

Position and temperature controller for mechatronic application

Embedded Systems course project report
University of Trento, A.Y. 2021/2022

Tommaso Canova, 209270,
Lisa Santarossa, 209386

Abstract—Al giorno d'oggi, sempre più applicazioni biomediche richiedono l'implementazione di sistemi embedded, per garantire performance e affidabilità mantenendo bassi i costi. Attualmente, presso i laboratori di Pro-M Facility è in via di sviluppo un impianto di calibrazione per un macchinario utilizzato nella terapia protonica. L'obiettivo di questa applicazione è quello di ridurre i tempi impiegati per la procedura di calibrazione del macchinario, potendo rendere il processo del tutto autonomo e privo di revisione, diminuendo così la durata dell'operazione da un paio d'ore ad alcune decine di minuti. Lo scopo di questo report è quello di descrivere lo sviluppo della parte di misura e controllo, a supporto del disegno innovativo proposto dall'azienda, realizzato mediante l'utilizzo di: un microcontrollore STM32H7, un encoder magnetico assoluto e dei sensori di temperatura. Durante l'intero percorso è stato utilizzato un approccio *bottom-up*, partendo quindi dallo sviluppo firmware ed hardware dei singoli componenti, per poi creare delle interconnessioni tra di essi. Il progetto è stato completato con successo ed, al momento presente, è del tutto integrato all'interno del macro progetto di Pro-M Facility.

I. INTRODUCTION

In questo report è stato dettagliatamente riportato lo sviluppo del progetto del corso di Embedded Systems, svolto durante l'attività di tirocinio presso i laboratori di Pro-M Facility, società specializzata in prototipazione meccanica, elettronica ed informatica.

Questa azienda, in concomitanza con il periodo di stage, ha sviluppato un sistema efficiente di calibrazione, per un macchinario utilizzato nella terapia protonica, situato presso il Centro di Protonterapia di Trento. Da questa collaborazione nasce l'esigenza di proporre una nuova soluzione per rendere più veloce la calibrazione del sistema rispetto allo standard attuale, il quale richiede la supervisione di un operatore dedicato.

Il sistema proposto è composto da un parte di attuazione e una di misura, su quest'ultima viene posta particolare attenzione in quanto è stata oggetto di ricerca e sviluppo ai fini della realizzazione del progetto del corso. In questa occasione, sfruttando un'esigenza aziendale, è stato possibile esplorare dal lato pratico le nozioni teoriche apprese durante il corso.

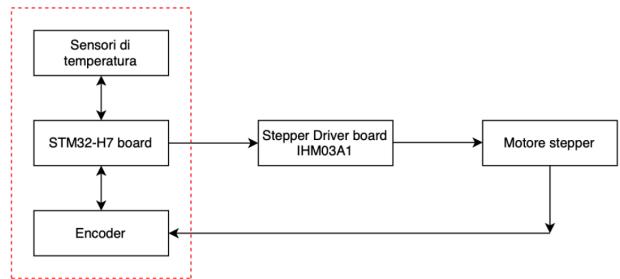


Fig. 1: Schema a blocchi dell'intero sistema. Nel riquadro rosso la sezione di misura e controllo

II. DESCRIPTION OF THE PROJECT WORK

A. Problem statement

Il sistema preso in esame prevede più componenti di misura ai fini di garantire una corretta automatizzazione del processo di calibrazione del macchinario; il microcontrollore gestirà autonomamente il motore basandosi sui dati rilevati dai sensori presenti sulla scheda. In questa sezione saranno oggetto di trattazione i componenti utilizzati, ponendo particolare attenzione all'hardware che li caratterizza per poi esporre la struttura e l'implementazione firmware del microcontrollore.

B. Hardware

1) Nucleo STM32H723ZG: L'intera logica del progetto è gestita dal microcontrollore *STM32H7*, presente sulla board di sviluppo *STM32H723ZG*. La board è stata scelta poiché dotata di interfaccia *ethernet*, sfruttata poi nel progetto completo, inoltre il microcontrollore è in grado di lavorare ad una frequenza di 550 MHz, che rende possibile la gestione immediata di interrupt ricorrenti. Il connettore *JP2* dimostra la versatilità della scheda, in termini di sorgenti di alimentazione, in quanto permette di selezionarne la tipologia mediante un jumper: esterna (connettendo 5V al pin 6 di *CN11*) oppure via *ST-LINK* (5V dati dal connettore USB).

2) Board di espansione: Ai fini di monitorare la temperatura e la posizione del motore, sono stati previsti due sensori di temperatura ed un encoder magnetico assoluto. Per far sì che componenti diversi potessero essere interconnessi tra loro

è stata progettata una board di supporto, la *PCB* dedicata funge dunque da alloggiamento per la scheda *Nucleo*, garantendo così le funzionalità richieste.

Poiché i componenti previsti necessitano di alimentazioni diverse, è stato utile prevedere più connettori per tali tensioni, ovvero: 24V, 5V e 3.3V. I connettori *J6* e *J7* sono dedicati rispettivamente alle alimentazioni esterne di 24V per l'encoder e 5V per la *Nucleo*; mentre la tensione di 3.3V può essere fornita sia esternamente, dal connettore *J8*, sia internamente della *Nucleo*. Si noti che la board presenta due tipi di banchi di connettori, *ZIO* (*CN5*, *CN4*, *CN3*, *CN2*) e *Morpho* (*CN12*, *CN11*), tuttavia in questa prima revisione il secondo non è totalmente sfruttabile, in quanto vengono solo utilizzati due pin di esso (per alimentare la scheda *Nucleo* a 5V).

