UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"TEMPERATURE CONTROL LABORATORY"

Relatore: Prof. Schenato Luca

Laureando: Bellato Filippo

ANNO ACCADEMICO 2021 – 2022

Data di laurea 14 novembre 2022

Contenuti

Abstract

Il punto di forza del sistema universitario è quello di fornire agli studenti una solida base teorica sugli argomenti trattati nelle materie previste del piano di studi. Per consentire tuttavia allo studente di avere una preparazione completa e potersi affacciare al mondo del lavoro in maniera competitiva, non viene trascurata la componente pratica e applicativa, che si concretizza nei laboratori didattici previsti da molti corsi. Può però risultare difficile predisporre un percorso di laboratorio che sia realizzabile (in termini sia economici che fisici) e allo stesso tempo utile per l'apprendimento.

Per questo motivo c'è stata negli ultimi anni una crescente ricerca di soluzioni low cost, pronte all'uso e di facile utilizzo. Tra le varie soluzioni molte università hanno cominciato a realizzare dei laboratori "take home", che offrono agli studenti la possibilità di avere dei kit economici e portatili. I dispositivi take home svincolano a tutti gli effetti lo studente dal contesto universitario (di laboratorio) e danno la possibilità di realizzare simulazioni ed esperimenti in qualsiasi luogo ed in qualsiasi momento.

Nella tesi, dopo una panoramica su questo tipo di dispositivi, che si differenziano tra di loro sia per architettura che per applicazione, ho analizzato in particolare il kit TCLab.

TCLab è un dispositivo low cost (acquistabile online per 40€) pensato per l'applicazione di nozioni teoriche sulla teoria dei controlli. Il chip, basato su Arduino, permette di controllare due riscaldatori e realizzare un controllore PID per controllarne la temperatura. Ho approfondito una serie di aspetti e ho potuto fare delle simulazioni, confrontandole allo stesso tempo con i risultati sperimentali.

Nello specifico ho creato dei modelli, realizzato un PID digitale e un meccanismo di feedforward e ho condotto diverse simulazioni di casi reali.

1. Introduzione

Le lezioni introduttive alla teoria dei controlli automatici richiedono che gli studenti acquisiscano concetti sia teorici che pratici. I laboratori richiedono specifiche attrezzature che, molto spesso, non sono facilmente accessibili per via dei costi e del numero di studenti. Per questo motivo prevedere un laboratorio per ogni corso risulta molto difficile, talvolta impossibile. L'utilizzo di simulazioni può essere d'aiuto, ma non potrà mai rimpiazzare l'esperienza pratica. Per ovviare a questa mancanza una soluzione valida è rappresentata da Arduino [1], uno strumento diffuso nei laboratori di controlli automatici. Esso presenta però delle criticità, tra le quali il fatto che molte delle sue applicazioni prevedono che lo studente assembli in autonomia il proprio kit. Di conseguenza, la combinazione di una soluzione low cost e pronta all'utilizzo insieme a del materiale gratuito non è facile da trovare.
Recentemente un kit molto interessante che soddisfa i requisiti appena citati è stato proposto da Apmonitor Dynamics and Control [2], per il controllo della temperatura, chiamato Temperature Control Laboratory (TCLab) [3]. TCLab, infatti, è un dispositivo low cost, pronto all'utilizzo e portatile che ha riportato numerosi casi di successo nei laboratori di insegnamento/apprendimento.
Nella mia tesi ho utilizzato il TCLab per condurre una serie di esperimenti legati alla teoria dei controlli, partendo dalla ricerca di un modello, passando al design di un controllore pere ottenere una serie di risultati sperimentali. Tramite questa tesi ho potuto provare direttamente l'efficacia didattica di uno strumento quale il TCLab, che mi ha permesso di comprendere a pieno concetti che fino ad ora avevo affrontato solo a livello teorico.

1.1 Obiettivi

In questa tesi ho focalizzato la mia attenzione su quattro aspetti principali. Il primo legato alle alternative possibili di un laboratorio low cost e gli altri tre riguardanti il TCLab. Nello specifico:

- Identificazione di un modello con approccio ”black box”, ovvero senza conoscere il modello fisico del sistema ma basandosi solamente sulla sua risposta ad un segnale in ingresso
- La realizzazione di un PID digitale
- Un confronto tra simulazioni e risultati sperimentali applicati sul TCLab

1.2 Struttura della tesi

La tesi è strutturata in 7 capitoli. Successivamente al capitolo introduttivo, il resto della tesi è organizzata come segue:

- Capitolo 2: Panoramica dei dispositivi portatili per i laboratori sulla teoria dei controlli
- Capitolo 3: Introduzione al kit TCLab approfondendo sia l'aspetto hardware che software
- Capitolo 4: Ricerca di un modello
- Capitolo 5: Realizzazione di un PID digitale
- Capitolo 6: Confronto tra simulazioni e risultati sperimentali
- Capitolo 7: Conclusione

2. Dispositivi portatili per i laboratori sulla teoria dei controlli

Lo scopo di questo capitolo è di presentare alcuni kit basati su Arduino per il controllo. In questi esempi è chiaro lo scopo di creare un dispositivo portatile, che permette agli studenti di non essere vincolati dagli orari di laboratorio e lezione.

2.1 Take home embedded control unit

Questo kit consiste in un circuito integrato che include un processore ARM 32-bit e numerose periferiche. E' un dispositivo portatile che può essere programmato e alimentato tramite cavo USB ed è molto semplice da configurare. Il punto di forza è la sua economicità, con un costo intorno a 50£, che permette agli studenti di acquistarlo anche autonomamente. Il chip incorpora inoltre diverse funzionalità tra cui memoria flash, unità di controllo, DAC/ADC e supporto per diversi protocolli di comunicazione. Il opera con il linguaggio C e utilizza Keil μvision, per via del suo uso industriale.
Lo scopo del kit è quello di introdurre l'architettura del dispositivo e i protocolli di comunicazione per poi focalizzarsi sui meccanismi di controllo hardware e software, per esempio gli interrupt e i sistemi operativi in real-time. [4]

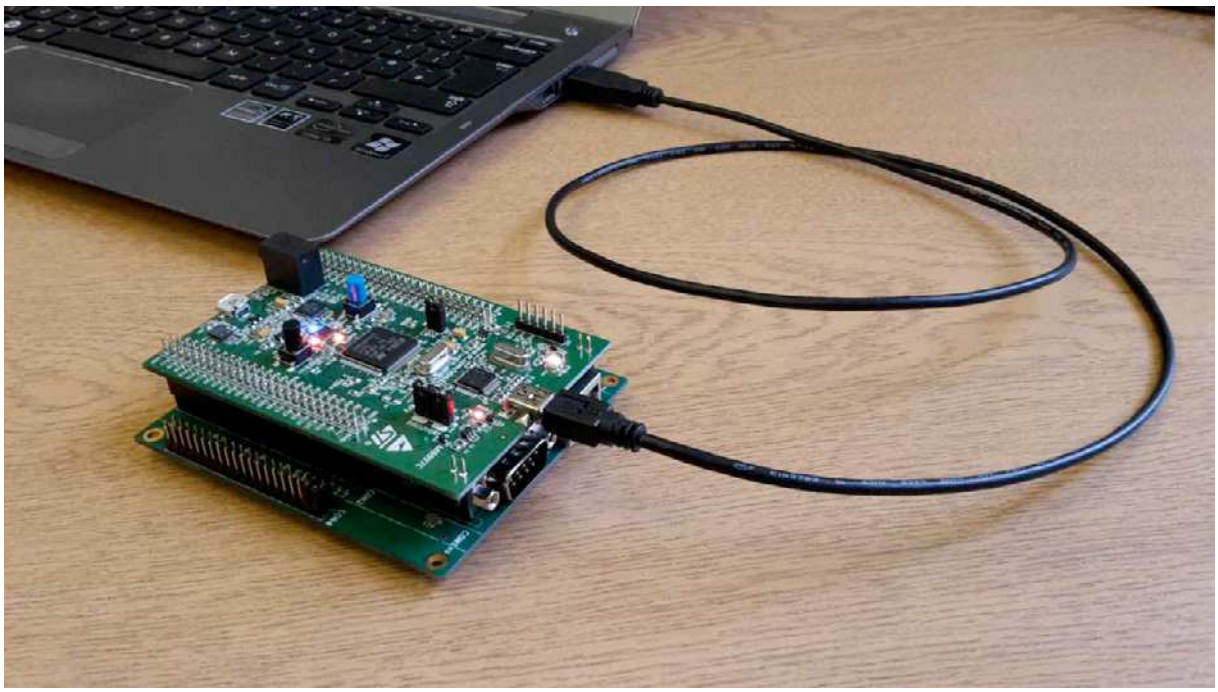

Figura 2.1: Take home embedded control unit

2.2 Take home helicopter emulator

Questo prodotto si focalizza sull'economicità al fine di essere di supporto per l'insegnamento di concetti avanzati di controllo e ingegneria dei sistemi. Il prodotto si concretizza in una giuntura meccanica che può ruotare con tre gradi di libertà in risposta a due attuatori (ventole). Ogni asse è dotato di sensore che ne misura la velocità angolare. Si tratta di un sistema multivariabile e non lineare, il che rende difficoltoso il controllo.

Lo scopo del kit è quello di applicare concetti di controllo e acquisizione dati, modellazione e simulazione di sistemi fisici multivariabile, identificazione del sistema e controllo multivariabile.

Il dispositivo è programmabile con Matlab e LabVIEW. [4]

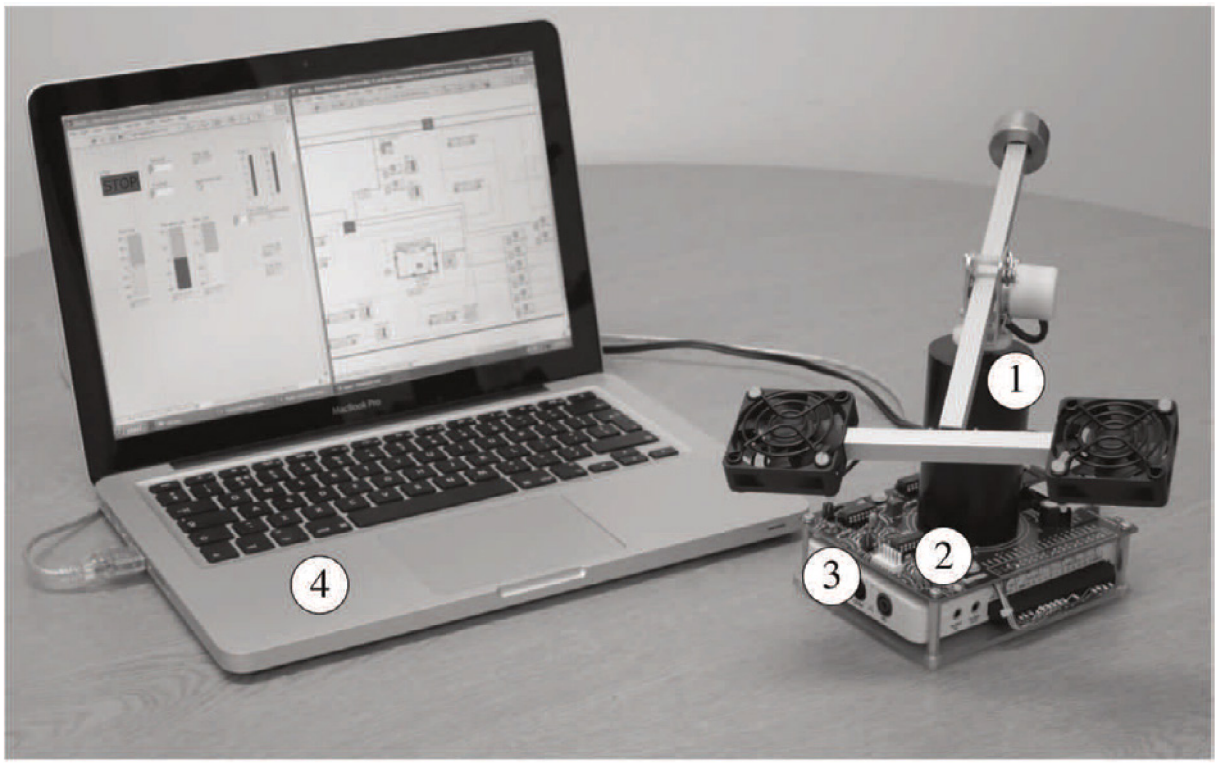

Figura 2.2: Take home helicopter emulator

2.3 Ball and Beam

Questo kit si compone di un Arduino, un'asse e una pallina, che possono essere acquistati oppure realizzati con una stampante 3D. Gli attuatori agiscono su un motore DC, che determina l'inclinazione dell'asse. Il laboratorio si focalizza sull'introduzione graduale di concetti teorici come la modellizzazione di un sistema fisico e l'acquisizione dei dati in tempo reale. I dati analogici sono letti tramite dei termoresistori ad infrarossi per poi essere campionati e convertiti in digitale.

Essendo anche questo dispositivo basato su Arduino può essere facilmente programmato utilizzando Matlab e Simulink. [5]

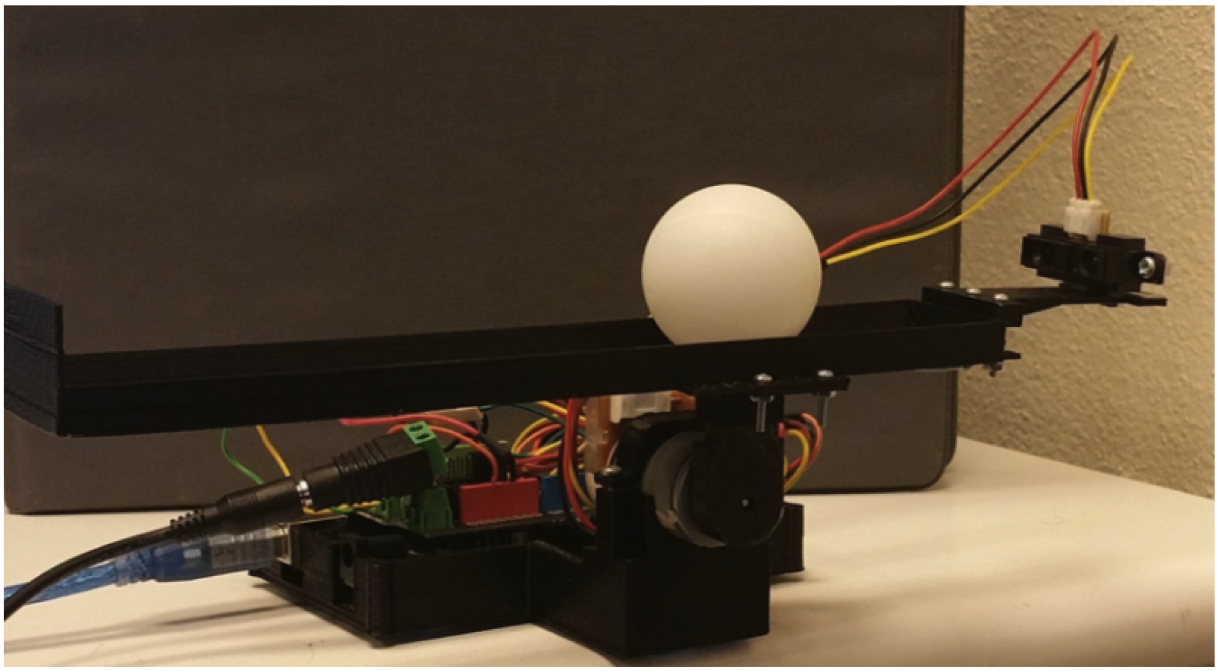

Figura 2.3: Ball and beam

3. Temperature Control Laboratory (TCLab)

Il "Temperature control laboratory" (TCLab) è un'applicazione del controllo in retroazione realizzata tramite un Arduino, un LED, due riscaldatori e due sensori di temperatura. La potenza in uscita dei riscaldatori è regolata per mantenere la temperatura ad un punto di riferimento (detto set point). L'energia termica dei riscaldatori è trasferita per conduzione, convezione e irraggiamento verso i sensori di temperatura. Il calore è dissipato anche nell'ambiente circostante. Questo dispositivo è una risorsa per l'identificazione di un modello e lo sviluppo di un controllore. E' un dispositivo tascabile con software disponibile in Python, MATLAB e Simulink allo scopo di rafforzare la teoria dei controlli per gli studenti.[6]

3.1 Architettura hardware

Il TCLab è un Arduino al quale sono stati aggiunti alcuni componenti. In particolare è composto da due sensori di temperatura, collegati a due pin analogici, due riscaldatori digitali, collegati a due pin digitali e un led che segnala che il dispositivo è acceso.
In aggiunta ho connesso i due riscaldatori con una barretta metallica. Lo scopo è quello permettere ai due riscaldatori di interferire l'uno con l'altro e creare dunque un elemento di disturbo. In questo modo, se un riscaldatore è acceso, l'aumento di temperatura sarà percepibile da entrambi i termometri.
Il TCLab dispone della tradizionale presa Jack per l'alimentazione e una presa micro USB. Lo scopo è quello di ottenere un sistema fisico controllabile, dotato di attuatori (i riscaldatori), sensori (i termometri) e un controllore (implementato via software sul pc collegato al TCLab). Questo consente di applicare le nozioni teoriche della teoria dei controlli automatici ad un caso concreto.
L'architettura del chip è mostrata in figura 3.1.

