Micro:bit e velocità

Prove per misurare i tempi di esecuzione di un semplice loop “per sempre”

Quanto impiega microbit a completare un ciclo?

Ci sono applicazioni di microbit che prevedo che accada qualcosa in seguito ad un evento rilevato da microbit.

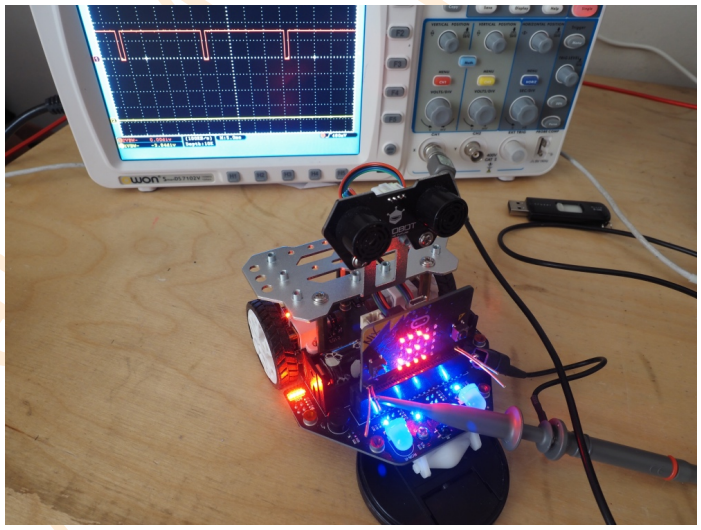
Si può immaginare di usare microbit in un termostato: quando la temperatura super 25° allora spegnere il riscaldatore.

I tempi lunghi del riscaldamento non destano di solito preoccupazioni.

Se invece si tratta di comandare un robot che avanza e deve rilevare un ostacolo per fermarsi, i tempi di rilevazione e reazione diventano importanti con conseguenze che dipendono dal tipo di ostacolo.

L’apparato

Per rilevare i tempi è stato usato un oscilloscopio



che permette di osservare i momenti di transizione del pin P0 da alto a basso e viceversa grazie a due blocchi appositamente inseriti nel ciclo.

Il codice è stato scritto usando MakeCode.

La sonda dell'oscilloscopio è posta fra il pin P0 e GND.

***

Nella fattispecie microbit è installato su un robot perché il problema è nato intorno alla necessità di fermare il robot usando una banda nera lungo il percorso posta di traverso per essere rilevata dai sensori di traccia.

In alcune circostanze il robot non si fermava in tempo e superava la barriera proseguendo oltre.

Ci sono due fatti che portano a questo risultato:

- i tempi di reazione,
- l'inerzia.

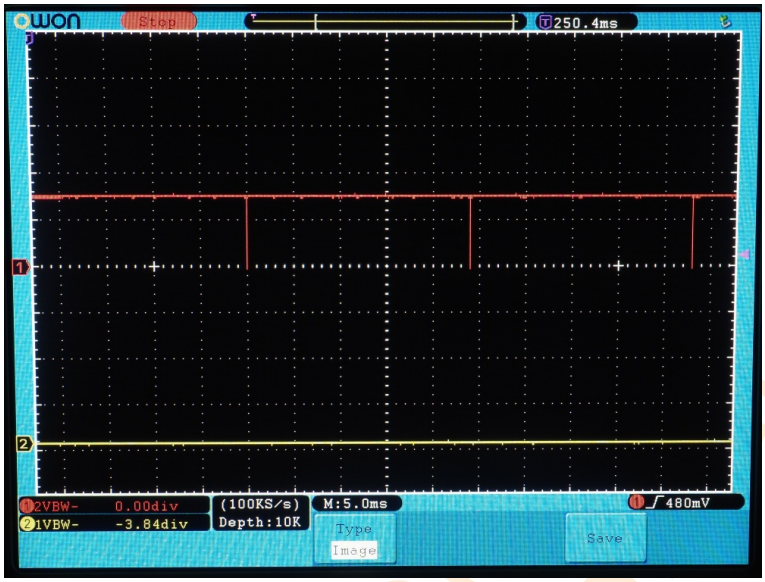
In questa scheda si indaga sul primo punto: esiste un intervallo di tempo minimo tra due successi rilevazioni fatte dal sensore?

Non è facile sapere quando il sensore fa una rilevazione e quando fa quella successiva ma è possibile avere indicazioni massima con degli esperimenti.

