Progettazione Sistemi Meccatronici 2021/2022

Sistema per ispezione visiva PCB tramite microscopio

Giacomo Bartolacci
Alessandro Zubani

Sommario

Scopo del progetto

Lo scopo del progetto è realizzare, con il recupero della stampante 3D MakerBot non più funzionante, un sistema di acquisizione immagini tramite microscopio digitale, per poter controllare la qualità delle saldature sulle PCB.
Nel progetto verrà bypassata la scheda non più funzionante della stampante, e i motori verranno pilotati direttamente dal microcontrollore NXP.
La stampante è dotata di tre motori stepper MS17HD2P4100, che permettono il movimento del microscopio lungo il piano xy e permettono lo spostamento della piattaforma su cui si appoggia la scheda PCB lungo l'asse z.

Elementi utilizzati

- Microcontrollore NXP S32K142-EVB.
- Tre Driver TMC2208 SilentStepStick per pilotare i motori stepper.
- Stampante 3D Makerbot, di cui si utilizza l'involucro e i tre motori stepper MS17HD2P4100.
- PC per fornire i valori degli spostamenti al microcontrollore tramite l'UART.
- CNC Arduino Shield V_3 per il cablaggio dei driver.
- Microscopio digitale con presa USB per trasferire l'immagine al PC.

Specifiche dei componenti:
I driver possono erogare 1.4A di corrente RMS (datasheet TMC2208), mentre i motori possono assorbire 1A RMS a 12V di alimentazione (estratto dalle curve del datasheet).
I driver inoltre possono essere usati in varie modalità di funzionamento, tramite il collegamento opportuno di alcuni shunt sui pin, per esempio ½ step e ¼ di step, in questo caso è stato scelto di utilizzarli nella modalità full step in quanto la risoluzione del movimento non deve essere così accurata.
Si può inoltre impostare la corrente erogata dai driver tramite un potenziometro.

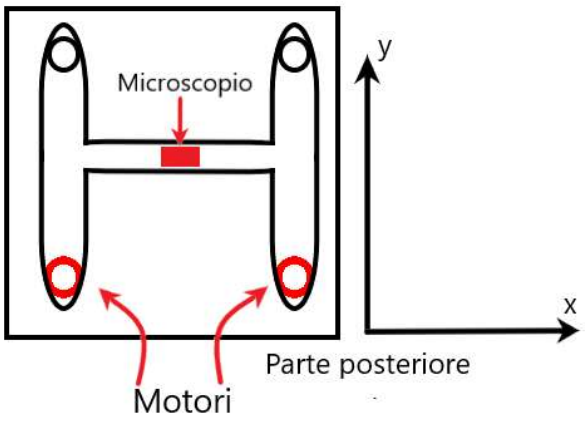
Del microcontrollore, oltre alla sua programmazione classica, verranno sfruttate in particolare la UART e il FTM (FlexTimer Module) PWM come contatore.

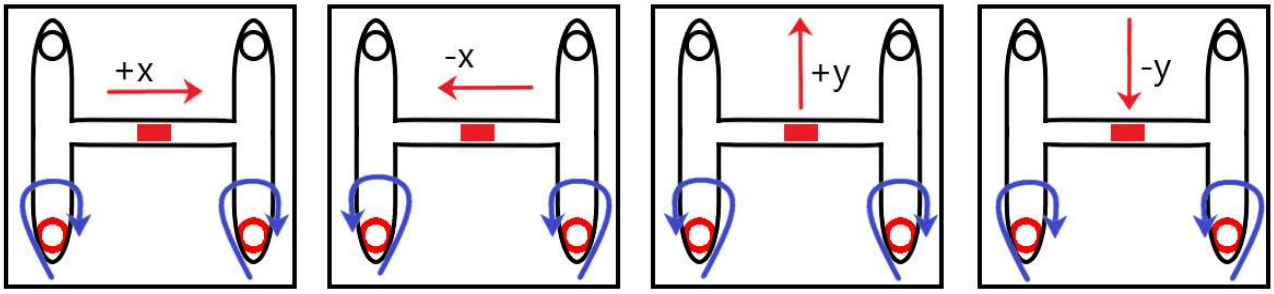
Funzionamento del sistema

I tre motori stepper della stampante vengono pilotati tramite tre driver TMC2208, i quali sono controllati dal microcontrollore NXP. L'utilizzo di questi driver facilita molto il controllo dei motori, poiché i driver sono essenzialmente pilotati da due segnali principali: Direction, che specifica il senso di rotazione, e Step, che ad ogni suo fronte in salita fa ruotare il motore di un passo. In questo modo basterà pilotare il pin direction con la direzione voluta e il pin step con un segnale PWM, generato su alcuni pin del microcontrollore.
È presente, inoltre, un segnale di enable attivo basso, che va abilitato per attivare il pilotaggio dei motori. Questo segnale verrà abilitato durante il pilotaggio, e disabilitato altrimenti, in modo da non lasciare attivi i motori e consumare potenza inutilmente, con il conseguente surriscaldamento dei driver (per questa applicazione non c'è bisogno di mantenere fissi, in blocco, i motori).

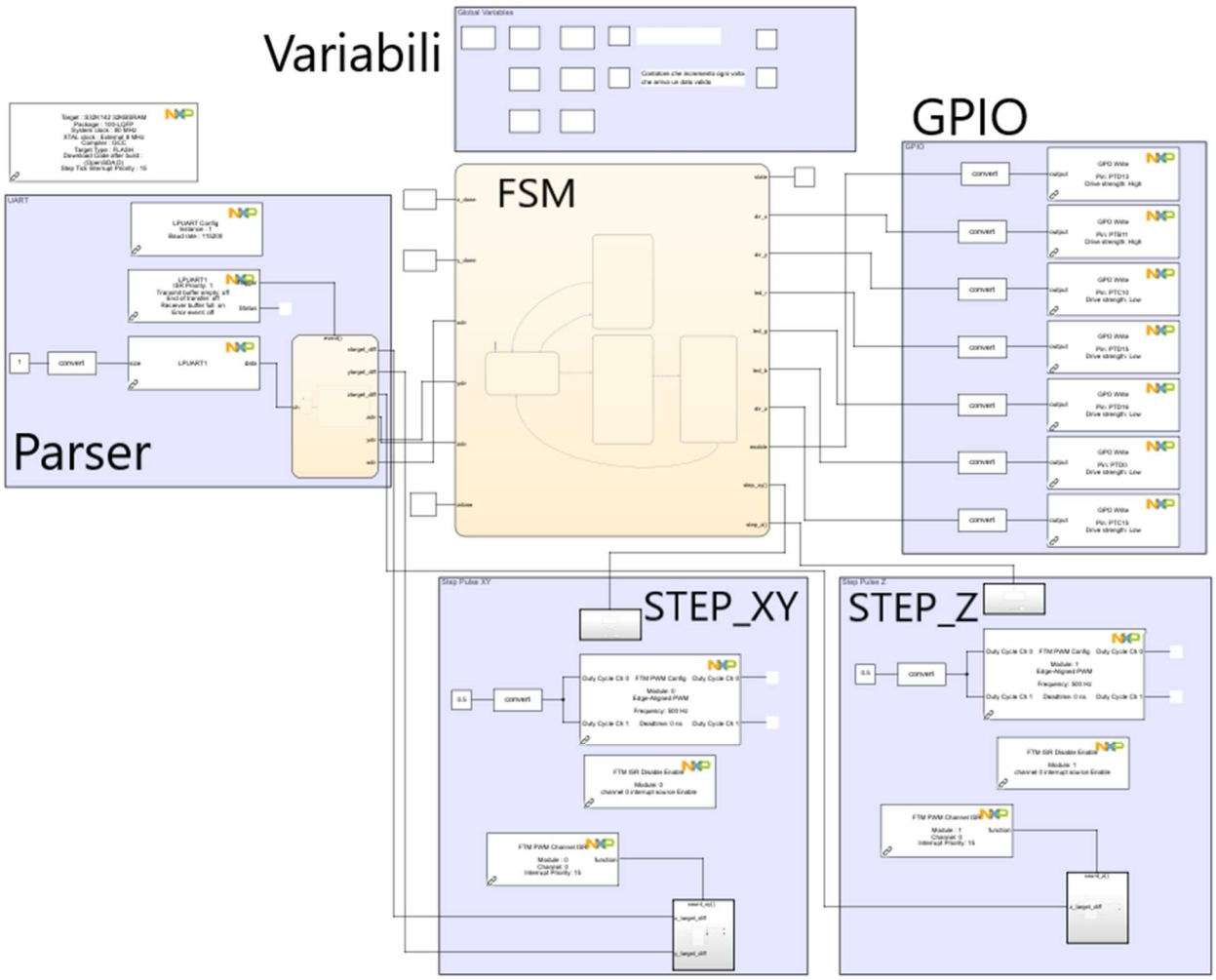
Per poter scegliere di quanto muovere il sistema, viene fornita una stringa dal PC al microcontrollore tramite la UART. Un parser traduce la stringa inviata nelle corrispondenti quantità di cui deve traslare il sistema lungo x,y,z.
Lo spostamento è di tipo differenziale, cioè data la stringa il sistema si muove della quantità specificata a partire dalla posizione in cui si trova. Un altro possibile modo è implementare uno spostamento assoluto, cioè indicando (x,y) il sistema si sposta a quella coordinata.

