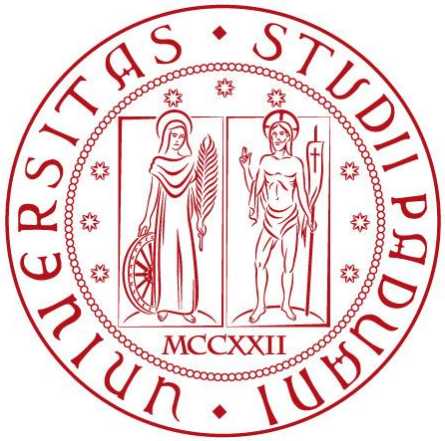


Corso di Laurea Magistrale in Ingegneria Meccatronica

Microcontrollori e DSP

Implementazione di un controllo PI per termo-resistenza

Matricola	Cognome	Nome
1197747	Braceschi	Marco
1197740	Rigon	Saverio
1205388	Zanrosso	Luca

Obiettivi del progetto:

- Controllare la temperatura associata ad un sistema termo-resistenza.
- Utilizzare un segnale PWM per regolare la potenza dissipata sul resistore e quindi la temperatura.
- Implementare un controllo di tipo PI su microcontrollore.
- Emulare il sistema termo-resistenza con un circuito RC.
- Confrontare la risposta ad anello aperto con quella ad anello chiuso.

Specifiche di progetto :

- Costante di tempo della termo resistenza: 6 secondi.
- Frequenza di attraversamento: 0.4Hz.

CODICE IMPLEMENTATO

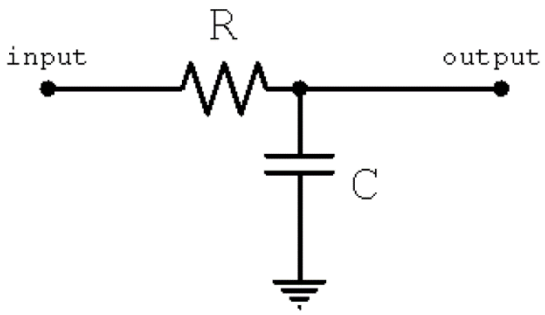
```
#include "MKL25Z4.h"  //device header, contiene un richiamo dei registri di configurazione delle periferiche

#define ki_norm 28795
#define kp_norm 30883
#define const T 32767/100

signed int tempADC;
signed int error=0; // e(k)
signed int Tref = 0; // duty x REF
int yI = 0; // yI(k)
int y = 0; //y(k)

void TPM_init(void) {  //genero il segnale di PWM

            SIM->SCGC6 |= (1UL<<SIM_SCGC6_TPM0_SHIFT); //abilito il clock della periferica
            TPM0->SC = (1UL<<1) | (1UL<<2) | (1UL<<3) | (1UL<<6); //abilito i'interrupt di overflow, abilito CMOD e prescaler pari a 64
            TPM0->CONTROLS[2].CnSC  = (1UL<<5) | (1UL<<3); //portante a dente di sega allineata a sinistra

            TPM0->CONTROLS[2].CnV = 0; // inizializzo il valore di match a zero
            PORTD->PCR[2] = (1UL<<10); //collego l'uscita del canale 2 del TPM0 al al pin D2
            NVIC_EnableIRQ(TPM0_IRQn); //abilito l'interrupt
}

void ADC_init(void){  //L'ADC mi serve per leggere il valore di tensione che è proporzionale al valore di temperatura
```

```
	SIM->SCGC6|=(1UL<<SIM_SCGC6_ADC0_SHIFT); // abilito il clock della periferica
	ADC0->CFG1 = (1UL<<4)|(1UL<<0)|(1UL<<3) | (1UL<<2); //opero in busclock/2, conversione in 16bit e longSample time
	ADC0->SC3 = (1UL<<2)|(1UL<<1)|(1UL<<0); //configuro il registro in modo da usare media su 32 acquisizioni, miglioro l'accuratezza
}

void LED_init(void){   //uso i led B18 e D1 come segnali visivi sincronizzati con la freq di acquisizione

	SIM->SCGC5 |= (1UL<<12) | (1UL<<10);  //abilito il clock porte B e D

	//selezione del pin corrispondente, MUX 10-8 al valore 001
	PORTD->PCR[1] = (1UL<<8);
	PORTB->PCR[18] = (1UL<<8);

	// selezione dei pin D1 e B18 come uscite, metto a 1 i relativi registri
	FPTD->PDDR = (1UL<<1);
	FPTB->PDDR = (1UL<<18);

	// accendo il led blu e spengo quello rosso
	FPTD->PCOR = (1UL<<1);
	FPTB->PSOR = (1UL<<18);

}

int main(void){
//richiama le tre routine di inizializzazione (quelle sopra)
	SystemCoreClockUpdate();//routine per aggiornare il clock
	LED_init();
	TPM_init();
	ADC_init();

	while(1){
		//loop infinito, starò qua in attesa di un interrupt
	}
}

int TPM0routine(void) {
```

```
        TPM0->SC |= (1UL << 7); //azzero il flag di overflow del TPM0

        // LED toggle routine, passo dal rosso al blu e viceversa
        FPTD->PTOR = (1UL<<1);
        FPTB->PTOR = (1UL<<18);

        ADC0->SC1[0] &= (!(1UL<<1)); //scrivo una serie di zeri su tale registro al
                                                    //fine di iniziare una nuova conversione sul canale DADP0P0

        while((ADC0->SC1[0] & ADC_SC1_COCO_MASK)== 0){} //attendo la fine della conversione monitorando il flag coco

        tempADC = ADC0->R[0];  //assegno alla variabile tempADC il valore acquisito
        tempADC = tempADC >> 1; //dato ADC a 16bit, shifto verso dx di uno per riportarmi in notazione Q15
        error = constT*Tref - tempADC; //calcolo l'errore, portando Tref in notazione decimale Q15, su 100° max

        /*
        Si calcola la risposta integrale: poiché bisogna riportarsi in notazione Q15 si shifta a destra di 15 posizioni perché il prodotto raddoppia il numero di  bit frazionari. Prima si era normalizzato il valore di Ki moltiplicandolo per 2^2 e Kp dividendolo per 2^4, però ora si necessita dell'uscita reale e quindi la costante deve tornare ad avere il suo valore originario. Complessivamente si ottiene uno shift a destra di 17 per ki_norm e 11 per kp_norm
        */
        yl = ((ki_norm*error) >> 17) + yl;
        y = ((kp_norm*error) >> 11) + yl;

        /*
        Controllo Anti WindUp: l'uscita del regolatore è limitata all'intervallo 0-1, quindi bisogna verificare che il valore non ecceda
        il valore max considerato in notazione Q15, cioè pari a 2^15 = 32767; lo stesso per il limite inferiore 0.
        */
        if (y > 32767) {
                yl = yl - ((ki_norm*error) >> 17);
                y = 32767;
        } else if (y < 0) {
                yl = yl - ((ki_norm*error) >> 17);
                y = 0;
```