A) Codice brevissimo

Si inseriscono solo i due blocchi per impostare i valori basso e alto del pin P0:

L'oscilloscopio restituisce il seguente oscillogramma:



Nell'oscillogramma si vedono tempi lunghi al valore e brevissimi all valore basso.

La permanenza al valore basso è data dal tempo necessario per eseguire il secondo blocco per forzare il pin P0 da 0 a Vcc mentre la permanenza al valore alto è data dal tempo necessario ad eseguire la chiusura del ciclo, la ripetizione dello stesso e l'esecuzione del primo blocco per forzare il pin P0 da Vcc a 0.

Si ricava he la maggior parte del tempo viene utilizzata dalla chiusura e ripetizione del ciclo che sono operazioni che esegue il codice del controllore sulla base delle indicazioni date dal codice scritto con MakeCode.

Il periodo ha una durata che si aggira intorno a 24[1] ms.

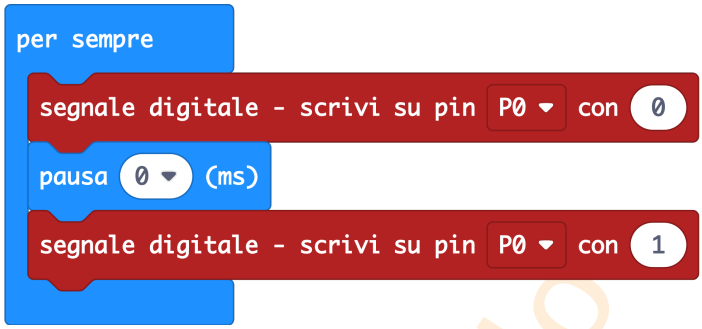
Per capire se sia tanto o poco bisogna pensare ad un robot che viaggia a 10 cm/s: un ostacolo può essere rilevato con un ritardo di 24 ms cioè dopo che il robot ha percorso 2,4 mm.

Velocità più elevate comportano spazi di latenza più lunghi

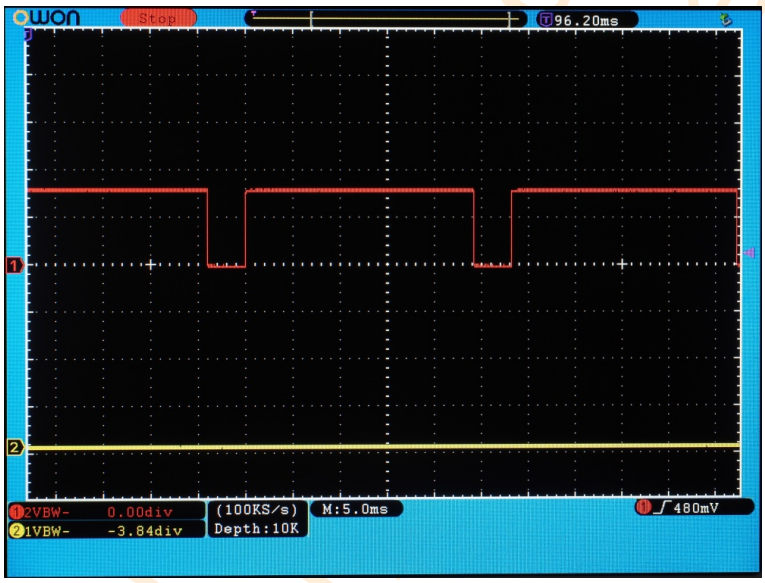
B) Codice con ritardo

Per osservare meglio si inserisce un ritardo di 0 ms tra i due blocchi; il blocco inserisce un ritardo nullo ma chiama una serie di istruzioni che comunque vanno eseguite:

[1] *È stata usata una scheda microbit V1.*

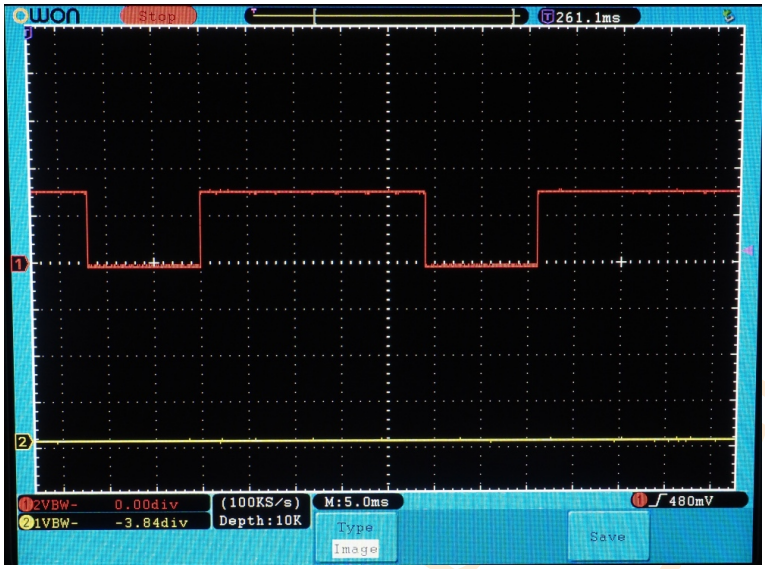


Con il seguente risultato:



Il semplice inserimento di un blocco di ritardo comporta un allungamento dello stato di P0 al valore basso della durata di 4 ms per un periodo complessivo di 28 ms.

Se il ritardo inserito è di 10 ms si ha:



con un periodo che è salito a oltre 35 ms come ci si poteva attendere.

***

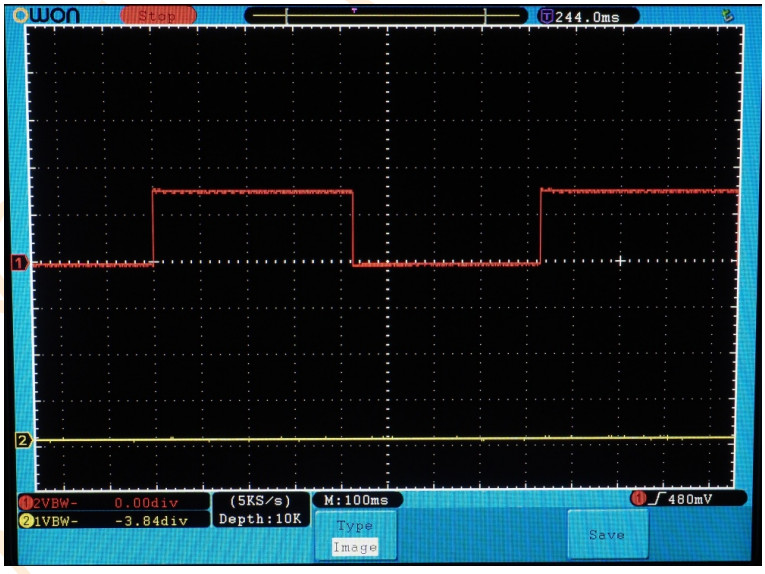
I tempi sono imposti dalla durata di esecuzione di ciascun blocco.

C) Attivazione del display

Con il codice:


per sempre
segnale digitale - scrivi su pin P0 con 0
mostra LED
segnale digitale - scrivi su pin P0 con 1
mostra LED

si ottiene:


Stop
244.0ms
2VBW- 0.00div
1VBW- -3.84div
(5KS/s)
Depth:10K
M:100ms
480mV
Type
Image
Save

Lo stato alto viene mantenuto per 420 ms e lo stato basso per 400 ms per un totale di 820 ms.

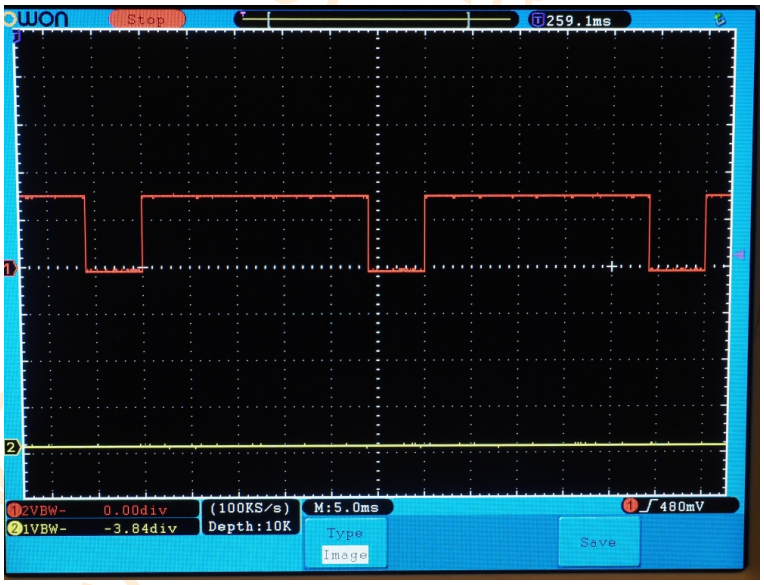
Un robot che viaggia alla velocità di 10 cm/s e si permette il lusso di comandare il display percorre 8,2 cm fra una rilevazione e la successiva: la guida è praticamente fuori controllo.

