# Events

**Event Blocks**

# 

The set-up block is required by all programs as it works in the background to import the library required for our programs to run. This block is the equivalent of Arduino's Void Setup function and contains functions that you only want to run once at the beginning of the program.

In the Blockly windows we can't see much but if we switch to the </> Python window we see the following Micropython code.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

We can see in this code snippet that this block imports the three core libraries required by all programs and sets the initial screen colour to an almost black colour. By changing the value highlighted in red, we can change the initial screen colour. For more information on setting colours, check out the colour blocks.

## Loop

**Event Blocks**

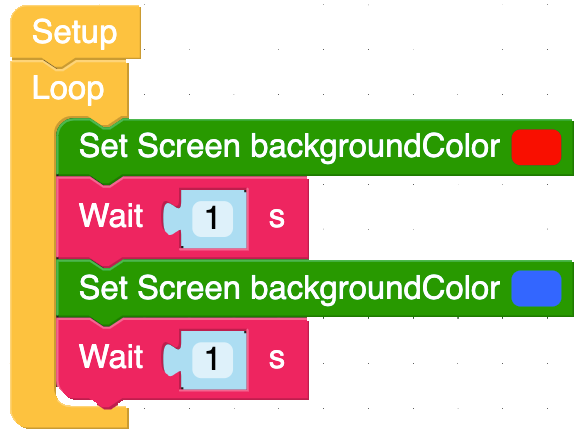
## 

After this we have the main program loop. Inside of this block we place the main program that we want to continuously repeat.

The Micropython code for this block is

**while True:**

Example



In the following example I am using a loop to continuously step through the Set screen Background Colour block. If this is run without the wait blocks then the screen will look purple because the screen is changing faster then the human eye can see. To be able to see the colour change I have added a wait block to make the program wait one second before moving between the blocks.

**Event Blocks**

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

**while True:**

**setScreenColor(0xff0000)**

**wait(1)**

**setScreenColor(0x3366ff)**

**wait(1)**

**wait\_ms(2)**

For more information on the wait block, check out the timer section of the book.

## Button Loop

**Event Blocks**

## 

As well as the main loop, we have the button loop that continuously watches the three main buttons on the front of the M5Stacks and then runs the code inside it when it detects a triggering even.

The four events the button loop waits for are as follows.

**Was Pressed -** This is the normal mode of operation and waits until a single short press of the button is detected before running the code inside it.

**Was Released -** This waits until the button is pressed and released before running the code placed inside it.

**Long Press -** Like the "Was Pressed" block, it waits until the button is pressed however, this block on activates if the button is pressed and held down for a few seconds.

**Was Double Pressed -** This function checks to see if the put-on was pressed and released twice within a few milliseconds before running the code inside it.

The Micropython code for the Button Loop is as follows.

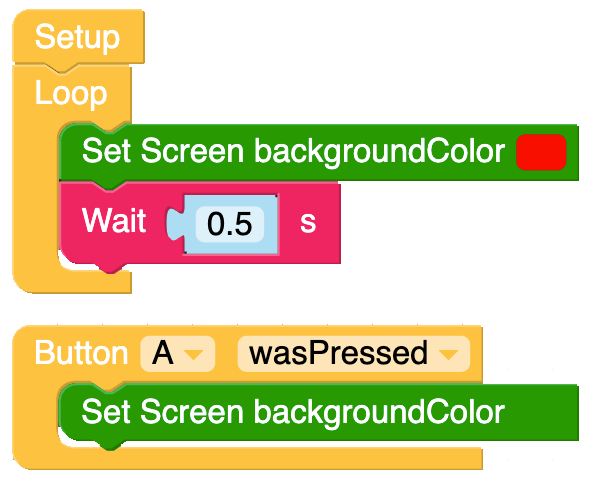
**def buttonA\_wasPressed():**

**# global params**

**pass**

**btnA.wasPressed(buttonA\_wasPressed)**

Example



**Event Blocks**

In this example I have two loops. The main loop set the screen to red and the button loop changes the screen to green when the button is pressed. The main loop is required to reset the screen colour after the button has been released.