La stampante, per muoversi sul piano xy, sfrutta due cinghie che realizzano una struttura ad H, controllata da due motori stepper. Facendo girare un unico motore, il microscopio si sposta in diagonale (sia lungo x che y). Invece, facendo girare entrambi i motori, il microscopio si sposta lungo un asse. Di seguito sono riportate delle immagini che schematizzano le cinghie e il verso con cui far girare i motori per ottenere uno spostamento lungo un certo asse e con un dato verso.





Realizzazione del controllo tramite Matlab/Simulink



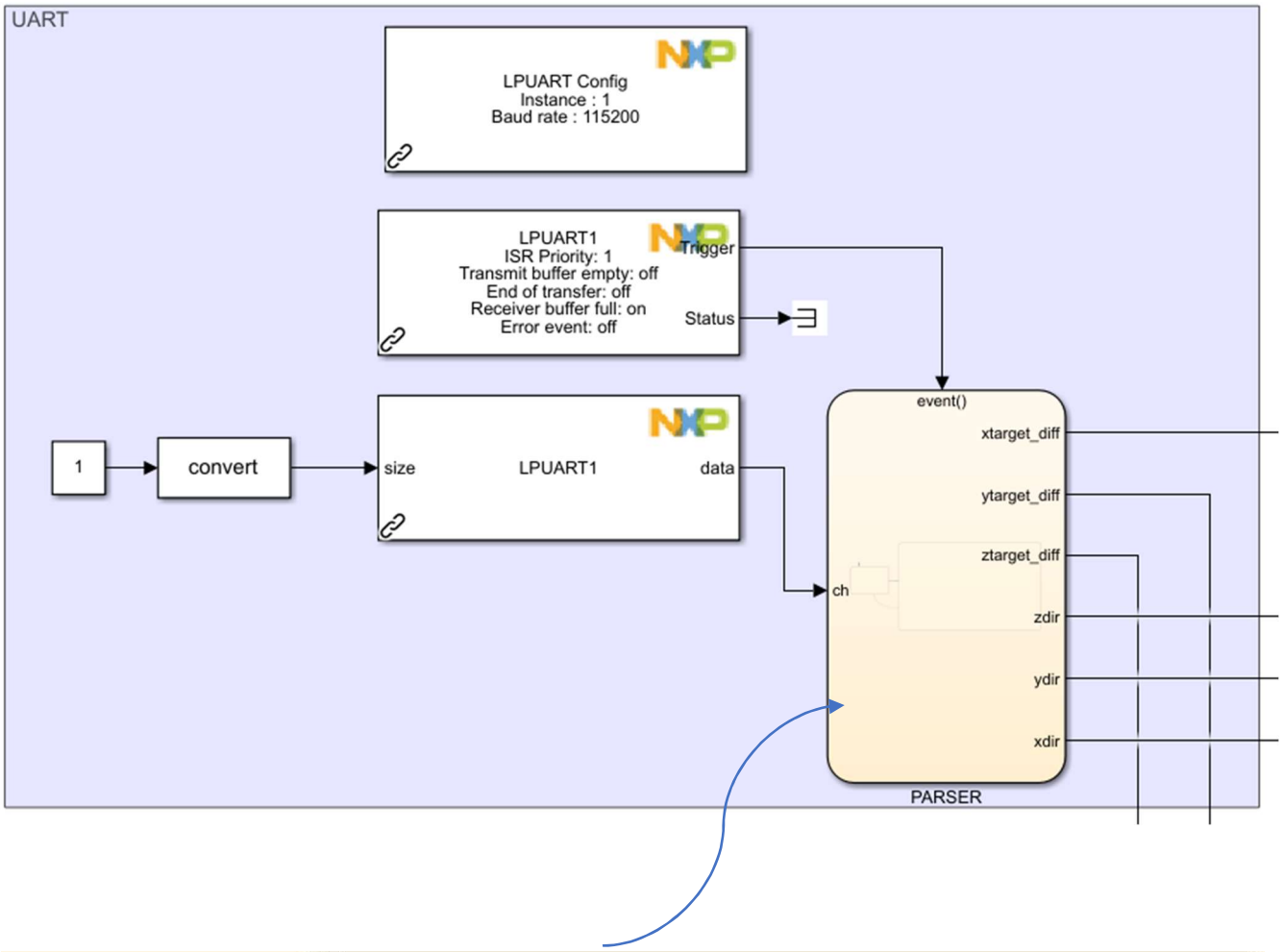
Lo schema è diviso in tre moduli principali:

1. Parser
2. FSM
3. STEP_XY e STEP_Z

Il Parser si occupa della comunicazione con la UART, la FSM si occupa di generare i segnali di direzione, enable, e dell'attivazione dei moduli STEP_XY/STEP_Z, i quali forniscono il PWM per controllare il segnale Step per un tempo opportuno (dipendente dalla lunghezza del comando dato con la UART).

Sono presenti delle variabili globali, usate in più moduli del sistema. In questo modo, i vari moduli possono comunicare tra loro tramite l'utilizzo delle variabili.

UART e Parser



Ad ogni invio di un carattere, viene attivata un'interrupt, che chiama la macchina a stati PARSER.
Inizialmente si rimane nello stato Accumulate, per ogni carattere viene incrementato un contatore ed il carattere è salvato in una stringa, nella locazione indicata dal contatore. Quando viene inviato il carattere terminatore ';' si passa nello stato di Output, e la stringa così ottenuta accumulando i caratteri viene analizzata per fornire i segnali di uscita alla FSM.

Le stringhe da inviare possono avere questi formati, con i vari segni:

M+xxxxx,+yyyyy; *con x,y,z numeri*
Z+zzz;

Tramite il primo carattere il parser capisce se muoversi nel piano xy (*M*), oppure se muovere la stampante lungo l'asse z (*Z*), con *x,y,z* si sceglie di quanto spostarsi e con + o – si stabilisce la direzione.
Per poter fornire un'ulteriore stringa, va mandato nuovamente il carattere ';' per tornare nello stato iniziale di Accumulate, e a quel punto si può inviare un nuovo comando.