Vietato usare il display quando si guida!

D) Pulsanti A e B

Vengono aggiunti al caso esaminato in B) i pulsanti A e B senza blocchi al loro interno solo per sapere se questa aggiunta influenza i tempi:

Il risultato è:

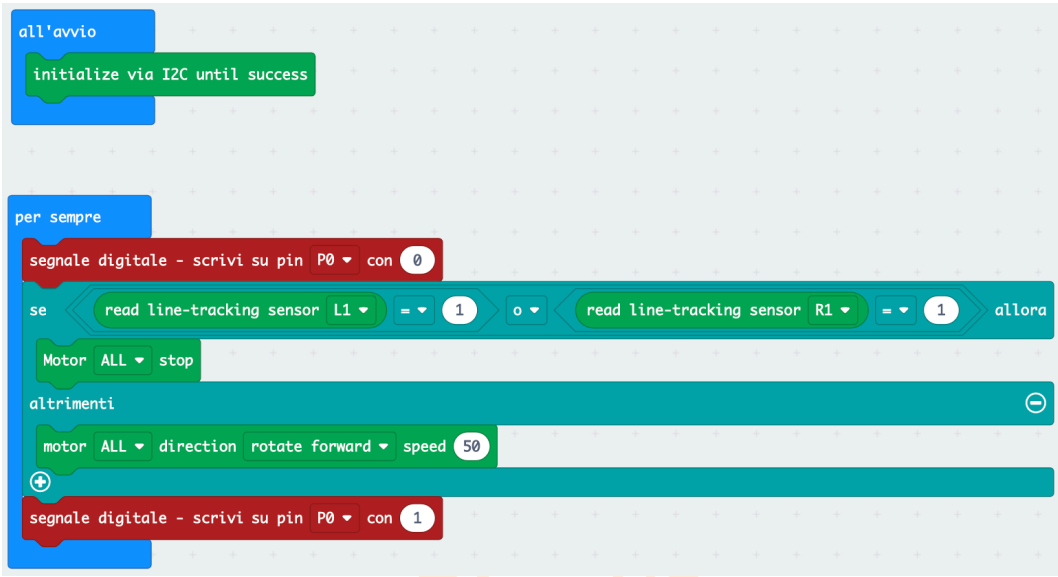


che è identico al caso B) con 0 ms di ritardo.

Si è indotti a pensare che la gestione dei pulsanti A e B sia esterna al ciclo per sempre forse avviene con gli interrupt.

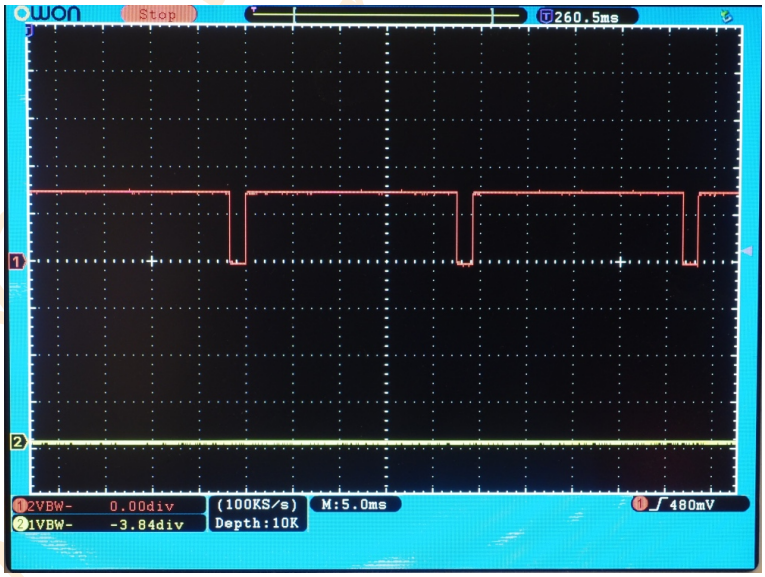
E) Arresto del robot

Viene inserito il codice per arrestare il robot “Maqueen plus" quando si trova su una banda trasversale non riflettente utilizzando i sensori di traccia S2 e R2 che distano 3 cm:



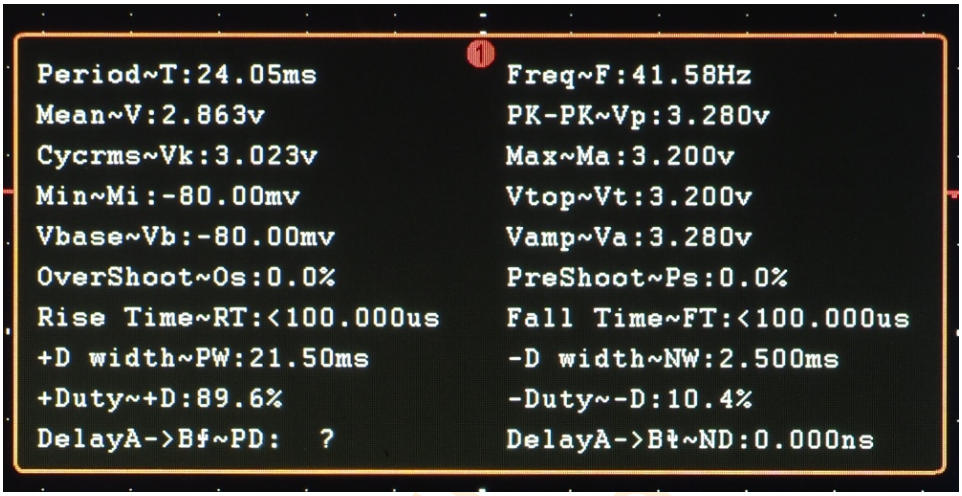
Questa volta il codice è un po’ più complesso:

ci sono comandi per la lettura dei sensori, un comando di selezione condizionale, operazioni logiche e comandi per l’azionamento dei motori.



Dall'oscillogramma si ricava che il periodo è di 24 ms con una durata del tempo di elaborazione (parte bassa del segnale) che significa un tempo di elaborazione ancora breve rispetto al tempo di chiusura e ripetizione del ciclo.

Per la precisione vedere la figura seguente:



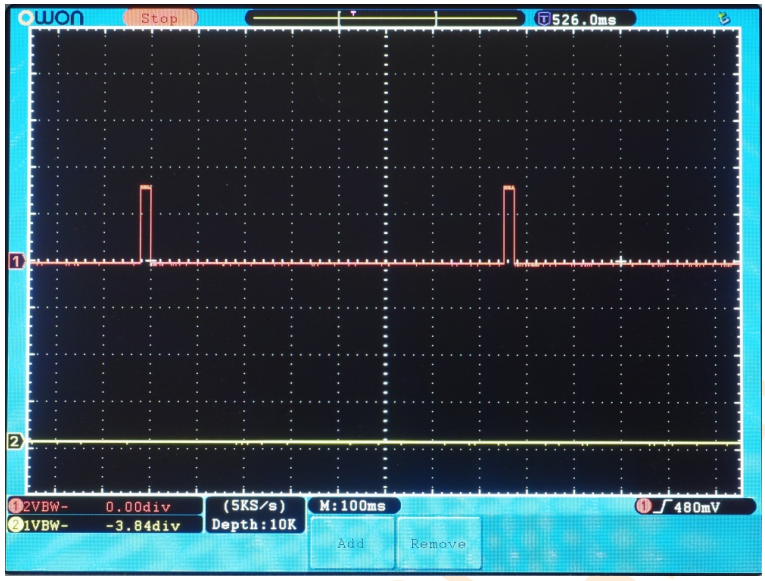
F) Arresto a velocità variabile

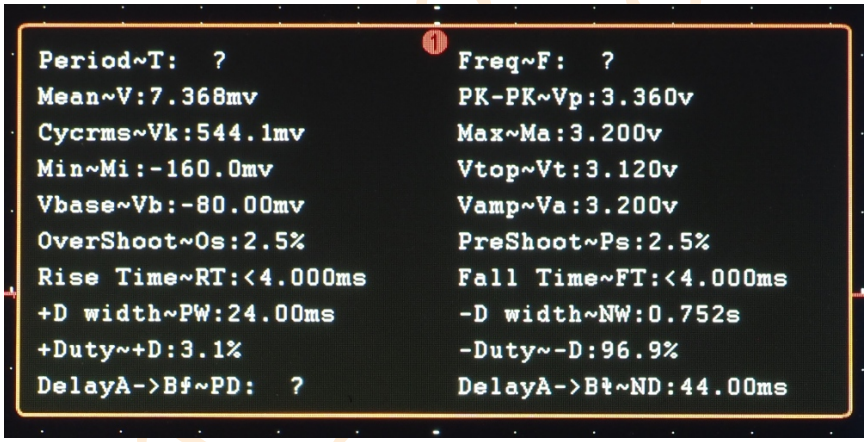
Per misurare lo spazio di arreso del robot a diverse velocità si costruisce un codice che prevede la possibilità di variare la velocità a gradini di 20 premendo il pulsante A e di diminuirla della stessa quantità con il pulsante B.