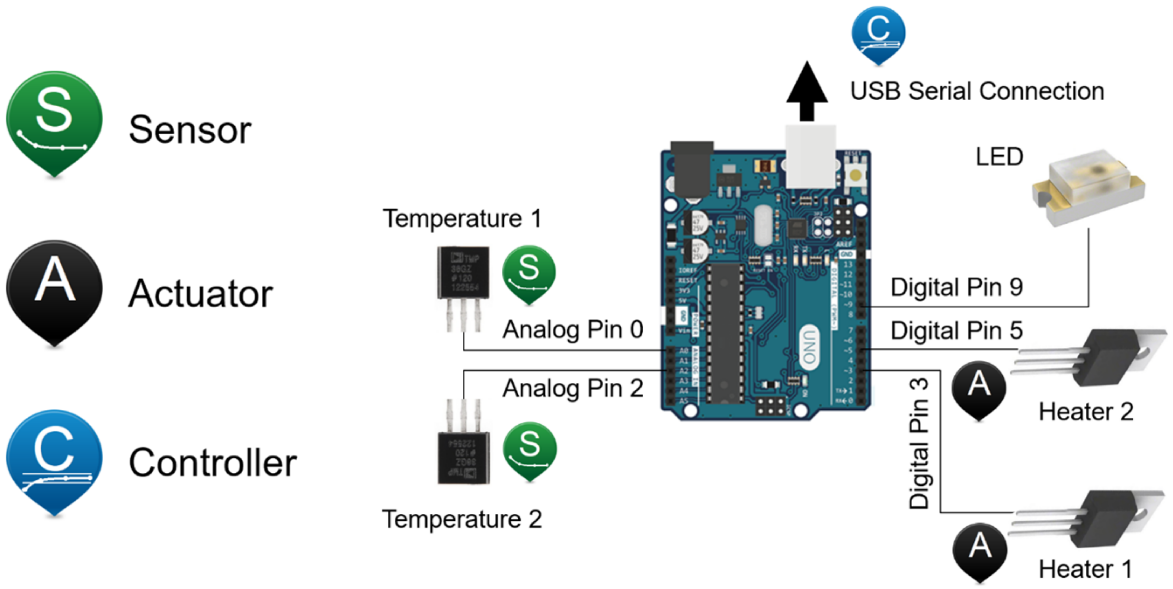

Figura 3.1: Architettura del TCLab

3.2 Software

Oltre al dispositivo hardware, il progetto fornisce una libreria Python per interfacciarsi con il TCLab. Il package tclab fornisce una serie di strumenti Python per interfacciarsi con il dispositivo. L'insieme del package tclab e del chip TCLab forniscono una piattaforma sperimentale low-cost per implementare algoritmi comunemente usati per il controllo di un processo.
Il package si installa con un normale comando pip della libreria python

```
pip install tclab
```

Una volta installato, il package può essere importato ed è possibile creare un'istanza

```
import tclab
lab = tclab.TCLab()
```

Le temperature dei due sensori sono accessibili tramite gli attributi $T1$ e $T2$

```
T1 = lab.T1
T2 = lab.T2
```

I due riscaldatori possono essere modificati tramite l'attributo $Q1$ e $Q2$. L'intervallo di utilizzo per i due riscaldatori è da 0% a 100%

```
lab.Q1(100)
lab.Q2(100)
```

4. Modellizzazione

Il TCLab è un dispositivo multivariabile, dove due ingressi (i riscaldatori) agiscono rispettivamente su due uscite (i sensori di temperatura). L'aggiunta della barretta metallica, che collega le due uscite, impone che le due uscite non siano indipendenti l'una dall'altra. Se il riscaldatore 1 viene acceso, il sensore 1 rileverà un aumento di temperatura. Tuttavia il calore arriverà per conduzione anche al sensore 2, che rileverà un aumento di temperatura ritardato rispetto al sensore 1.
Lo schema è quello in figura 4.1.

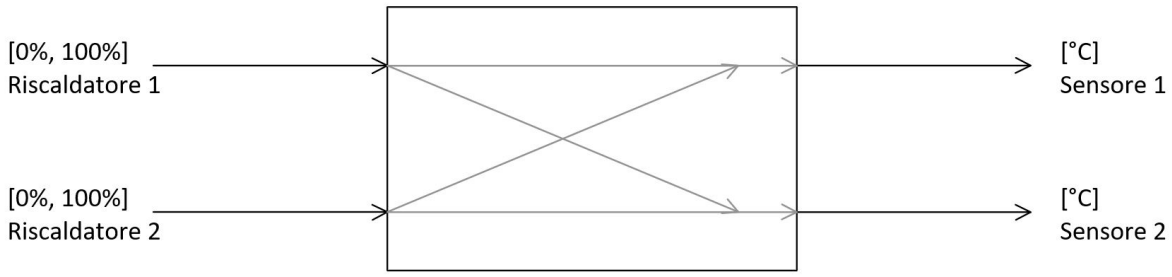

Figura 4.1: Schema a blocchi del TCLab multivariabile

Nel mio caso utilizzerò il riscaldatore 1 come variabile da controllare (tramite il PID) e il riscaldatore 2 come variabile di disturbo. Questo mi permette di impostare manualmente il disturbo desiderato (riscaldatore 2) e verificare che il controllore compensi correttamente, modificando il valore del riscaldatore 1. In aggiunta implementerò un meccanismo di feedforward che, misurando il valore del disturbo (riscaldatore 2), agirà direttamente sul processo senza il bisogno dell'intervento del PID. La temperatura che voglio controllare è quella del sensore 1. Il sensore di temperatura 2 non sarà utilizzato.
Lo schema a blocchi in figura 4.2 mostra come utilizzerò il TCLab.

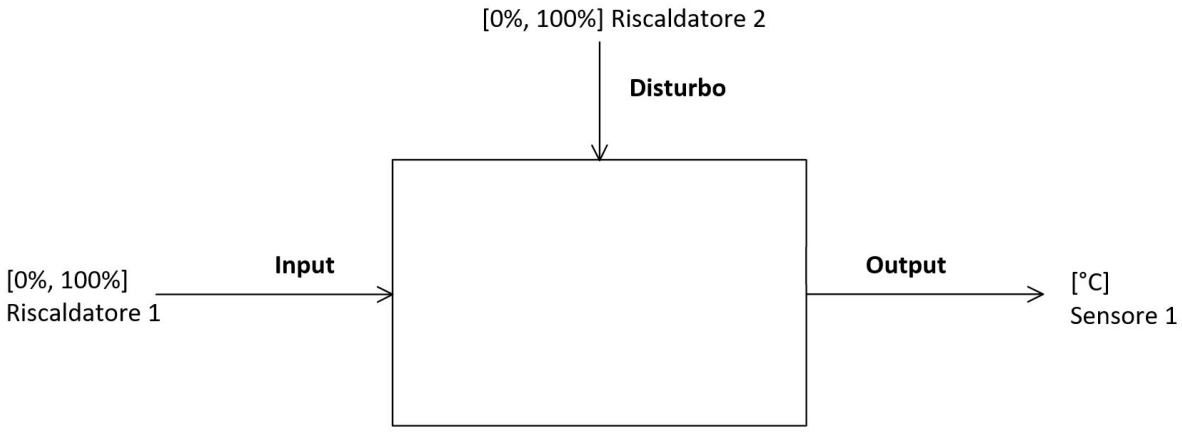

Figura 4.2: Schema a blocchi del TCLab con disturbo

4.1 Schema a blocchi

In questa sezione analizzerò nello specifico lo schema a blocchi dell'intero processo. Considererò prima lo schema a blocchi semplice composto dal processo $P(s)$, il disturbo $D(s)$ ed il controllore. In seguito considererò lo schema a blocchi completo composto anch'esso dal processo $P(s)$, il disturbo, $D(s)$, il controllore e, in aggiunta, il blocco di feedforward. Quest'ultimo permette di eliminare il disturbo anticipandolo, prima che riesca a raggiungere il processo.

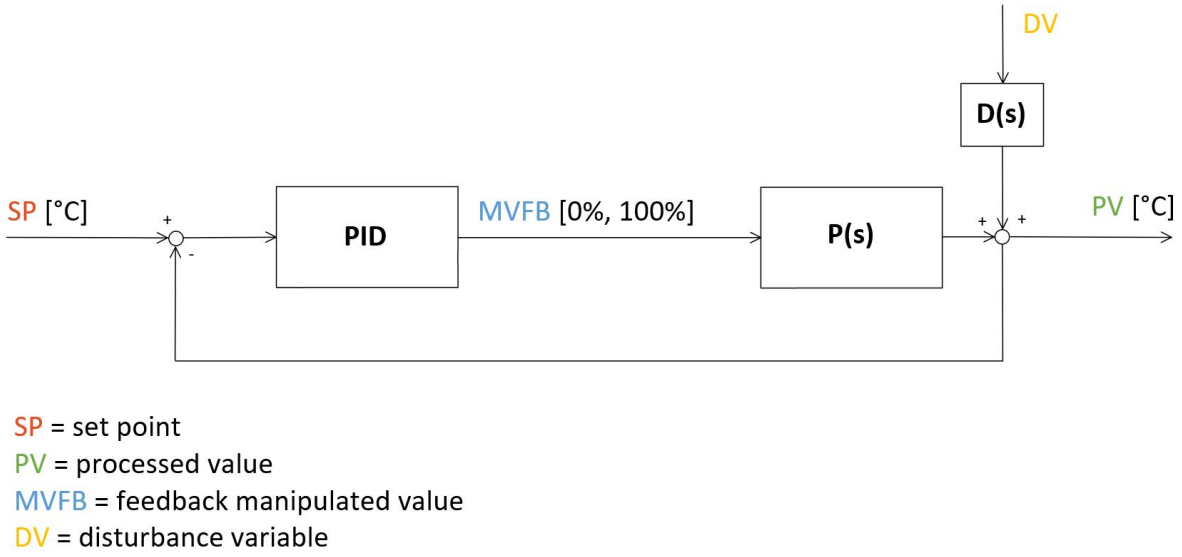

Figura 4.3: Schema a blocchi del sistema senza feedforward

In figura 4.3 è mostrato lo schema a blocchi dove il set point (SP) è la temperatura da raggiungere e mantenere. Il processed value (PV) è il valore rilevato dal sensore. Il manipulated

value (MV) è il valore processato dal PID che va ad interagire con il riscaldatore. L'ingresso del PID è la differenza tra il set point e il processed value, cioè l'errore (E) tra la temperatura rilevata e la temperatura da raggiungere. In questo modo, quando la temperatura desiderata sarà raggiunta, l'errore sarà 0. Il processed value viene modificato anche dalla disturbance variable (DV), che modificherà il valore del processed value. Ancora una volta, sarà compito del PID compensare il disturbo per mantenere costante la temperatura.

4.1.1 Feedforward

Lo scopo del blocco di feedforward è duplice:

1. Eliminare gli effetti dei disturbi prima che essi agiscano sul processo
2. Assicurare le prestazioni del sistema in presenza di disturbi, anticipandoli utilizzando le informazioni a monte

Per raggiungere questi scopi è necessario uno strumento che rilevi e misuri il disturbo con una discreta precisione e, soprattutto, prima che raggiunga il processo.
I vantaggi del feedforward sono molteplici:

- Le azioni correttive sono prese prima che il processo sia disturbato
- Non influenza la stabilità del sistema

Ma porta con sé anche alcuni svantaggi:

- Il disturbo deve essere misurato (costi aggiuntivi)
- Richiede un ottimo modello del sistema da controllare (non è sempre facile dato che il sistema potrebbe cambiare nel tempo)
- Non è robusto in caso di variazioni nel processo

Per questo motivo il feedforward non è quasi mai utilizzato da solo ma viene affiancato dal tradizionale feedback. Il feedback, infatti, assicura la stabilità del sistema, corregge i disturbi non misurabili ed è robusto di fronte a cambiamenti nel processo.

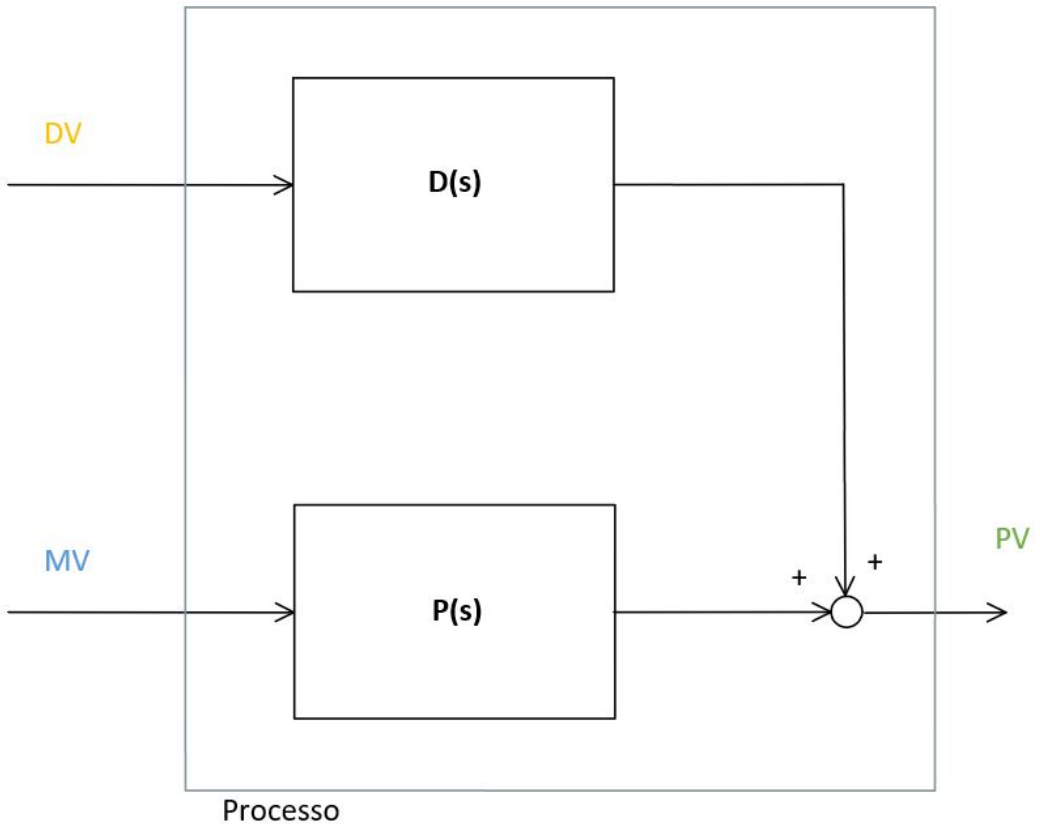

Figura 4.4: Schema a blocchi del processo

Per poter inserire il blocco di feedforward è necessario conoscere la dinamica del processo $P(s)$ e del disturbo $D(s)$. Questo viene fatto a priori, tramite un'analisi del comportamento del processo. Nel mio caso ho costruito i modelli osservando sperimentalmente la risposta a gradino del sistema.

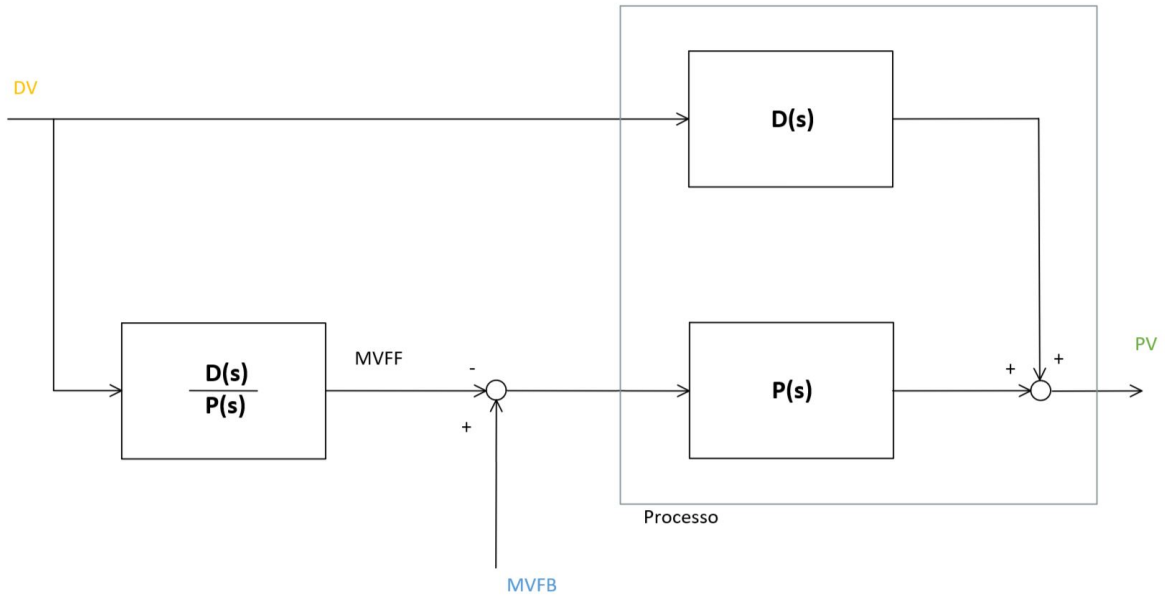

Figura 4.5: Schema a blocchi del sistema con feedforward

In figura 4.5 è mostrato lo schema a blocchi del sistema con feedforward. Il blocco si trova prima dell'ingresso sul processo $P(s)$ e si va a sommare con il valore determinato dal controllore.

Modellando entrambi i blocchi come sistemi del primo ordine con ritardo otteniamo:

$$PV_D = K_D \cdot \frac{1}{T_D s + 1} \cdot e^{-\theta_D s} \tag{4.1}$$

$$PV_P = K_P \cdot \frac{1}{T_P s + 1} \cdot e^{-\theta_P s} \tag{4.2}$$

A questo punto possiamo creare il blocco di feedforward come rapporto tra il processo $D(s)$ e $P(s)$.

$$MV_{FF} = K_{FF} \cdot \frac{T_P s + 1}{T_D s + 1} \cdot e^{-\theta_{FF} s} \tag{4.3}$$

$$K_{FF} = \frac{K_D}{K_P}$$

$$\theta_{FF} = max(\theta_D - \theta_P, 0)$$

Da notare che se il ritardo del processo $P(s)$ è maggiore del ritardo del processo $D(s)$ il feedforward non può operare in tempo. In questo caso la compensazione non è immediata e il disturbo avrà effetto sul processo. Ciò significa che non è possibile un controllo perfetto del sistema.

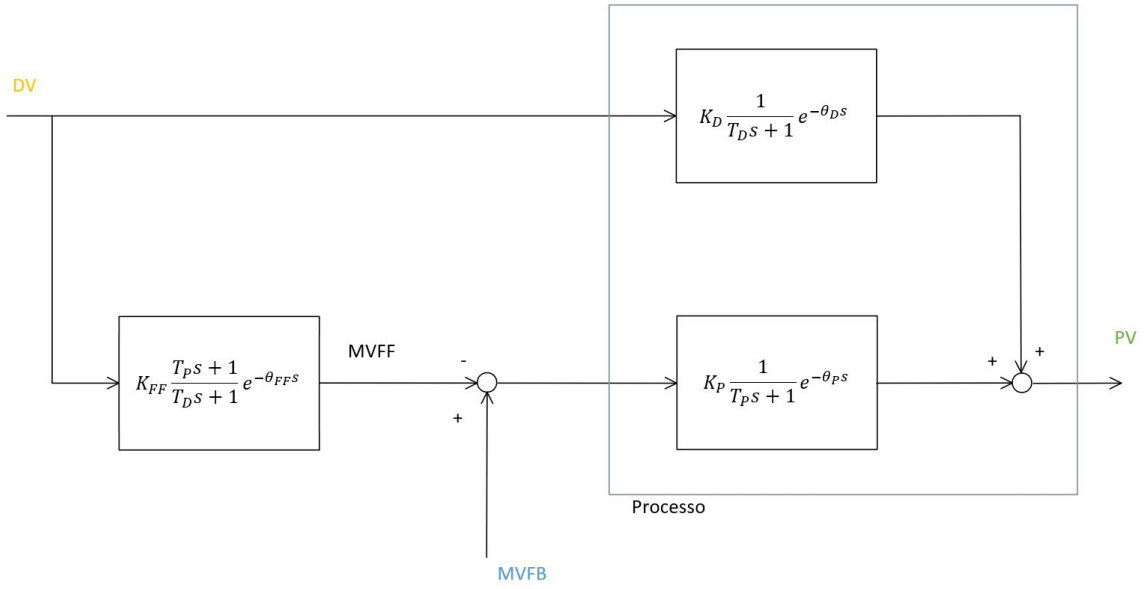

Figura 4.6: Schema a blocchi del sistema con feedforward

A questo punto il blocco di feedforward sarà in grado di rilevare il disturbo e correggerlo. Ignorando la componente feedback possiamo verificare che il feedforward elimini effettivamente il disturbo. Consideriamo il blocco di feedforward, il blocco $P(s)$ ed il blocco $D(s)$. Consideriamo il valore all'uscita $PV(s)$

$$-\frac{D(s)}{P(s)} \cdot P(s) + D(s) = PV(s) \tag{4.4}$$

Semplificando $P(s)$ otteniamo

$$-D(s) + D(s) = PV(s) \tag{4.5}$$

$$\Rightarrow PV(s) = 0 \tag{4.6}$$

L'equazione è verificata e il disturbo è compensato.

