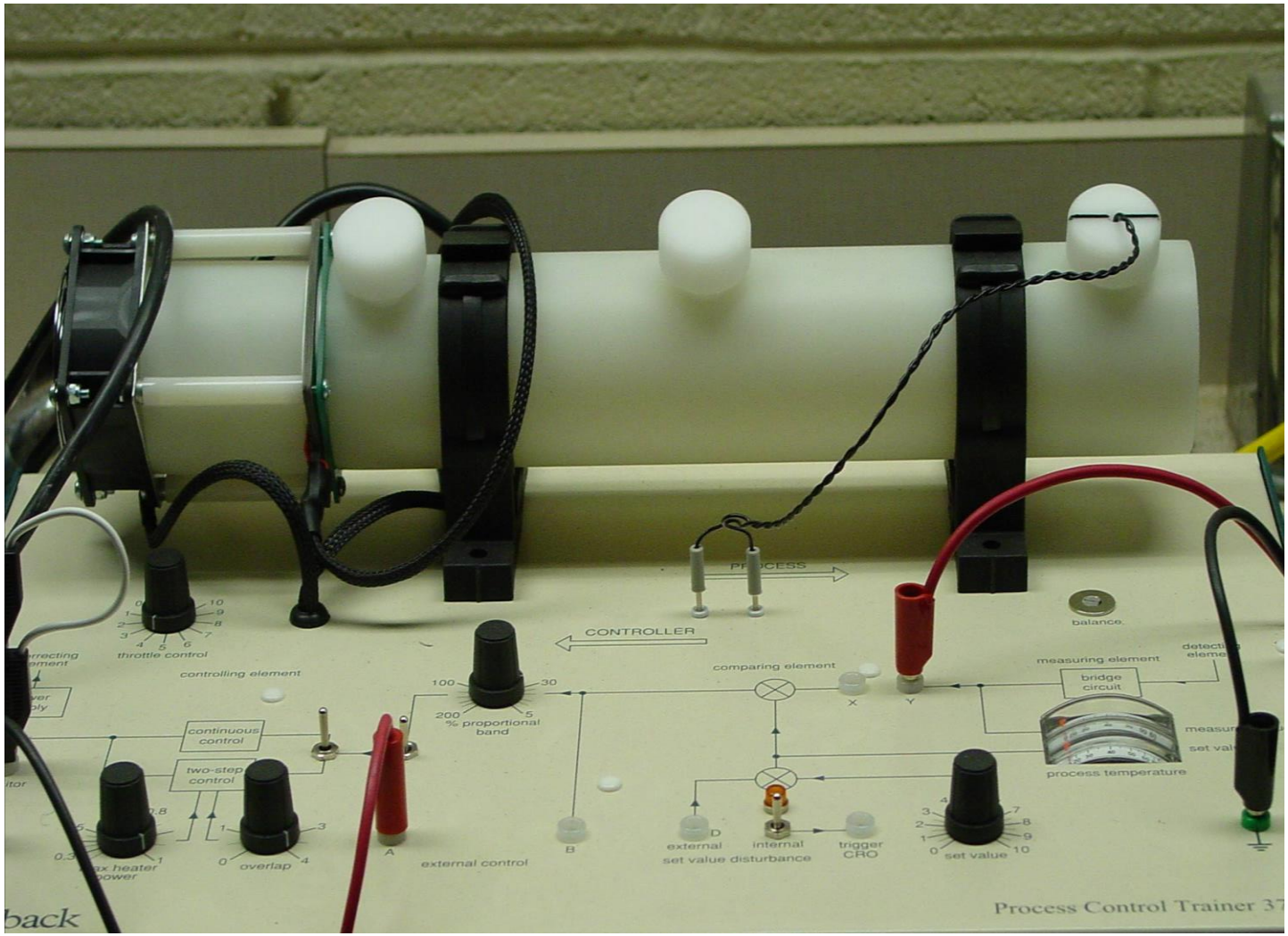
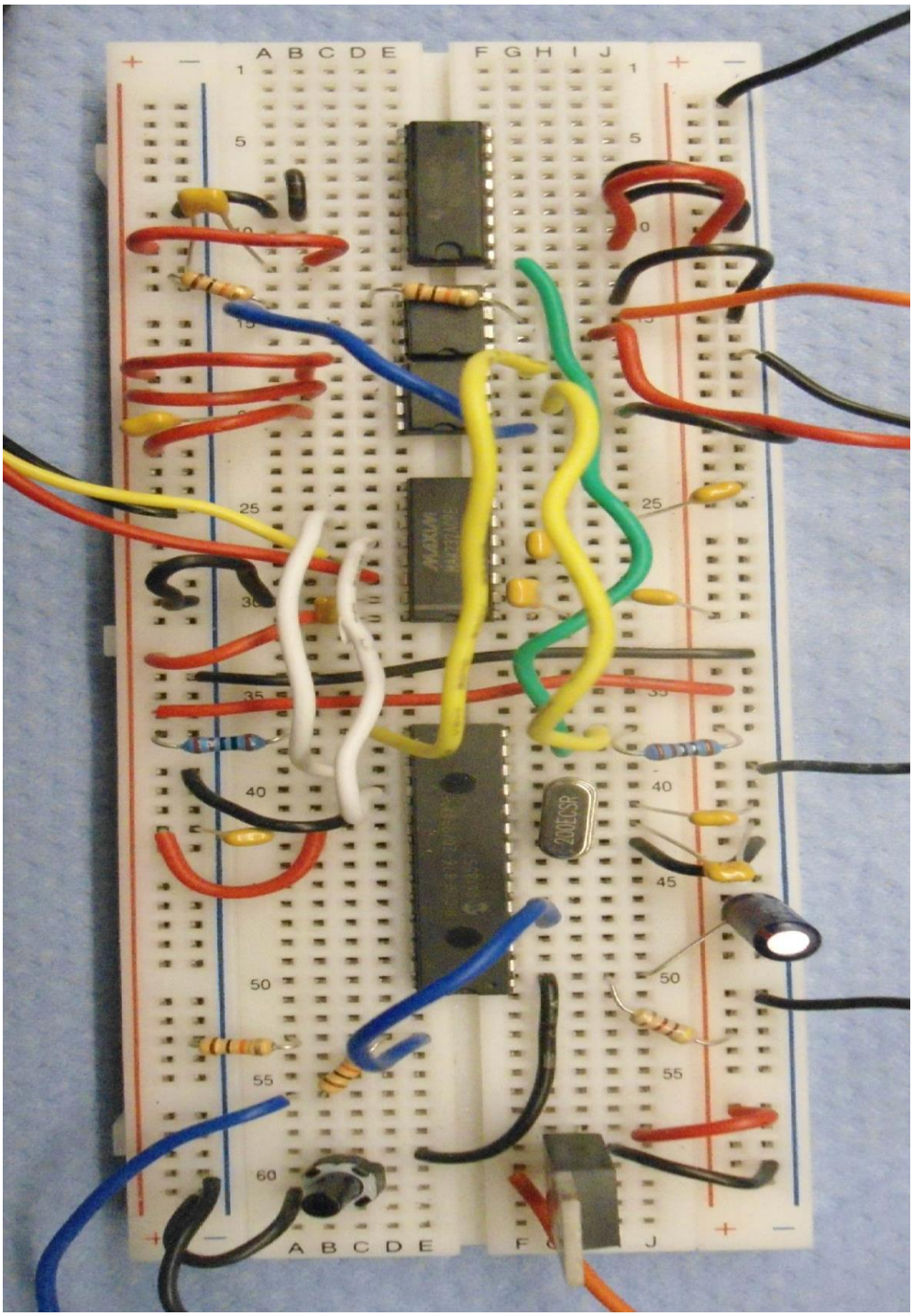
AUTOMATIC CONTROL ENGINEERING

