# Eventi

**Event Blocks**

# 

Il blocco di configurazione è richiesto da tutti i programmi poiché funziona in background per importare le librerie necessaria per l'esecuzione dei nostri programmi. Questo blocco è l'equivalente della funzione Void Setup di Arduino e contiene funzioni che si desidera eseguire solo una volta all'inizio del programma.

Nelle finestre Blockly non vediamo molto ma se passiamo alla finestra </> Python possiamo vedere il seguente codice Micropython.

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

Possiamo vedere in questo frammento di codice che il blocco importa le tre librerie principali richieste da tutti i programmi e imposta il colore dello schermo iniziale su un colore quasi nero. Modificando il valore evidenziato in rosso, possiamo cambiare il colore della schermata iniziale. Per ulteriori informazioni sull'impostazione dei colori, controlla i blocchi di colore.

## Loop

**Event Blocks**

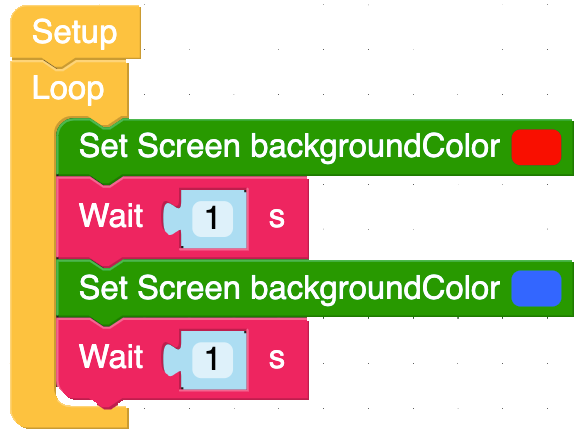
## 

Dopo questo abbiamo il ciclo principale del programma. All'interno di questo blocco posizioniamo il programma principale che vogliamo ripetere continuamente. (Loop continuo)

Il codice Micropython per questo blocco è:

**while True:**

Esempio:



Nell'esempio seguente usiamo un ciclo per scorrere continuamente il blocco del colore di sfondo della schermata Imposta. Se viene eseguito senza i blocchi di attesa, lo schermo apparirà viola perché lo schermo cambia più velocemente di quanto l'occhio umano possa vedere.

Per poter vedere il cambio di colore viene aggiunto un blocco di attesa per far aspettare il programma un secondo prima di spostarsi tra i blocchi.

**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)**

Per ulteriori informazioni sul blocco di attesa, controlla la sezione del timer.

## Loop Pulsanti

**Event Blocks**

## 

Oltre al loop principale, abbiamo il loop di pulsanti che guarda continuamente i tre pulsanti principali sulla parte anteriore di M5Stacks e quindi esegue il codice al suo interno quando rileva un evento di trigger.

I quattro eventi che il loop di pulsanti attende sono i seguenti:

**Was Pressed (pulsante premuto):** questa è la normale modalità di funzionamento e attende fino a quando non viene rilevata una singola pressione breve del pulsante prima di eseguire il codice al suo interno.

**Was Released (**pulsante rilasciato): attende che il pulsante venga premuto e rilasciato prima di eseguire il codice inserito al suo interno.

**Long Press (**pressione lunga) - Come il blocco "Was Pressed", attende fino a quando il pulsante viene premuto, tuttavia questo blocco si attiva se il pulsante viene premuto e tenuto premuto per alcuni secondi.

**Was Double Pressed** (Doppia pressione) - Questa funzione controlla se il pulsante è stato premuto e rilasciato due volte entro pochi millisecondi prima di eseguire il codice al suo interno.

Il codice Micropython per Button Loop è il seguente.