The Micropython code for this example is as follows.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

**def buttonA\_wasPressed():**

**# global params**

**setScreenColor(0x009900)**

**pass**

**btnA.wasPressed(buttonA\_wasPressed)**

**while True:**

**setScreenColor(0xff0000)**

**wait(0.5)**

**wait\_ms(2)**

## Obtain Button Value,

**Event Blocks**

The obtain button block works in a similar mode to the Button loop but instead of running code placed inside it, it is a value block that returns true or false when one of the button events have been triggered.

The four events the button value block waits for are as follows.

**Was Pressed -** This is the normal mode of operation and waits until a single short press of the button is detected before running the code inside it.

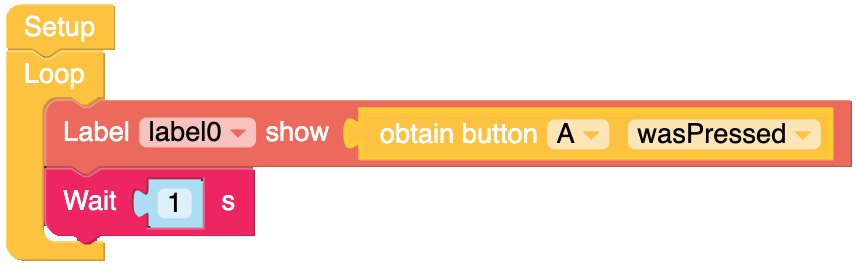
**Was Released -** This waits until the button is pressed and released before running the code placed inside it.

**Long Press -** Like the "Was Pressed" block, it waits until the button is pressed however, this block on activates if the button is pressed and held down for a few seconds.

**Was Double Pressed -** This function checks to see if the put-on was pressed and released twice within a few milliseconds before running the code inside it.

The Micropython code for the Button Loop is as follows.

**btnA.wasPressed()**

Example

**Event Blocks**

In this example I am using a Label block to show the status of the buttons.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

**label0 = M5TextBox(95, 100, "Text", lcd.FONT\_Default,0xFFFFFF, rotate=0)**

**while True:**

**label0.setText(str(btnA.wasPressed()))**

**wait(1)**

**wait\_ms(2)**

## Button A+B Press Loop

**Event Blocks**

Button A+B press block expands on the function of the button loop by waiting until it detects that any two buttons on the front of the m5stack that have been selected in the drop down boxes have been pressed at the same time.

The Micropython code for the Button Loop is as follows.

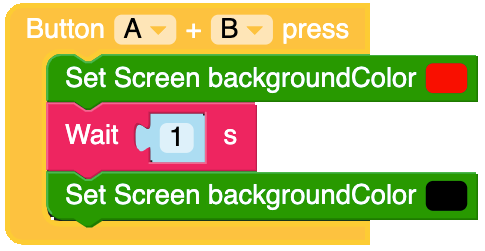
**def multiBtnCb\_AB():**

**# global params**

**pass**

**btn.multiBtnCb(btnA,btnB,multiBtnCb\_AB)**

Example



In the following example the screen will turn red when both A and B is pressed and then if the buttons have been released for more than a second, will turn black.

The Micropython code for this example is as follows.

**Event Blocks**

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x111111)**

**def multiBtnCb\_AB():**

**# global params**

**setScreenColor(0xff0000)**

**wait(1)**

**setScreenColor(0x000000)**

**pass**

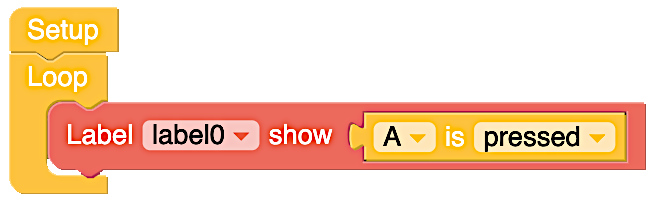
**btn.multiBtnCb(btnA,btnB,multiBtnCb\_AB)**

## Button Value

**Event Blocks**

## 

Similar to the Obtain Button function, this block also monitors buttons A, B, and C on the front panel of the M5Stacks but is simplified to only having options for Pressed and Released.

Example 1,

In the following example a label with show false until button A is pressed, the label will show true until A button is released.