Le sonde di temperatura previste rientrano nella famiglia dei sensori analogici, mentre l'encoder in quella dei sensori

digitali, ragion per cui, i segnali emessi devono essere interpretati in maniera differente. Le sonde emettono una tensione ai loro capi in funzione del calore, pertanto è richiesto un circuito di condizionamento in grado di misurare, filtrare e amplificare questa grandezza. Invece, per quanto riguarda l'encoder, il modello scelto comunica attraverso due linee seriali differenziali e due singole. Le linee differenziali sono adibite a clock e dati e richiedono un transceiver, il *MAX488E*, per comunicare con la *Nucleo*. In particolare, è necessario l'integrato *CD4050BD* in serie con la linea dati per limitare il valore logico alto a 3.3V evitando che il pin *GPIO* della *Nucleo* si danneggi.

La scheda di supporto prevedrebbe un ulteriore sensore di temperatura, un termistore in grado di comunicare mediante protocollo *SPI*, il quale non è stato montato ed implementato nella revisione trattata.

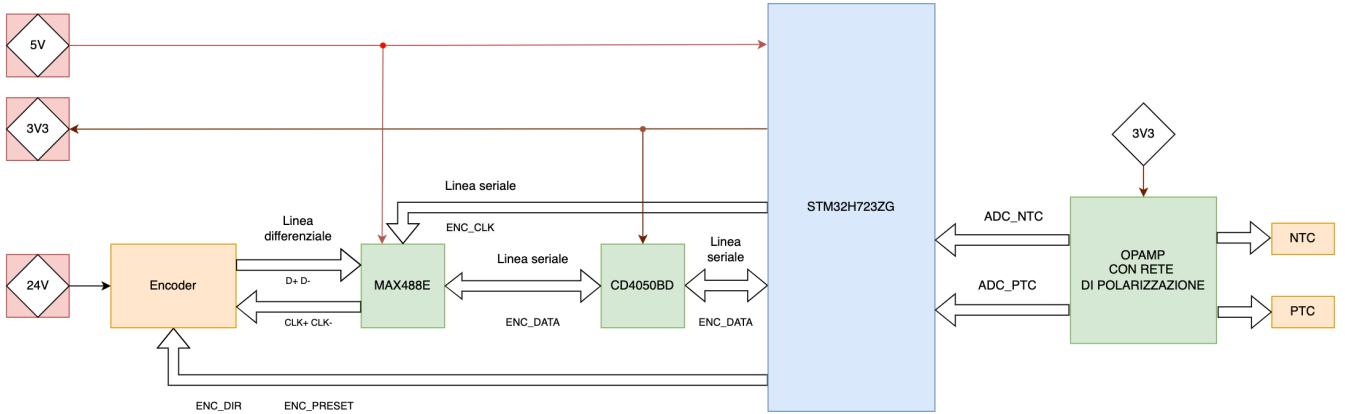


Fig. 2: Schema a blocchi della scheda di supporto

3) *Sensori di temperatura*: Il primo sensore utilizzato per la rilevazione delle temperature è una termistore *PT1000*, della famiglia dei sensori *PTC* (Positive Temperature Coefficient), in quanto tale aumenta la propria resistività in funzione della temperatura. Il coefficiente di temperatura è 3850 ppm/K.

$$\alpha = \frac{R_{100} - R_0}{100 \times R_0} \text{ [K}^{-1}\text{]} \text{ according to the IEC60751, 2009-05 numerical value of } 0.00385 \text{ K}^{-1}.$$

Fig. 3: Costante di temperatura del sensore PTC

Il produttore fornisce inoltre un foglio di calcolo in cui vengono riportati i valori di resistenza in relazione alla temperatura; sfruttando questi dati è stato possibile scegliere la configurazione del circuito di condizionamento.

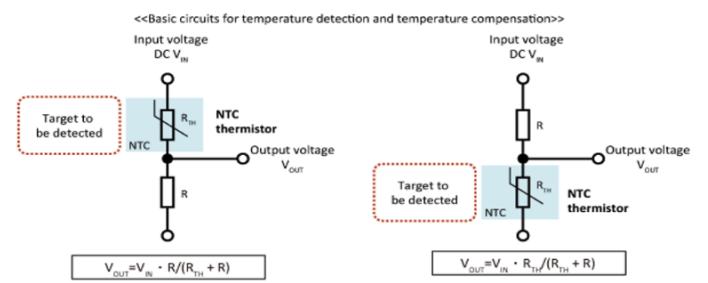


Fig. 4: Disposizione alternativa top e bottom per i termistori

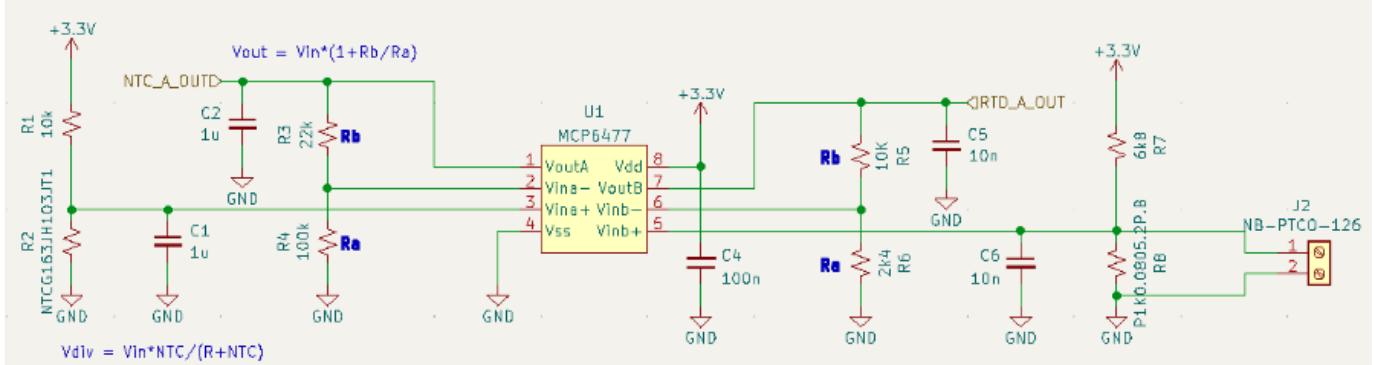


Fig. 5: Circuito di condizionamento per i sensori di temperatura

La lettura dei sensori di temperatura, avviene mediante l'utilizzo dei canali di *ADC* del microcontrollore, il range accettato da tale periferica è [0-3.3V], pertanto si è cercato di avere un'amplificazione adatta, per poter sfruttare al meglio questo intervallo di valori.