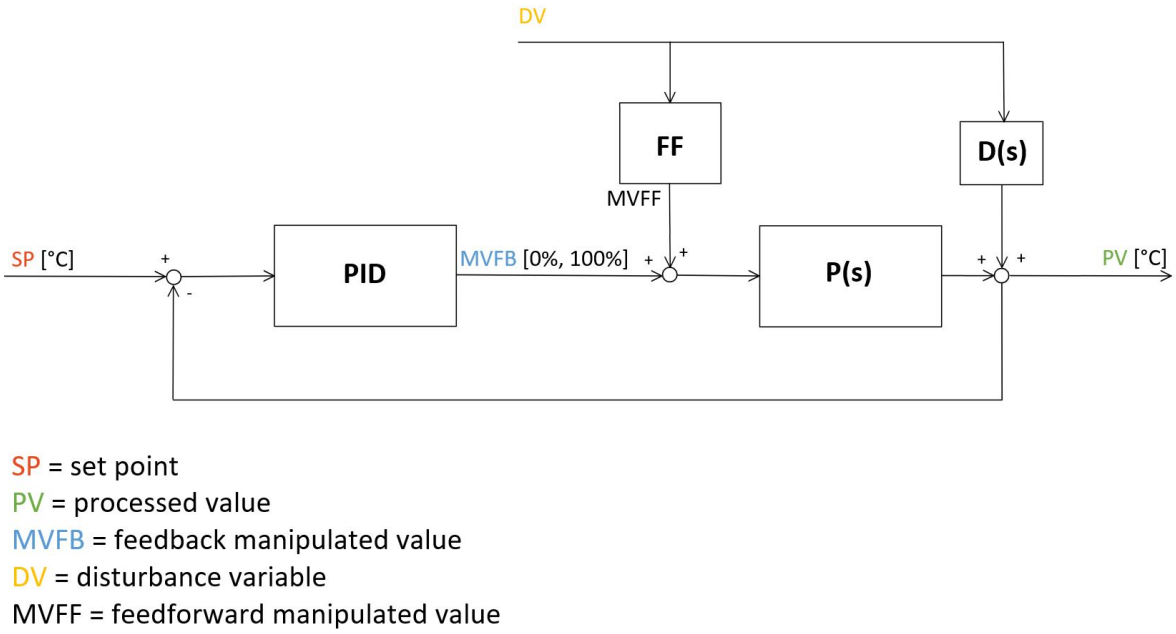

Figura 4.7: Schema a blocchi del sistema con feedforward

Lo schema in figura 4.7 è simile allo schema in figura 4.3, con la sola aggiunta del blocco di feedforward, che interviene sul processo sommato al valore del PID. Questo permette di misurare il cambiamento della disturbance variable e intervenire immediatamente sul processo, senza aspettare l'intervento del PID. Il feedforward è un ottimo meccanismo per correggere il disturbo in maniera diretta.

4.2 Modello del sistema

Il primo passo per conoscere un processo è conoscere il suo comportamento. Ci sono diversi aspetti dietro la conoscenza di un processo: la fisica di base, non linearità, disturbi e restrizioni. Questi aspetti influenzano sia la struttura che il tipo di controllo da applicare ovvero: rifiuto dei disturbi e tracciamento del set point. Nella mia tesi ho tenuto conto di tutti questi aspetti al di fuori di quello fisico, che per questo caso possono essere ignorati.

Ho identificato un modello del sistema con un approccio black box, ignorando le equazioni fisiche che descrivono il modello. Ho analizzato solamente la risposta del sistema ad un segnale d'ingresso a gradino e osservando la sua risposta.

Poiché il TCLab non è un sistema lineare è stato necessario lavorare intorno ad un punto di lavoro, attorno al quale il sistema può essere approssimato con un sistema lineare. Ho deciso di usare come punto di lavoro la potenza dei riscaldatori a 50% (sia per MV che per DV) che corrisponde ad un PV di circa $70C$. I limiti su MV (specifici per il TCLab) corrispondono alla

potenza dei riscaldatori, che varia da 0% a 100%. Il limite superiore del *PV* non è stato testato, per non rischiare di danneggiare il chip. Naturalmente il limite inferiore del *PV* è la temperatura dell'ambiente (intorno ai 23*C*).

4.2.1 Modellizzazione di *MV*

Per prima cosa ho creato un programma per l'identificazione del modello.
Questo programma porta il TCLab vicino al suo punto di lavoro e, una volta raggiunto, applica un segnale a gradino ad *MV*. Nel mio caso ho usato un delta di 20 intorno al punto di lavoro. Una volta che *MV* sarà stabilizzato intorno a 30% il segnale a gradino porterà *MV* a 70%.

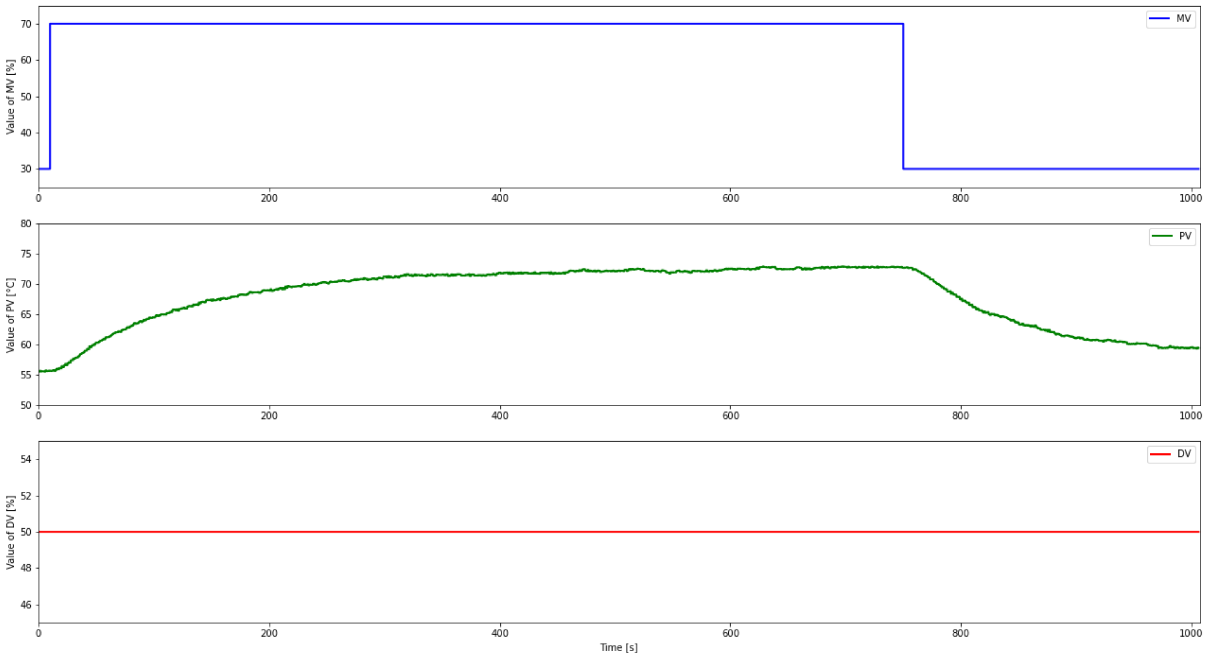

Figura 4.8: Risposta a gradino di *MV*

A partire dal grafico sperimentale in figura 4.8 è possibile ricavare i parametri per un sistema del secondo ordine con ritardo. I parametri sono calcolati automaticamente con una funzione che ottimizza la funzione di costo.

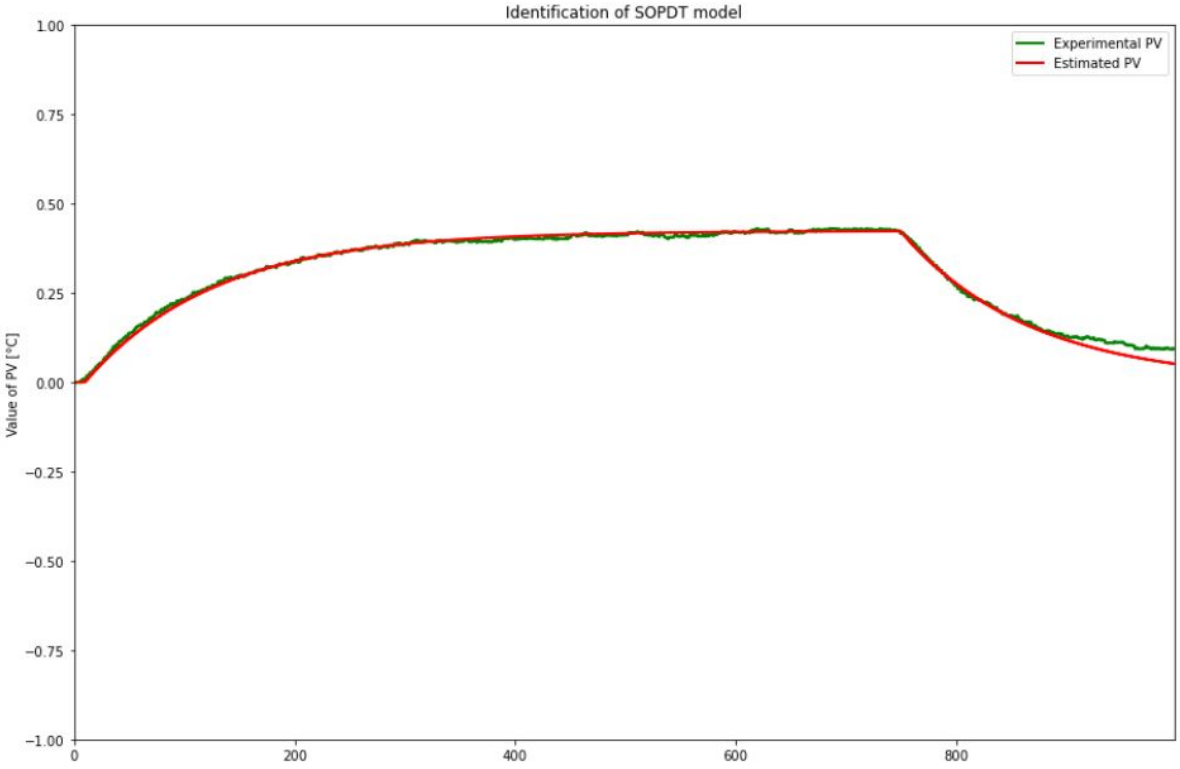

Figura 4.9: Identificazione di un modello del secondo ordine per *MV*

I parametri calcolati sono:

$$K_C = 0.42339070549814495$$

$$T_1 = 117.67680265705938$$

$$T_2 = 0.0002015037782807489$$

$$\theta = 8.299701290784041$$

4.2.2 Modellizzazione di *DV*

Il secondo passo è stato fare lo stesso procedimento per *DV*. Come nel primo caso il programma porta il dispositivo al punto di lavoro e poi applica un segnale a gradino. In questo caso *MV* rimane fisso a 50% mentre *DV* va da 30% a 70%. Il risultato è mostrato in figura 4.10.

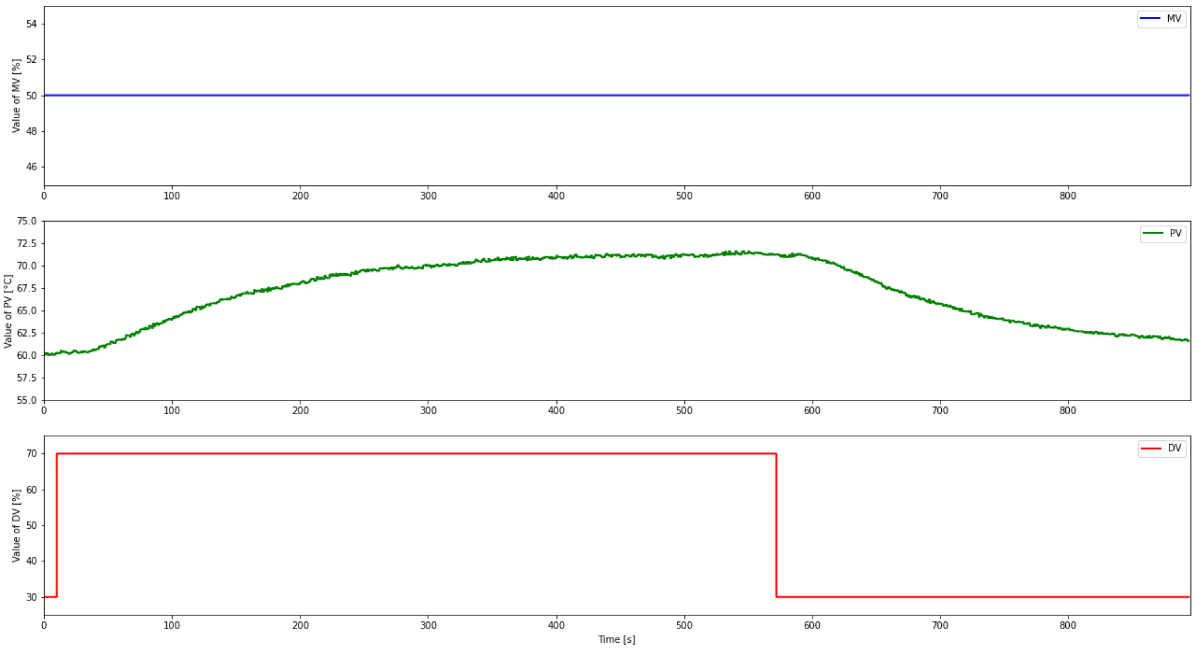

Figura 4.10: Risposta a gradino di *DV*

Anche per *DV* ho ricavato i parametri grazie al programma di ottimizzazione.

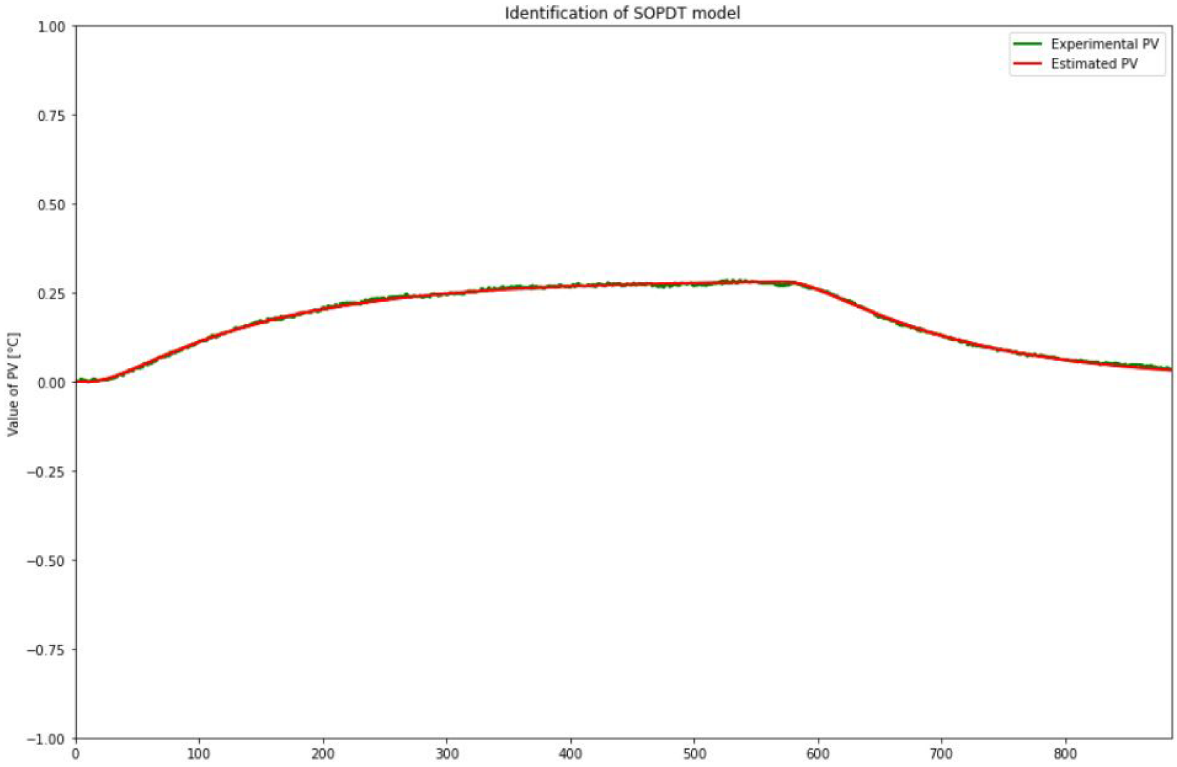

Figura 4.11: Identificazione di un modello del secondo ordine per *DV*

I parametri calcolati sono:

$$K_C = 0.28564594943666616$$

$$T_1 = 131.77153519132057$$

$$T_2 = 21.609531382173707$$

$$\theta = 10.342897290161249$$

Il prossimo step è stato quello di creare la funzione PID_RT(), già spiegata precedentemente. Per calcolare i parametri del PID ho creato una funzione IMC_tuning() che, basandosi sui parametri del processo, ritorna i valori ottimali per le tre azioni proporzionale, integrale e derivativa del PID.

5. Progettazione del PID digitale

Ora che è stato modellato il sistema può cominciare lo sviluppo di un controllore digitale. Ho utilizzato il PID in forma parallela partendo con le equazioni delle tre azioni proporzionale, integrale e derivativa. In seguito, tramite la trasformata z inversa, ho ottenuto le equazioni in tempo discreto, implementabili direttamente nel software del PID digitale.