The Micropython code for this example is as follows.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

**label0 = M5TextBox(95, 100, "Text", lcd.FONT\_Default,0xFFFFFF, rotate=0)**

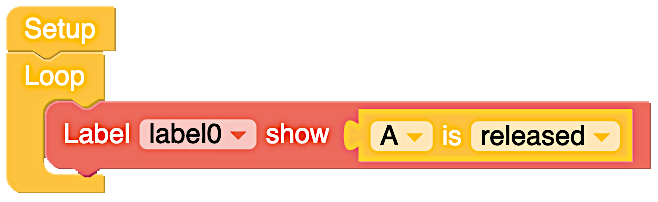
**while True:**

**label0.setText(str(btnA.isPressed()))**

**wait\_ms(2)**

Example 2,

**Event Blocks**



In this following example a label with show false while button A is pressed, the label will show true when A button is released.

The Micropython code for this example is as follows.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

**label0 = M5TextBox(95, 100, "Text", lcd.FONT\_Default,0xFFFFFF, rotate=0)**

**while True:**

**label0.setText(str(btnA.isReleased()))**

**wait\_ms(2)**

# Timers (Part 1)

**Event Blocks**

## Timer Callback Loop.

## 

The timer callback look creates a timer that can contain actions that are set to run under certain time critical events. Click on the light box with **"timer1"** in it to change the name of the timer.

The Micropython code for the loop is as follows:

**@timerSch.event('timer1')**

**def ttimer1():**

**# global params**

**pass**

## Set timer Period



Defines the time the timer will run for. Periodic mode set the timer to continuously run for the defined period where as "One Shot" will set it run once. The time value can be set with a maths block or a variable block. The operating mode can be changed by clicking the down arrow on the right of the block.

The Micropython code for set timer Period block in periodic mode is

**timerSch.setTimer('Example', 100, 0x00)**

and for "One Shot" mode,

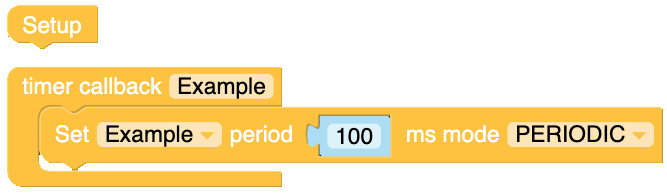
**timerSch.setTimer('Example', 100, 0x01)**

It is worth noting that the only difference in the code is the 0x00 used to set "Periodic" mode and 0x01 to set "One Shot" mode in Micropython.

**Event Blocks**

Examples

The following examples show how to set the different modes.



The Micropython code for this example is as follows.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x111111)**

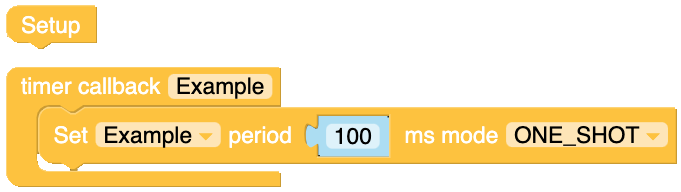
**@timerSch.event('Example')**

**def tExample():**

**# global params**

**timerSch.setTimer('Example', 100, 0x00)**

**pass**

****

The Micropython code for this example is as follows.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x111111)**

**@timerSch.event('Example')**

**def tExample():**

**# global params**

**timerSch.setTimer('Example', 100, 0x01)**

**pass**

## Start Timer

**Event Blocks**



Starts the timer for a defined period of time. The time value can be set with a maths block or a variable block. The operating mode can be changed by clicking the down arrow on the right of the block.

The Micropython code for the stop block is:

**timerSch.run('Example', 100, 0x00)**

For "Periodic" mode and

**timerSch.run('Example', 100, 0x01)**

For "One Shot" Mode.

Example.

## Stop Timer

**Event Blocks**

## 

Stops the timer selected in the drop down box.

The Micropython code for the stop block is:

**timerSch.stop('Example')**

Example.

The following example (as mentioned earlier) has been created to show how the various time blocks react with each other but also rely on each other to work.

The Micropython code for this is quite long and is more advanced then previous demos'