Sviluppando ulteriori calcoli aggiuntivi svolti sul foglio di lavoro è stato scelto il circuito di partizione con il termistore in posizione *bottom* ed un'amplificazione di 5.4, ottenendo quindi un range da [1.96-3.17V]. L'amplificazione è data da:

$$1 + \frac{R_B}{R_A}$$

(guadagno tipico di un amplificatore non invertente).

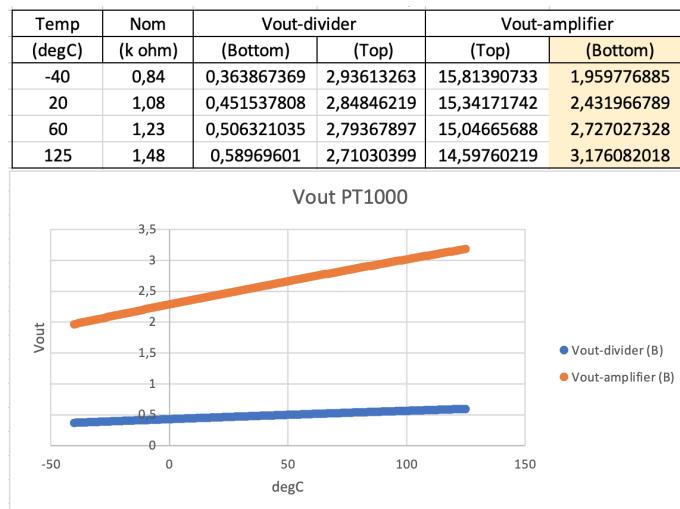


Fig. 6: PTC excel

Il secondo sensore di temperatura utilizzato è un *NTC* (Negative Temperature Coefficient), termistore che, al contrario del primo, diminuisce la sua resistività all'aumentare della temperatura. A 25°C la resistenza nominale è pari a 10kΩ, il coefficiente di temperatura definito dalla formula in figura 7, prendendo in considerazione i valori di temperatura di 25°C e 85°C si ottiene un valore pari a 3435 K.

B constant calculation formula

$$B = \frac{\ln R_1 - \ln R_2}{(1/T_1) - (1/T_2)}$$

B: B constant (K)
T1: Arbitrary temperature (K)
T2: Arbitrary temperature different from T1 (K)
R1: Zero-load resistance value at temperature T1 (Ω)
R2: Zero-load resistance value at temperature T2 (Ω)
Each temperature is measured in absolute temperature. 0°C=273.15K

Fig. 7: Costante di temperatura del sensore NTC

Con un ragionamento analogo al precedente è stato scelto il circuito di condizionamento per il termistore NTC.

Temp (degC)	Nom (k ohm)	Vout-divider		Vout-amplifier	
		(Bottom)	(Top)	(Top)	(Bottom)
-40	188,5	3,133753149	0,166246851	0,202821	3,823179
20	12,09	1,806111363	1,493888637	1,822544	2,203456
60	3,019	0,765243106	2,534756894	3,092403	0,933597
125	0,534	0,167286881	3,132713119	3,82191	0,20409

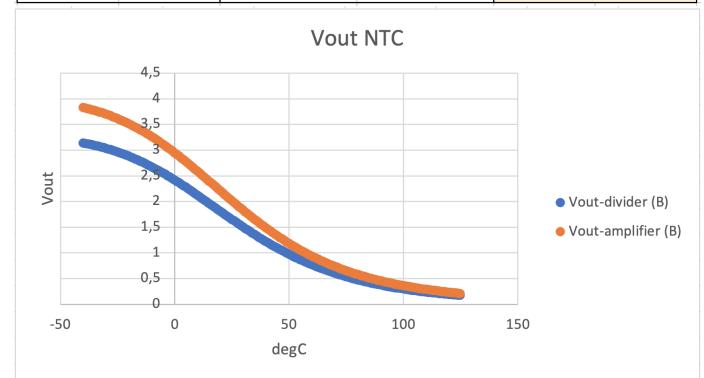


Fig. 8: NTC excel

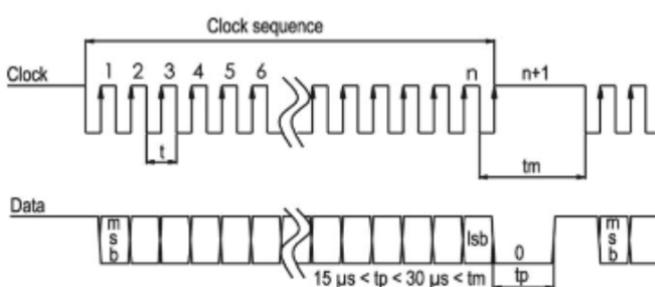
4) Encoder: Per monitorare la posizione del motore è stato scelto encoder magnetico assoluto, modello *WDGA 36A* del produttore *Wachendorff*. Questo dispositivo comunica utilizzando il protocollo *SSI* (Synchronous Serial Interface) e pertanto è stato necessario utilizzare un componente integrato (*MAX488E*) per interfacciarlo con il microcontrollore.

L'encoder è dotato di una linea dati differenziale, che fornisce un pacchetto di dimensione pari a 25 bit, di cui 13 indicano il numero dei giri e 12 la risoluzione in un giro, ottenendo così una precisione di $\frac{360^\circ}{2^{12}} = 0.09^\circ$. Il dispositivo comunicando mediante una linea sincrona seriale necessita di un segnale di clock generato dal microcontrollore, con

frequenza massima di 500 kHz e con un numero di fronti pari alla dimensione del dato. Il frame è composto inoltre da uno o più bit di stop e segue lo standard *Big Endian*.

