A simple stopwatch using micro:bit & Microsoft Javascript Blocks Editor

The basic micro:bit device does not have a real time clock but it does have a neat feature that counts, in milliseconds (*thousandths of a second*), how long the device has been running since it was switched on or reset.

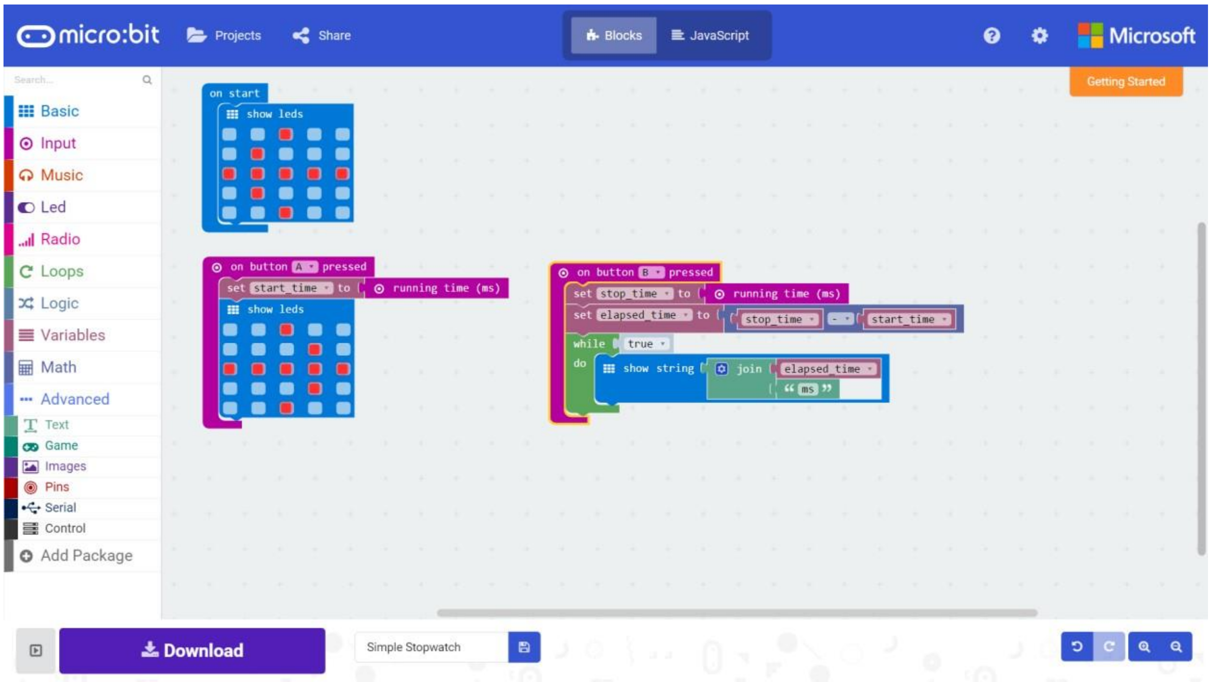
By using the **Microsoft Javascript Blocks** editor at www.microbit.org you can access this 'running time' information and use it to count time. With a little **maths** and the use of **variables** an accurate stopwatch can be made that uses the **buttons** on the micro:bit to stop and start and the LED display to show the time.

USING YOUR STOPWATCH:

Press A to start. Press B to stop. As soon as you press B the micro:bit will repeatedly display the time in milliseconds. *(You could divide this by 1000 to display seconds but that might not be accurate enough for rocket cars!!!)*

To take another reading, simply press the 'reset' button on the back of the micro:bit and a moment later you will be ready to record another time. Easy!