Per evidenziare il valore della velocità in atto deve essere mostrato sul display il suo valore; si sceglie di mostrare la velocità diviso 20 così da avere visualizzata una sola cifra per non peggiorare le cose con l'uso del display.

In un primo caso il display viene chiamato all'interno del ciclo "per sempre".

Il risultato è dato dall'oscillogramma:





Il valore alto (+D width) dura i soliti 24 ms mentre il valore basso (-D width), quello del codice da eseguire, dura 0,752s.

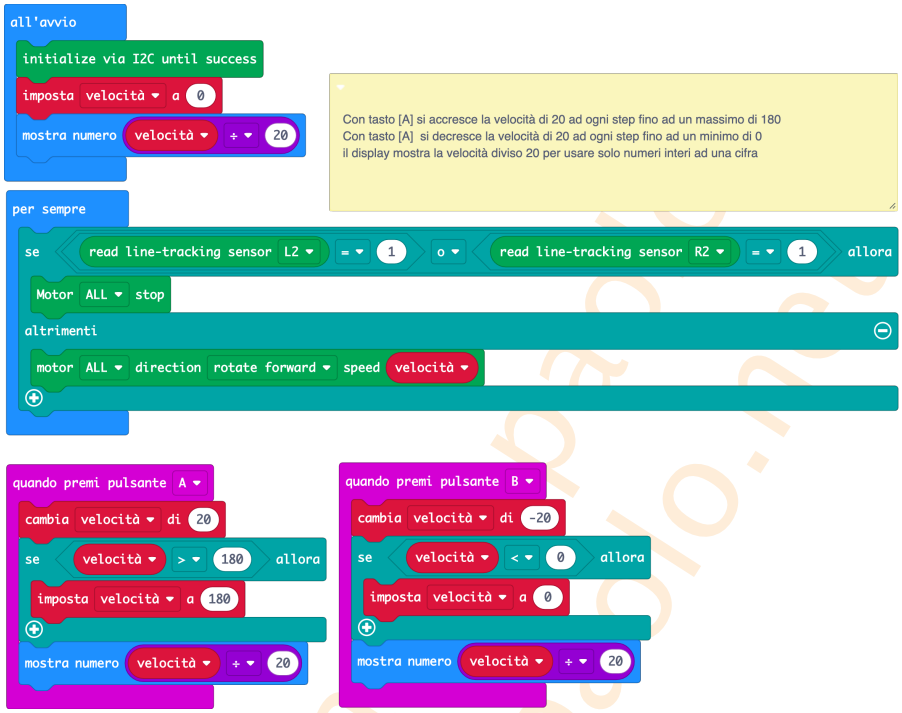
Troppo per un robot che si muove e deve essere controllato.

Si conferma la precedente osservazione sul display: non deve essere chiamato entro il ciclo “per sempre” perché ne rallenta troppo l’esecuzione.

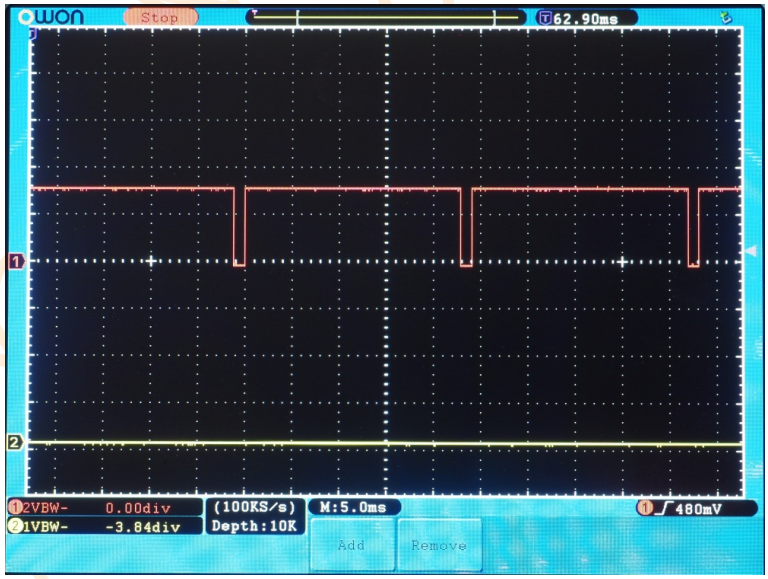
***

Dato che si vuole avere la lettura della velocità in atto tramite un indice mostrato sul display di microbit, si adotta la soluzione di spostarne la funzione entro il codice dei pulsanti così che viene aggiornato quando serve cioè quando si preme uno dei due pulsanti rimanendo fisso fino a quando non si preme nuovamente uno dei due pulsanti.