Questo tipo di encoder può lavorare in due modalità: *single transmission* e *multipath transmission*. In entrambi i casi il dato viene fornito dopo il primo fronte di discesa del clock. Nel primo caso tra due *frame* è necessario tenere alto il livello del clock per un tempo *tm*, in genere maggiore di 30us, in modo tale che possa essere aggiunto un singolo bit di stop al dato e che la linea dati possa ritornare ad un valore alto. La trasmissione *multipath*, invece, permette di memorizzare nel registro a scorrimento dell'*encoder* il dato memorizzato e di mantenerlo fino al successivo *tm*. In questo caso il clock viene fornito in modo continuativo ed ad ogni *frame* verranno aggiunti due bit di stop.

Transmission protocol SSI Single transmission:



Transmission protocol SSI Multipath transmission:

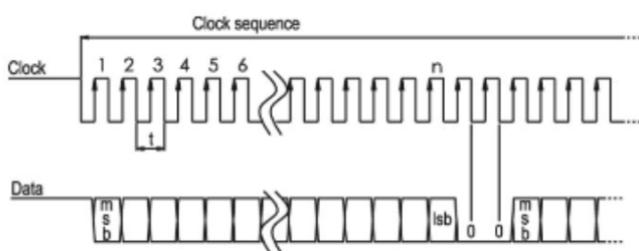


Fig. 9: Modalità di comunicazione encoder

Sono presenti due ulteriori segnali d'ingresso per l'*encoder*, *RESET* e *DIR*. Per azzerare il conteggio dell'*encoder*, il segnale di *RESET* deve rimanere alto per più di due secondi, dopodiché può assumere valore logico basso, mentre il segnale *DIR*, se posto alto permette di contare in modo incrementale i giri in senso antiorario, altrimenti se tenuto basso, in senso orario.

Assignments	
	CB8
GND	1
(+) Vcc	2
SSI CLK+	3
SSI CLK-	4
SSI DATA+	5
SSI DATA-	6
RESET	7
DIR	8
Shield	housing

Fig. 10: Pinout connettore encoder

C. Firmware

1) Ambiente di sviluppo e funzionamento generale del codice:

Per il progetto è stato utilizzato l'ambiente di sviluppo *STM-CubeIDE*, affiancato a *STM CUBE MX*, mentre per il software di controllo della versione è stato scelto *Git*.

Il codice si può suddividere in due macro sezioni, una dedicata alla lettura dell'*encoder* e l'altra dedicata alla misura della temperatura. Nel primo caso il funzionamento del codice è scandito da un segnale di clock a 125 kHz, generato tramite timer; la lettura del dato viene effettuata a metà di un livello logico alto, ovvero, nell'intervallo di tempo che intercorre tra un fronte di salita e il suo successivo fronte di discesa. In seguito, tre pacchetti di dato successivi vengono interpretati e trasmessi via *UART*.

Il codice relativo alla misura di temperatura, è più semplice, ed è basato su un *ADC* che gestisce la lettura dei valori emessi dalle due sonde. Ogni lettura viene poi convertita in un valore di tensione e trasmessa anch'essa via *UART*.

2) File di configurazione ioc:

Sono riportati di seguito i principali pin utilizzati per il progetto e la loro funzionalità:

- **PF4:** NTC (*ADC3_CHANNEL_9*)
- **PF5:** PTC (*ADC3_CHANNEL_4*)
- **PG3:** SSI_ENC_CLK (*GPIO_OUTPUT*)
- **PG2:** SSI_ENC_DATA (*GPIO_INPUT*)
- **PC12:** ENC_RESET (*GPIO_OUTPUT*)
- **PC11:** ENC_DIR (*GPIO_OUTPUT*)
- **PD2:** GPIO_DEBUG (*GPIO_OUTPUT*)

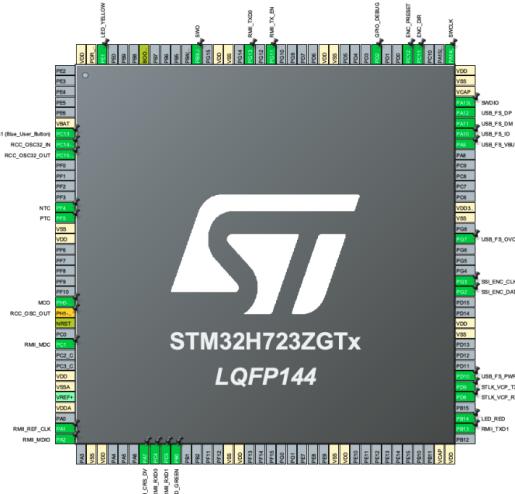


Fig. 11: Pinout del microcontrollore

3) Lettura dell'encoder:

L'encoder per la trasmissione del dato utilizza il protocollo SSI, pertanto per rispettare le specifiche è stato utilizzato un segnale di clock a 125 kHz, il quale, in stato di attesa, si trova a livello logico alto.

A livello implementativo si utilizza un timer che al suo scadere genera un interrupt, nella cui callback viene incrementato il contatore `enc_clk_counts`. In base al valore di quest'ultimo si eseguono diverse operazioni:

- se `enc_clk_counts % 4` è 0 viene letto il bit di dato emesso dall'encoder;
- se `enc_clk_counts % 2` è 0 non viene eseguita nessuna istruzione;
- altrimenti la linea di clock viene commutata;

È stata utilizzata la periferica Timer7 con prescaler pari a 110 e *counter period* di 10, dato che il clock della rispettiva periferica APB1 è di 275 MHz.

$$\bullet \quad f_{timer} = \frac{APB1_{clk}}{\text{Prescaler} \cdot \text{Counterperiod}} = 250 \text{kHz}$$

$$\bullet \quad f_{SSI} = \frac{f_{timer}}{2} = 125 \text{kHz}$$

Il dato è composto da 27 bit, ogni bit viene letto dopo ogni fronte di salita del clock e salvato all'interno dell'iesimo buffer di memoria dell'array `enc_bits`. In questo array verranno salvati 3 (quantità settata da `AVG_MEAS`) pacchetti dati completi, i quali una volta acquisiti verranno mascherati e interpretati come segue:



Fig. 12: Configurazione del clock del microcontrollore

```

for (i = 0; i < AVG_MEAS; i++)
{
    enc_bits[i] >>= 2;
    //13 bits angle
    angle[i] = (int32_t) ((enc_bits[i] & 0x00001FFF) * 44);
    //12 bits multi-turn
    turn[i] = (int16_t)((enc_bits[i] & 0x01FFE000) >> 13);
}

```

Fig. 13: Snippet di mascheramento del pacchetto dati

- i due bit di stop si scartano;
- i 13 bit finali rappresentano l'informazione sull'angolo e moltiplicando il valore per 44 si ottiene l'angolo in *mDeg*, infatti $\frac{360}{2^{13}} \cdot 1000 = 44mDeg$;
- i 12 bit iniziali indicano il numero di giri compiuti.