Esempi di stringhe inviabili, successive:

M+01200,-00650;;Z-333;;M-00800,+00345;;

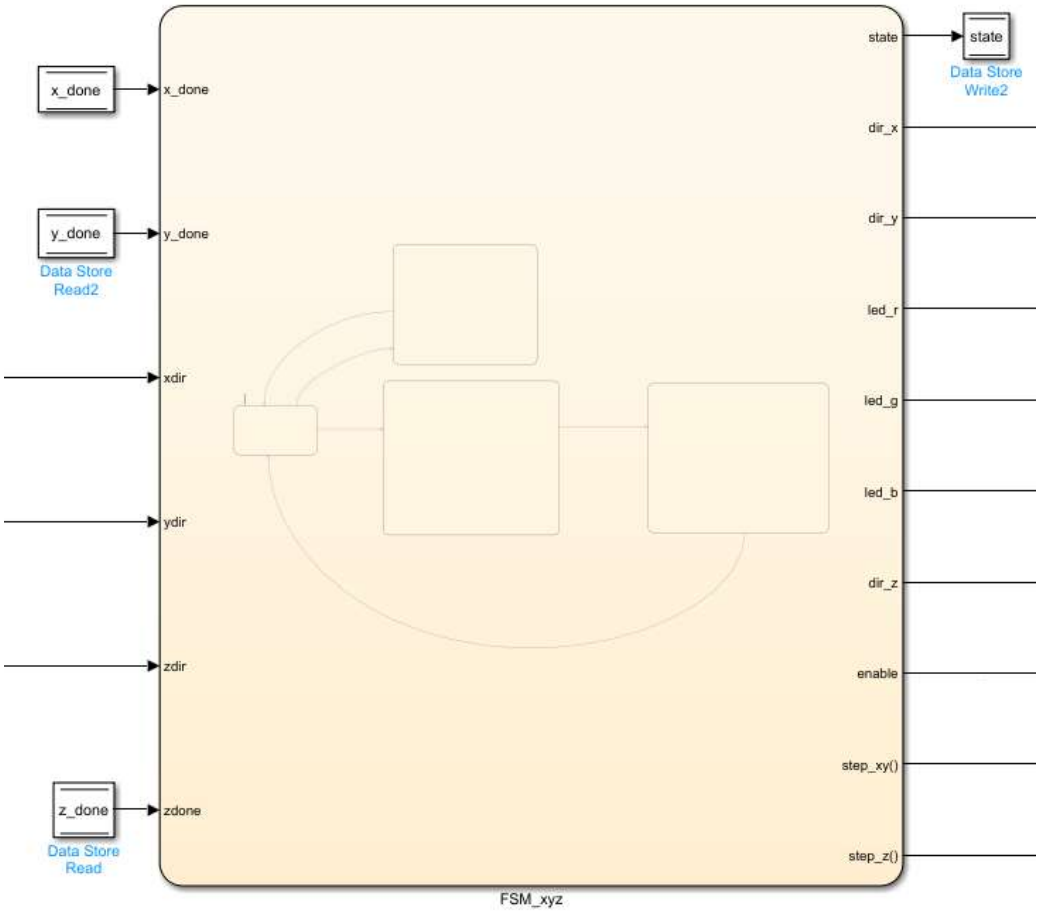
Se la stringa viene acquisita correttamente, la macchina a stati del Parser genera vari segnali per la FSM:

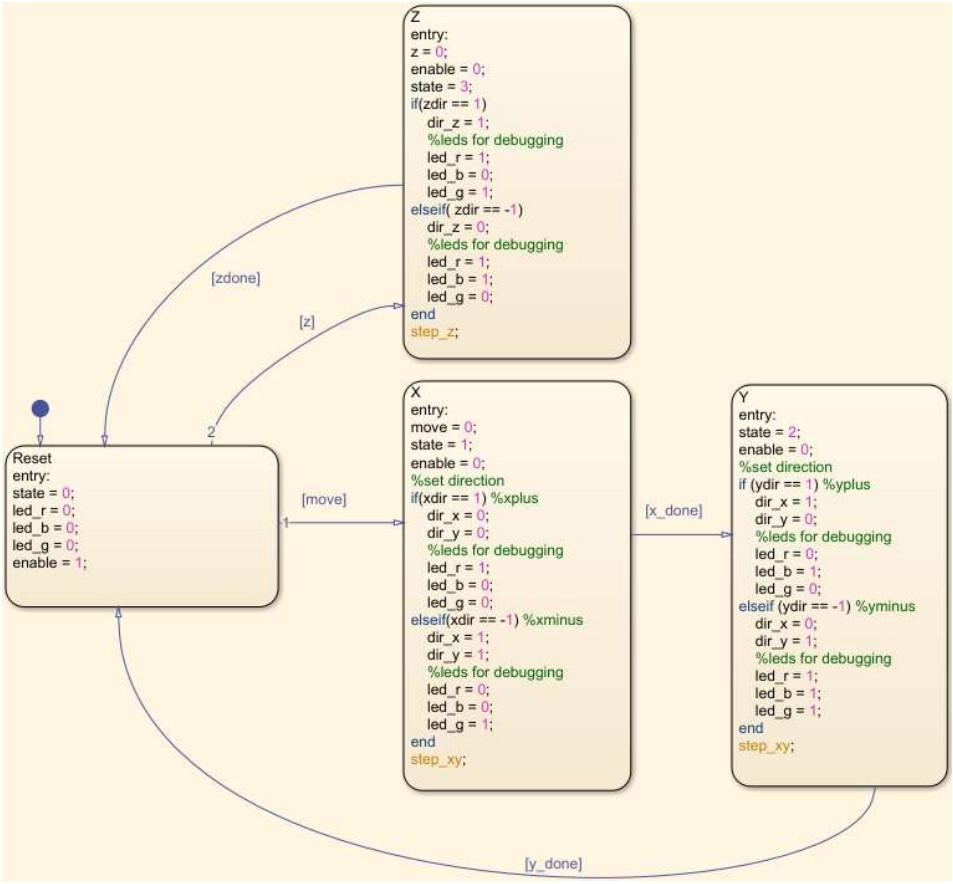
1. Xtarget_diff, Ytarget_diff, Ztarget_diff: Valori differenziali degli spostamenti.
2. xdir, ydir, zdir: il senso di rotazione degli stepper, valore binario (0 o 1, orario o antiorario).
3. move = 1 o z = 1, per far uscire la FSM dallo stato di reset.

Per il corretto funzionamento di questo modulo è necessario inviare i caratteri tramite la Uart non troppo velocemente, altrimenti essi vengono interpretati male.

I caratteri inviati tramite la UART sono dei byte in formato ASCII, per convertirli in double si può usare la funzione double(str(n)), con cui si converte dall'ASCII al numero, sottrarre 48, poiché lo zero è codificato come 48, e moltiplicare per il peso che hanno all'interno della stringa.