5.1 PID

Per l'implementazione del PID utilizzerò la forma in parallelo

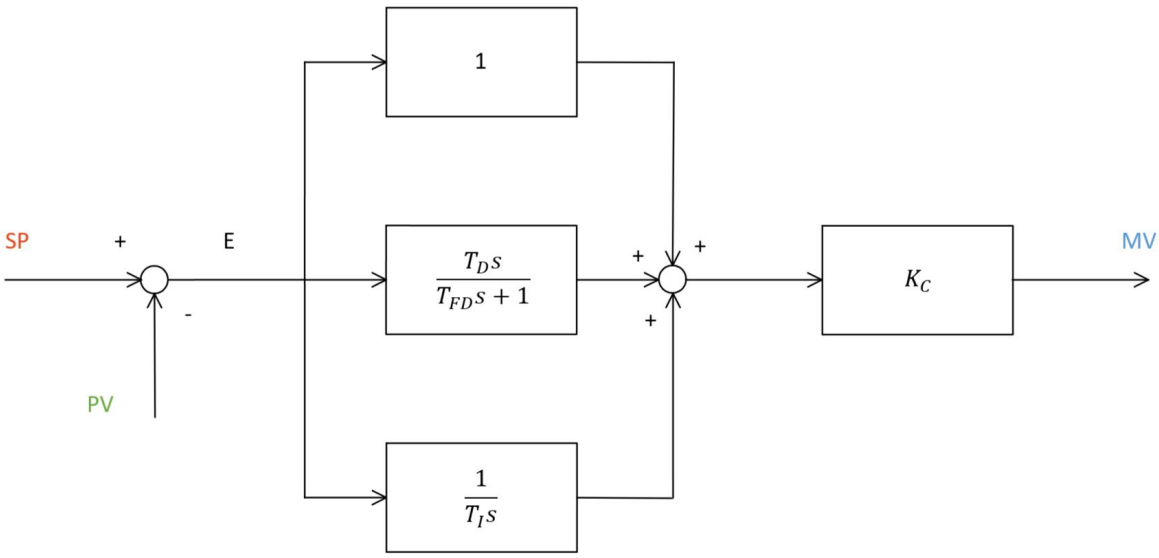

Figura 5.1: PID in forma parallela

La cui formula è

$$MV = K_C \left(1 + \frac{1}{T_I s} + \frac{T_D s}{T_{FD} s + 1}\right) \cdot E \tag{5.1}$$

Dove T_{FD} è un filtro del primo ordine per la componente derivativa con costante di tempo

$$T_{FD} = \alpha T_D \tag{5.2}$$

Questo filtro limita il guadagno ad alta frequenza a $1/\alpha$.

La versione a tempo discreto del PID si ottiene usando il metodo di Eulero all'indietro. Il metodo di Eulero in avanti è da evitare in quanto la stabilità del PID non può essere garantita. Utilizzo l'approssimazione del metodo di Eulero all'indietro

$$s = \frac{z-1}{zT_s} \tag{5.3}$$

Sostituendo

$$MV(z) = K_C \cdot \left(1 + \frac{\frac{T_s}{T_I}z}{z-1} + \frac{\frac{T_D}{T_{FD}+T_s}(z-1)}{z-\frac{T_{FD}}{T_{FD}+T_s}}\right) \cdot E(z) \tag{5.4}$$

L'equazione che descrive la componente proporzionale è

$$\frac{MV_P(z)}{K_C E(z)} = 1$$

Prendiamo la trasformata z inversa e otteniamo la componente integrale in tempo discreto, utilizzabile in un circuito digitale

$$MV_P[k] = K_C \cdot E[k] \tag{5.5}$$

L'equazione che descrive la componente integrale è

$$\frac{MV_I(z)}{K_C E(z)} = \frac{\frac{T_s}{T_I}z}{z-1} = \frac{\frac{T_s}{T_I}}{1-z^{-1}} \tag{5.6}$$

Prendiamo la trasformata z inversa e otteniamo la componente integrale in tempo discreto, utilizzabile in un circuito digitale

$$MV_I[k] = MV_I[k-1] + \frac{K_C T_s}{T_I} E[k] \tag{5.7}$$

L'equazione che descrive la componente derivativa è

$$\frac{MV_D(z)}{K_C E(z)} = \frac{\frac{T_D}{T_{FD}+T_s}(z-1)}{z-\frac{T_{FD}}{T_{FD}+T_s}} = \frac{\frac{T_D}{T_{FD}+T_s}(1-z^{-1})}{1-\frac{T_{FD}}{T_{FD}+T_s}z^{-1}} \tag{5.8}$$

Prendiamo la trasformata z inversa e otteniamo la componente derivativa in tempo discreto

$$MV_D[k] = \left(\frac{T_{FD}}{T_{FD}+T_s}\right) MV_D[k-1] + \left(\frac{K_C T_D}{T_{FD}+T_s}\right)(E[k]-E[k-1]) \tag{5.9}$$

A questo punto con la (5.5), (5.7) e la (5.9) abbiamo tutte le componenti e possiamo integrarle nel PID digitale

$$MV_P[k] = K_C E[k]$$

$$MV_I[k] = MV_I[k-1] + \frac{K_C T_s}{T_I} E[k]$$

$$MV_D[k] = \left(\frac{T_{FD}}{T_{FD}+T_s}\right) MV_D[k-1] + \left(\frac{K_C T_D}{T_{FD}+T_s}\right)(E[k]-E[k-1])$$

Queste sono le formule necessarie per creare il PID a tempo discreto. Ad ogni istante k il valore del manipulated value è la somma dei tre contributi.

Ci sono però ancora due aspetti da considerare:

1. L'anti wind-up dell'integratore
2. La saturazione del MV

5.2 Meccanismo di anti wind-up

Il wind up dell'integratore si verifica quando il PID è in modalità manuale. Cioè quando il valore di MV è deciso dall'utente e il PID non è in funzione. In questo caso l'errore continua ad essere integrato dal PID. Quando il PID viene riacceso il valore della componente integrale è così alto che serve molto tempo prima che torni a zero. Per questo motivo è necessario creare un meccanismo di anti wind-up.

Se il PID è in modalità manuale, il valore di MV è deciso dall'utente (MAN). In pseudocodice diventa:

$$if \quad MAN \quad then \quad MV[k] = MAN$$

Il valore di MV_I dell'istante k è uguale al valore dell'istante precedente. In questo modo non c'è wind-up

$$MV_I[k] = MV_I[k-1]$$

5.3 Saturazione

Se il valore di MV sale sopra il 100% significa che è in saturazione: è stato superato il limite fisico di MV (questo per il caso specifico del TCLab). In questo caso è sufficiente verificare con delle condizioni il valore di MV. Se esso è maggiore di 100% o minore di 0% il valore viene fissato al valore di saturazione. Lo pseudocodice è il seguente:

$$if \quad MV[k] > MV_{MAX} \quad then \quad MV[k] = MV_{MAX}$$

$$if \quad MV[k] < MV_{MIN} \quad then \quad MV[k] = MV_{MIN}$$

Nel caso in cui il feedforward sia attivato il suo valore sarà aggiunto al valore di MV. Il valore di MV sarà la somma delle tre azioni proporzionale, integrale, derivativa e della componente di feedforward.

$$MV[k] = MF_P[k] + MV_I[k] + MV_D[k] + MV_{FF}[k]$$

5.4 Implementazione software

Ora che abbiamo tutte le formule necessarie possiamo creare il nostro controllore. Il PID digitale si concretizza in una funzione che prende come parametri una serie di variabili

```
def PID_RT(SP, PV, Man, MVMan, MVFF, Kc, Ti, Td, alpha, Ts, MVMin, MVMax
    , MV, MVP, MVI, MVD, E, ManFF=False, PVInit=0, methods='EBD-EBD'):
```

Dove SP, PV, Man, $MVMan$, $MVFF$, MV, MVP, MVI, MVI ed E sono dei vettori. Il funzionamento dell'intero programma di simulazione e sperimentazione sul chip sono costruiti intorno a questi vettori. Per ogni istante k la funzione aggiunge un valore al vettore. Alla fine del programma i vettori avranno lunghezza pari al quoziente del tempo di funzionamento (in secondi) per il periodo di campionamento (in secondi). Ogni vettore conterrà una cronologia del valore, dove ad ogni istante viene salvato il valore nel vettore.
I vettori sono rispettivamente:

- SP = vettore del set point
- PV = vettore del processed value
- Man = vettore del PID in modalità manuale. Il vettore contiene solo valori true e false. Se il PID è sempre acceso il vettore conterrà solo il valore false
- $MVMan$ = vettore dei valori di MV imposto dall'utente. Quando il PID è in modalità manuale, il MV seguirà questo vettore
- $MVFF$ = vettore del blocco di feedforward
- MV = vettore del manipulated value
- MVP = vettore della componente proporzionale
- MVI = vettore della componente integrale
- MVD = vettore della componente derivativa

- E = vettore dell'errore (SP–PV)

Kc, Ti e Td sono i parametri del PID, calcolati con un'opportuna funzione di tuning.
$Alpha$ è un valore tra 0 e 1, relativo al filtro per la componente derivativa.
$MVMin$ e $MVMax$ sono i valori massimi di saturazione del PID (nel caso del TCLab 0 e 100).
$PVInit$ è il valore di partenza del processed value (in °C).
$Method$ è una stringa che determina il metodo utilizzato per la componente integrale (MVI) e quella derivativa (MVD). I valori possono essere:

1. EBD = metodo di Eulero all'indietro
2. $TRAP$ = metodo del trapezio

5.4.1 La funzione PID_RT()

Per prima cosa definisco la variabile Tfd come prodotto tra $alpha$ e Td

```
Tfd = alpha * Td
```

Il vettore dell'errore E viene costruito prendendo la differenza dell'ultimo valore nel vettore di SP e del vettore di PV

```
#initialization of E
if (len(PV) == 0):
E.append(SP[-1] - PVInit)
else:
E.append(SP[-1] - PV[-1])
```

Il valore della componente proporzionale viene calcolato con la formula (5.5)

```
#initialization of MVP
MVP.append(Kc * E[-1])
```

Il valore della componente integrativa viene calcolato con la formula (5.7)

```
#initialization of MVI
if (len(MVI) == 0):
MVI.append((Kc * Ts / Ti) * E[-1])
else:
if (methodI == 'TRAP'):
MVI.append(MVI[-1] + (0.5 * Kc * Ts / Ti) * (E[-1] + E[-2]))
else:
MVI.append(MVI[-1] + (Kc * Ts / Ti) * E[-1])
```

Il valore della componente derivativa viene calcolato con la formula (5.9)

```
#initialization of MVD
if (len(MVD) == 0):
MVD.append((Kc * Td) / (Tfd + Ts) * (E[-1]))
else:
if (methodD == 'TRAP'):
MVD.append((((Tfd - Ts * 0.5) / (Tfd + Ts * 0.5)) * MVD[-1]) + ((Kc * Td
    / (Tfd + Ts * 0.5)) * (E[-1] - E[-2])))
else:
MVD.append((((Tfd) / (Tfd + Ts)) * MVD[-1]) + (((Kc * Td) / (Tfd + Ts))
   * (E[-1] - E[-2])))
```

Il valore di *MV* viene calcolato sommando le tre componenti del PID. La funzione tiene conto di saturazione di *MV*, presenza o meno del blocco di feedforward e del wind-up della componente integrale

```
# calculate MV, saturation and anti wind-up
if (Man[-1] == True):
if (len(MVI) == 1):
MVI[-1] = 0
else:
MVI[-1] = MVI[-2]
if (ManFF):
if (MVMan[-1] + MVFF[-1] > MVMax):
MV.append(MVMax)
else:
MV.append(MVMan[-1] + MVFF[-1])
else:
MV.append(MVMan[-1])
elif (MVP[-1] + MVI[-1] + MVD[-1] > MVMax):
MV.append(MVMax)
elif (MVP[-1] + MVI[-1] + MVD[-1] < MVMin):
MV.append(MVMin)
elif (MVP[-1] + MVI[-1] + MVD[-1] + MVFF[-1] > MVMax):
MV.append(MVMax)
elif (MVP[-1] + MVI[-1] + MVD[-1] + MVFF[-1]< MVMin):
MV.append(MVMin)
else:
if (ManFF):
MV.append(MVP[-1] + MVI[-1] + MVD[-1] + MVFF[-1])
else:
MV.append(MVP[-1] + MVI[-1] + MVD[-1])
```

La funzione va inserita all'interno di un ciclo che iteri per tutta la durata della simulazione/sperimentazione.

5.4.2 Test della funzione PID_RT()

Per testare la funzione inizializzo tutti i parametri con dei valori arbitrari

```
SP = []
PV = []
Man = [1]
MVMan = []
MVFF = [1]
Man[0] = False
MVFF[0] = 0

Kc = 0.4
Ti = 3
Td = 5
alpha = 0.1
Ts = 0.01

TSim=50
t = []
DV = []

MVMin = 0
MVMax = 100

MV = []
MVP = []
MVI = []
MVD = []
E = []

method = ''

SPPath = {0: 0, 5:1}
DVPath = {0: 0, 50: 0}
```

Il tempo di simulazione $TSim$ è di 50 secondi. Il periodo di campionamento Ts è di 0.01. Questo significa che i vettori finali avranno una dimensione di 5001.

Il vettore t è il vettore del tempo. Ogni elemento di t contiene un istante di campionamento. In questo caso t conterrà i valori 0, 0.01, 0.02, 0.03... fino ad arrivare a 50. Il vettore t è necessario al fine di disegnare i grafici.

Infine, $SPPath$ e $DVPath$ sono dei dizionari che indicano il percorso che SP e DV devono seguire durante la simulazione. In questo caso DV rimane sempre a zero. SP parte a zero e prende il valore 1 dopo 5 secondi.

```
for i in range(0,int(TSim/Ts)+1):
t.append(i*Ts)
SelectPath_RT(SPPath,t,SP)
SelectPath_RT(DVPath,t,DV)
PID_RT(SP, PV, Man, MVMan, MVFF, Kc, Ti, Td, alpha, Ts, MVMin, MVMax, MV
    , MVP, MVI, MVD, E, ManFF=False, PVInit=0, method='EBD-EBD')
```

A questo punto possiamo creare il ciclo su cui verrà chiamata la funzione. Il for partirà da 0 e arriverà fino a 5001, aggiungendo man mano ogni valore nel vettore corrispondente. Una volta finito il ciclo possiamo utilizzare i vettori costruiti per creare il grafico

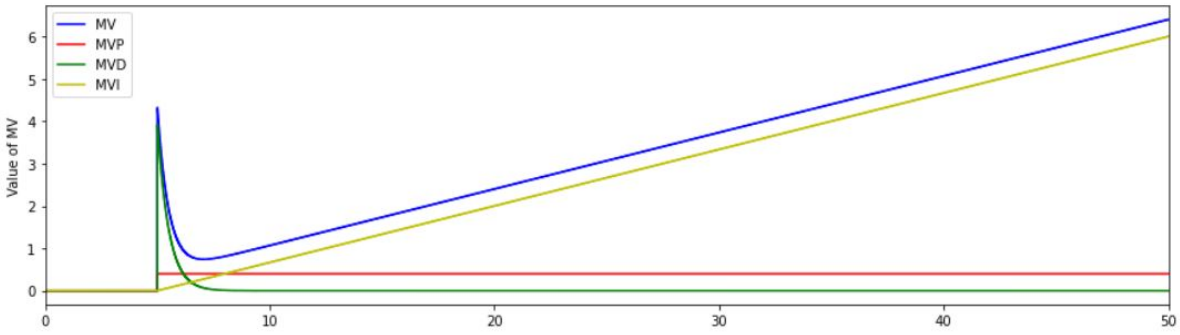

Figura 5.2: Risposta a gradino della funzione PID_RT

Nel grafico in figura 5.2 possiamo vedere le tre componenti in azione su MV. La componente proporzionale (in rosso) rimane costante. La componente derivativa (in verde) ha un picco all'inizio del gradino per poi riassestarsi sullo zero. La componente integrale (in giallo) integra l'errore costantemente. L'errore rimane costante per tutta la durata della simulazione quindi la componente integrale cresce linearmente.

5.4.3 Test del meccanismo di anti wind-up

Il meccanismo di anti wind-up va testato quando il controllore è in modalità manuale

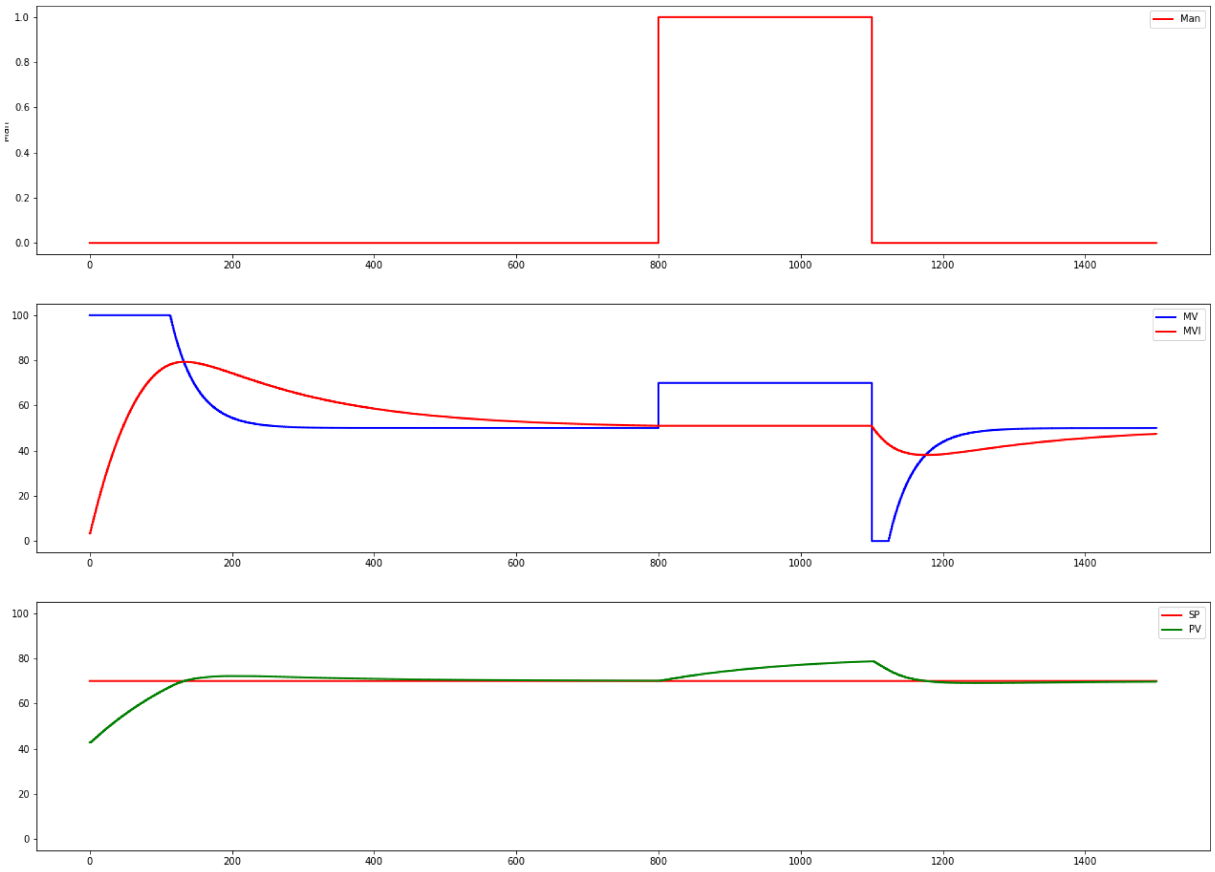

Figura 5.3: Wind up correttamente implementato

Il grafico in figura 5.3 mostra che da 800 a 1100 il PID è in modalità manuale. Dal secondo grafico si vede l'azione integrale e il MV totale. Quando il PID è in modalità manuale il valore di MV è fisso a 70%. Mentre il PID è in modalità manuale la componente integrale rimane costante, come previsto. In questo modo quando il PID torna in modalità automatica la componente integrale non influisce più di quello che è necessario