4) Lettura sensori di temperatura:

Per la lettura dei sensori di temperatura è stata utilizzata la periferica *ADC3*, ad una risoluzione pari a 12 bit, in modalità *Single Rank*. Tipicamente, quando si leggono più sensori analogici, è buona prassi abilitare la modalità *Scan Conversion Mode*, mediante la quale, previa inizializzazione di più canali di conversione, detti *Rank*, viene eseguita un'unica lettura che coinvolge tutti i *Rank* dichiarati (uno dopo l'altro in modo sequenziale). Tuttavia, volendo rendere tra loro indipendenti le librerie dei sensori, non è stata adottata questa modalità di lettura. Di conseguenza ogni sensore di temperatura, nella sua funzione di acquisizione, configura manualmente la propria struct di configurazione (*ADC_ChannelConfTypeDef*), impostando il proprio canale di scansione, permettendo così all'*ADC* di eseguire una singola scansione, evitando di coinvolgere altre periferiche e rendendo questa operazione del tutto indipendente dalle altre possibili letture.

```

ADC_ChannelConfTypeDef sConfig = {0};
sConfig.Channel = NTC_CHANNEL;
sConfig.Rank = ADC_REGULAR_RANK_1;

sConfig.SingleDiff = ADC_SINGLE_ENDED;
sConfig.OffsetNumber = ADC_OFFSET_NONE;
sConfig.Offset = 0;
sConfig.OffsetSign = ADC3_OFFSET_SIGN_NEGATIVE;
sConfig.OffsetSaturation = DISABLE;

sConfig.SamplingTime = ADC3_SAMPLETIME_6CYCLES_5;
if(HAL_ADC_ConfigChannel(&adc, &sConfig) != HAL_OK){
    Error_Handler();
}

```

Fig. 14: Configurazione ADC del sensore PTC

Ogni sensore di temperatura viene definito dal corrispettivo file *.c* e *.h*, nei quali vengono definiti: i valori delle soglie di temperatura, il proprio canale di *ADC* e le funzioni di lettura di stima del livello di temperatura. I livelli di temperatura vengono definiti come *enum* nel file *temp.h*, il quale è comune ad ogni sensore di temperatura.

```
typedef enum{
    COLD = 0U,
    NORMAL,
    HOT,
    DANGER,
    TEMP_ERROR
} temperature_level;
```

Fig. 15: Livelli di temperatura codificati nel codice

La periferica *ADC3* offre 4096 livelli di quantizzazione, con una risoluzione di $805\mu\text{V}/\text{bit}$, ciononostante per i sensori di temperatura non viene utilizzato l'intero range di risoluzione. Di seguito sono riportate le differenze di tensioni equivalenti tra la temperatura minima e massima rilevabili.

- PTC 1.053V, equivalente al 32% del range *ADC*
- NTC 2.340V, equivalente al 71% del range *ADC*

5) Trasmissione seriale:

La comunicazione tra il microcontrollore ed il computer avviene mediante la linea seriale della periferica *USART3*. Non essendo stato previsto alcun *hardware flow control* è stato necessario fissare il *baud rate* a 115200 Bits/s per poter ricevere correttamente i dati in uscita.

III. EXPERIMENTAL RESULTS

In questa sezione vengono elencate diverse considerazioni, ottenute in seguito a test sperimentali.

A. Scelta microcontrollore

Originariamente il microcontrollore di riferimento per il progetto sarebbe dovuto appartenere alla famiglia *F4*, tuttavia è stato necessario utilizzarne uno della serie *H7*; poiché la routine principale in esecuzione nella funzione *main* veniva continuamente interrotta dalla *callback* del timer. Dunque, nonostante il microcontrollore *F4* stesse lavorando ad una frequenza di 84 MHz, non riusciva ad eseguire abbastanza velocemente le istruzioni per gestire ogni singolo evento legato alla *callback*. È stato quindi necessario utilizzare un microcontrollore più potente, avente una frequenza di lavoro di 550 MHz.

B. Problemi di EMC

Sulla scheda si sono manifestati disturbi elettromagnetici, per esempio nella configurazione in cui l'alimentazione del microcontrollore viene fornita via USB, è stato rilevato un disturbo sinusoidale che si è poi protratto sui pin di output del microcontrollore, generando malfunzionamenti generali, per questo motivo è stata scelta la configurazione con alimentazione esterna.

C. Oscillazioni dell'encoder in posizione zero

L'encoder quando si trova in posizione di partenza trasmette valori che oscillano intorno lo zero, questa imprecisione è dovuta in primo luogo dalla struttura del sistema meccanico. L'encoder è, infatti, connesso all'albero del motore tramite un sistema di trasmissione a cinghia che presenta del gioco, pertanto queste oscillazioni, seppur minime, vengono rilevate e amplificate dalla circuiteria dell'encoder. Questa imprecisione sui dati trasmessi è dovuta in parte anche da rumore elettronico, il quale potrebbe essere ridotto utilizzando cavi di segnali più corti e schermati.