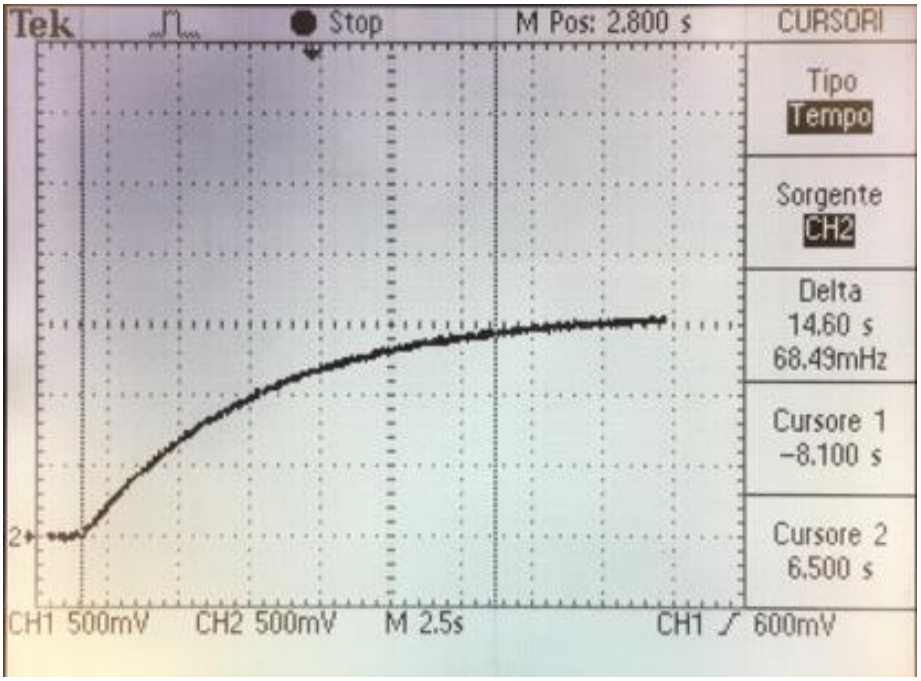
```
        }
        /*
        Assegniamo il valore dell'uscita al modulatore PWM che si
occupa poi di pilotare il circuito.
        E' necessario comunque allineare il dato su 16bit, quindi
shiftiamo verso sinistra di una posizione
        */
        TPM0->CONTROLS[2].CnV = y << 1;


        return 1;
}
```