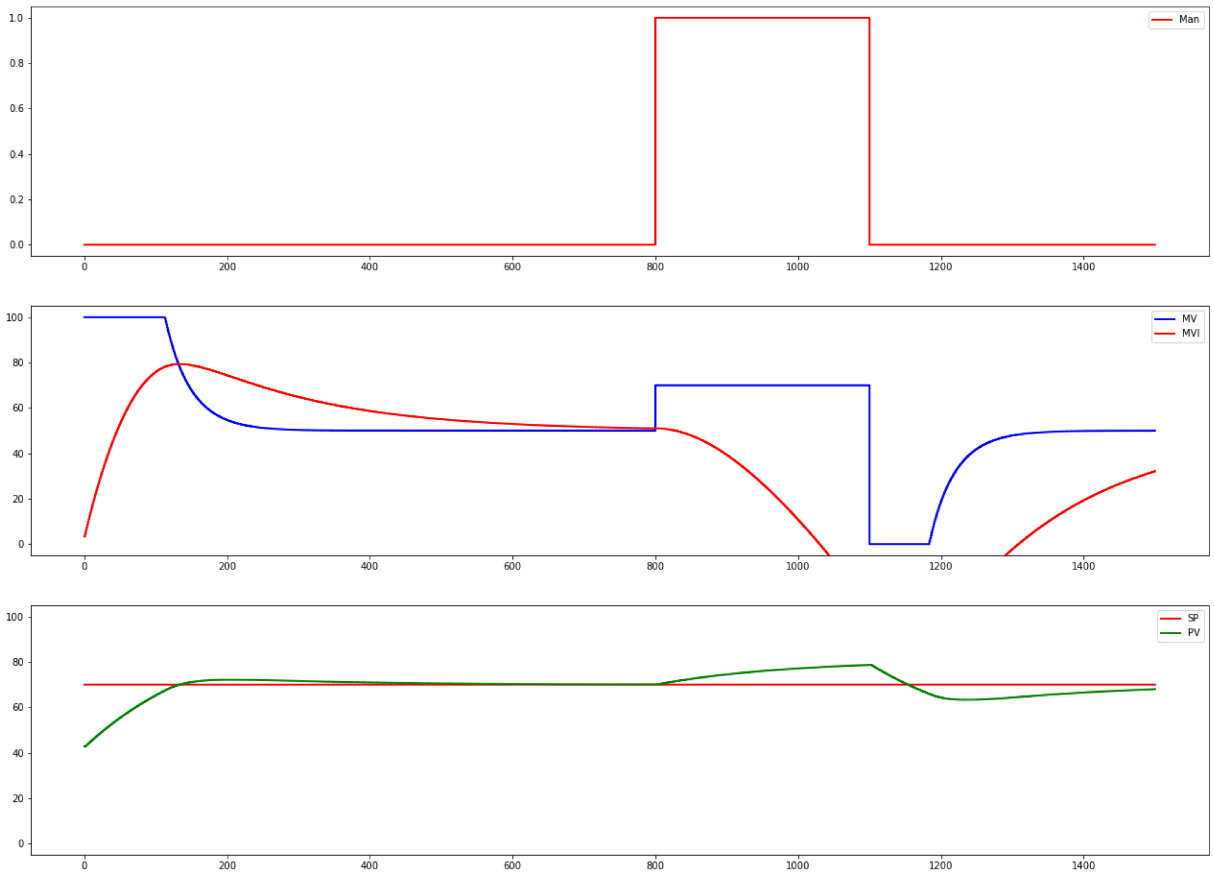

Figura 5.4: Wind up non implementato

In questa simulazione ho volontariamente evitato di implementare il meccanismo di wind up. Come è possibile vedere nel grafico di figura 5.4, quando il PID torna in modalità automatica (dopo il secondo 1100), il valore di MVI è estremamente elevato in valore assoluto. Questo perché l'errore è stato integrato nonostante il PID fosse in modalità manuale. Di conseguenza ci è voluto più tempo per il de-winding e per tornare ad un controllo ottimale.

5.4.4 Test meccanismo di saturazione

Per ottenere il controllo perfetto MV dovrebbe poter raggiungere valori arbitrariamente grandi. Per ragioni fisiche questo non è possibile. Nel mio caso MV può andare solo da 0% a 100%. Il grafico in figura 5.5 mostra che quando MV dovrebbe avere un valore superiore a 100% entra in azione il meccanismo di saturazione, che fissa MV al massimo valore possibile cioè 100%.

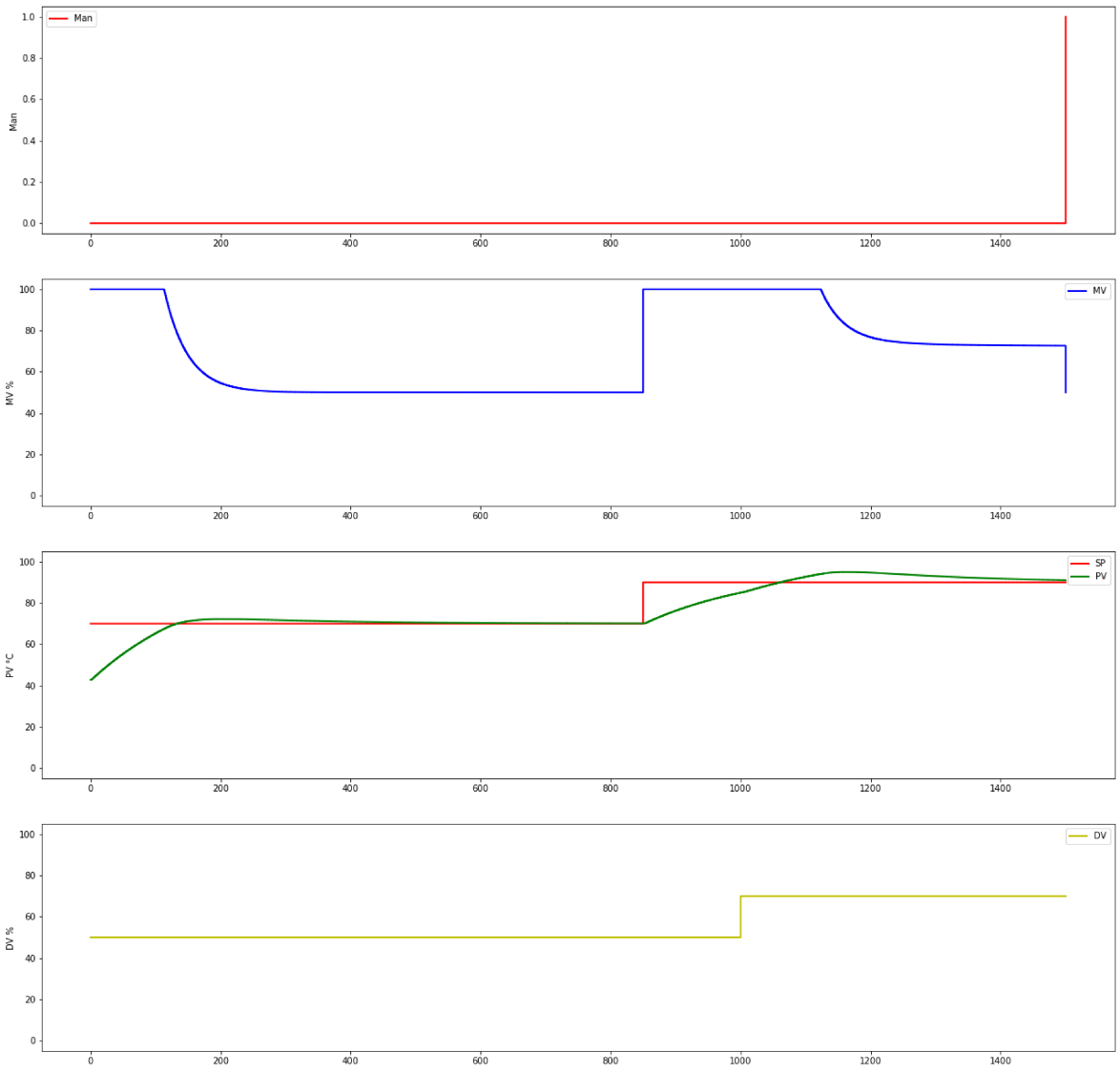

Figura 5.5: saturazione

6. Simulazioni e risultati sperimentali

Ora che la funzione di PID è pronta, insieme all'IMC tuning, i parametri dei processi *MV* e *DV* sono stati trovati, ho iniziato le simulazioni/sperimentazioni. Ho deciso di analizzare 6 casi:

1. Risposta ad anello chiuso di un disturbo esterno
2. Risposta ad anello chiuso ad un cambio nel set point (PID attivo)
3. Risposta ad un cambio in DV: FF disattivato e PID disattivato
4. Risposta ad un cambio in DV: FF attivato e PID disattivato
5. Risposta ad un cambio in DV: FF disattivato e PID attivato
6. Risposta ad un cambio in DV: FF attivato e PID attivato

Ad ogni simulazione corrisponde un risultato sperimentale. Questo permette di osservare direttamente il comportamento del sistema simulato rispetto al sistema reale e di confrontarli.

6.1 Risposta ad anello chiuso di un disturbo esterno

```
# Closed loop response to a SP change
ManPath = {0: False, TSim: False}
MVManPath = {0: MV0, TSim: MV0}
SPPath = {0: PV0, TSim: PV0}
# SPPath = {0: PV0, 850: PV0 + 10, 1000: PV0 - 10, TSim: PV0 - 10}
DVPath = {0: DV0, 1000: DV0, TSim: DV0}
FF = True
ManFF = True
```

Ho deciso di verificare per prima cosa la risposta del PID di fronte ad un disturbo esterno. In questo caso il *SP* rimane costante. Durante la sperimentazione ho soffiato sul dispositivo al fine di abbassarne la temperatura ed osservare il comportamento. Nel grafico in figura 6.1 è

possibile vedere come al secondo 600 e 1100 la temperatura rilevata dal sensore (in verde) si abbassi improvvisamente, discostandosi da *SP* (in rosso). In entrambi i casi il PID è entrato in azione riportando la temperatura a ridosso del *SP*. La sovra elongazione del PID non supera i $10°C$ ed è accettabile al fine di ottenere un tempo di risposta maggiore. Questa sperimentazione mi ha permesso di capire che il PID è stato correttamente implementato.
In questo esempio ho inoltre attivato il meccanismo di feedforward. Quest'ultimo, tuttavia, non è mai entrato in azione, dato che il disturbo è esterno e non proveniente da *DV*.

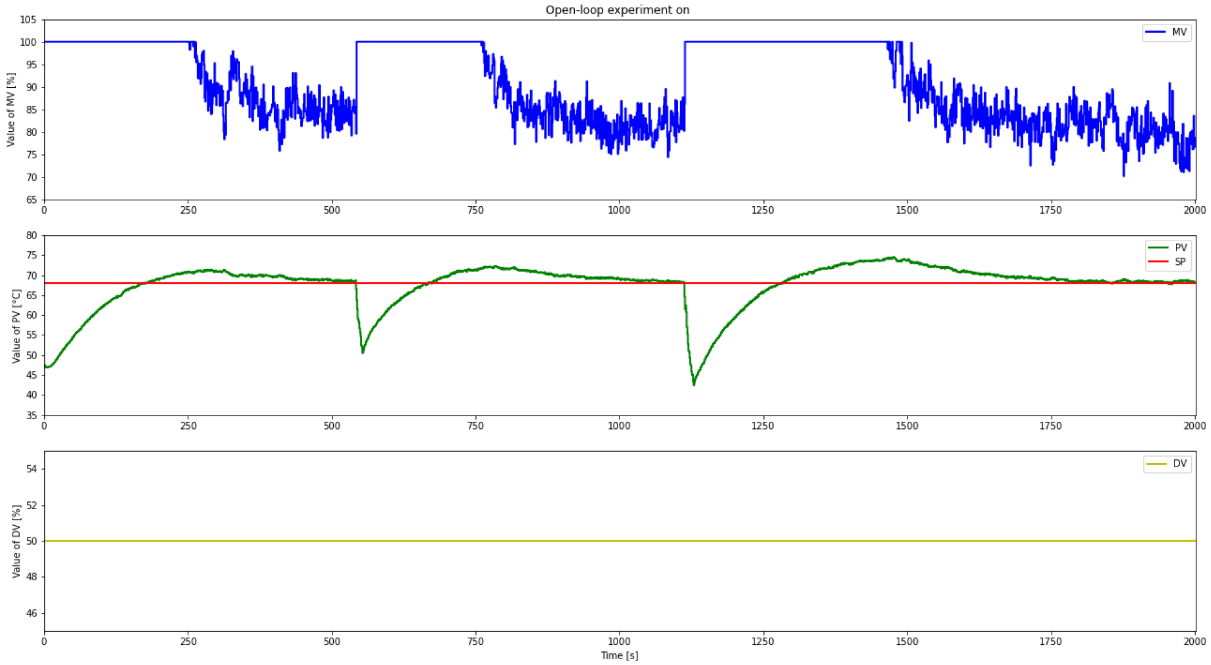

Figura 6.1: Risposta ad anello chiuso di un disturbo esterno

6.2 Risposta ad anello chiuso ad un cambio nel set point

```
# Closed loop response to a SP change
ManPath = {0: False, TSim: False}
MVManPath = {0: MV0, TSim: MV0}
# SPPath = {0: PV0, 850: PV0 + 10, TSim: PV0 + 10}
SPPath = {0: PV0, 850: PV0 + 10, 1000: PV0 - 10, TSim: PV0 - 10}
DVPath = {0: DV0, 1000: DV0, TSim: DV0}
FF = False
ManFF = False
```

In questo caso ho usato solo *SPPath*, essendo l'unico parametro che cambia. In questi due casi il solo compito del PID è di far corrispondere il *PV* (in verde) con il *SP* (in rosso).

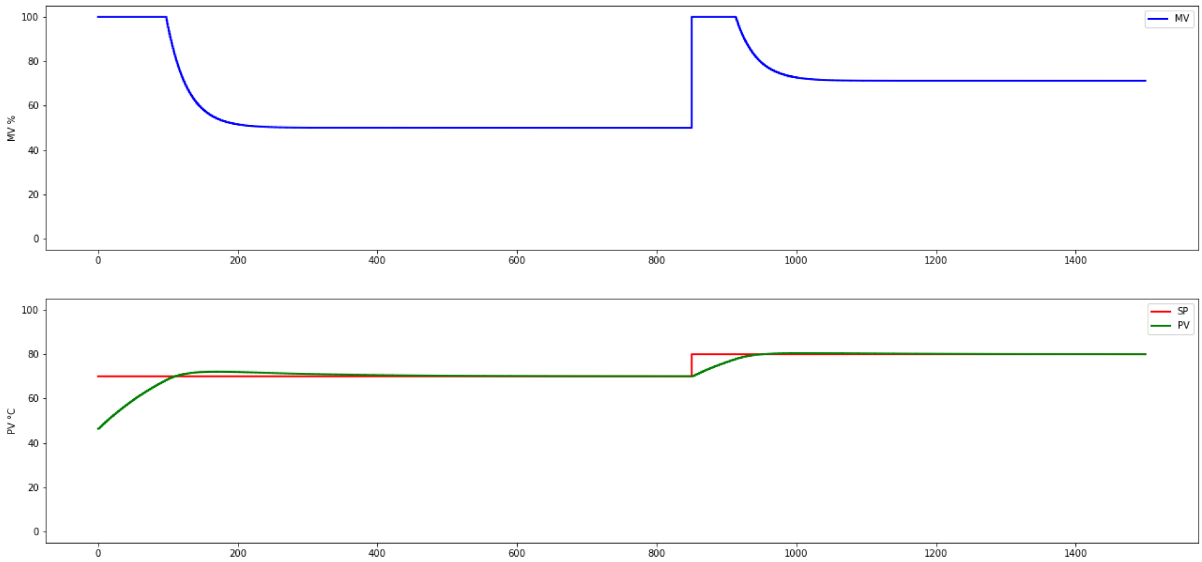

Figura 6.2: Simulazione della risposta ad anello chiuso ad un cambio nel set point

Nel grafico in figura 6.2 il *SP* rimane costante fino al secondo 850. In quell'istante applica un gradino di $10°C$. Il PID risponde correttamente e si assesta in tempo relativamente breve alla temperatura desiderata, con sovra elongazione accettabile. In blu è visibile il valore di *MV* calcolato dal PID.