In questo modo non si influisce sulla durata del ciclo “per sempre” riducendo di molto i tempi di reazione del robot:



L’effetto di questa modifica è dato dall’oscillogramma seguente:



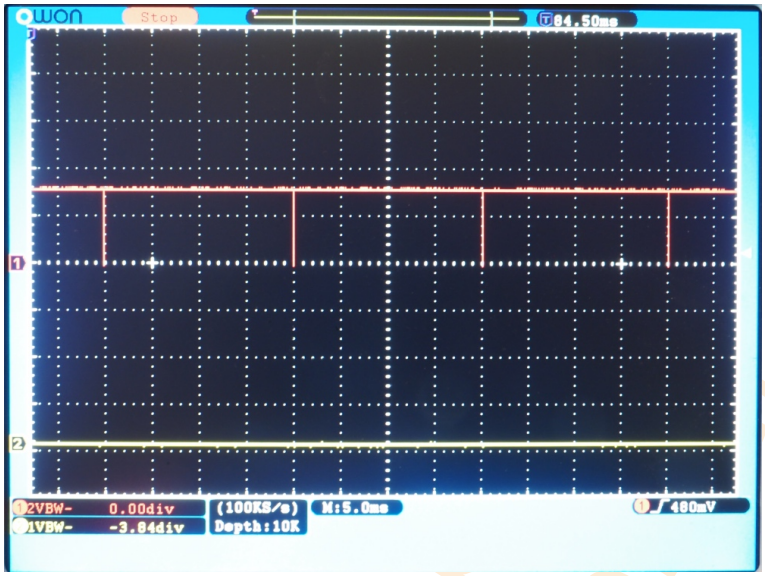
Il periodo è 28 ms e viene ripristinato un valore accettabile del tempo di reazione.

G) Velocità senza maqueen plus

Per provarle proprio tutte sono stati tolti tutti i blocchi riguardanti maqueen nell'eventualità che questi possano produrre ulteriori rallentamenti utilizzando questo codice:

nel qual sono state lasciate le stesse operazioni logiche, ovviamente basate con i dati derivanti dal solo microbit in modo da escludere gli effetti prodotti dal codice scritto per il robot.

Il risultato è il seguente:



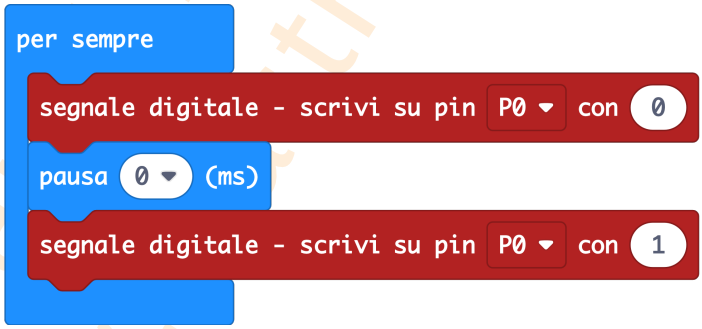
da cui risulta una durata del ciclo di 20[2] ms.

Sembra che con questo codice di programmazione non si possa compier un ciclo in meno di 20ms.

Accelerare il ciclo con Python

Si prova a scrivere il codice usando Python supponendo che il periodo di esecuzione del ciclo "per sempre" possa diventare più breve.

Si riprende il codice con inserito un blocco di ritardo di 0 ms di cui al **punto B**:

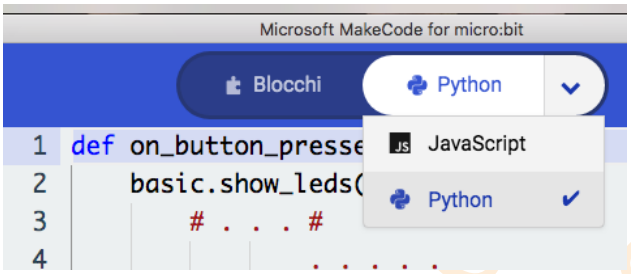


che, come si è visto, impiega 28 ms a compiere un ciclo.