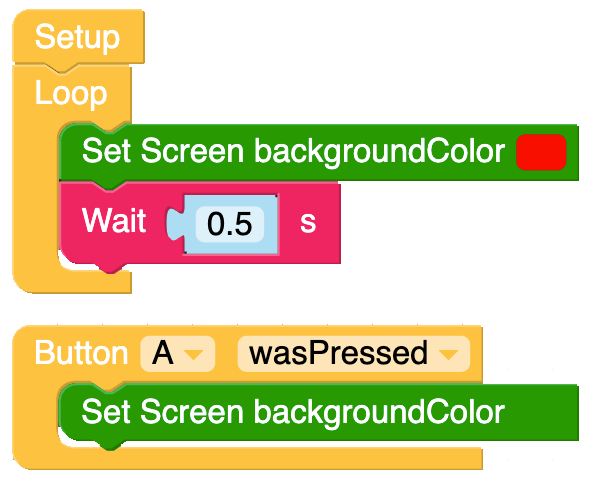
**def buttonA\_wasPressed():**

**# global params**

**pass**

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

**Esempio**



**Event Blocks**

In questo esempio abbiamo due loop. Il loop principale imposta lo schermo in rosso ed il loop dei pulsanti cambia lo schermo in verde quando viene premuto il pulsante. Il ciclo principale è necessario per ripristinare il colore dello schermo dopo che il pulsante è stato rilasciato.

Il codice Micropython per questo esempio è il seguente:

**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)**

## Acquisizione del valore del pulsante,

**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.

Il blocco del pulsante di acquisizione funziona in una modalità simile al ciclo del pulsante, ma invece di eseguire il codice inserito al suo interno, è un blocco di valori che restituisce **Vero** o **Falso** quando uno degli eventi del pulsante è stato attivato.

I quattro eventi che il blocco del valore del pulsante attende sono i seguenti:

**Was Pressed (**pulsante premuto):

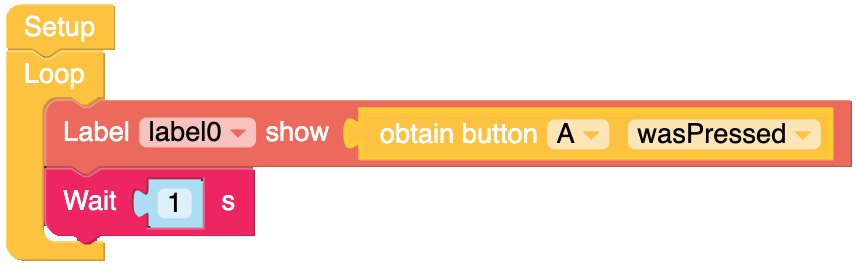
**Was Released (**pulsante rilasciato):

**Long Press (**Pressione lunga)

**Was Double Pressed (**doppia pressione

Il codice Micropython per Obtain Loop è il seguente:

**btnA.wasPressed()**

**Esempio**

**Event Blocks**

In questo esempio stiamo usando un blocco Etichetta per mostrare lo stato dei pulsanti.

**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)**

## Ciclo pressione pulsante A+B

**Event Blocks**

Il blocco con il ciclo di pressione dei pulsanti A + B espande la funzione del loop di pulsanti aspettando fino a quando non rileva che due pulsanti qualsiasi sulla parte anteriore di m5stack ( selezionati nelle caselle a discesa del blocco), sono stati premuti contemporaneamente.

Il codice Micropython è il seguente.

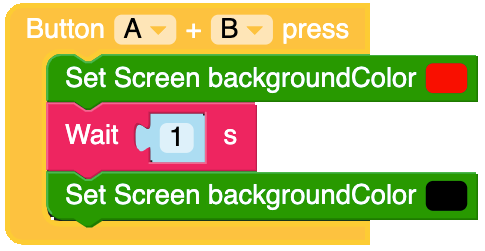
**def multiBtnCb\_AB():**

**# global params**

**pass**

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

Esempio:



Nell'esempio seguente lo schermo diventerà rosso quando vengono premuti sia A che B e quindi, se i pulsanti sono stati rilasciati per più di un secondo, diventerà nero. Il codice Micropython per questo esempio è il seguente.

**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**

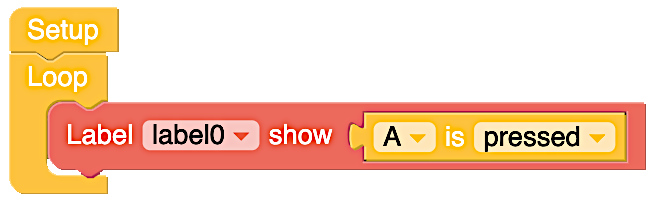
**btn.multiBtnCb(btnA,btnB,multiBtnCb\_AB)**

## Valore del pulsante

**Event Blocks**

## 

Simile alla funzione Acquisisci pulsante, questo blocco monitora anche i pulsanti A, B e C sul pannello frontale di M5Stacks, ma è semplificato per avere solo le opzioni Premuto e Rilasciato.