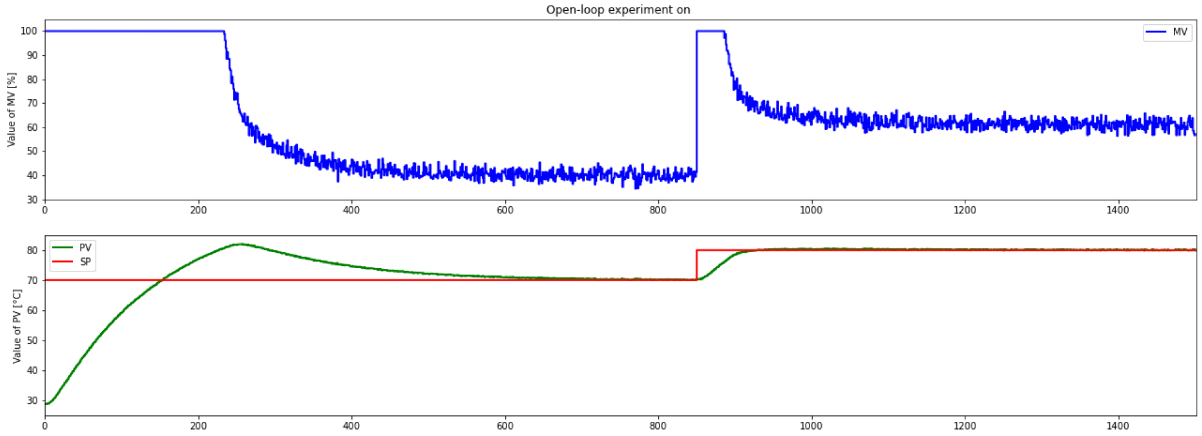

Figura 6.3: Applicazione della risposta ad anello chiuso ad un cambio nel set point

L'applicazione sul TCLab corrisponde alle aspettative precedentemente osservate nella simulazione. Il grafico della sperimentazione è mostrato in figura 6.3

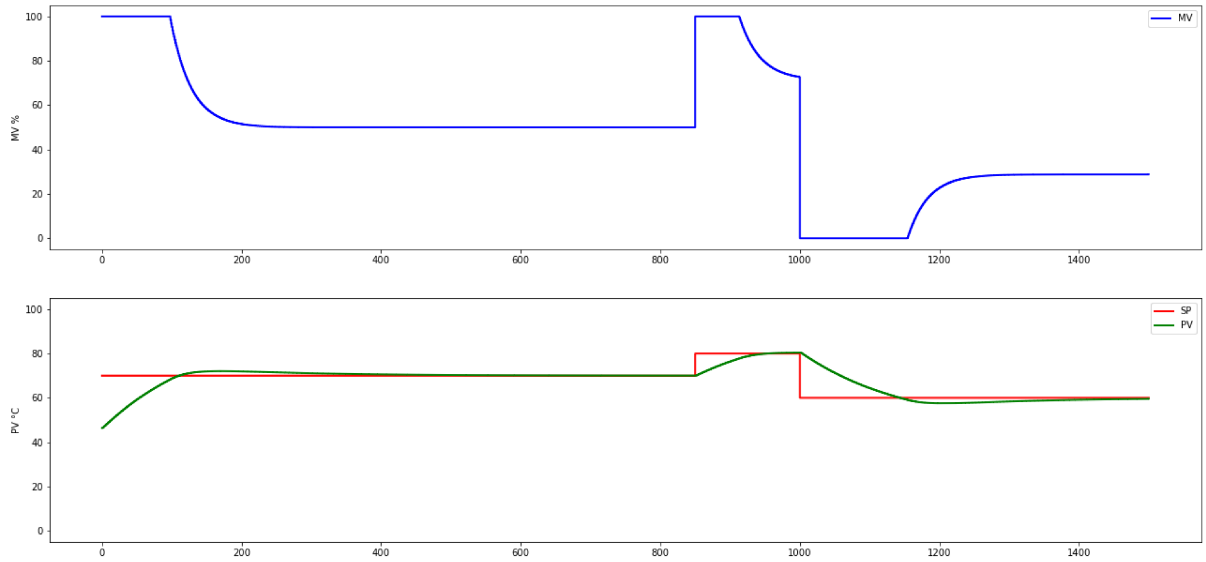

Figura 6.4: Simulazione della risposta ad anello chiuso ad un cambio nel set point

Anche in questo secondo esempio ho applicato un segnale a gradino di $+10°C$ al secondo 850. Al secondo 1000 ho applicato un secondo gradino, questa volta di un delta di $-20°C$. Anche in questo caso il PID risponde come previsto. Il tempo di assestamento è accettabile.

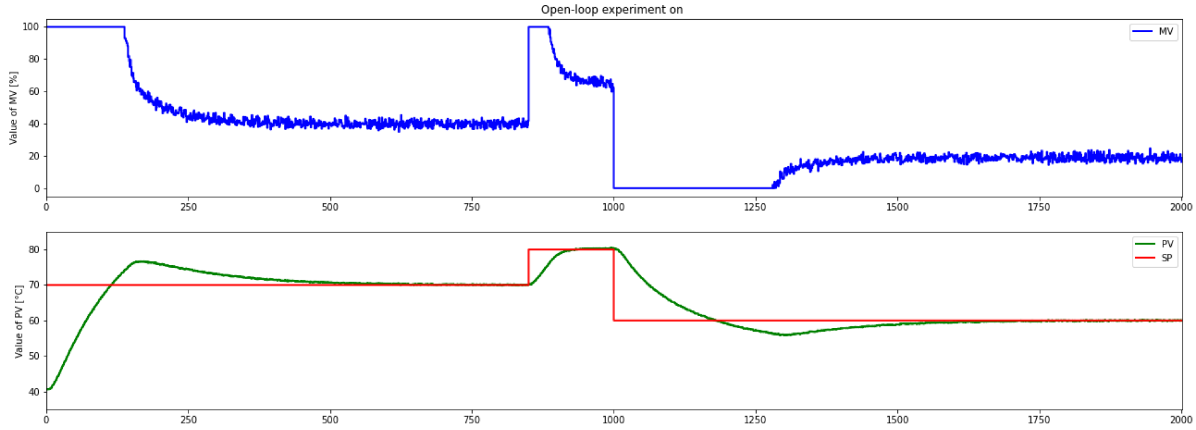

Figura 6.5: Applicazione della risposta ad anello chiuso ad un cambio nel set point

Il grafico in figura 6.5 mostra la sperimentazione precedentemente simulata. Anche questa volta il grafico non si discosta molto dalla simulazione. La sovra elongazione è maggiore rispetto alla simulazione. Questo significa che il modello non è esatto dato che è solo un'approssimazione. Nel contesto attuale, però, il modello ha un errore accettabile.

Un'altra osservazione riguarda il valore di MV (in blu). Durante le sperimentazioni sul chip il periodo di campionamento è di 1 secondo. Questo significa che ad ogni secondo viene calcolato un nuovo valore di MV. Per questo motivo nei grafici in figura 6.3 e 6.5 il valore di MV è molto più instabile e non è una linea continua.

6.3 Risposta ad un cambio di DV: FF disattivato e PID disattivato

```
# Response to DV : No FF and controller in manual mode
ManPath = {0: True, TSim: True}
MVManPath = {0: MV0, TSim: MV0}
SPPath = {0: PV0, TSim: PV0}
# DVPath = {0: DV0, 1000: DV0 + 20, TSim: DV0 + 20}
DVPath = {0: DV0, 1000: DV0 + 20, 2000: DV0, TSim: DV0}
FF = False
ManFF = False
```

In questo caso i parametri sono impostati nel modo seguente:

- *ManPath*: indica se il PID è in modalità manuale o meno. In questo caso il PID è sempre disattivato quindi l'intero vettore è *true*
- *MVManPath*: indica il valore di *MV* (dato che il PID è disattivato). In questo caso il valore è impostato a 50%
- *DVPath*: indica il percorso di *DV*
- *FF*: indica se il feedforward è attivato (no)
- *ManFF*: indica se il feedforward è attivato in modalità manuale (no)

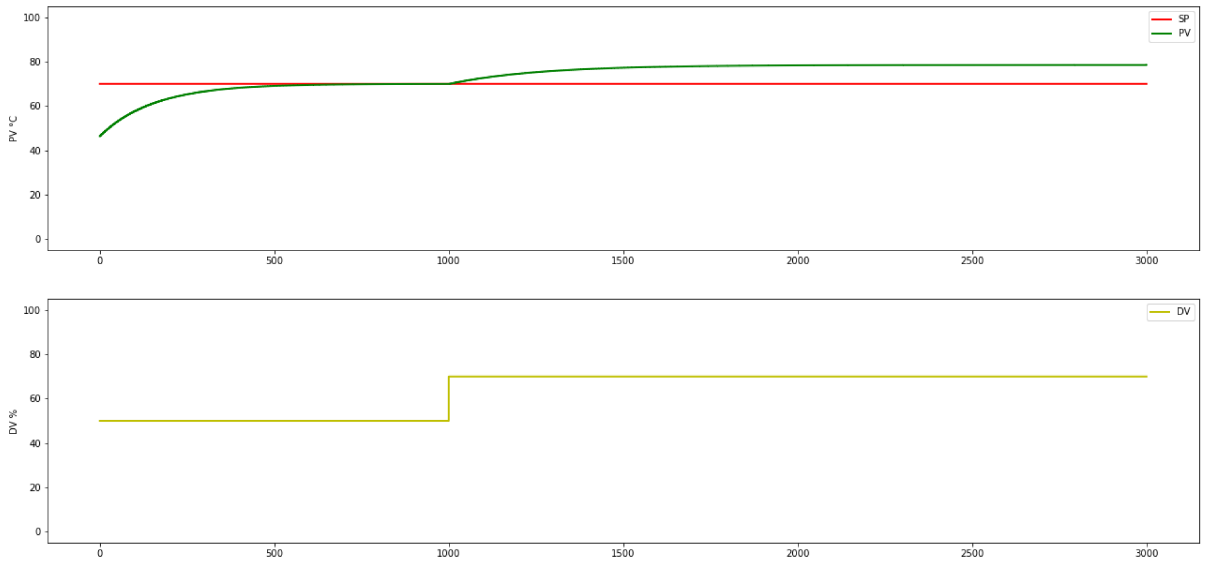

Figura 6.6: Simulazione della risposta ad anello aperto ad un cambio di *DV*

Dal grafico in figura 6.6 si vede che quando *DV* (in giallo) cambia esso influenza il *PV*. Questo perché, ricordando l'architettura del TCLab, *DV* controlla il secondo riscaldatore. Quando viene azionato, infatti, influenza la temperatura finale.

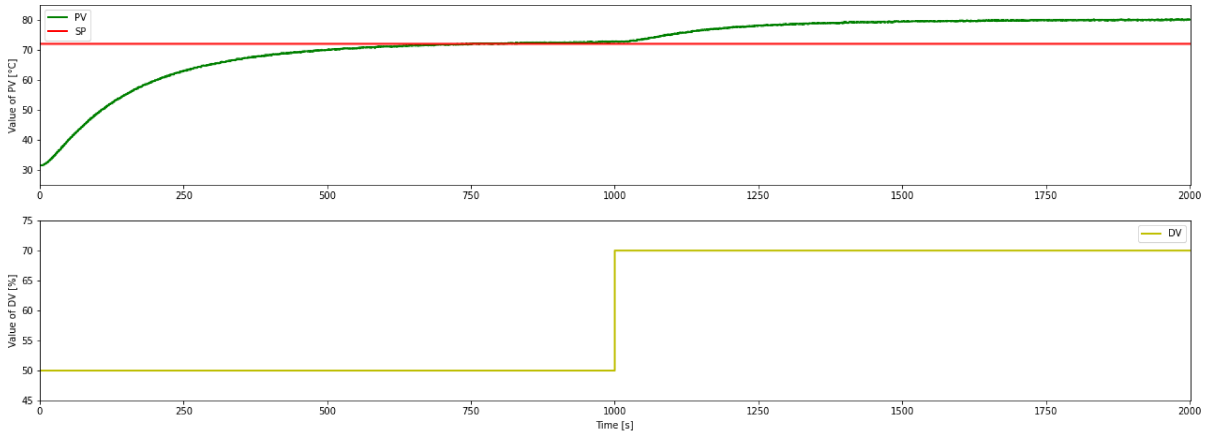

Figura 6.7: Applicazione della risposta ad anello aperto ad un cambio di *DV*

Il grafico in figura 6.7 dell'applicazione reale mostra un andamento identico a quello simulato in figura 6.6

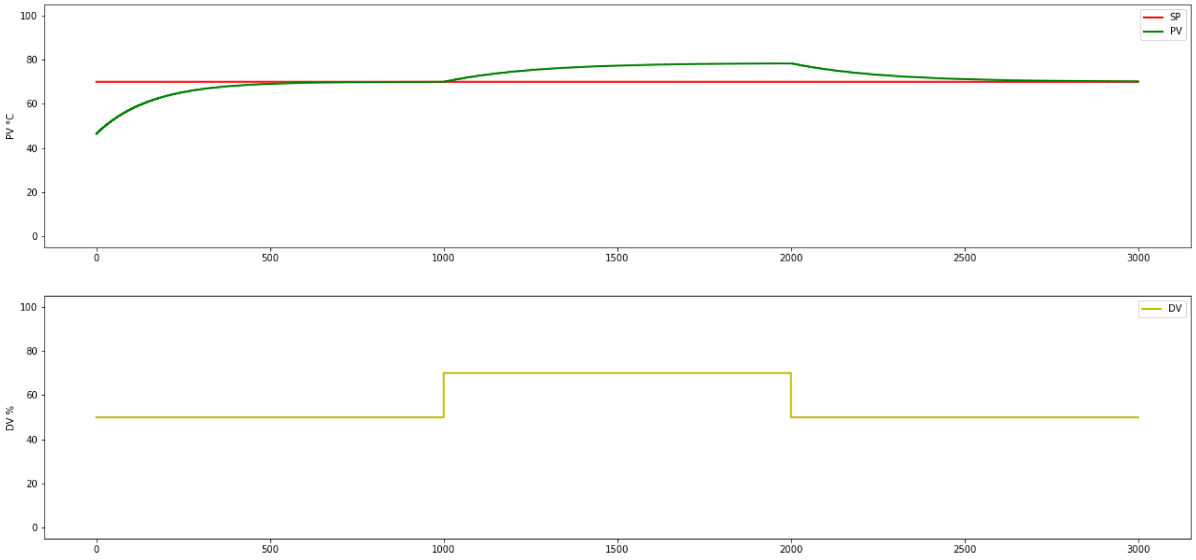

Figura 6.8: Risposta in catena chiusa ad un cambio nel set point

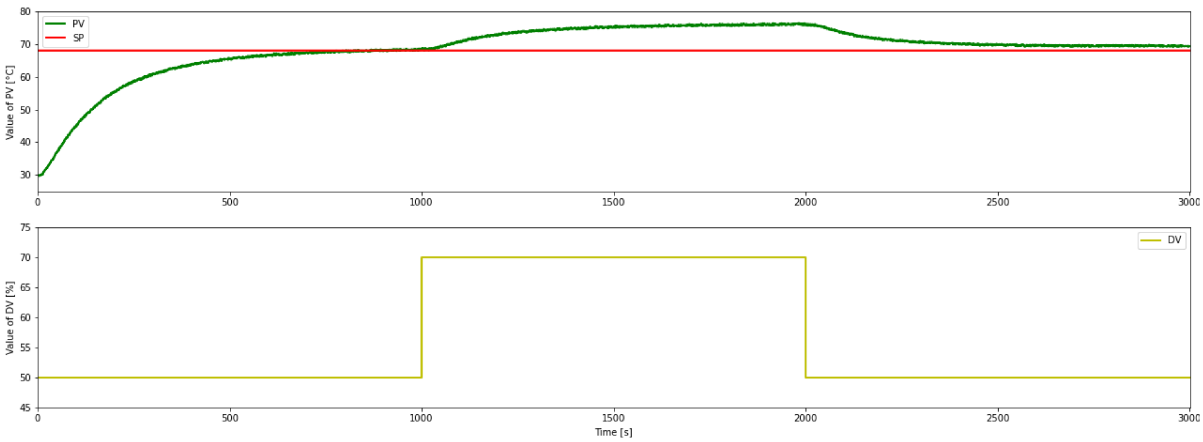

Figura 6.9: Risposta in catena chiusa ad un cambio nel set point

Anche in quest'altro esempio la simulazione in figura 6.8 e l'applicazione in figura 6.9 combaciano quasi perfettamente.

6.4 Risposta ad un cambio in DV: FF attivato e PID disattivato

```
# Response to DV : FF and controller in manual mode
ManPath = {0: True, TSim: True}
MVManPath = {0: MV0, TSim: MV0}
SPPath = {0: PV0, 850: PV0, TSim: PV0}
# DVPath = {0: DV0, 1000: DV0 + 30, TSim: DV0 + 30}
DVPath = {0: DV0, 1000: DV0 + 30, 2000: DV0, TSim: DV0}
FF = True
ManFF = True
```

In questo caso i parametri sono gli stessi del caso precedente, con la sola differenza che le variabili FF e $ManFF$ sono a $true$, dato che il meccanismo di feedforward è attivato. Il PID è disattivato ed il valore di MV è fissato a 50%.

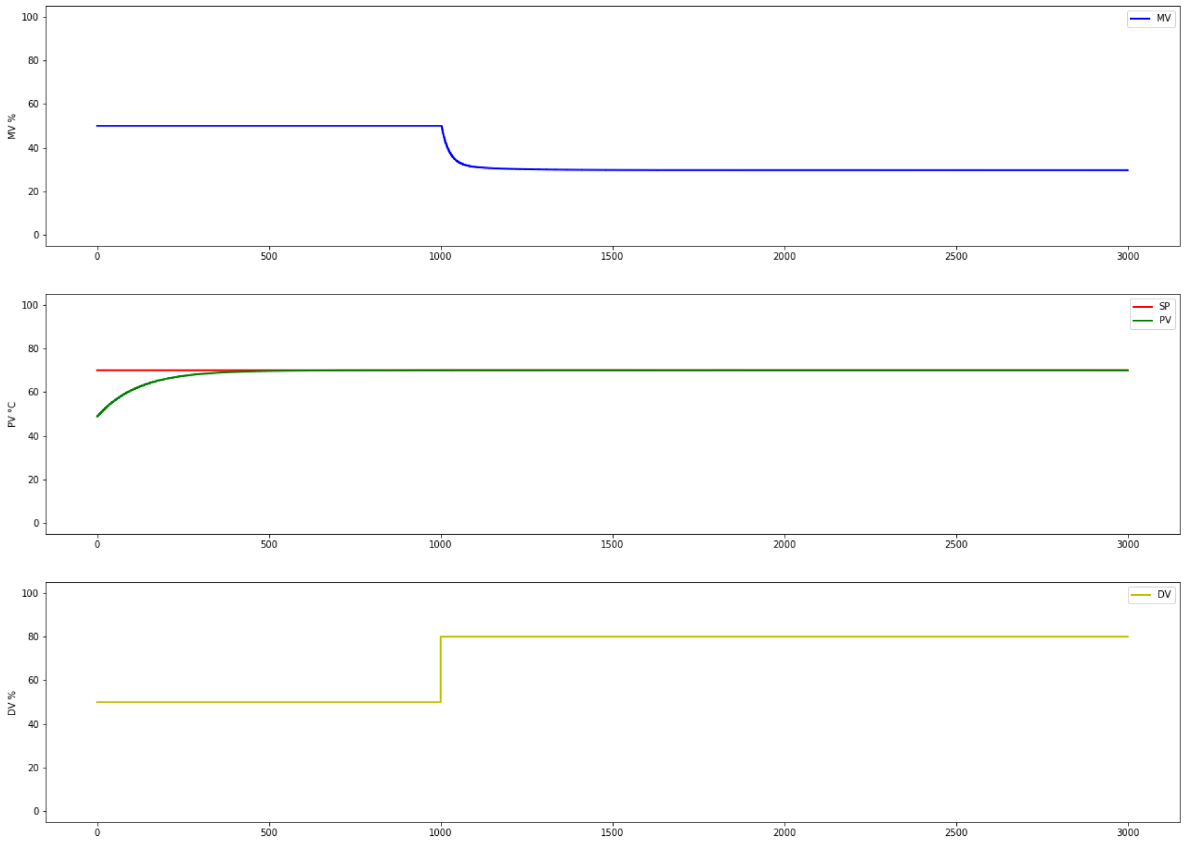

Figura 6.10: Simulazione della risposta ad anello aperto con feedforward attivato

Dal grafico in figura 6.10 è evidente l'efficacia del feedforward. Non appena avviene il cambio nel DV al secondo 1000 il feedforward interviene immediatamente per correggerlo. Nel grafico centrale si vede come il *PV* non si discosta dal *SP* neanche al momento del disturbo. Questo indica che il feedforward è stato implementato correttamente rispetto al modello.