[2] *In questo caso è stata usata una scheda microbit V2.*

Lo stesso codice in Python

MakeBlock permette di passare dal codice a blocchi ad uno scritto con Python o con JavaScript semplicemente scegliendo l'opzione che interessa.



Nel caso in esame passando a Python si ottiene il codice:

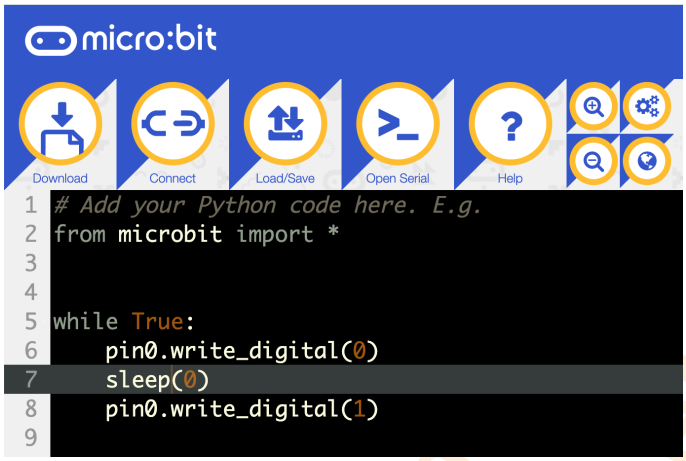
```
def  on_forever():
    pins.digital_write_pin(DigitalPin.P0,  0 )
    basic.pause( 0 )
    pins.digital_write_pin(DigitalPin.P0,  1 )
basic.forever(on_forever)
```

Se si carica questo codice da questa pagina su microbit non si ha alcun vantaggio in termini di velocità.

Questa versione di Python, che si chiama **MakeCode Python**, è solo un traduzione del codice a blocchi per cui non porta ad alcun miglioramento in termini di velocità.

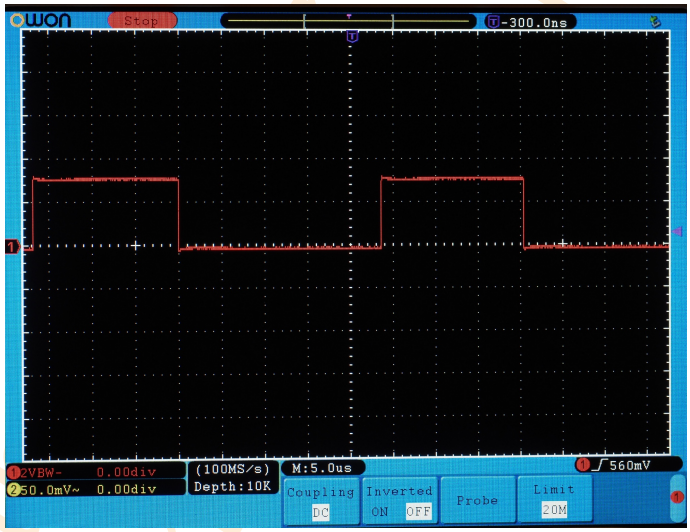
Python editor for microbit

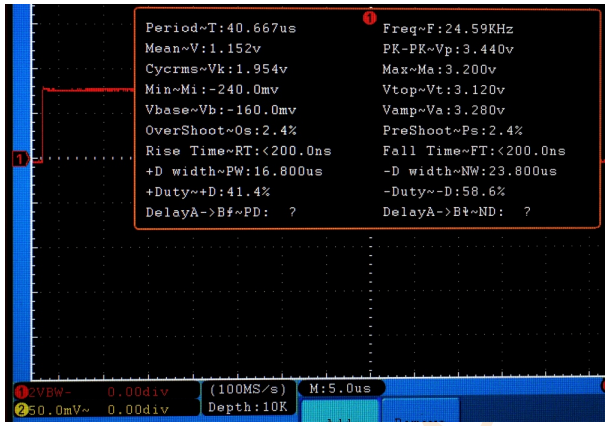
Cambia tutto se si scrive lo stesso programma usando "**Python editor for microbit**":



Nota. Come si vede, cambiano anche le parole del codice.

Con **MicroPython**, così si chiama, viene utilizzato un interprete che lavora da un livello più basso molto vicino al codice del microcontrollore con il seguente risultato:





La durata del ciclo è di 40,7 μs (microsecondi) quasi mille volte più breve.
Per sapere di più sul confronto fra le due implementazioni di Python, vedere **qui**.