FSM


x_done
y_done
Data Store Read2
xdir
ydir
zdir
z_done
Data Store Read
zdone
state
state
Data Store Write2
dir_x
dir_y
led_r
led_g
led_b
dir_z
enable
step_xy()
step_z()
FSM_xyz

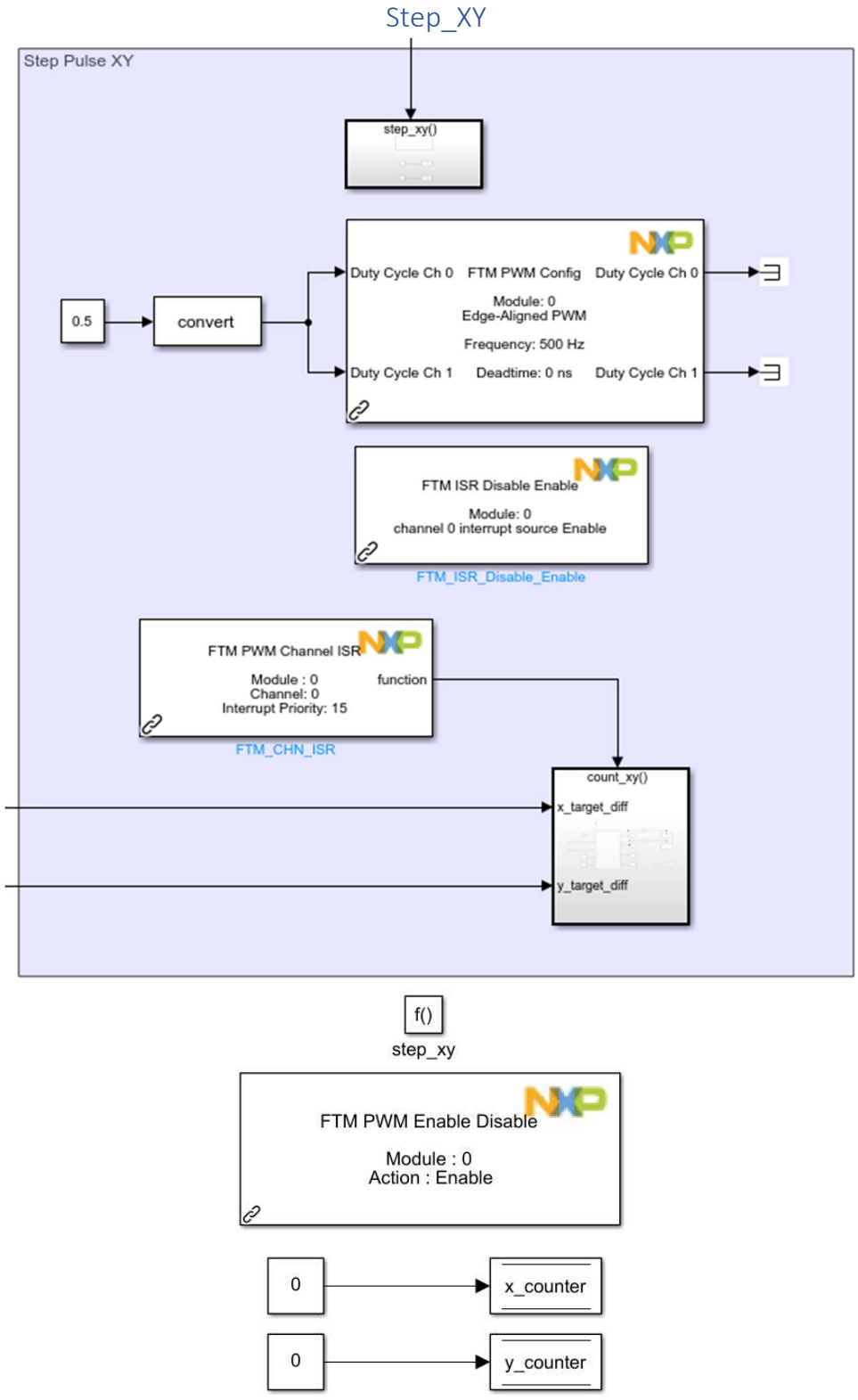

Z
entry:
z = 0;
enable = 0;
state = 3;
if(zdir == 1)
dir_z = 1;
%leds for debugging
led_r = 1;
led_b = 0;
led_g = 1;
elseif(zdir == -1)
dir_z = 0;
%leds for debugging
led_r = 1;
led_b = 1;
led_g = 0;
end
step_z;
[zdone]
[z]
2
Reset
entry:
state = 0;
led_r = 0;
led_b = 0;
led_g = 0;
enable = 1;
[move]
1
X
entry:
move = 0;
state = 1;
enable = 0;
%set direction
if(xdir == 1) %xplus
dir_x = 0;
dir_y = 0;
%leds for debugging
led_r = 1;
led_b = 0;
led_g = 0;
elseif(xdir == -1) %xminus
dir_x = 1;
dir_y = 1;
%leds for debugging
led_r = 0;
led_b = 0;
led_g = 1;
end
step_xy;
[x_done]
Y
entry:
state = 2;
enable = 0;
%set direction
if (ydir == 1) %yplus
dir_x = 1;
dir_y = 0;
%leds for debugging
led_r = 0;
led_b = 1;
led_g = 0;
elseif (ydir == -1) %yminus
dir_x = 0;
dir_y = 1;
%leds for debugging
led_r = 1;
led_b = 1;
led_g = 1;
end
step_xy;
[y_done]

La macchina a stati esce dallo stato di Reset quando move = 1 oppure z = 1, e questi segnali vengono forniti dal Parser.
Nel caso di move = 1, si passa negli stati per muoversi nel piano xy, mentre nel caso di z = 1 si passa nello stato per muoversi lungo z.
Per il movimento nel piano xy, il sistema si sposta prima lungo l'asse X e poi lungo l'asse Y.
Quando si entra nello stato, vengono settate correttamente le direzioni da fornire ai driver, acceso il led presente sulla scheda NXP con un colore dipendente dallo stato, attivato l'enable per i driver e inviato l'evento step_xy o step_z. Questi eventi attivano il PWM (altrimenti spento di default), che va a pilotare i pin Step dei vari driver. Ad ogni fronte su questi pin, si ha un passo del corrispondente motore.
Quando tutti i passi sono stati fatti, tramite i segnali x_done, y_done e z_done (proveniente dai blocchi Step XY e Step Z rispettivamente) si torna nello stato di reset, aspettando una nuova stringa.
Si vede che per muoversi lungo l'asse X i motori vanno pilotati in modo concorde (dir_x = dir_y) mentre per muoversi lungo l'asse y i motori vanno pilotati in modo discorde (dir_x != dir_y).

Esempio di funzionamento, data la stringa:

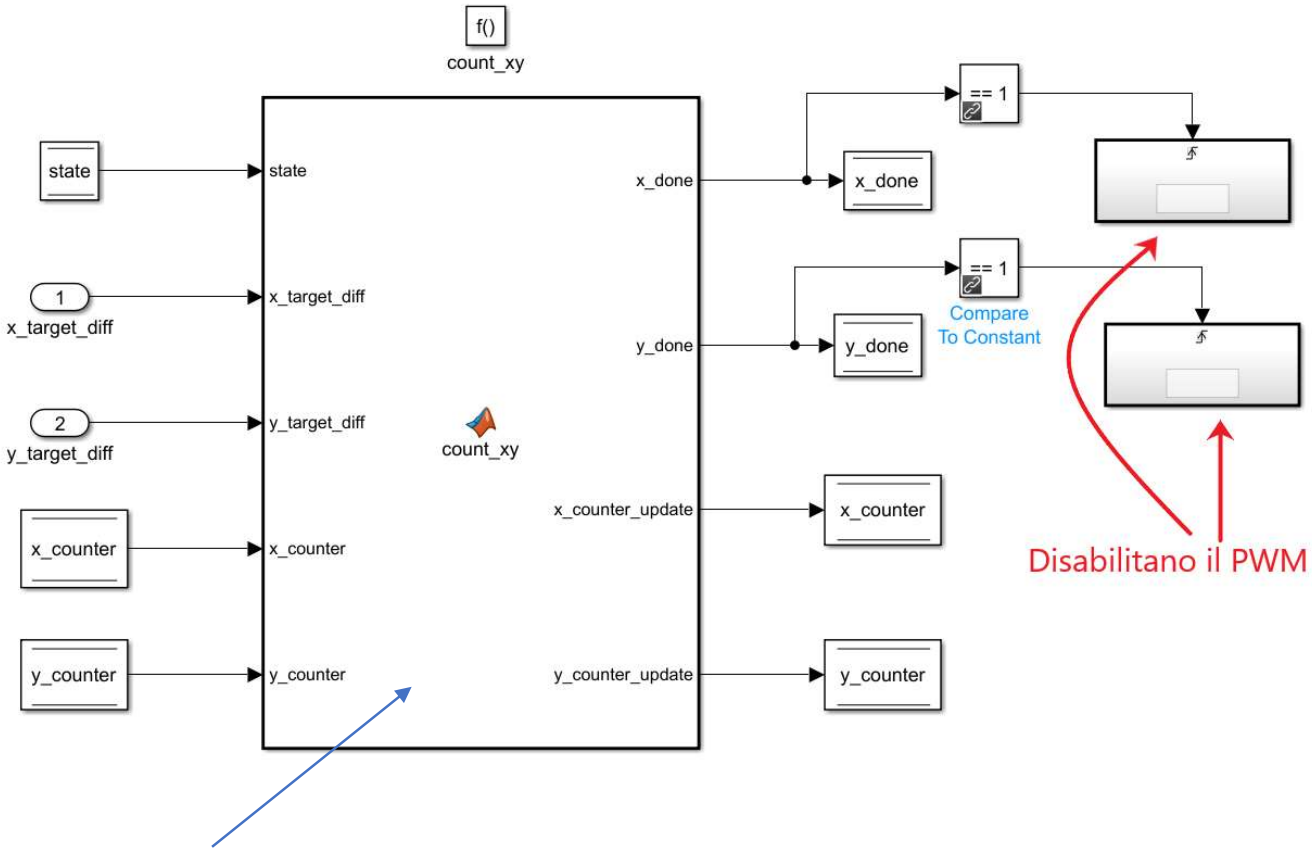
M+00900,-01200; *deve spostarsi di 900 passi lungo +x e di 1200 lungo -y*