Duty cycle e PWM

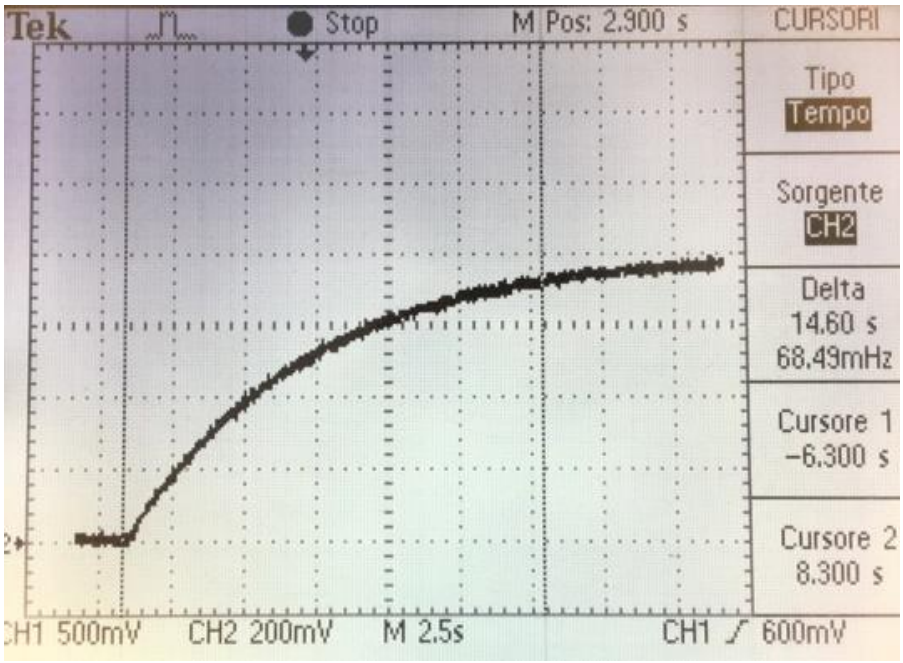


$$V_{out} = \frac{1}{1 + sRC} V_{in}$$

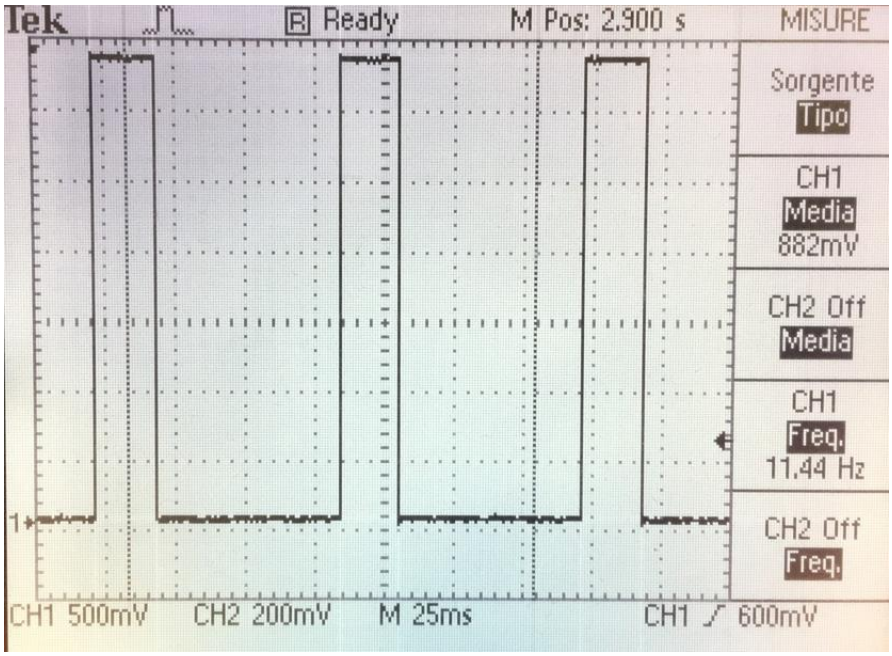
Il sistema termo-resistenza può essere emulato mediante il circuito in figura, dove R = 270kΩ e C = 22µF, ove τ = R*C = 5.94s. La tensione d'ingresso V_{IN} è il valore medio della PWM, mentre V_{OUT} rappresenta la temperatura della termo resistenza espressa in tensione (da 0 a 3.3V).



Nella figura sopra riportata è raffigurata la risposta del sistema ad un duty cycle del 50%, corrispondente ad un valore medio di ingresso pari a circa 1.5V. Il tempo necessario per arrivare al 90% del valore massimo è di 14.6s.