Esempio 1:

Nell'esempio seguente un'etichetta con show false fino a quando non viene premuto il pulsante A, l'etichetta mostrerà true fino al rilascio del pulsante A.

Il codice Micropython per questo esempio è il seguente:

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x222222)**

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

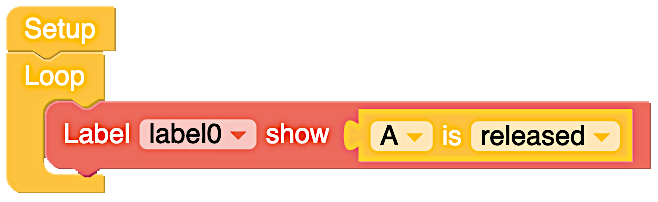
**while True:**

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

**wait\_ms(2)**

Esempio 2,

**Event Blocks**



Nell'esempio seguente un'etichetta con show false mentre si preme il pulsante A, l'etichetta mostrerà true quando viene rilasciato il pulsante A.

Il codice Micropython per questo esempio è il seguente.

**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 (Parte 1)

**Event Blocks**

## Timer Callback Loop.

## 

L'aspetto del callback del timer crea un timer che può contenere azioni impostate per l'esecuzione in determinati eventi critici. Fare clic sulla casella contenente "**timer1**" per modificare il nome del timer.

Il codice Micropython per il ciclo è il seguente:

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

**def ttimer1():**

**# global params**

**pass**

## Set timer Period



Definisce il tempo di esecuzione del timer. La modalità “Periodic” imposta il timer in modo che funzioni continuamente per il periodo definito, "One\_Shot" lo imposterà per una sola volta. Il valore temporale può essere impostato con un blocco matematico o un blocco variabile. La modalità operativa può essere modificata facendo clic sulla freccia in basso a destra del blocco.

Il codice Micropython per impostare il blocco del periodo del timer in modalità periodica è:

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

E per la modalità "One Shot":

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

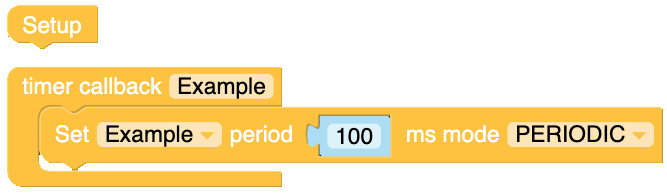
Vale la pena notare che l'unica differenza nel codice è lo 0x00 utilizzato per impostare la modalità "Periodica" e 0x01 per impostare la modalità "One Shot" in Micropython.

**Event Blocks**

**Esempi**

I seguenti esempi mostrano come impostare le diverse modalità.

PERIODIC



Il codice Micropython per questo esempio è il seguente:

**from m5stack import \***

**from m5ui import \***

**from uiflow import \***

**setScreenColor(0x111111)**

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

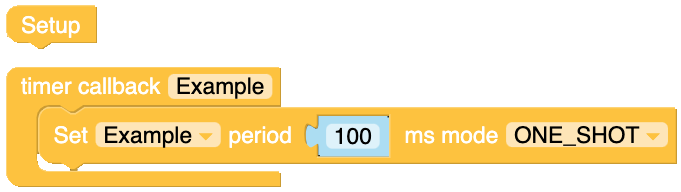
**def tExample():**

**# global params**

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

**pass**

**ONE\_SHOT**

****

Il codice Micropython per questo esempio è il seguente.

**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**



Avvia il timer per un periodo di tempo definito. Il valore temporale può essere impostato con un blocco matematico o un blocco variabile. La modalità operativa può essere modificata facendo clic sulla freccia in basso a destra del blocco.

Il codice Micropython per il blocco stop è:

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

Per la modalità "Periodic"

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

Per la modalità "One\_Shot"

**Esempio**

## Stop Timer

**Event Blocks**

## 

Arresta il timer selezionato nella casella a discesa.

Il codice Micropython per il blocco stop è:

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

Esempio.

Il seguente esempio (come accennato in precedenza) è stato creato per mostrare come i vari blocchi temporali interagiscono tra loro, ma si basano anche l'uno sull'altro per funzionare.

Il codice Micropython per questo è piuttosto lungo ed è più avanzato rispetto alle precedenti demo '