THE COMPLETE PROGRAM:

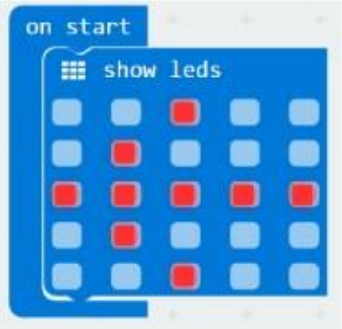


```
on start
    show leds

on button A pressed
    set start_time to running time (ms)
    show leds

on button B pressed
    set stop_time to running time (ms)
    set elapsed_time to stop_time - start_time
    while true
    do
        show string join elapsed_time "ms"
```

See the next page to see how to make the program yourself

Code a stopwatch using a micro:bit

PUTTING THE PROGRAM TOGETHER

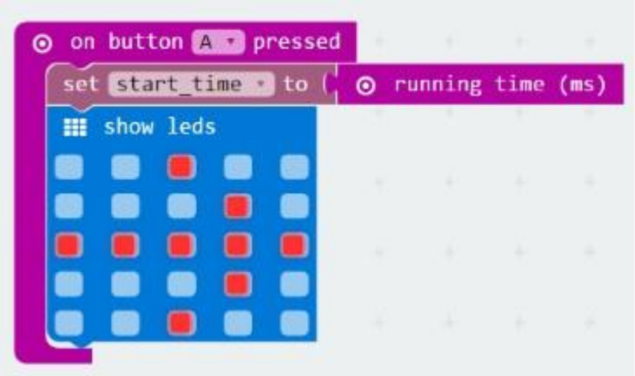


```
on start
    show leds
        □ □ ■ □ □
        □ ■ □ □ □
        ■ ■ ■ ■ ■
        □ ■ □ □ □
        □ □ ■ □ □
```

From the **Basic** section, drag a **show leds** block out and put it in the **on start** block on the screen. Click the small squares to make an arrow shape that will appear on the micro:bit.

As it is not attached to an **input** or other event, this first block will work as soon as the micro:bit is switched on or reset.

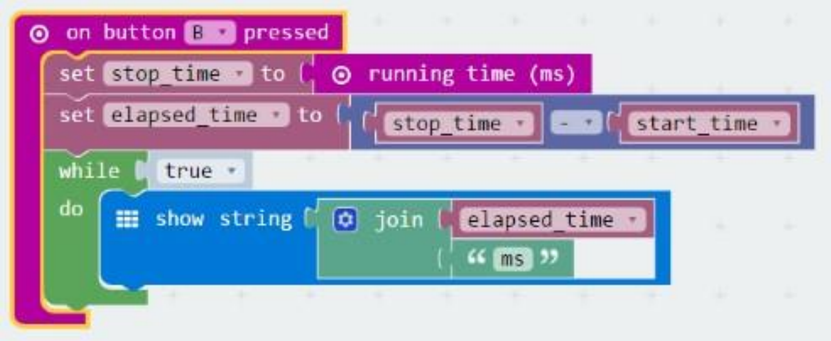
It will display an arrow pointing at the A button (on the left) encouraging the user to press the button to start the timer.



```
on button A pressed
    set start_time to running time (ms)
    show leds
        □ □ ■ □ □
        □ □ □ ■ □
        ■ ■ ■ ■ ■
        □ □ □ ■ □
        □ □ ■ □ □
```

This section of code starts the stopwatch.

Drag out an **on button ___ pressed_do** block from the **Input** section. Set it to button A. From the **Variables** section drag out a **set ____ to** block and attach **running time (ms)** from the **Input** > **More** section to it. Name the variable *'start_time'*. Put both inside the button block. Add another **show leds** block and set it to draw and arrow pointing the other way.



```
on button B pressed
    set stop_time to running time (ms)
    set elapsed_time to stop_time - start_time
    while true
    do
        show string join elapsed_time
                         " ms "
```

Finally construct this code to stop the timer and display the time.

When the B button is pressed another variable called *stop_time* is set to the running time value.

The next line uses a **Maths** block and calculates the time elapsed between the presses by simply subtracting the *start_time* from the *stop_time* to make a new variable*, elapsed_time*

The final part of this block performs a continuous loop that displays the calculated *stopwatch* time and the units (ms) combined as a text string. The **join** block to do this is in the **Advanced > Text** section.

Now download your program and give it a go!

TAKING IT FURTHER:

You could use external switches connected to the pins to stop and start the stopwatch or maybe design a case. You could also add a buzzer to make beeps when buttons are pressed.