In questa seconda figura invece viene riportato il medesimo test con duty cycle pari al 25%; questo implica un V_{IN} = 0.25 * 3.3 = 0.825V, per ricavare il $t_{90\%}$ ci siamo posizionati circa a 0.74V, ricavando un tempo identico a quello precedente.



La forma d'onda in ingresso al circuito analogico equivalente è riportato sopra, si nota che esso varia da 0 a 3.3V, in tal caso abbiamo un duty del 25%, come si può notare contando i quadrati temporali riportati sull'oscilloscopio. La frequenza dell'onda quadra è proprio quella decisa a priori impostando il prescaler a 64, cioè pari a 11.44 Hz.

Progettazione Regolatore PI

Per la progettazione del regolatore PI, è stato considerato il seguente guadagno d'anello, con

$$R(s)T(s) = \frac{k_i}{s}\left(1 + s\frac{k_p}{k_i}\right)\left(\frac{1}{1+s\tau}\right)$$

retroazione unitaria. E' stato deciso di utilizzare lo zero del regolatore PI al fine di cancellare il polo della termoresistanza; inoltre, la banda passante del sistema retroazionato deve essere di 0.4Hz. Da questi due vincoli sono stati calcolati i coefficienti Ki e Kp del regolatore:

1) $\frac{k_p}{k_i} = \tau$
2) $|R(i2\pi f_c)T(i2\pi f_c)| = 1 \text{ con } f_c = 0.4Hz$

Imponendo il modulo del guadagno d'anello pari a 1, dopo la cancellazione polo-zero, è stato ricavato il valore di Ki = 2.5133 = w_C , successivamente Kp = Ki * τ = 15.0796.

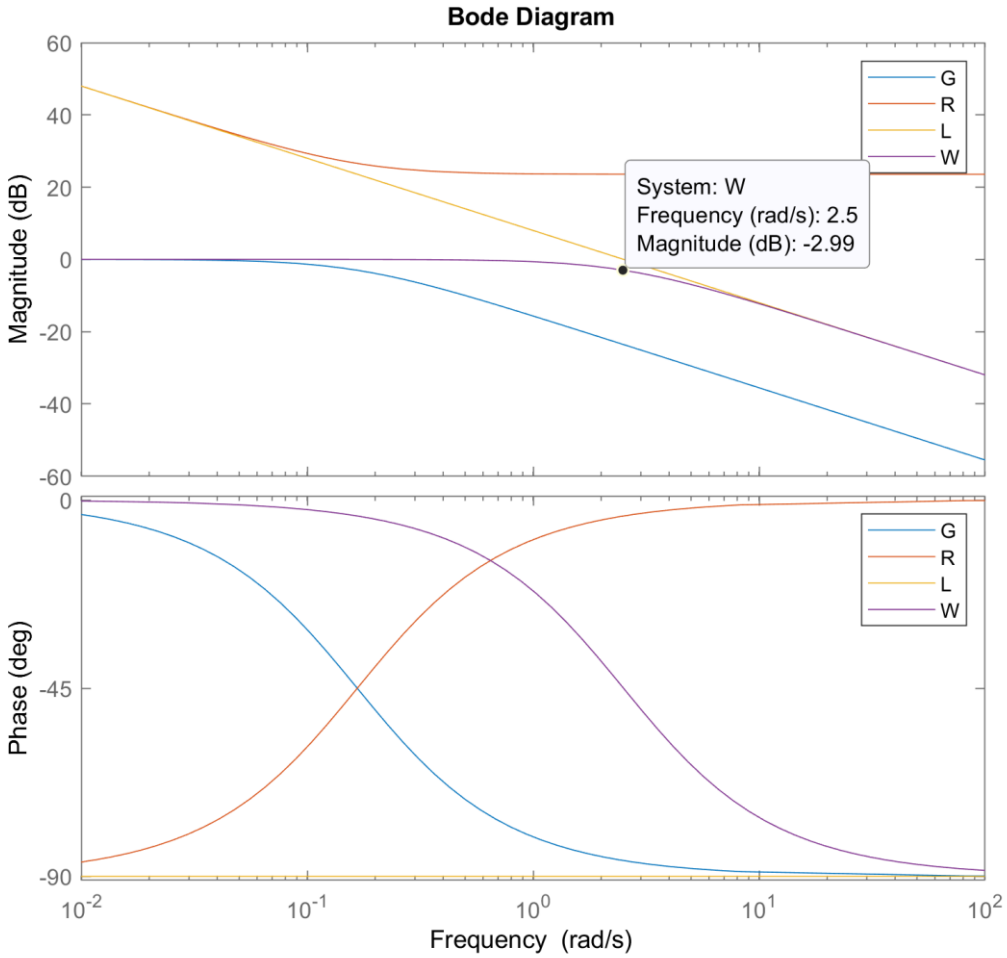
Con l'ausilio di Matlab, è stato verificato che la funzione retroazionata W rispetti le specifiche, infatti dal diagramma del modulo si evince una banda passante di 0.4Hz ed un margine di fase di 90°. (G = sistema del primo ordine della termo resistenza, R = regolatore PI, L = guadagno d'anello, W = funzione retroazionata).