Dopo aver ricevuto il carattere terminatore ';', il parser imposta move = 1, dunque la FSM entra nello stato X, imposta le direzioni affinché il sistema si muova lungo +x e fa partire l'evento Step_XY. Questo evento accende il PWM, e dopo aver fatto un numero pari a 900 passi, imposta x_done = 1. A seguito di ciò si entra nello stato Y, in cui si fanno le stesse operazioni, impostando però le direzioni per muoversi lungo -y, e per eseguire 1200 passi. Dopo i 1200 passi arriva y_done = 1, e la FSM torna nello stato di reset.



Questo modulo si occupa di gestire il counter del microcontrollore, cioè il FTM (FlexTimer Module) PWM. In questo caso viene utilizzato come un contatore per contare il numero di step da eseguire. A seguito dell'evento Step_XY fornito dalla FSM, viene abilitato il PWM e azzerate le variabili counter. L'evento viene fornito quando si entra nello stato X, Y (o Z, nel blocco Step_Z). Il PWM viene disabilitato tramite il blocco Count_XY al termine del conteggio dei passi.
La frequenza del PWM, che è la frequenza a cui si pilotano i motori, è impostata a 500 Hz. A frequenze più elevate gli stepper non funzionano correttamente.

Count_XY



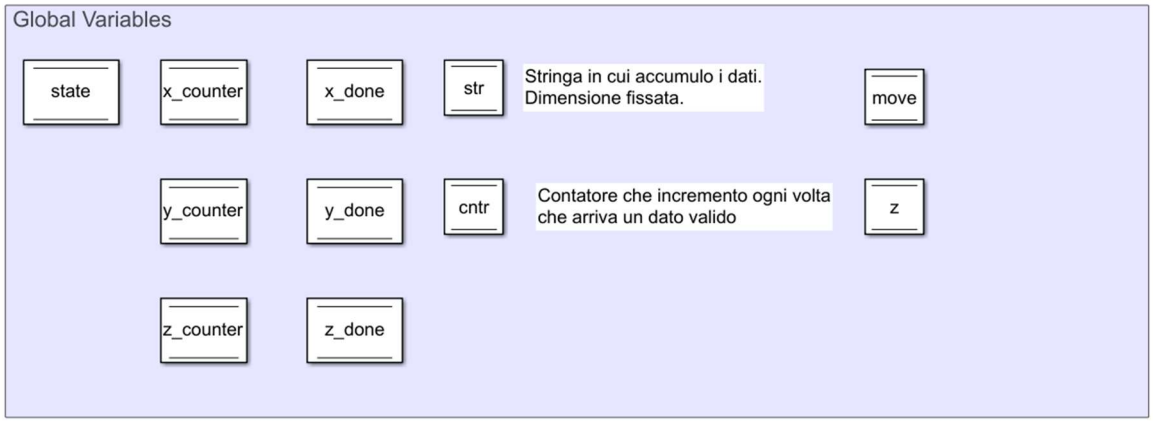

```
function [x_done, y_done,x_counter_update, y_counter_update] = count_xy(state,
x_target_diff, y_target_diff, x_counter, y_counter)

    if( state == 1 ) %Muovo lungo asse x
        if ( x_counter < x_target_diff )
            x_counter_update = x_counter + 1;
            y_counter_update = 0;
            x_done = 0;
            y_done = 0;
        else %Se arrivo all'ultimo passo, azzero i counter e metto x_done a 1
            x_counter_update = 0;
            y_counter_update = 0;
            x_done = 1;
            y_done = 0;
        end
    elseif ( state == 2 ) %Muovo lungo asse y
        if ( y_counter < y_target_diff )
            y_counter_update = y_counter + 1;
            x_counter_update = 0;
            y_done = 0;
            x_done = 0;
        else %%Se arrivo all'ultimo passo, azzero i counter e metto y_done a 1
            y_counter_update = 0;
            x_counter_update = 0;
            y_done = 1;
            x_done = 0;
        end
    else
        x_counter_update = x_counter;
        y_counter_update = y_counter;
        x_done = 0;
        y_done = 0;
    end
end
```

Questo blocco si occupa dell'invio di un numero di fronti sui pin di step pari a quelli specificati dalla stringa, utilizzando il PWM del microcontrollore.
Questa funzione viene eseguita ad ogni ciclo del timer (FTM) dell'NXP, poiché ad ogni ciclo viene inviata l'interrupt.
Viene utilizzata una funzione Matlab che ad ogni chiamata della funzione, cioè ad ogni periodo del PWM, incrementa le variabili x_counter e y_counter, e verifica quando queste hanno raggiunto il valore x_target_diff e y_target_diff, fornito nella stringa.

Il funzionamento del modulo complessivo (Step_xy e count_xy) è il seguente:
A seguito dell'evento step_xy fornito dalla FSM, vengono azzerate le variabili counter_x, counter_y, e viene abilitato il PWM.
Grazie all'attivazione del PWM, il microscopio viene pilotato lungo l'asse x, e quando tutti i passi sono stati fatti, x_done è posto a 1, evento che disabilitata il PWM. Grazie a x_done = 1, al tick successivo del sistema (quindi al massimo dopo 0.1 s), nella FSM si entra nello stato Y, e di nuovo viene inviato l'evento step_xy e si esegue quindi il pilotaggio lungo l'asse y. Quando vengono fatti tutti i passi lungo y, si ha y_done = 1, il PWM viene disabilitato e la FSM torna nello stato di reset.

Il funzionamento per spostarsi lungo l'asse z è analogo a questo, ma semplificato, poiché un unico motore controlla l'asse z.