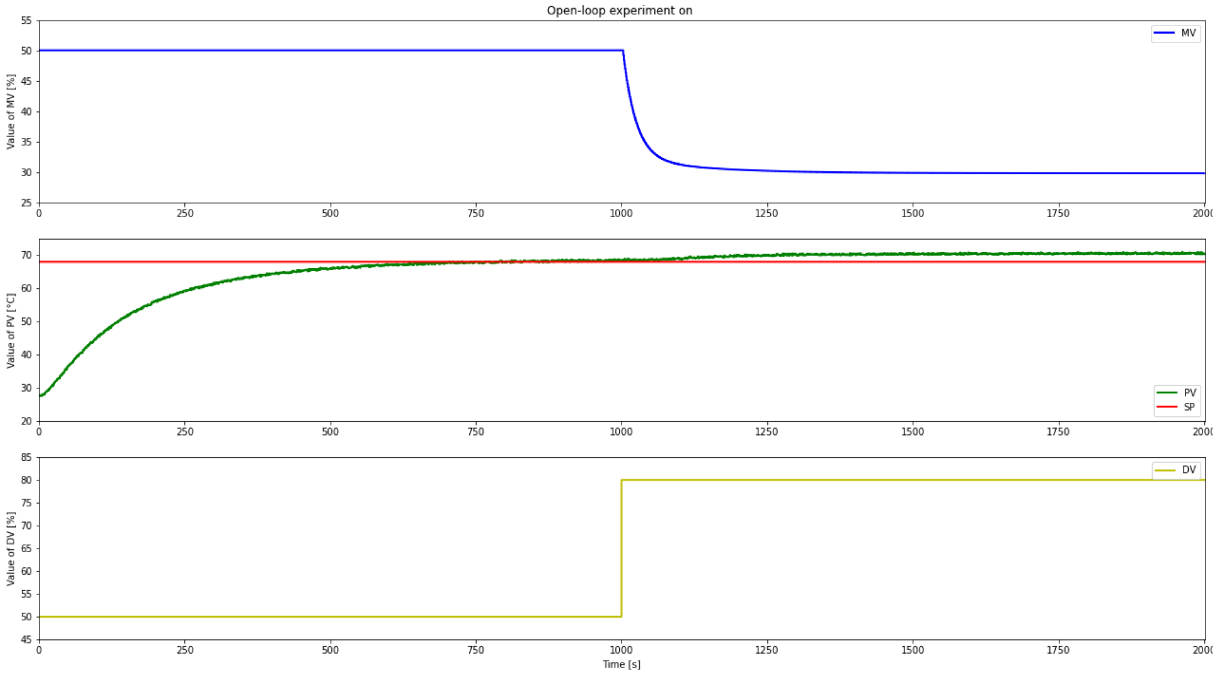

Figura 6.11: Applicazione della risposta ad anello aperto con feedforward attivato

Nel grafico in figura 6.11 ci sono delle discrepanze: il feedforward non compensa totalmente il disturbo. Questo succede perché il modello non è esatto. La differenza della temperatura ambiente tra il momento della creazione del modello e quello della sperimentazione sono diversi. Per questo motivo è importante che il feedforward sia accompagnato da un meccanismo di feedbeck, perché quest'ultimo permette di eliminare l'errore a regime. Nonostante la non perfetta compensazione del disturbo, l'errore a regime risulta inferiore ai $4°C$ e per il mio caso di interesse è accettabile.

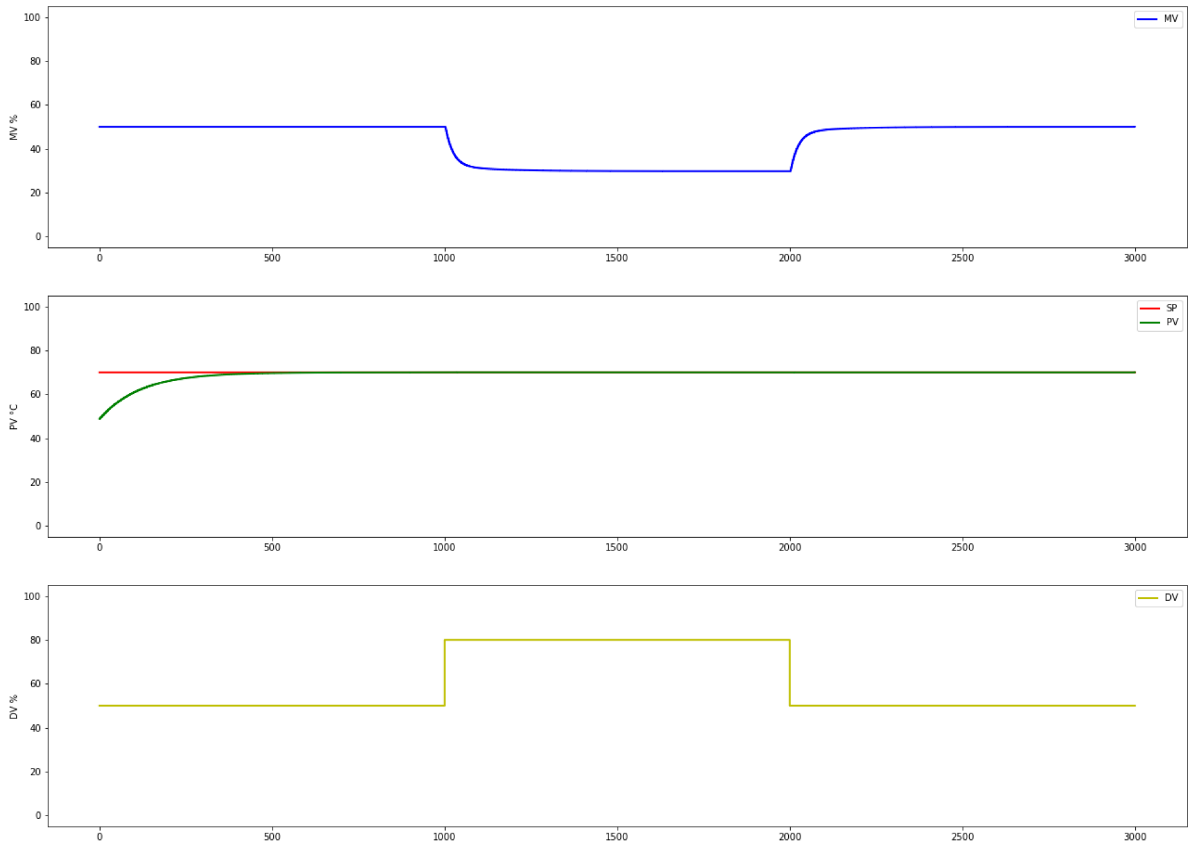

Figura 6.12: Simulazione della risposta ad anello aperto con feedforward attivato

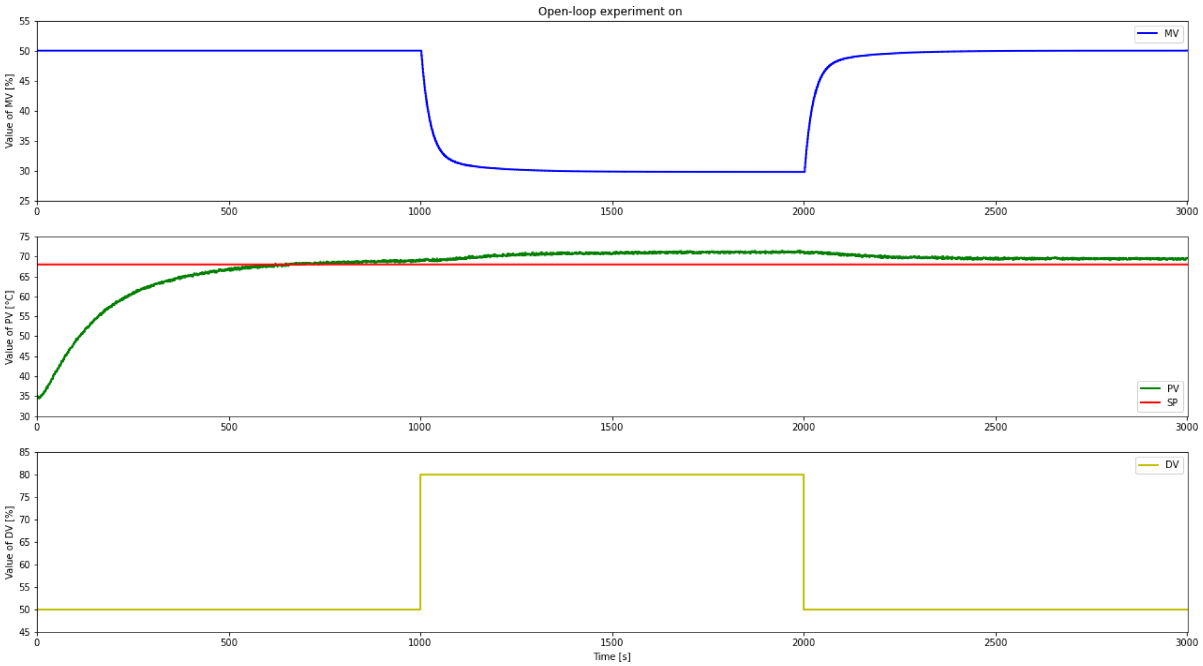

Figura 6.13: Applicazione della risposta ad anello aperto con feedforward attivato

Anche in questo secondo caso, nel grafico in figura 6.13, il disturbo non è compensato in maniera esatta.

6.5 Risposta ad un cambio in DV: FF disattivato e PID attivato

```
# Response to DV : No FF and controller in automatic mode
ManPath = {0: False, TSim: False}
MVManPath = {0: MV0, TSim: MV0}
SPPath = {0: PV0, 850: PV0, TSim: PV0}
# DVPath = {0: DV0, 1000: DV0 + 30, TSim: DV0 + 30}
DVPath = {0: DV0, 1000: DV0 + 30, 2000: DV0, TSim: DV0}
FF = False
ManFF = False
```

Ancora una volta i parametri sono simili. FF e $ManFF$ sono a $false$ in quanto il meccanismo di feedforward è disattivato. Questa volta il PID è attivato quindi $ManPath$ è sempre $false$.

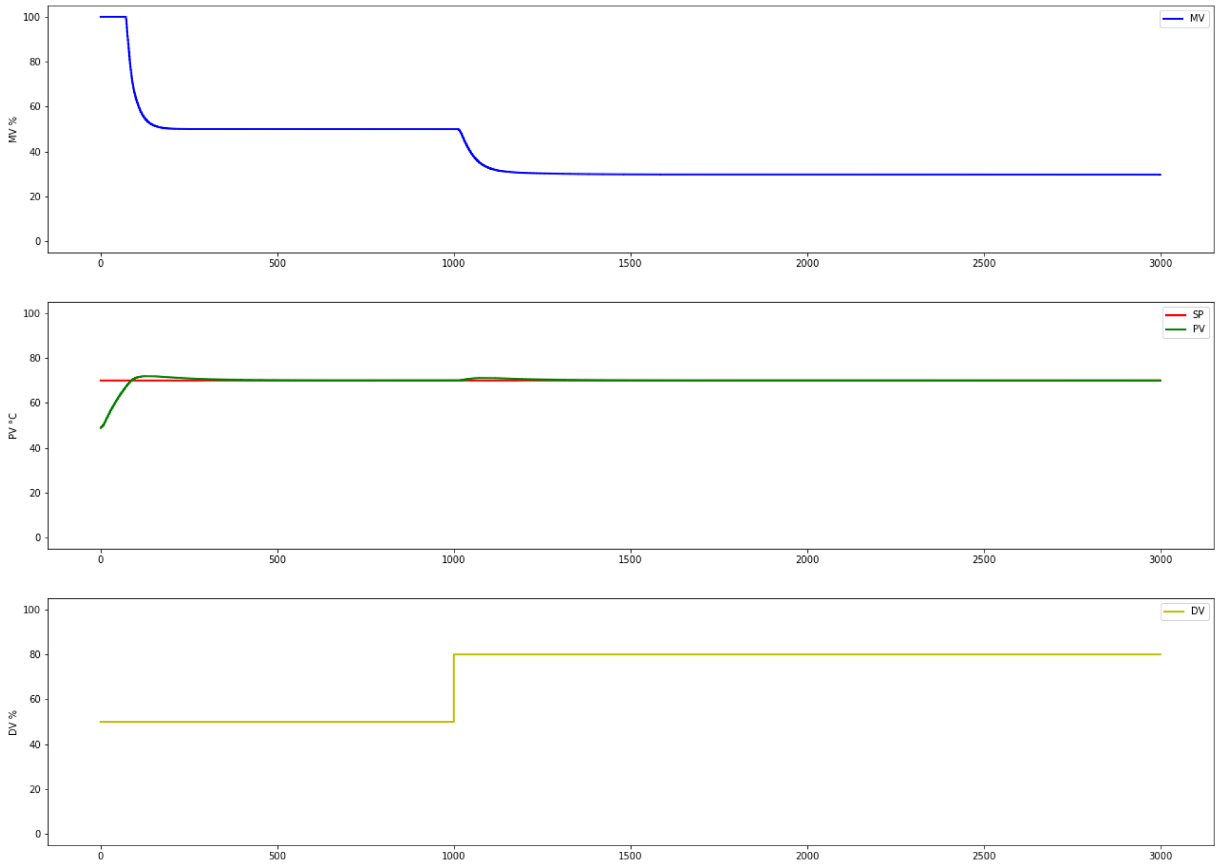

Figura 6.14: Simulazione della risposta ad anello chiuso senza feedforward

Anche in questo caso il PID risponde in maniera corretta al disturbo. Tuttavia non è efficiente come il meccanismo di feedforward. All'istante in cui avviene il disturbo si può infatti vedere una leggera deviazione del PV rispetto al SP. Il feedback non è reattivo come il feed-

back, dato che quest'ultimo è costruito su misura per compensare il disturbo proveniente da *DV*.

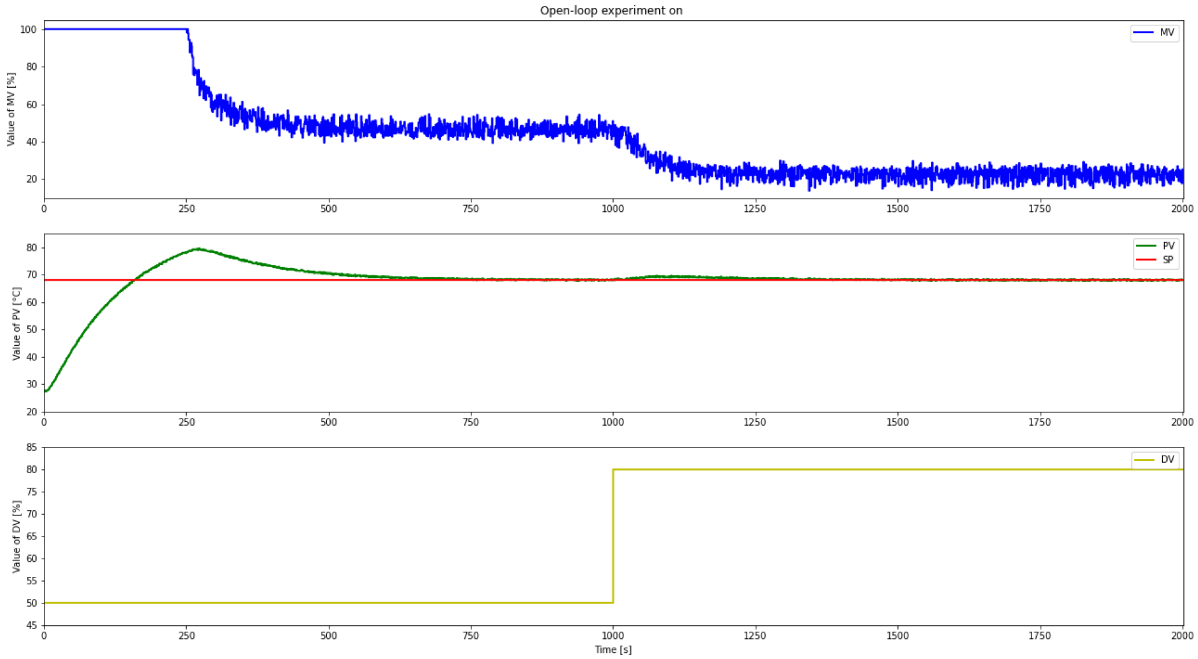

Figura 6.15: Applicazione della risposta ad anello chiuso senza feedforward

Nel grafico in figura 6.15 è comunque visibile la variazione creata dal disturbo. Nonostante il tempo di reazione minore rispetto al feedforward, il feedback assicura un errore a regime nullo.

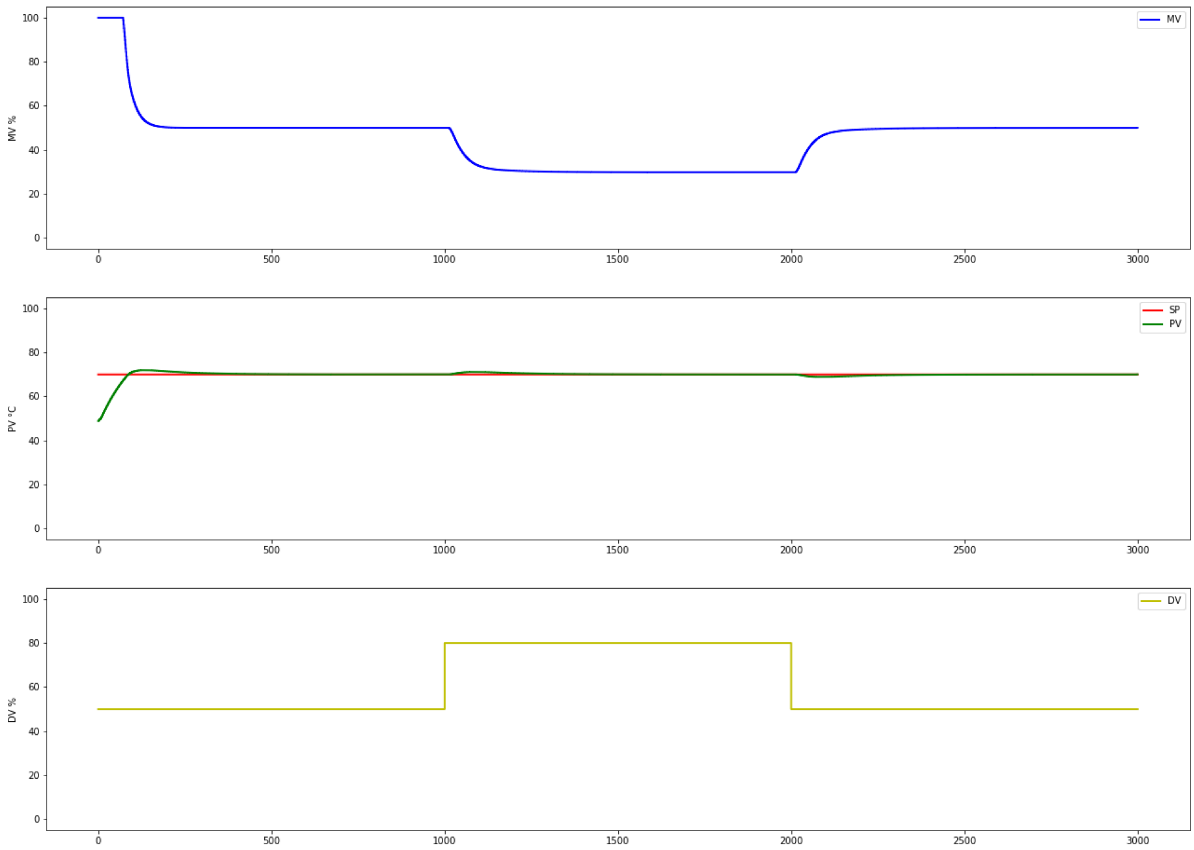

Figura 6.16: Simulazione della risposta ad anello chiuso senza feedforward

Come nel caso precedente il PID risponde in maniera corretta ma, al momento del cambio in *DV* è visibile una deviazione tra *PV* e *SP*.

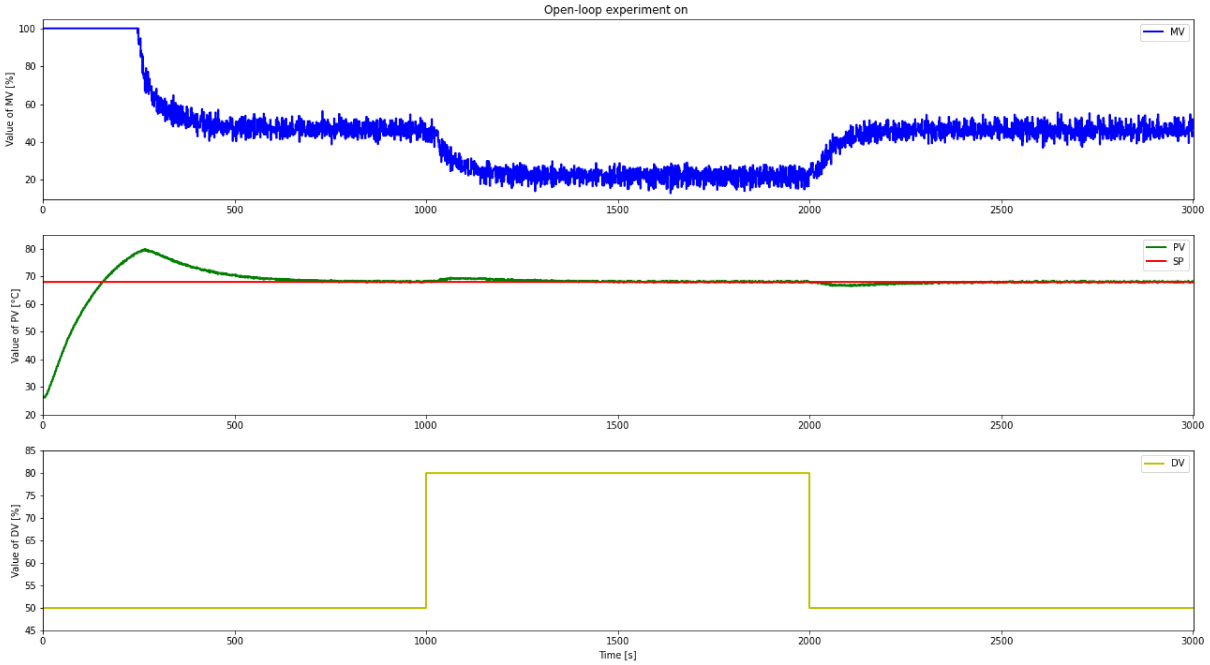

Figura 6.17: Applicazione della risposta ad anello chiuso senza feedforward

Lo stesso si verifica per l'applicazione sul chip.

Dall'analisi di questi grafici risultano evidenti i vantaggi e svantaggi dei due meccanismi di feedback e feedforward:

1. Il feedforward permette un controllo reattivo, al costo di non avere un errore nullo a regime
2. Il feedback ha un tempo di reazione più lento, ma consente di avere un errore nullo a regime

E' chiaro che la combinazione dei due meccanismi consente il miglior risultato possibile: un tempo di reazione breve ed un errore a regime nullo. Questo caso è analizzato nell'ultima sezione.

6.6 Risposta ad un cambio in DV: FF attivato e PID attivato

```
# Response to DV : FF and controller in automatic mode
ManPath = {0: False, 1100: False, TSim: False}
MVManPath = {0: MV0 + 20, TSim: MV0 + 20}
SPPath = {0: PV0, 850: PV0, TSim: PV0}
# DVPath = {0: DV0, 1000: DV0 + 30, TSim: DV0 + 30}
DVPath = {0: DV0, 1000: DV0 + 30, 2000: DV0, TSim: DV0}
FF = True
ManFF = True
```

In questa ultima simulazione entrambi PID e feedforward sono attivati. Questa soluzione è ottimale in quanto permette una ottima risposta al cambiamento in *DV* (grazie al feedforward), una buona risposta ad un disturbo esterno e un errore a regime nullo. Questo sarà visibile molto bene nelle sperimentazioni sul chip.

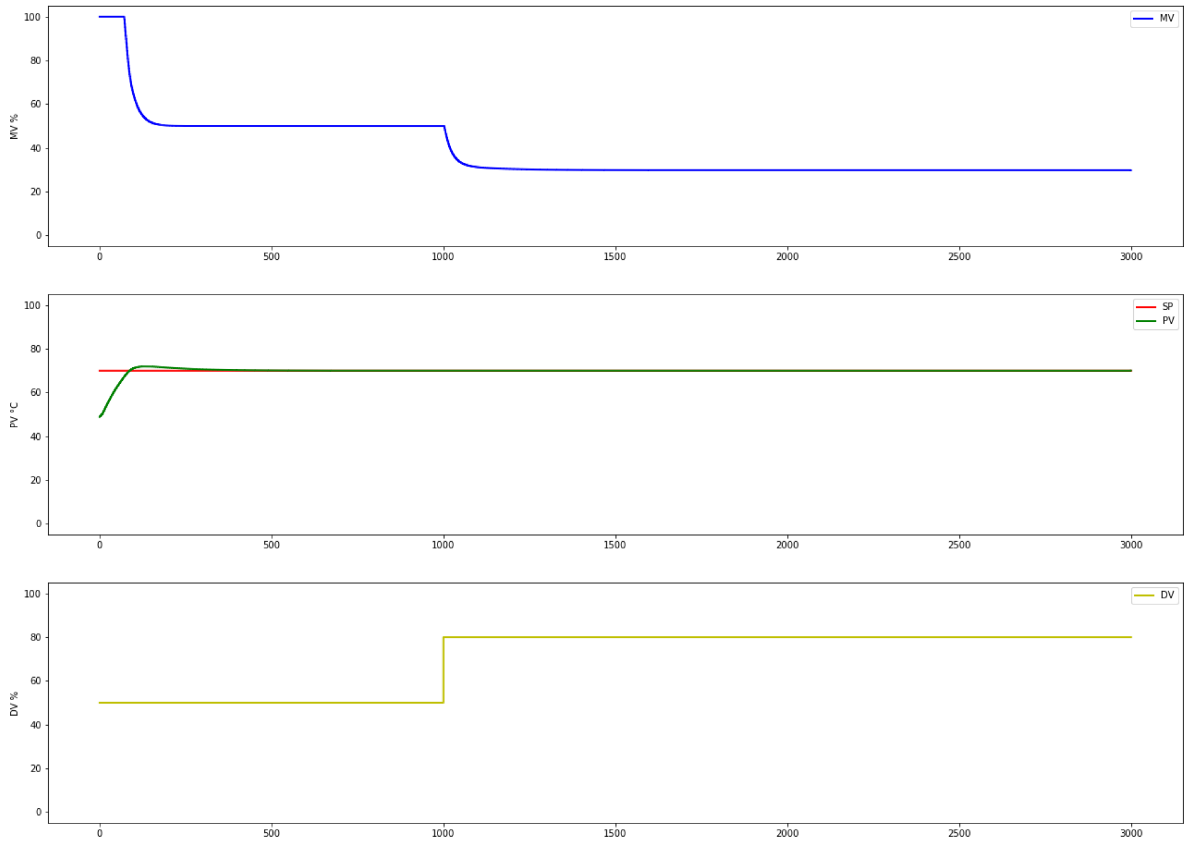

Figura 6.18: Simulazione della risposta ad anello chiuso con feedforward

L'azione combinata di feedforward e PID permettono un ottimo controllo sul *PV*. Al momento del disturbo non è visibile una discrepanza tra *PV* e *SP*.

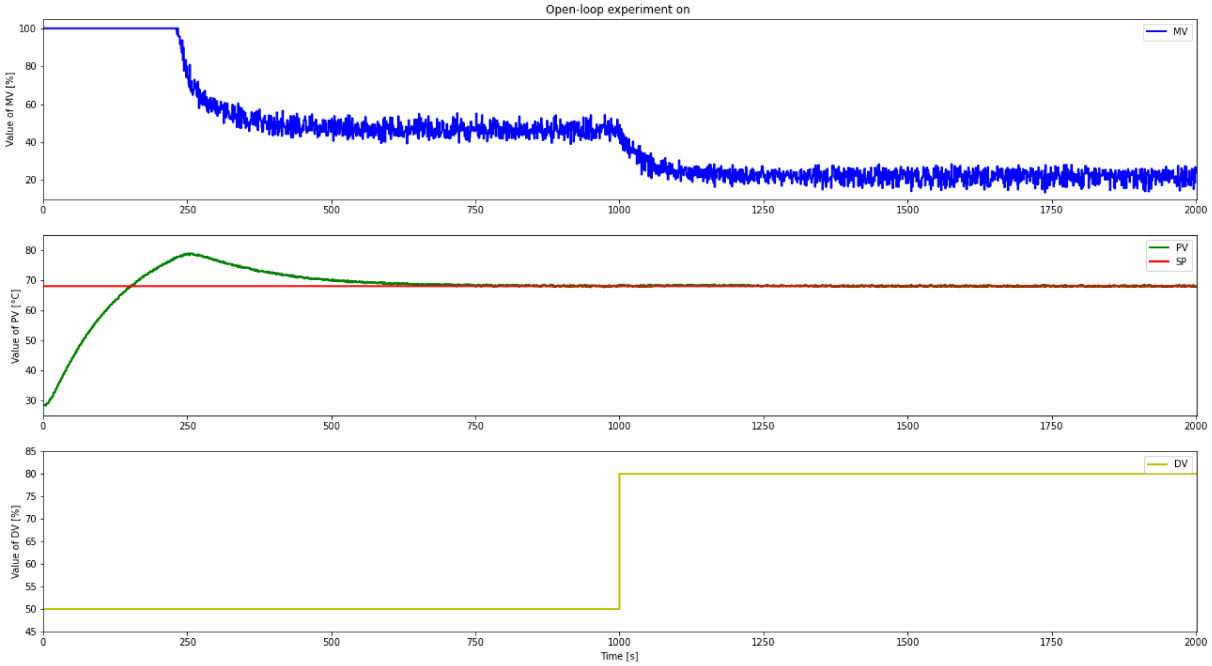

Figura 6.19: Applicazione della risposta ad anello chiuso con feedforward

Nel grafico in figura 6.19 è possibile vedere come l'azione combinata di feedforward e feedback portino ad un controllo ottimo. Non c'è, infatti, discrepanza tra *SP* e *PV*: il disturbo è stato completamente compensato.

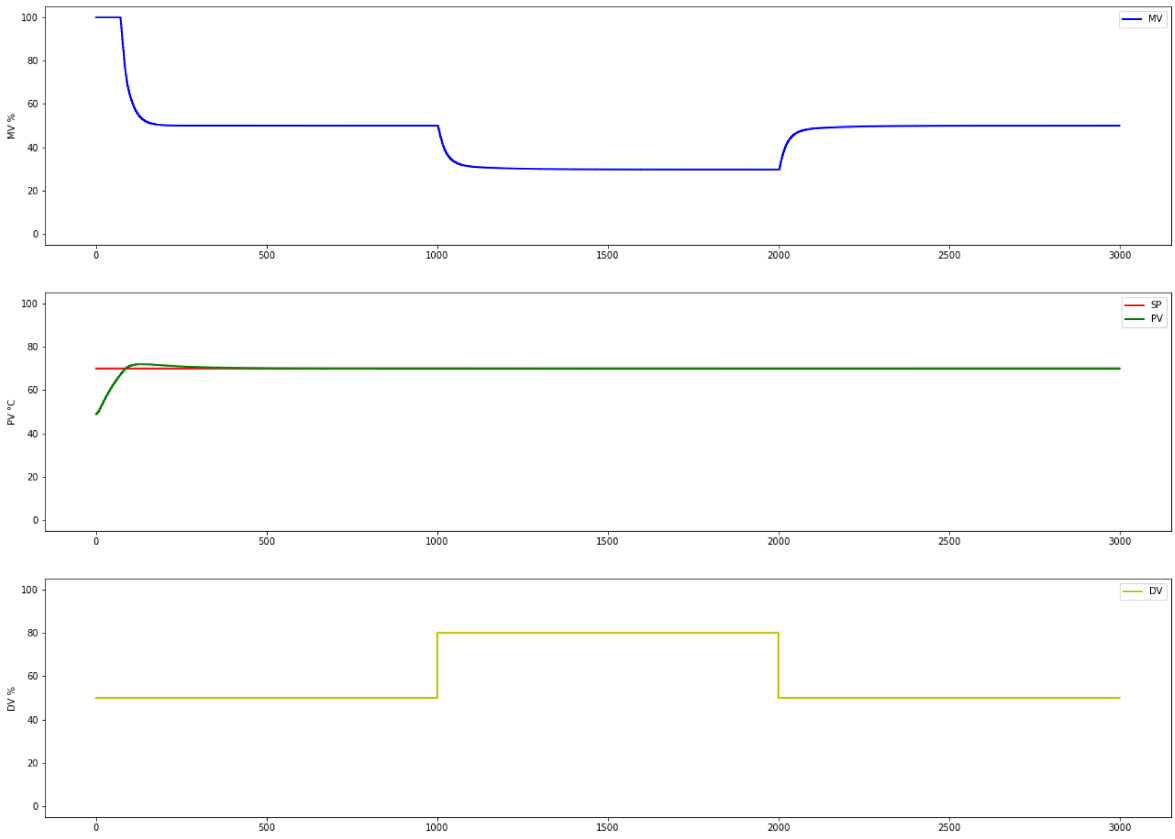

Figura 6.20: Simulazione della risposta ad anello chiuso con feedforward

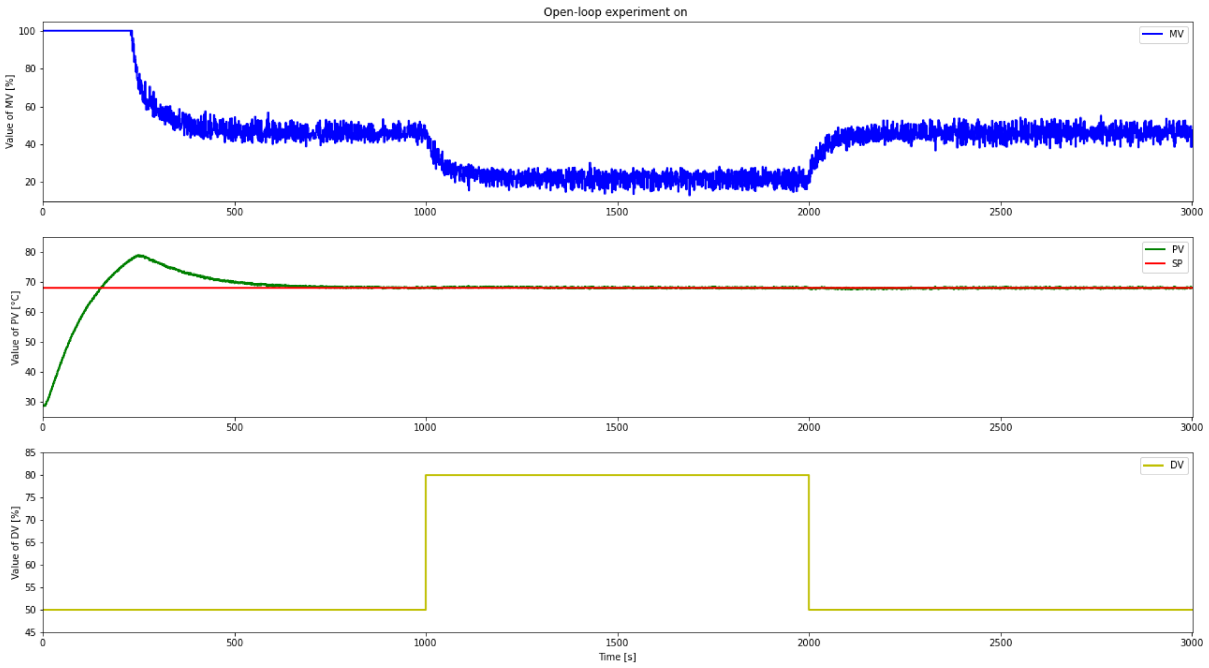

Figura 6.21: Applicazione della risposta ad anello chiuso con feedforward

Anche in questo secondo caso il disturbo è stato completamente compensato.

7. Conclusione

Questa tesi mi ha permesso di provare in prima persona l'efficacia di un laboratorio take home nell'apprendimento della teoria dei controlli automatici. I laboratori portatili possono essere utilizzati ovunque e in qualsiasi momento e facilitare notevolmente l'insegnamento di questa materia. TCLab si è dimostrato una risorsa molto valida per lo sviluppo di un controllore digitale e l'identificazione di un modello.

Mettere in pratica i concetti utilizzate attraverso questo dispositivo è molto semplice grazie alla documentazione fornita dai ricercatori di Apmonitor Process Dynamics and Control. Tutte le piattaforme utilizzate sono facilmente accessibili e utilizzabili, anche per gli studenti che non possiedono molte competenze informatiche.

Gli obiettivi prefissati per questo progetto riguardo alle funzionalità e alle applicazioni pratiche che TCLab offre sono stati raggiunti. Nello specifico, l'analisi dei dati sperimentali per osservare i punti di forza e di debolezza del controllore PID e del feedforward e rimarcare l'efficacia del loro uso combinato per la reiezione di un disturbo sul processo.

Bibliography

[1] McRoberts, M., n.d. ”Beginning Arduino”. 2nd ed. pp.1-20.

[2] Hedengren, J., 2019. ”Process Dynamics and Control”. URL: http://apmonitor.com/pdc/index.php/Main/HomePage

[3] Hedengren, J., 2019. ”Temperature Control Lab”. URL: http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl

[4] Rossiter, J., Pope, S., Jones, B. and Hedengren, J., 2019. ”Evaluation and demonstration of take home laboratory kit”. ScienceDirect, 52(9), pp.56-61.

[5] Carion Pelton, 2019. ”Take home labs and Ball and Beam”. Oklahoma State University. Stillwater, Oklahoma.

[6] Kantor, J. and Sandrock, K., 2018. ”TCLab Overview — TCLab”. https://tclab.readthedocs.io/en/latest/notebooks/01_TCLab_Overview.html.