Nel passaggio da continuo a discreto il Ki viene moltiplicato per Tc = 1/11.444 = 0.0874, quindi si ottiene Ki' 0.21962. Ora è necessario riportare tali valori in una notazione Q15. Per sfruttare tutti i bit a disposizione, per Kp è stato scelto come fattore di normalizzazione 2^4, mentre per Ki' 2^-2.

I valori in formato Q15 decimale risultano:

- Kp = 0.942475

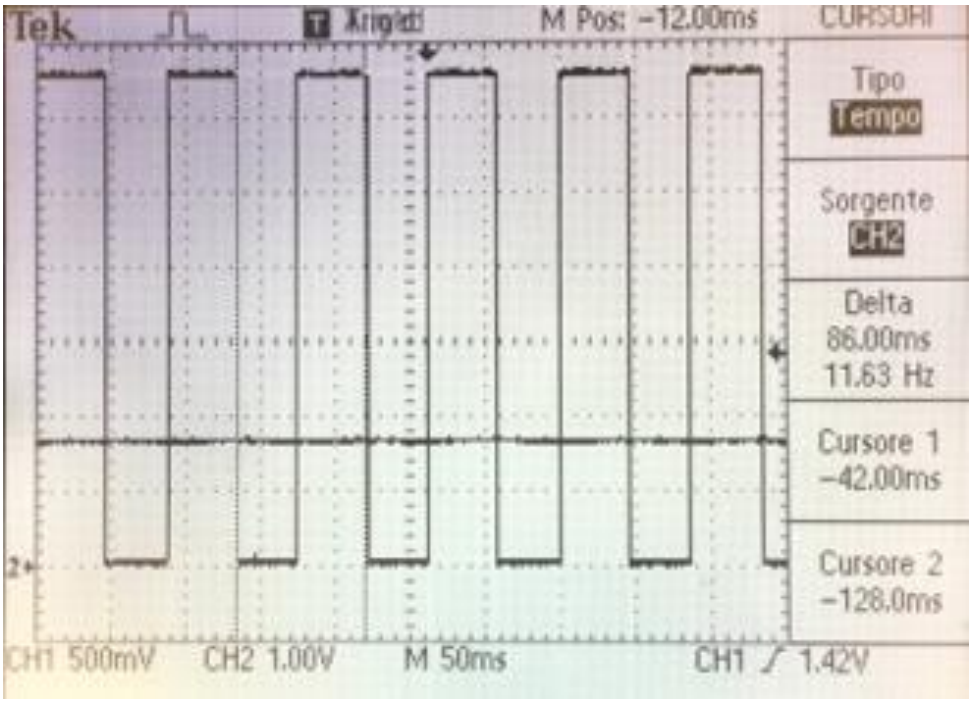
- Ki = 0.87846



In binario: primo bit del segno e gli altri la parte frazionaria

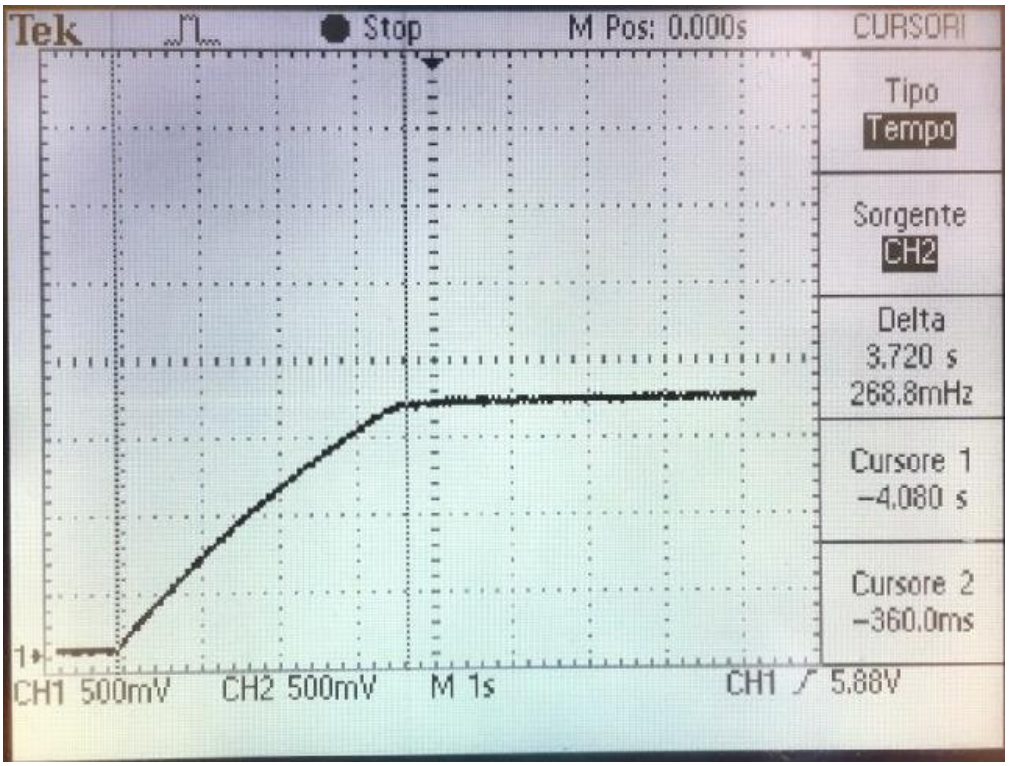
- Kp 0111 1000 1010 0011 (30883)
- Ki 0111 0000 0111 0001 (28785)