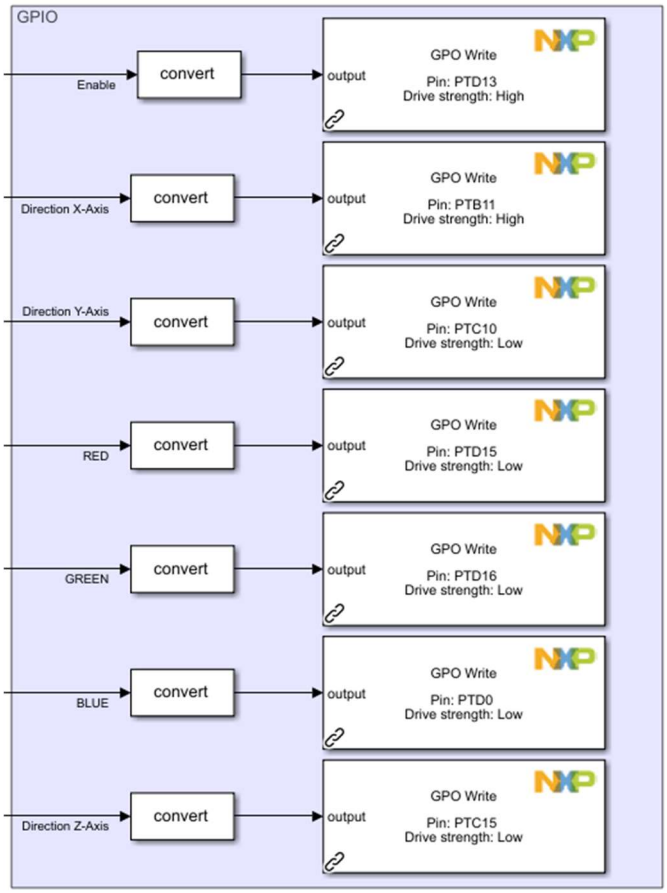
Variabili Globali



Variabili globali condivise tra i vari moduli:

1. State: variabile che mantiene lo stato per la FSM. Viene usato anche dal blocco count_xy, per capire quale contatore (X o Y) incrementare.
2. x_counter, y_counter, z_counter: Contatori usati per fare un numero di passi pari a quello specificato dalla stringa.
3. x_done, y_done, z_done: segnali utilizzati per cambiare stato nella FSM.
4. Cntr: contatore utilizzato dal Parser per immagazzinare i caratteri nella stringa.
5. Str: stringa in cui vengono immagazzinati i caratteri, analizzata dal Parser.
6. Move, z: Segnali forniti dal Parser ed utilizzati dalla FSM per uscire dallo stato di reset.

GPIO

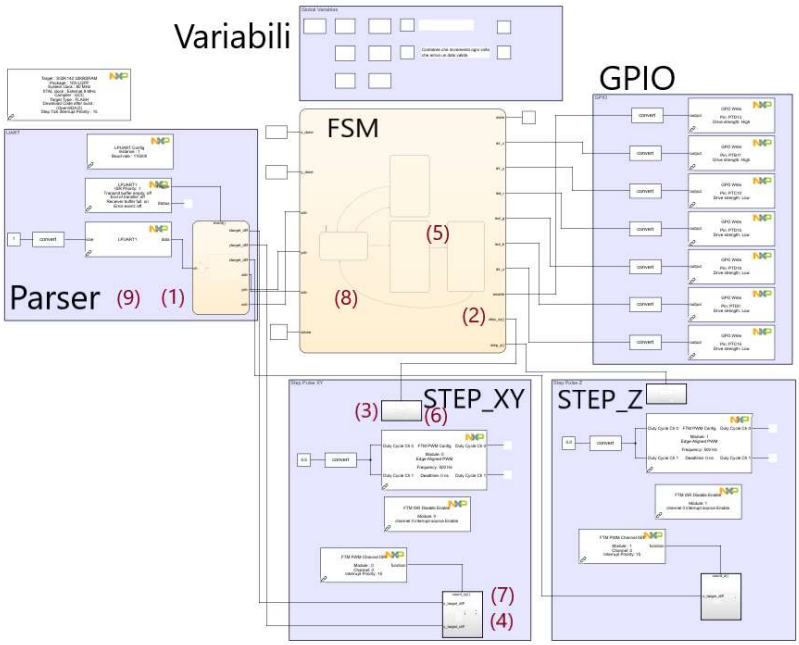


Riepilogo del funzionamento

Nelle sezioni precedenti sono state analizzate le singole parti costituenti del progetto. Per spiegare in modo completo il funzionamento, vengono riportate le varie fasi di esecuzione del programma, con questo comando di esempio: *m+12500,-00800;;*

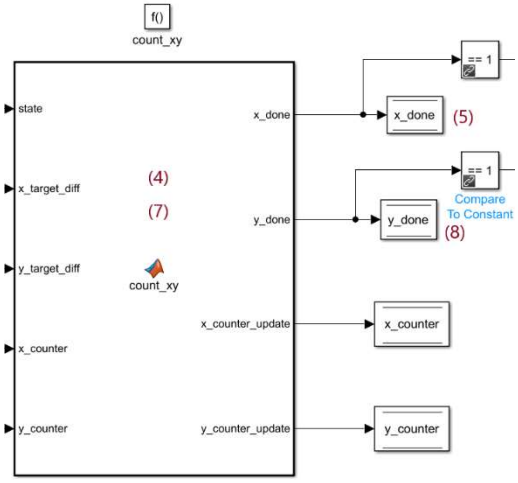
1. Il Parser accumula la stringa nella variabile str, e al primo carattere ';' decodifica la stringa, generando i segnali xdir, ydir, move, i quali vanno alla FSM, e i valori xtarget_diff e ytarget_diff i quali vanno alla sezione STEP_XY. La sezione che riguarda z non viene considerata.

2. La FSM passa dallo stato di Reset allo stato X, poiché move = 1. In questo stato abilita i motori, e genera le direzioni corrette affinché il sistema si muova lungo +x, ed abilita l'evento step_xy().



3. A seguito dell'evento step_xy (), viene abilitato il modulo STEP_XY, il quale azzera le variabili x_counter ed y_counter, ed attiva il PWM del microcontrollore con f = 500Hz e δ = ½ . Il PWM andrà a pilotare i pin di step dei driver, che muovono i motori. Ad ogni ciclo del PWM il motore esegue un passo.

4. Viene elaborata per prima la coordinata 'x'. Ad ogni ciclo del PWM è generata l'interrupt che esegue la funzione Matlab del blocco counter_xy. Questa incrementa la variabile x_counter, fin quando non risulta uguale a xtarget_diff, ovvero in questo caso 12500.



5. Quando x_counter = xtarget_diff, il modulo count_xy pone la variabile x_done = 1, e questo causa la disabilitazione del PWM e il passaggio della FSM dallo stato X allo stato Y.

6. Si entra nello stato Y della FSM, vengono generate le direzioni per spostarsi lungo -y, e viene riattivato il modulo STEP_XY tramite l'evento step_xy().

