx > 300:
        display.set_pixel(3,0,9)
    if agx > 400:
        display.set_pixel(4,0,9)
    if agx > 500:
        display.set_pixel(0,1,9)
    if agx > 600:
        display.set_pixel(1,1,9)
    if agx > 700:
        display.set_pixel(2,1,9)
    if agx > 800:
        display.set_pixel(3,1,9)
    if agx > 900:
        display.set_pixel(4,1,9)
```

Wearable Tech

A tour of some Micro:bit possibilities using MicroPython.

Compass Code Example –

```
from microbit import *
compass.calibrate()
x = True
while x:
    head = compass.heading()
    if head >= 330:
        display.show(Image.CLOCK1)
    elif head >= 300:
        display.show(Image.CLOCK2)
    elif head >= 270:
        display.show(Image.CLOCK3)
    elif head >= 240:
        display.show(Image.CLOCK4)
    elif head >= 210:
        display.show(Image.CLOCK5)
    elif head >= 180:
        display.show(Image.CLOCK6)
    elif head >= 150:
        display.show(Image.CLOCK7)
    elif head >= 120:
        display.show(Image.CLOCK8)
    elif head >= 90:
        display.show(Image.CLOCK9)
    elif head >= 60:
        display.show(Image.CLOCK10)
    elif head >= 30:
        display.show(Image.CLOCK11)
    elif head >= 0:
        display.show(Image.CLOCK12)
    else:
        display.clear()
    sleep(500)
```

Game Code Example –

```
from microbit import *
import random

def displayBinaryNumber():
    num = random.randint(1,31)
    binNum = bin(num)
    r = list(reversed(binNum[2:]))

    for i in range(len(r)):
        bit = r[i]
        x = 4 - i
        intensity = 1 if bit == "1" else 0
        display.set_pixel(x, 0, intensity)
    return num

while True:
    if button_b.is_pressed():
        num = displayBinaryNumber()
        sleep(3000)
        display.show(str(num))
        sleep(5000)
        display.clear()
```

Game Example Code

```
from microbit import *
 import random

 def displayBinaryNumber():
    num = random.randint(1,31)
    binNum = bin(num)
    r = list(reversed(binNum[2:]))

    for i in range(len(r)):
        bit = r[i]
        x = 4 - i
        intensity = 1 if bit == "1" else 0
        display.set_pixel(x, 0, intensity)
    return num

 while True:
    if button_b.is_pressed():
        num = displayBinaryNumber()
        sleep(3000)
        display.show(str(num))
        sleep(5000)
        display.clear()
```