SWITCHING CONTROLLER LAB

PURPOSE: The purpose of this lab is to give you some experience with switching control of a simple system.

PROCEDURE: Operate the setup using the PIC control code pipe.c. From the sensor signal trace estimate the amplitude $\mathbf{E_o}$ and the period $\mathbf{T_o}$ of the limit cycles generated by the following nonlinear controllers: (1) proportional with saturation (2) ideal relay switching (3) relay with deadband. When proportional is acting alone the borderline gain and period are $\mathbf{K_P}$ and $\mathbf{T_P}$. For ideal relay case, compare $\mathbf{E_o}$ with $[4Q_o]/[\pi \mathbf{K_P}]$. For relay with deadband case, compare critical deadband $\mathbf{B^*}$ with $[2Q_o]/[\pi \mathbf{K_P}]$. For each case, compare the period $\mathbf{T_o}$ with $\mathbf{T_P}$.


PROCESS
CONTROLLER
balance
throttle control
controlling element
comparing element
measuring element
bridge circuit
process temperature
continuous control
two-step control
% proportional band
external control
external set value
disturbance
internal
trigger CRO
set value
overlap
X
Y
A
B
D
Process Control Trainer 37


A B C D E
F G H I J
MAXIM
20.00 CSR

LIMIT CYCLE OSCILLATIONS

Systems controlled by nonlinear controllers often undergo finite amplitude oscillations which neither grow nor decay. It seems the gain of the controller is equal to the borderline proportional gain. One can form a gain for the controller by dividing the amplitude of the fundamental component in its output by the amplitude of its input. This gain is known as a Describing Function. Consider the ideal relay controller. Its Describing Function is

$$DF = [4Q_o/\pi]/E_o = [4Q_o]/[\pi E_o]$$

Note that small E_o oscillations have high gain and thus grow whereas large E_o oscillations have low gain and thus decay. At the limit cycle

$$\mathbf{DF} = [4Q_o]/[\pi \mathbf{E_o}] = \mathbf{K}$$

$$\mathbf{E_o} = [4Q_o]/[\pi \mathbf{K}]$$

```
/*************************************
    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=0.0;gi=0.0;gd=0.0;
      bias=6.0*s;gs=0.0;
      a=3.5*s;b=8.5*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();
}
```