WEINTEK LABS., INC.

オーブン温度制御

FB Weintek_PID 応用例

サンプルプロジェクト

目次

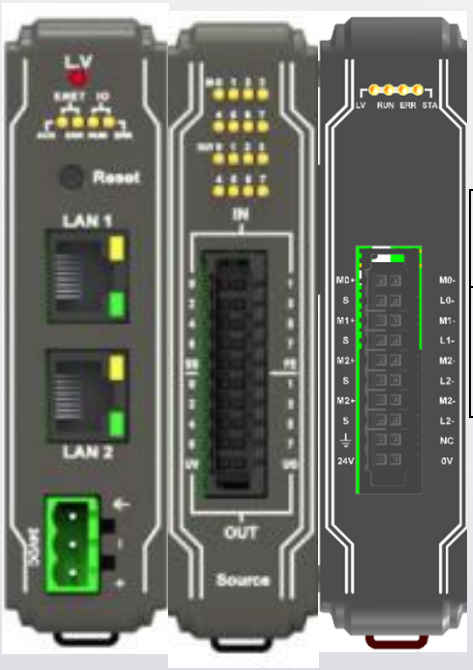
1. 概要及びシステム操作

概要

以下の例では、ソリッドステートリレー(SSR)で AC 電源を制御して PWM 信号に達することで温度を制御し、Weintek 独自開発の PID 機能ブロックと合わせて自動校正を実行することを実演します。

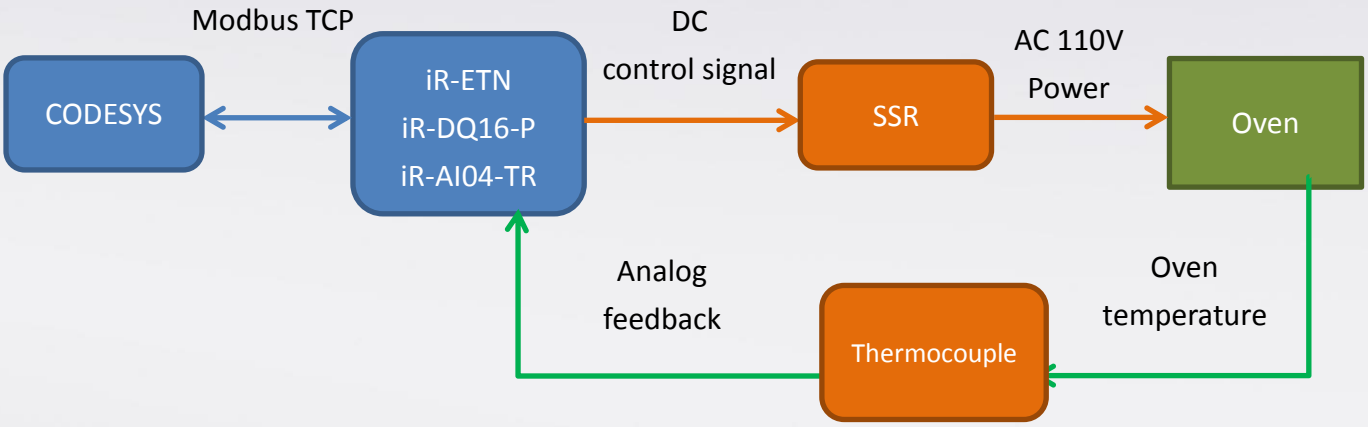
システム操作

iR-ETN
iR-DM16-P
iR-AI04-TR

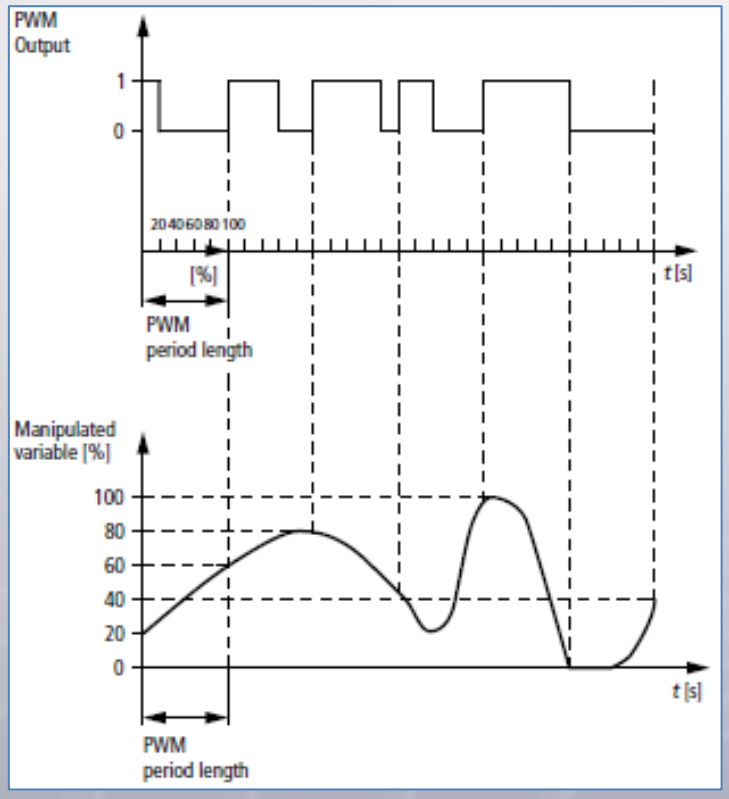


順番	0	1	2
モジュール名	iR-ETN	iR-DM16-P	iR-AI04-TR

オーブン信号のフローチャート



CODESYS がイーサネットを通じて iR-ETN に接続し、iR-DQ16-P のデジタル出力信号を制御します。
iR-DQ16-P は AC110V の通電時間を制御します。
熱電対(Thermocouple)で温度値を iR-AI04-TR にフィードバックし、PID から次回の通電周期を判断させます。
※PWM(Pulse Width Modulation：パルス幅変調)：
PID は Duty cycle を計算することができます。この Duty cycle とは、出力信号 = True の周期(cycle)です。



2. Weintek library 機能ブロックの紹介

Weintek library をインストールする方法については、《Weintek 関数ライブラリ》アニュアルの第 2 章をご参照ください。

Weintek_PID

Weintek_PID とは、Weintek 独自開発の PID コントローラ機能ブロックで、これをアナログモジュールと合わせて PID 制御を実行可能です。本 PID 機能ブロックの関連公式については《Weintek 関数ライブラリ》マニュアルをご参照ください。本文では、使用する方法について説明します。

PID の入力方式は、手動/自動と分けられています。手動モード(Manual=TRUE)では、MV(Manipulated Value)は手動出力値(Mout)に等しいで、他のパラメータは出力値に影響を与えません。

自動モード(Manual=FALSE)では、PID 制御公式のパラメータにより自動的に値を出力します。自動出力に影響を与えるパラメータは Kp、Ki、Kd、Tf、BIAS、Time_Base、Error_Deadband があります。

PID パラメータをどう入力すればいいかのが分からない場合、PID 自動調整(Autotune)機能も提供されています。有効にするには Run & Autotune = TRUE の状況下で有効されます。有効にした後、全力で(MV=MV_Max)PV が SV に達したまで出力し、そして解放し(MV=MV_Min)、PV が SV より小さくなった後、再度に全力で出力し、二度目の PV が SV に達した場合、Autotune が完成しました(Autotune=FALSE)。即ち、Kp、Ki、Kd、Tf の調整が完成しました。PV 曲線は《4.温度制御曲線》の Autotune ゾーンにご参照ください。

PWM

PWM(Pulse Width Modulation)信号の機能ブロックを使用します。PWM の週期(Period)&責任比例(Duty)を入力し、(Q)を出力して TRUE から出力始めます。

例：一個の周期が 10 秒で、Q=TRUE の時間が 2.5 秒の PWM 信号を出力したい場合：Period = T#10S ；Duty = 25 。

3. プログラムを設定する

宣言

```
PROGRAM PLC_PRG
VAR
    PID : weintek.PID;
    xRun, xDir, xQE, xDE, xAutoTune : BOOL ;
    rSV,rPV,rAuto_Deadband,rBIAS,rMV,rI_MV : REAL ;
    rTime_Base : REAL := 0.1 ;
    rError_Deadband : REAL := 0.0 ;
    rMV_Max : REAL := 100.0 ;
    rMV_Min : REAL  := 0.0 ;
    PWM : weintek.PWM ;
    rTemp : REAL ;
```

プログラムで必要な変数を宣言し、初期値を設定します。

ST プログラム

```
1   rTemp2 :=INT_TO_REAL(TR0) ;
2   rPV := rTemp2 / 10 ;
3   PID(
4       Manual:= xManual,
5       Run:= xRun,
6       SV:= rSV,
7       PV:= rPV,
8       Dir:= xDir,
9       MV_Manual:= rMV_Manual,
10      MV_Max:= rMV_Max,
11      MV_Min:= rMV_Min,
12      Auto_Deadband:= rAuto_Deadband,
13      Bias:= rBIAS,
14      Time_Base:= rTime_Base,
15      Error_Deadband:= rError_Deadband,
16      MV=> rMV,
17      I_MV=> rI_MV,
18      Kp:= PVar.rKp,
19      Ki:= PVar.rKi,
20      Kd:= PVar.rKd,
21      Tf:=PVar.rTf,
22      AutoTune:= xAutoTune);
23
24  PWM(Enable:=xRun,Period:=timPeriod,Duty:=rMV,Q=> xOut);
25  DO0 := xOut ;
26
```

1~2： INT(温度入力)を REAL に変換する

3~22：ブロックで変数を入れる

24~25： PID.MV を PWM に変換し、そして DO0 で出力する

Mapping

DO0 => %QX0.0 (SSR の出力を制御する)

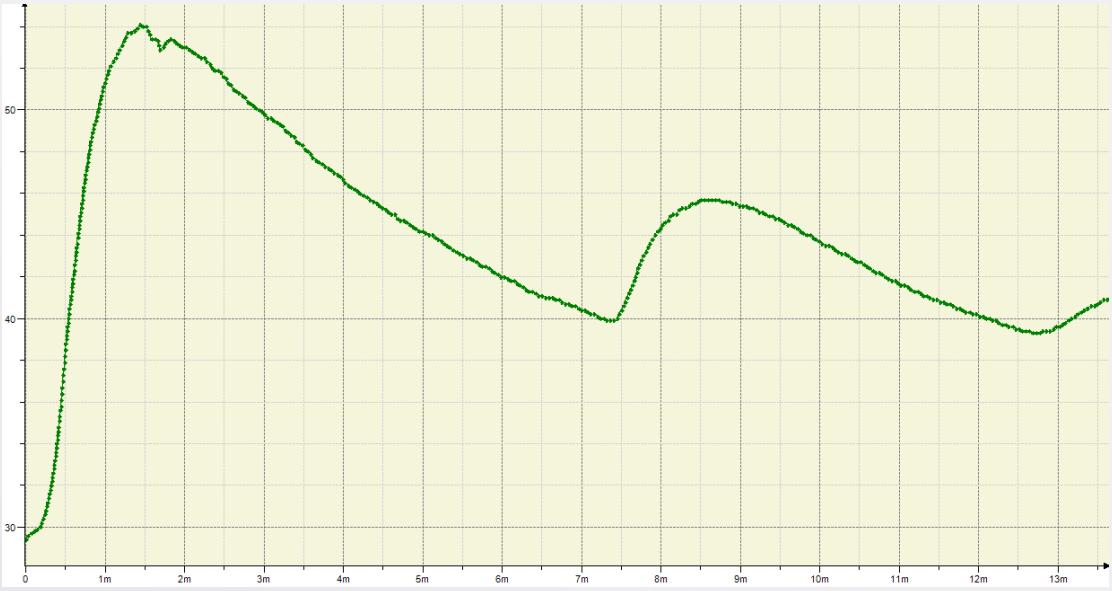
TR0 => %IW0 (温度入力)

4. 制御曲線図

挙げられた例での温度制御曲線は以下の通りで、緑線は温度です。開始温度は常温でやく 28℃で、目標温度 SV は 40℃です。

Auto Tuning 温度曲線

まずは Auto Tune を実行し、**Kp**、**Ki**、**Kd**、**Tf** の参考値を取り出します。下図は Auto Tune 期間内の温度曲線です。



Auto Tune の流れ：

Full power で温度が SV に達したまで出力します。

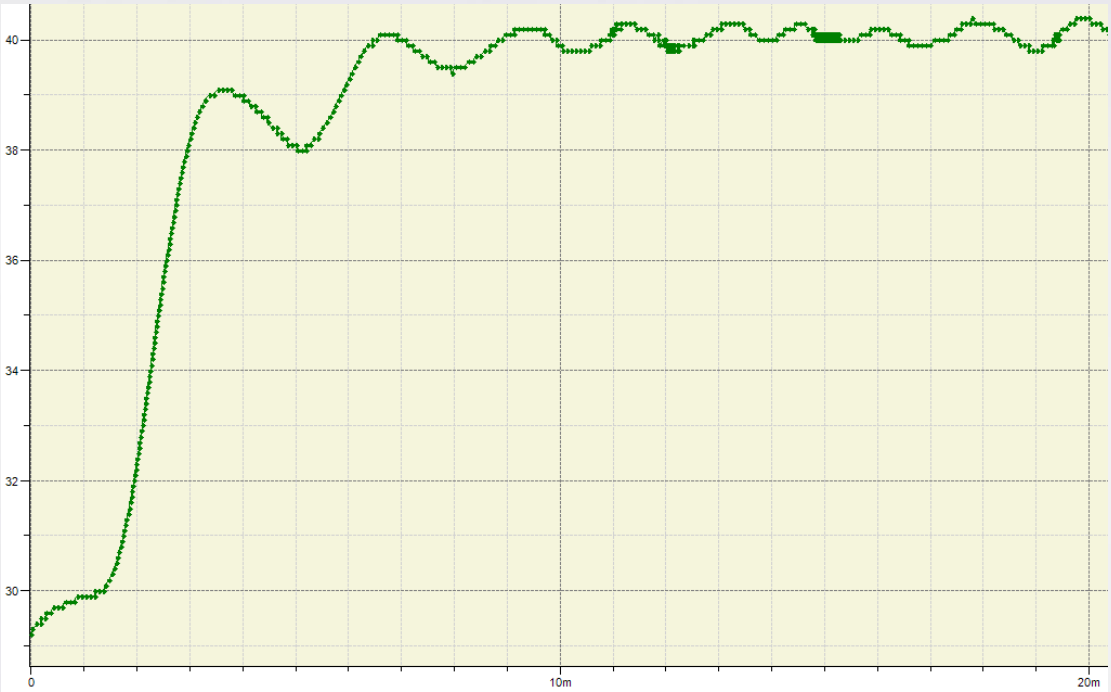
温度が SV まで下がってから、再度 Full power で出力します。

二度目の温度 SV 以下に下がったら、Auto Tune が完成したと示しています。

この時、システムは自動的に **Kp**、**Ki**、**Kd**、**Tf** を計算し、タグに出力します。

Auto Tune 原始制御曲線

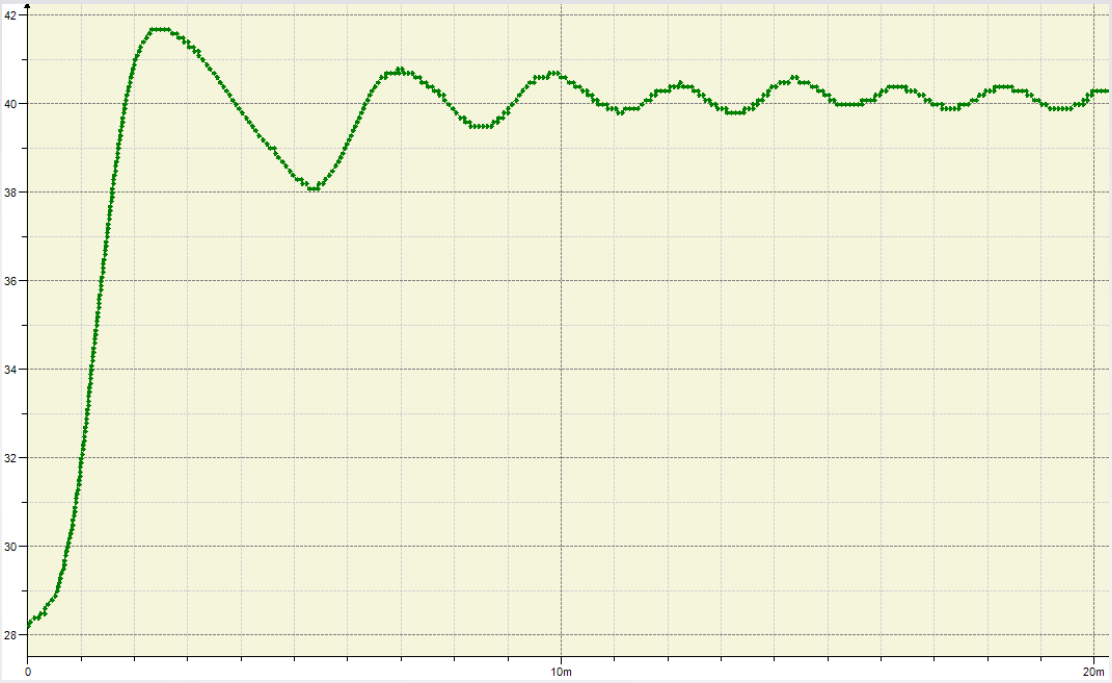
そして、このパラメータを再利用して改めて加熱すると、温度曲線が以下のようになります：



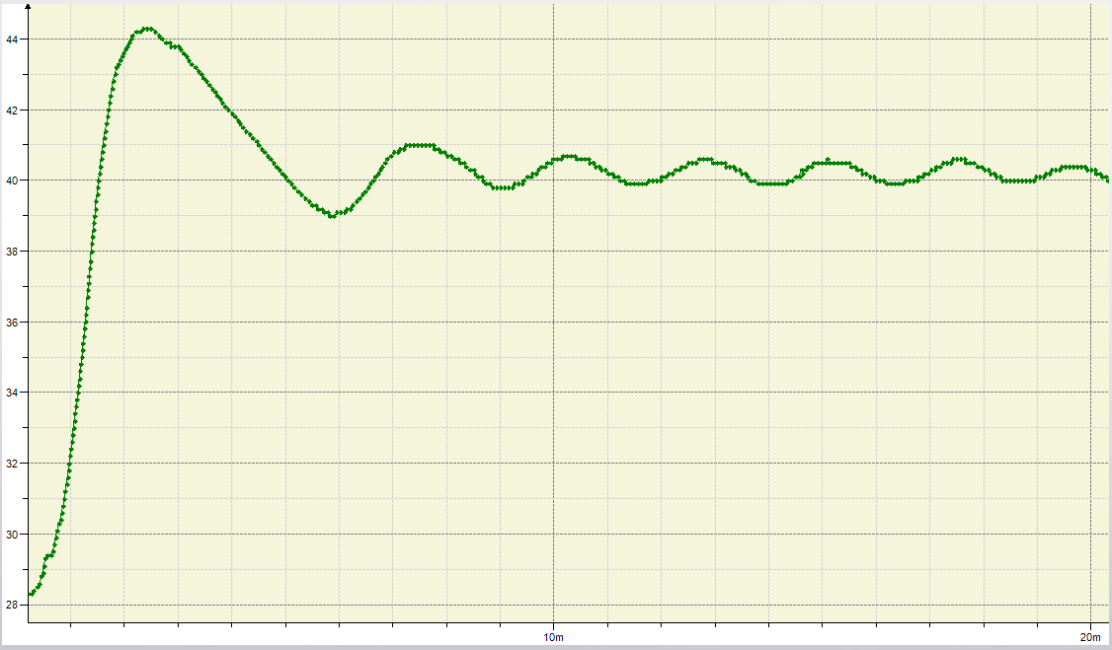
最大オーバーシュートが目標値 SV より低いので、手動でパラメータを調整します。

Auto Tune 手動で制御曲線を調整する

Kp パラメータを下げてから改めて加熱すると、温度曲線が下記のようになります：



上図が示した通り、最大オーバーシュートは目標値 SV を超えたが、コンバージェンス時間がより長かった(10 分間以上)ので、Ki パラメータを調整して改めて加熱すると、温度曲線は下記のようになります：



調整した後の温度曲線