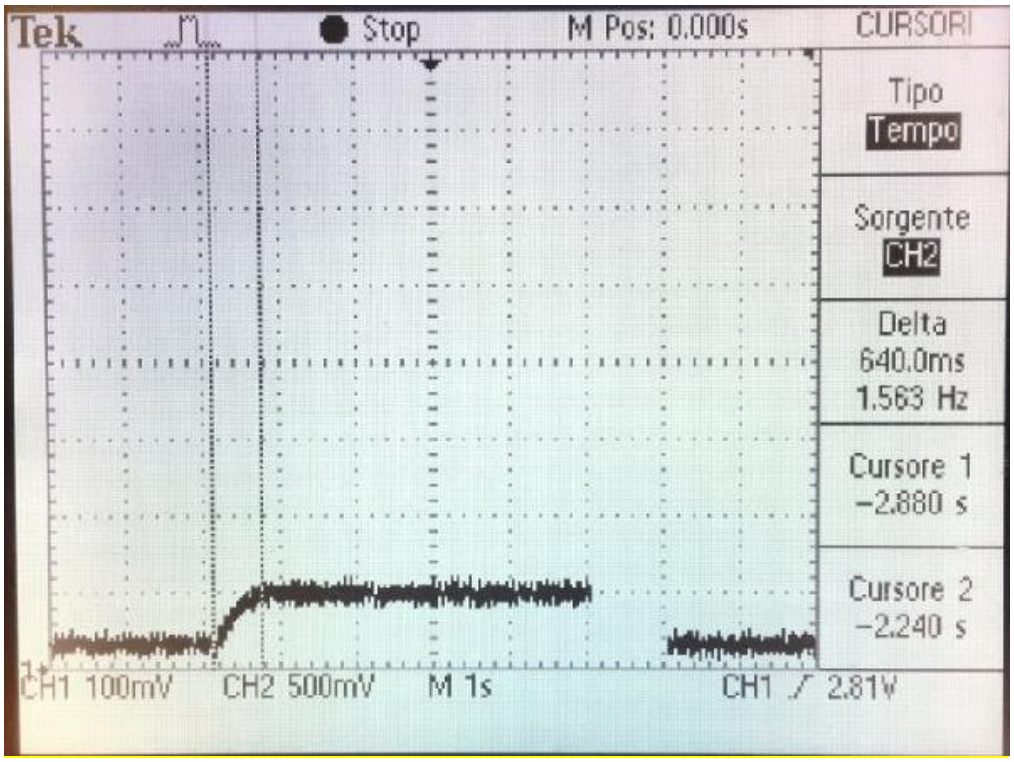
Test Sperimentale



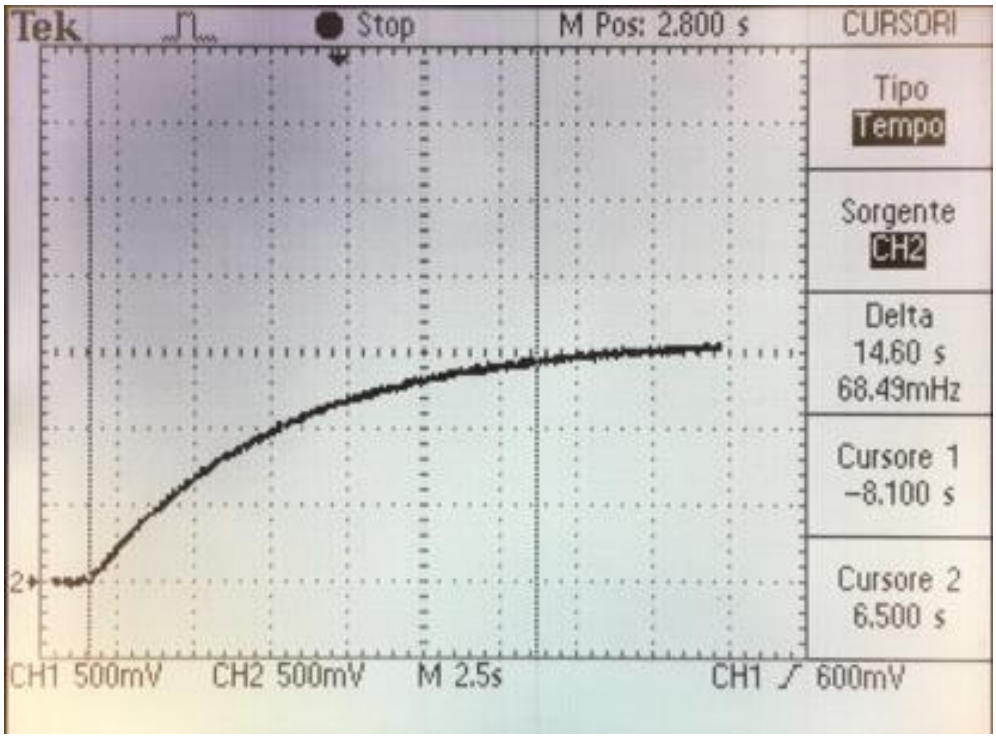
L'immagine sopra riportata mette a confronto la PWM sul canale 1 con cui pilotiamo il circuito, e l'ingresso dell'ADC sul canale 2 che rappresenta l'uscita del circuito. La PWM è realizzata con un duty del 50%, infatti si nota dal confronto con l'immagine seguente che il tempo del livello alto corrisponde a quello del livello basso; la forma d'onda varia da 0 a 3.3V. Per quanto riguarda l'ingresso dell'ADC, il segnale risulta costante e pari proprio a metà della sua massima escursione, cioè circa 1.5V.