D. Taratura sonde

Durante la taratura delle sonde di temperatura: il sensore *PTC* e il relativo circuito di condizionamento si è rivelato poco preciso, pertanto il range dell'*ADC* sfruttato durante la sua lettura è risultato assai limitato (32%). Per ovviare a questa imprecisione, sarebbe stata più adatta la scelta di componenti con tolleranza inferiore. Mentre, il termistore *NTC* si è rivelato eccessivamente instabile, con oscillazioni della tensione ai suoi capi nell'ordine dei 300 mV, pertanto è stato necessario sostituire i suoi condensatori di filtro da 10 nF con quelli da 1uF.

Per entrambi i sensori la taratura è avvenuta per tentativi, adattando il guadagno e confrontando l'output rispetto ad un sensore di temperatura a sé stante fornito dal tutor aziendale (utilizzato come riferimento).

Tipicamente, per una taratura professionale, si analizzano i singoli componenti presenti nel circuito di condizionamento, in questo caso sarebbe stato corretto confrontare i valori nominali di resistenza corrispondenti ai vari livelli di temperatura in un ambiente a temperatura ed umidità controllata. Dopodiché si sarebbe dovuto verificare l'effettivo guadagno dell'amplificatore, basandosi sui valori di tolleranza dei resistori.

IV. CONCLUSION

ENCODER: 0 0 0 mDeg 0 0 0 turns	ENCODER: 18524 18524 18524 mDeg 13 13 13 turns
Temperatures: PTC 2.294531 [V] PT1000 TEMP ZONE: NORMAL	Temperatures: PTC 2.209131 [V] PT1000 TEMP ZONE: NORMAL

NTC 2.274390 [V] NTC TEMP ZONE: NORMAL	NTC 2.271973 [V] NTC TEMP ZONE: NORMAL
---	---

Fig. 16: Output dopo il reset dell'encoder (sx) e dopo alcuni giri (dx)

Il progetto è stato completato con successo ed implementato all'interno del sistema di calibrazione di *ProM Facility*. Passo dopo passo, sono state applicate le nozioni teoriche e gli esempi pratici appresi durante il corso di *Embedded Systems*, facendo fronte agli imprevisti, dovuti all'inesperienza.

Ai fini della realizzazione del *PCB* sono risultate fondamentali anche le competenze apprese durante il corso di *Progettazione di Sistemi Elettronici*, grazie alle quali si è potuto progettare, prototipare e debuggare correttamente l'hardware.

Questo tracciato, come tutti i percorsi di apprendimento, si è rivelato lungo e ricco di imprevisti, ciononostante è stato un ottimo strumento per approfondire ed applicare gli argomenti trattati a lezione, esplorandone anche di nuovi mediante diversi problemi ingegneristici.

ALLEGATI

Repository del progetto:

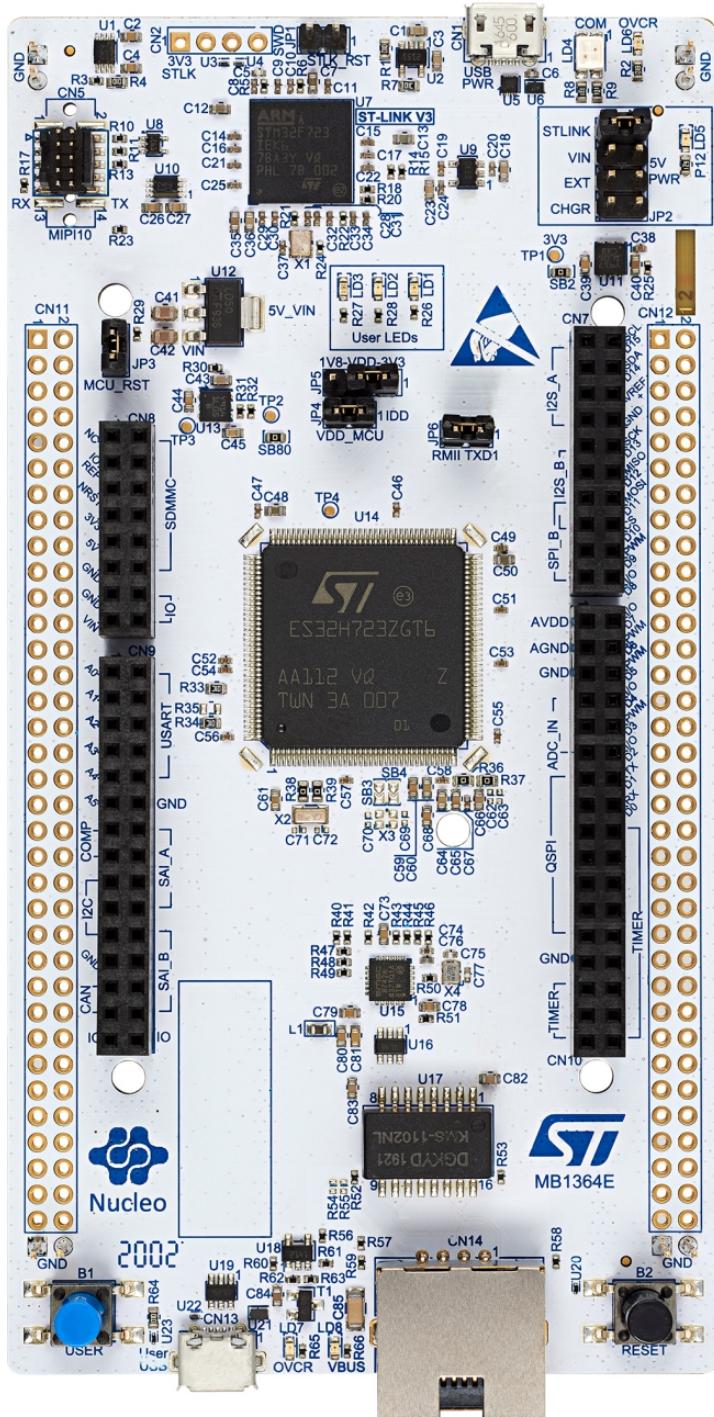
- Github repository

Datasheet:

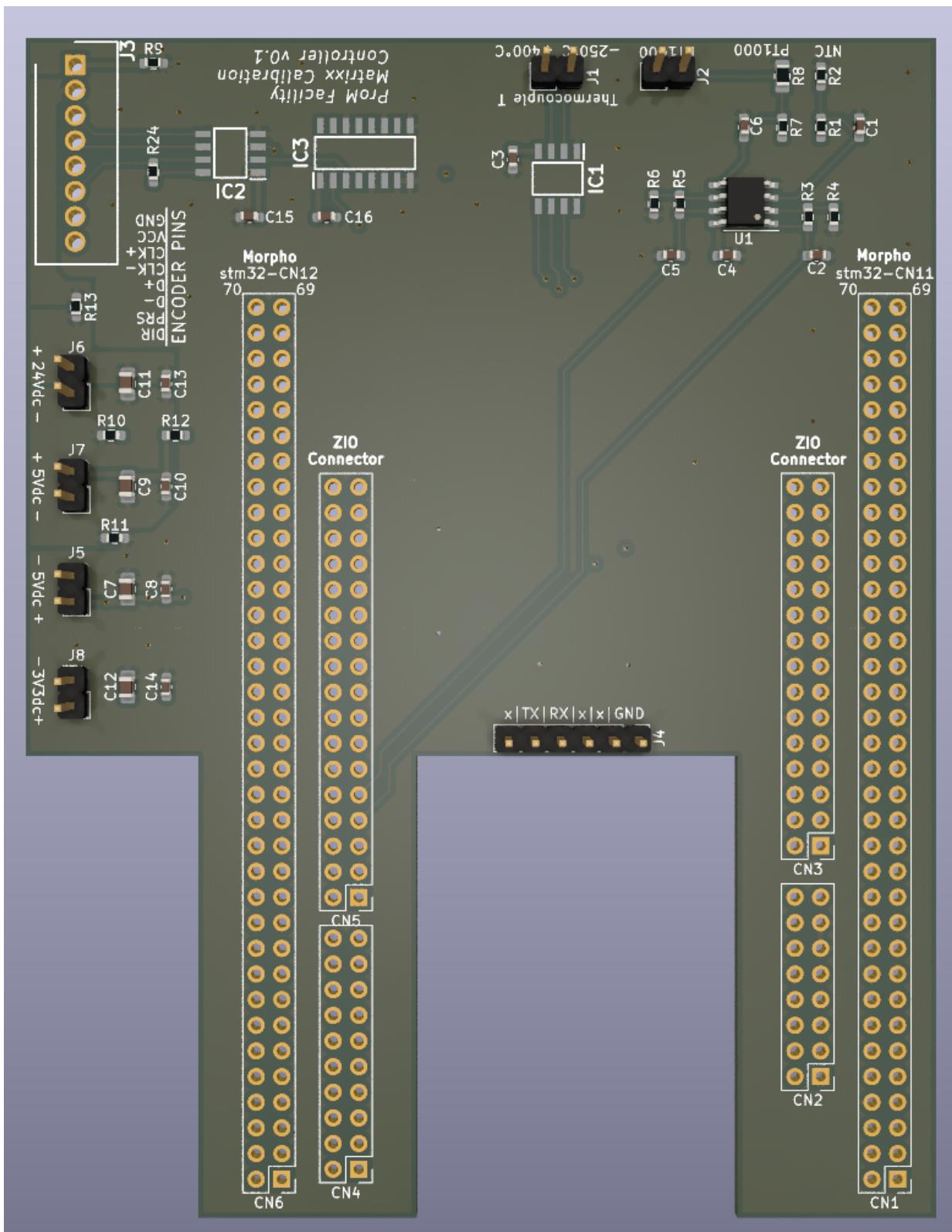
- CD40B .
- WDGA 36A
- WDGA 36A General technical Data
- STM32H723ZG
- Arm Corted M7
- MAX 488E
- OPAMP
- PTC
- NTC

IMMAGINI

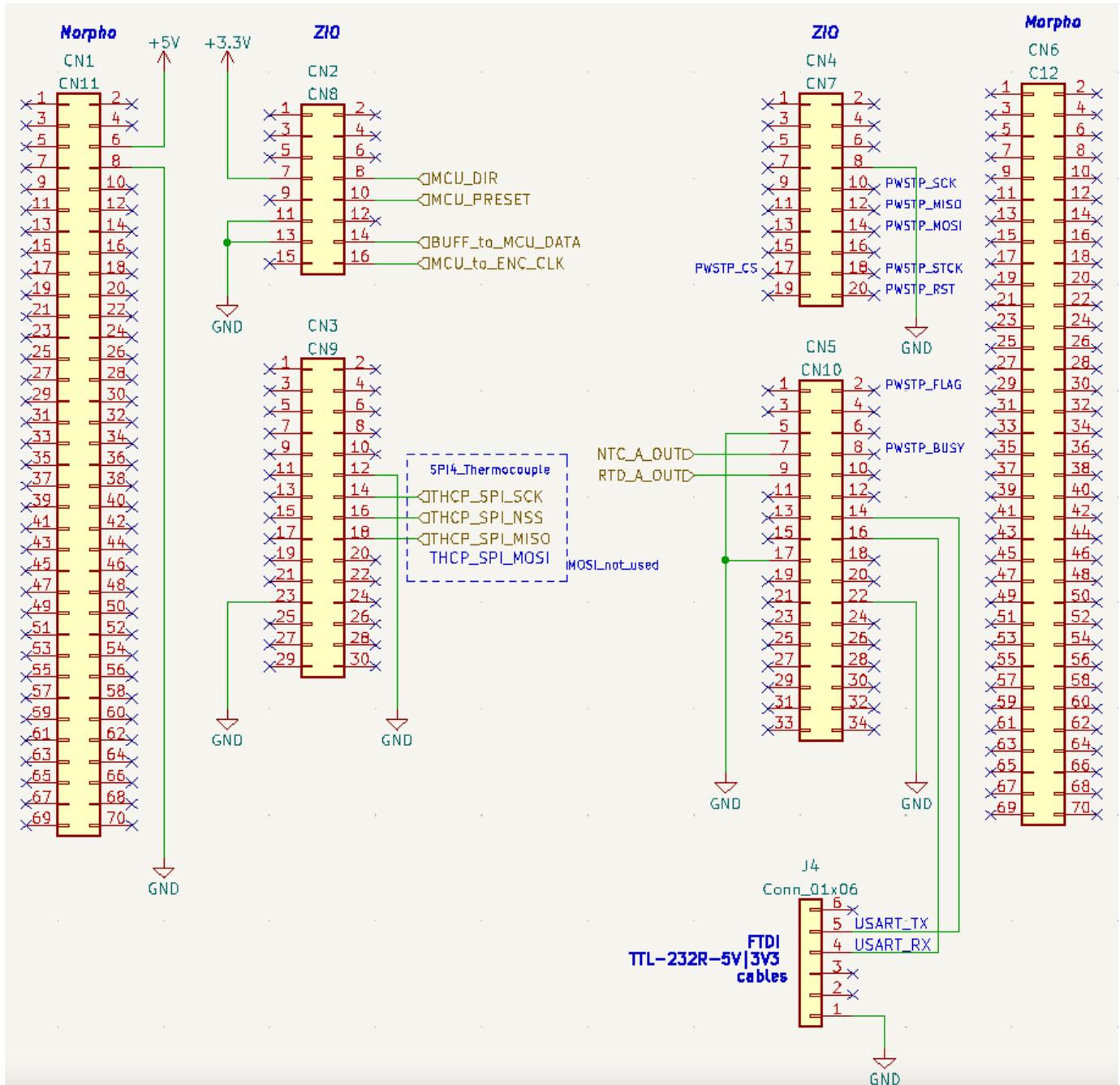
A - Scheda STM32H723ZG:



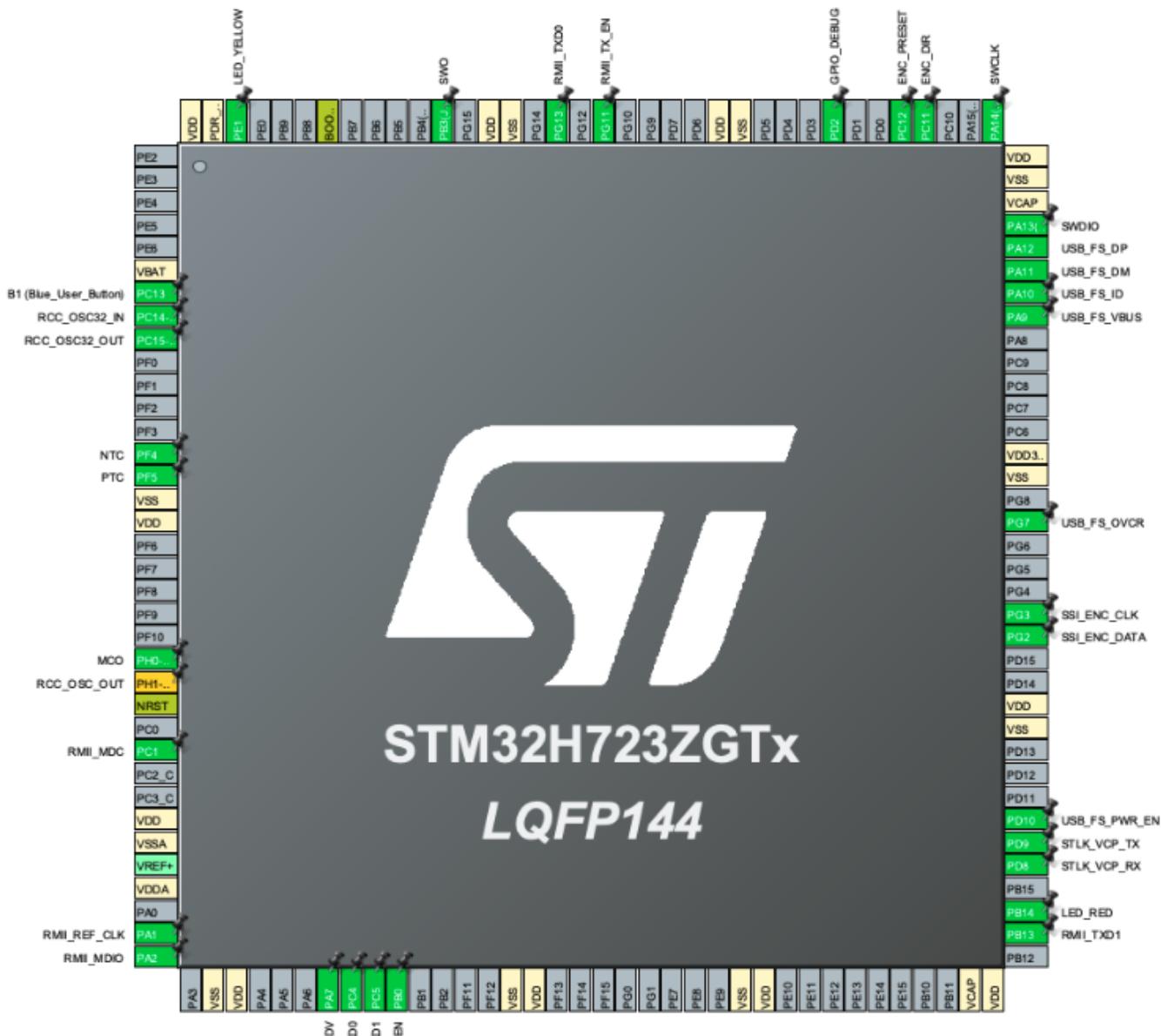
B - Board di espansione (o di supporto):



C - Connettori Morpho e Zio sulla board di espansione:



D - Pinout MCU:



E - Linea clock e linea dati durante una lettura dell'encoder:



F - Setup hardware:

