AUTOMATIC CONTROL ENGINEERING

ZIEGLER NICHOLS GAINS LAB

PURPOSE: The purpose of this lab is to give you some experience with getting the Ziegler Nichols gains of a simple system using a PIC microcontroller.

SETUP: A heater is used to adjust the temperature of air flowing down a pipe. A thermistor is used to sense the temperature. A PIC calculates a control signal and sends it to the heater using a DAC and an OP AMP. It can also use PWM and an H BRIDGE.

PROCEDURE: Operate the setup using the PIC control code pipe.c. Estimate the borderline gain $\mathbf{K_P}$ and period $\mathbf{T_P}$ when proportional is acting alone. Use these to calculate the Ziegler Nichols gains of the system. Examine the performance of the system with the Ziegler Nichols gains.

ZIEGLER NICHOLS GAINS

CLOSED LOOP PROCEDURE

Ziegler and Nichols, through a series of experiments on simple systems, developed criteria for picking gains in a controller that would give good tracking performance. For a system that can be made unstable with proportional acting alone, the procedure they recommend for getting proper gains is as follows. With proportional acting alone and the system operating closed loop, increase the proportional gain until the system becomes borderline stable. Let the borderline gain be $\mathbf{K_P}$ and its period be $\mathbf{T_P}$. According to Ziegler Nichols proper PID gains are:

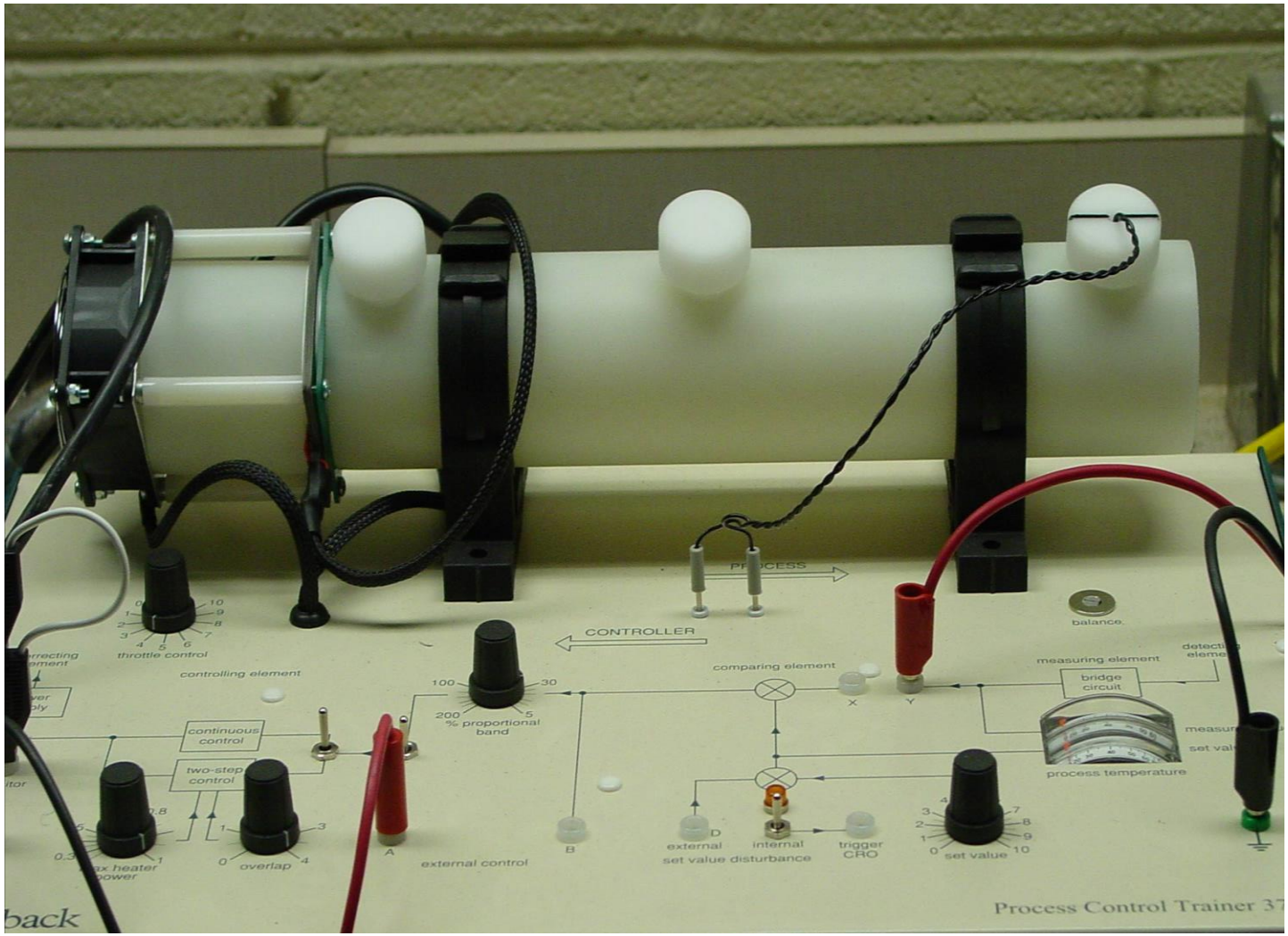
$$K_P = C\ \mathbf{K_P} \qquad K_I = K_P/T_I \qquad K_D = K_P * T_D$$

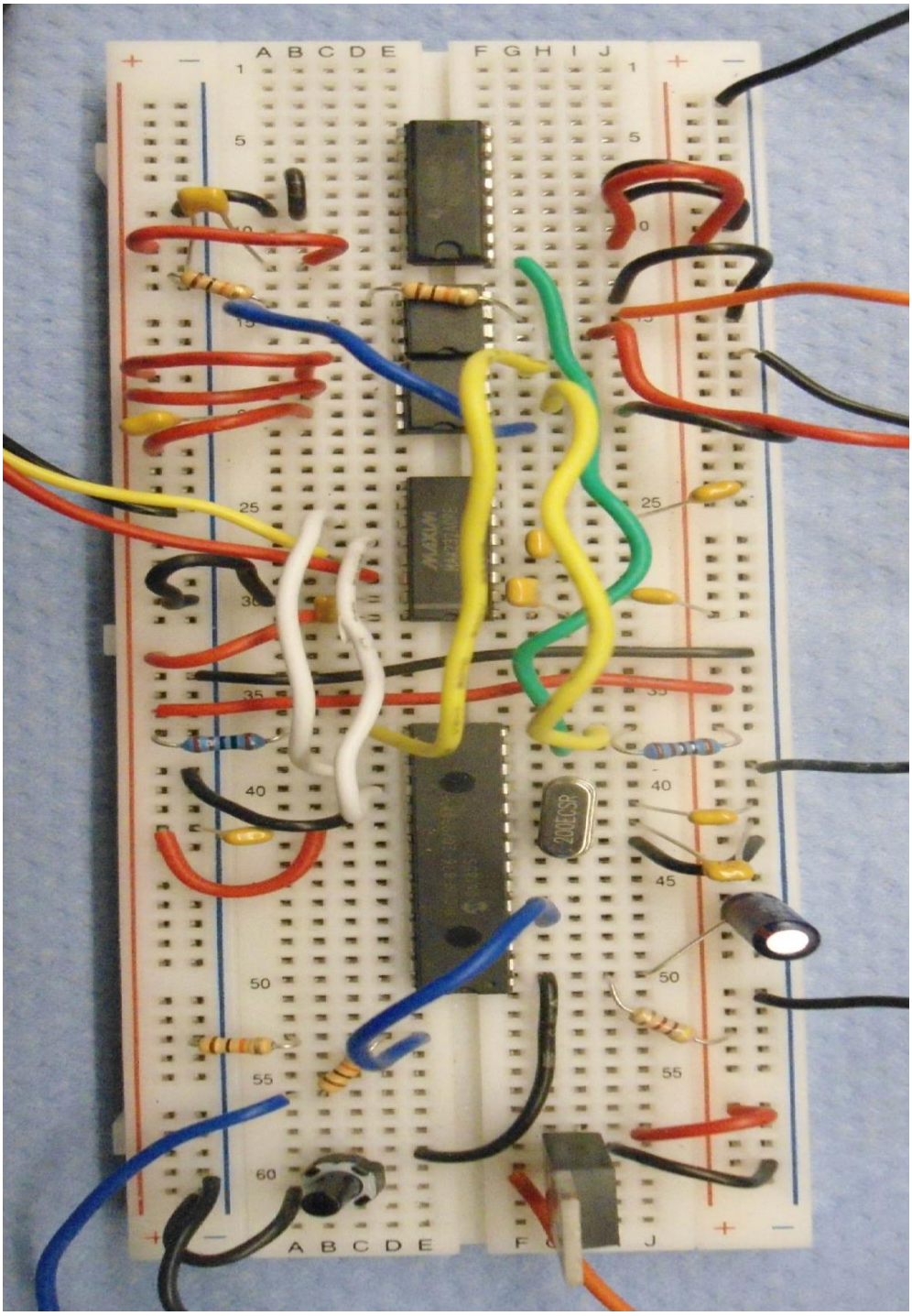
$$C=0.6 \qquad T_I = \mathbf{T_P}/2 \qquad T_D = \mathbf{T_P}/8$$

When only P and I are acting, they recommend:

$$K_P = C\ \mathbf{K_P} \qquad K_I = K_P/T_I$$

$$C=0.45 \qquad T_I = 5/6\ \mathbf{T_P}$$


CONTROLLER
balance
throttle control
controlling element
comparing element
measuring element
bridge circuit
continuous control
two-step control
% proportional band
process temperature
external control
external set value
disturbance
internal
trigger CRO
set value
overlap
Process Control Trainer


A B C D E
F G H I J
MAXIM
200ECSR

```
************************************
    PIC CODE FOR PIPE FLOW SETUP
              BREADBOARD
************************************/


/* header files */
#include <16f876.h>
#fuses HS,NOWDT
#fuses NOPROTECT,NOLVP
#fuses NOBROWNOUT,NOPUT
#device ADC=10 // 10 BIT
#use delay(clock=20000000)
#use i2c(master,sda=PIN_C4,scl=PIN_C3,slow)
#org 0x1F00,0x1FFF{}

/* declare types */
float target,data,s;
float control,sum,rate;
float error,wrong,sign;
float change,step,bias;
float gp,gi,gd,gs;
float band,size;
float signal;
float a,b,c;
int heater;
int power;

void heat(int bits);

void main()
{

/* set up ports */
   setup_adc_ports(ALL_ANALOG);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_timer_2(T2_DIV_BY_16,254,1);
   setup_ccp1(CCP_PWM);
   set_pwm1_duty(0);
   set_adc_channel(2);
   delay_us(21);

/*  set data */
    target=6.0;
    s=1023.0/10.0;
    target=target*s;
    sum=0.0;band=0.0;
    wrong=0.0;step=0.005;
    gp=10.0;gi=0.0;gd=0.0;
    bias=0.0*s;gs=0.0;
    a=0.0*s;b=10.0*s;

    delay_ms(7000);
```

```
while(TRUE)
{

/*  read sensor */
    data=read_adc();
    delay_ms(5);

/*  calculate error */
    error=target-data;
/*  proportional control */
    control=gp*error;

/*  intergral control */
    control=control+gi*sum;
/*  calculate error sum */
    sum=sum+error*step;

/*  calculate error rate */
    change=error-wrong;
    rate=change/step;
/*  derivative control */
    control=control+gd*rate;

/*  calculate error sign */
    size=abs(error);
    if(size<band) {sign=0.0;}
    if(error>+band)  {sign=+1.0;}
    if(error<-band)  {sign=-1.0;}
/*  switching control */
    control=control+gs*sign;

/*  saturation */
    signal=control+bias;
    if(signal<a) {signal=a;}
    if(signal>b) {signal=b;}

/*  heater signal */
    c=signal/1024.0;
    heater=c*c*254.0;
    power=c*254.0;
    set_pwm1_duty(heater);
    heat(power);

/*  store error */
    wrong=error;

}
}

void heat(int bits)
{
 i2c_start();
 i2c_write(0x5e);
 i2c_write(0);
 i2c_write(bits);
 i2c_stop();
}
```