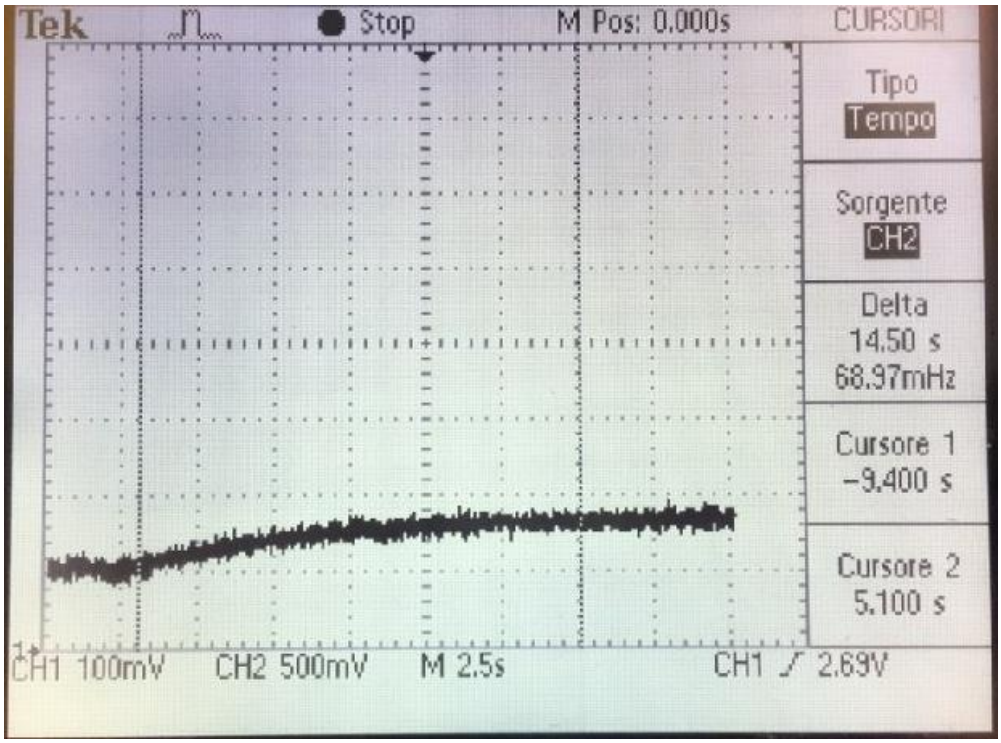
Dopo aver aggiunto sulla variabile Tref un watchpoint, con il codice in modalità free-running è stata modificata tale variabile per passare da 0° a 50° e da 50° a 52°. La prima figura si riferisce al transitorio 0 - 50° mentre la seconda da 50 – 52°. Si nota subito una sostanziale differenza della forma d'onda rispetto al transitorio senza regolatore PI. Inoltre anche i tempi per raggiungere la temperatura di riferimento sono molto più brevi. Attraverso l'oscilloscopio si è misurato un tempo di 3.72s per il transitorio 0 – 50°, mentre il tempo per il transitorio 50 – 52° è di 0.64s.