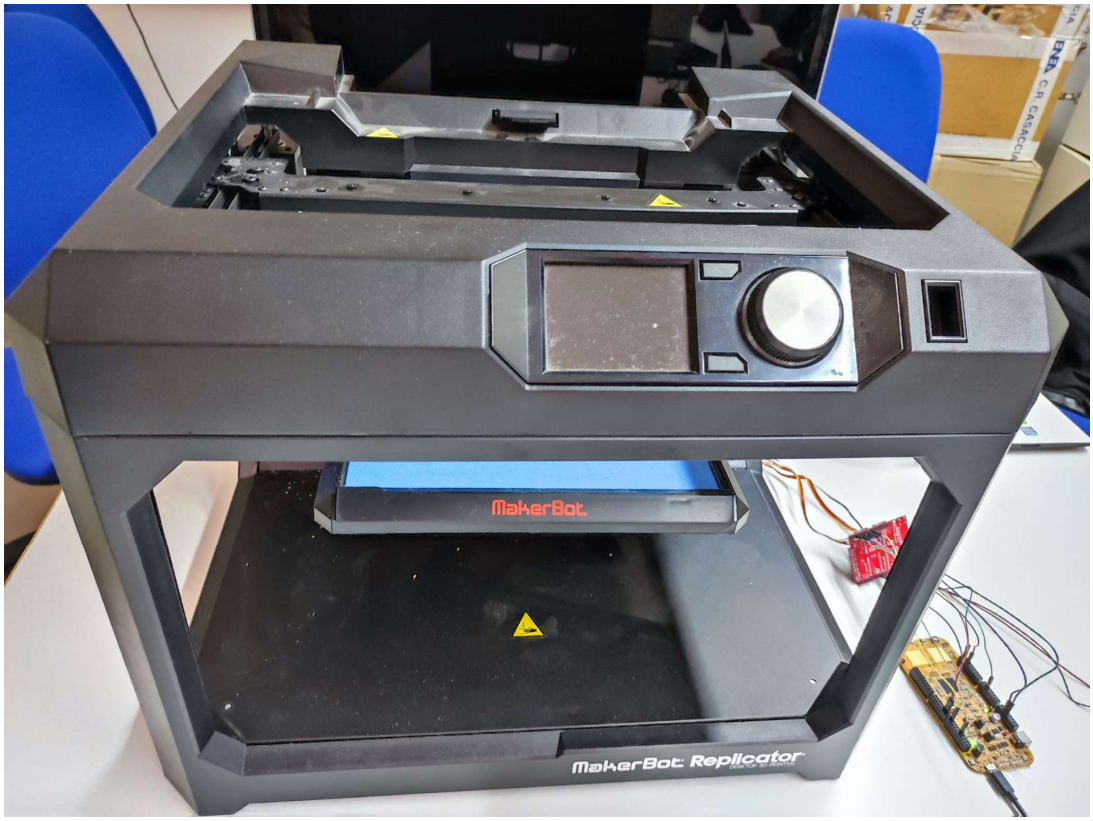
7. Viene riattivato il PWM con la stessa frequenza e δ, e viene elaborata la coordinata 'y'. Il funzionamento è analogo alla coordinata x. Il sistema si sposta di 800 passi lungo -y.

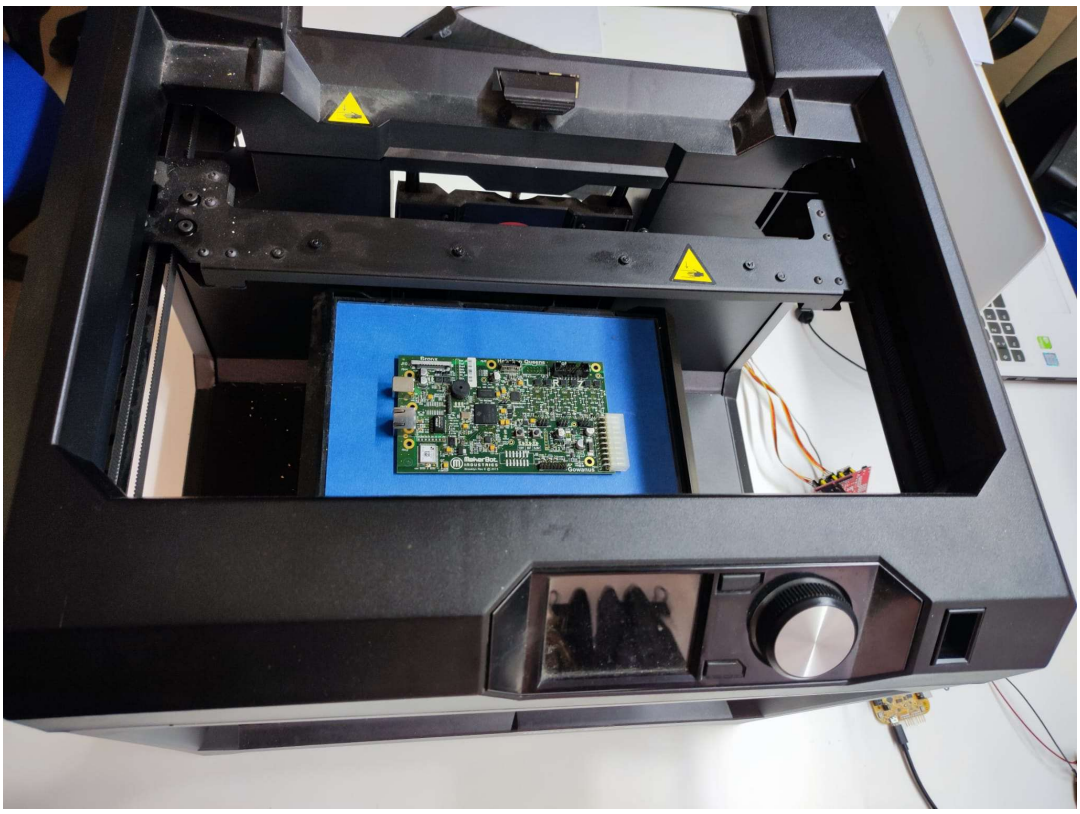
8. Quando anche y_counter = ytarget_diff, count_xy() genera y_done = 1, questo disabilita il PWM e riporta la FSM nello stato di Reset.

9. Il sistema si è spostato delle quantità desiderate, ed ora si trova in attesa di un nuovo comando, che può essere inviato solo dopo aver fornito un ulteriore carattere di ';'.

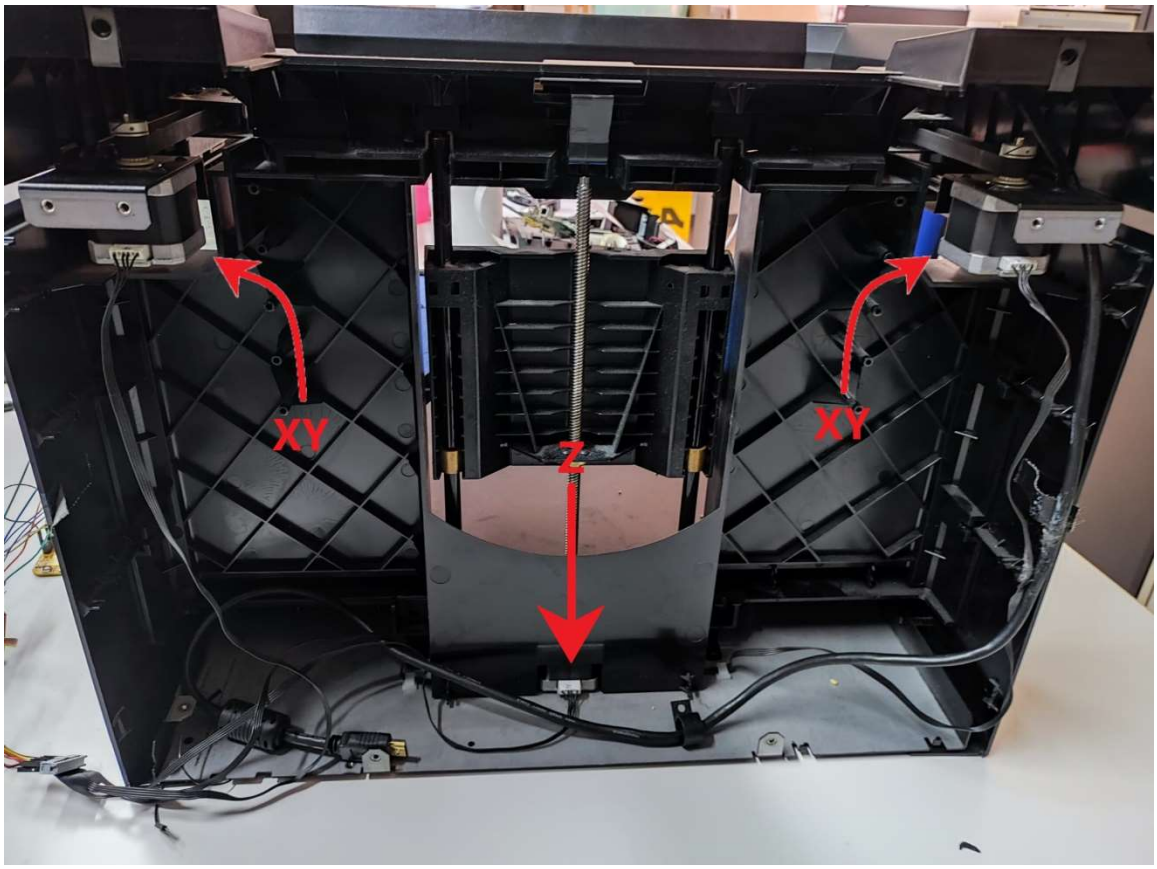
Immagini del sistema

Stampante

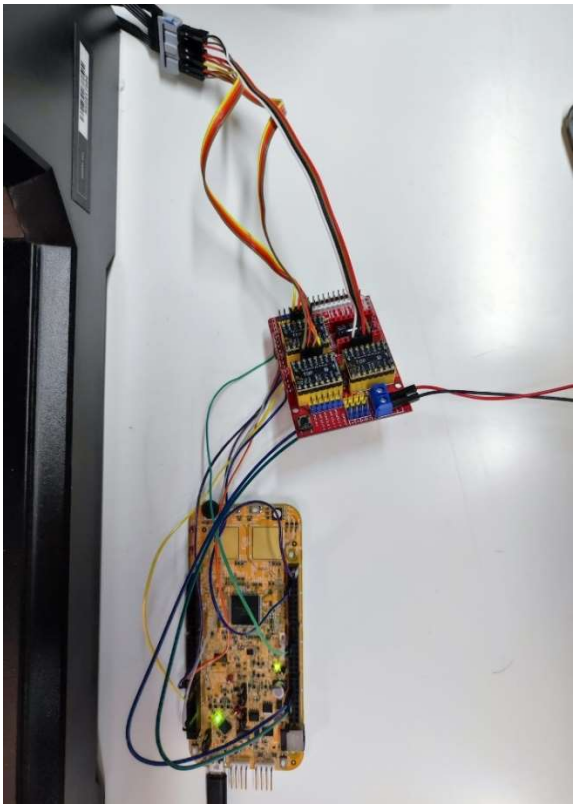




Posizione dei motori

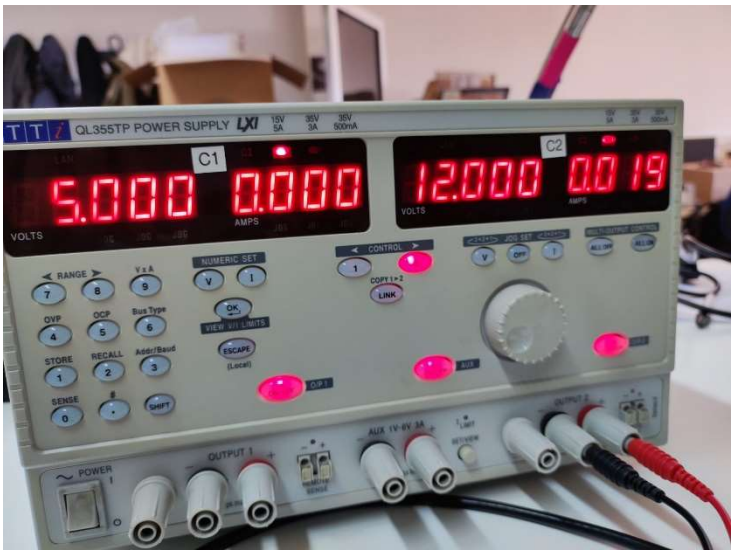


Microcontrollore e CNC Shield

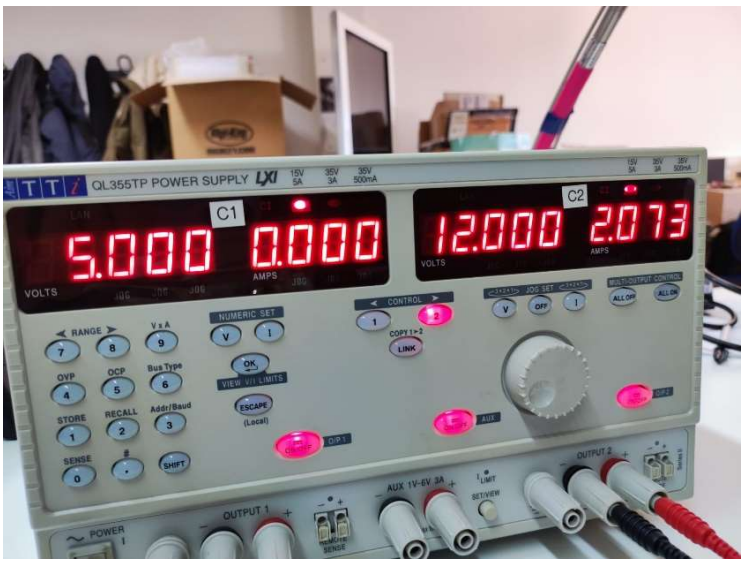


Connessione dei tre motori

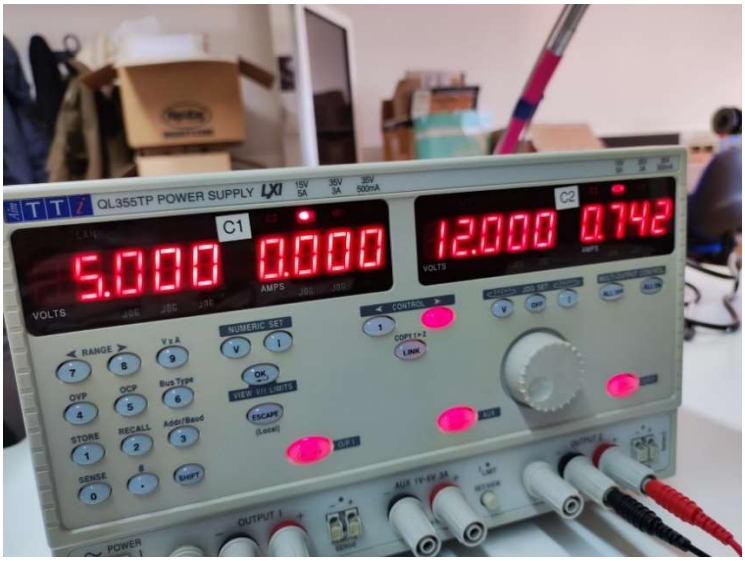
Assorbimento a riposo



Assorbimento per il movimento nel piano XY (due motori)



Assorbimento per movimento lungo asse z (1 motore)



Possibili miglioramenti

Il sistema funziona correttamente, ma potrebbero venire implementate ulteriori funzioni.

- Si potrebbe creare un programma ausiliario che a partire dai componenti sulla PCB, estratti dal gerber, ricava i vari spostamenti differenziali da inviare al sistema.
  Per fare questo bisogna conoscere la relazione tra numero di passi e spostamento, in modo da poter effettuare una conversione tra i cm sulla PCB e le stringhe da fornire tramite la UART.

- Si potrebbe aggiungere sul microscopio un ulteriore motore, per effettuare o il focus manuale, aggiungendo semplicemente nella FSM uno stato che si occupa di questo motore, oppure implementare l'auto-focus.
  Per l'auto focus il PC, a partire dall'immagine acquisita e processata, deve fornire i segnali corretti al microcontrollore in modo da regolare la messa a fuoco tramite il motore.

- Il sistema funziona alimentato tramite l'alimentatore da banco, che dispone delle protezioni da cortocircuito. Volendo passare ad un'alimentazione più standard, come l'alimentatore AC/DC Desktop 12V 100W, potrebbe essere conveniente aggiungere dei fusibili per evitare la rottura della parte di potenza.

- Potrebbero venire aggiunti dei fine corsa, in modo che il sistema non possa mai sbattere contro i bordi, sia nel piano XY che lungo l'asse Z.