Tek
Stop
M Pos: 0.000s
CURSORI
Tipo
Tempo
Sorgente
CH2
Delta
3.720 s
268.8mHz
Cursore 1
-4.080 s
Cursore 2
-360.0ms
CH1 500mV
CH2 500mV
M 1s
CH1 ∕ 5.88V

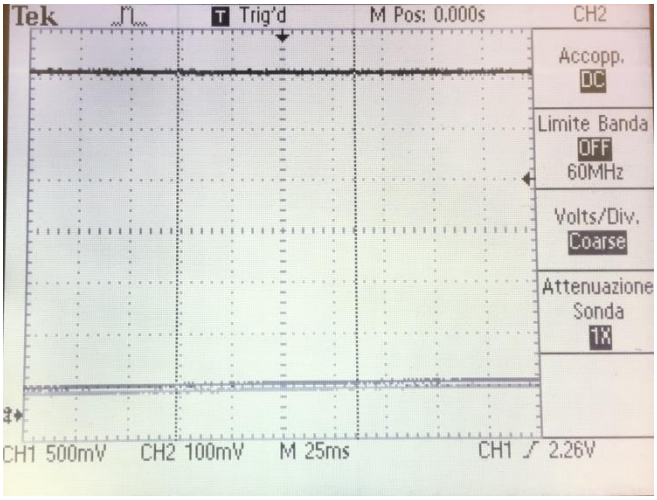

Tek
Stop
M Pos: 0.000s
CURSORI
Tipo
Tempo
Sorgente
CH2
Delta
640.0ms
1.563 Hz
Cursore 1
-2.880 s
Cursore 2
-2.240 s
CH1 100mV
CH2 500mV
M 1s
CH1 ∕ 2.81V

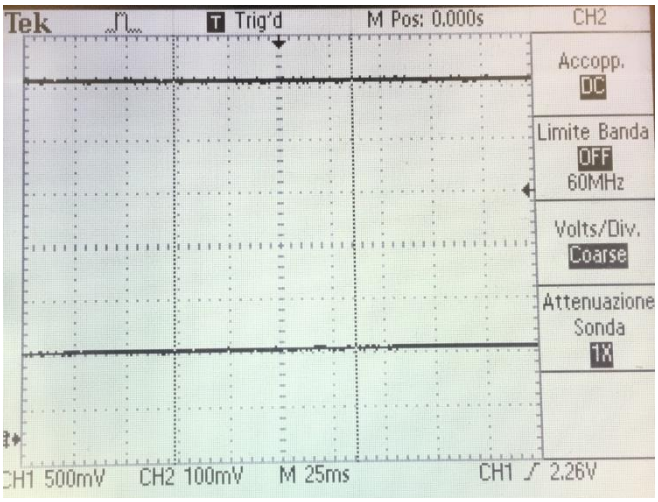
Ad anello aperto, invece, il tempo per raggiungere il 90% della temperatura desiderata è sempre attorno i 14.6s, anche nel caso 50 – 52°, come si può vedere dalle immagini sotto riportate.





Anti Wind-Up





In entrambi i casi, da 0° a 50° e da 50° a 52°, si nota l'innesco dell'anti wind-up. Ad anello aperto, una volta scelta la temperatura, viene applicato un duty-cicle direttamente collegato alla temperatura desiderata. Ad esempio, se la temperatura voluta è 50° viene applicato un duty-cicle di 0.5 a prescindere dalla temperatura di partenza. Ad anello chiuso, invece, la componente integrare del regolatore PI sottoposta all'anti wind-up fa in modo che venga applicato un duty-cicle pari a 1 o 0 fino a quando la temperatura non è prossima a quella desiderata, come si vede delle immagini soprastanti. In questo modo si raggiunge la temperatura voluta in un tempo minore (o uguale se la temperatura voluta è 100° o 0°) rispetto al sistema ad anello aperto. Al raggiungimento della temperatura verrà prodotto lo stesso duty-cicle del sistema ad anello aperto.