Laboratorio de Diseño de Robots Móviles
Practica No. 7
Control PID

Objetivo: Controlar la velocidad de los dos motores del robot construido en las practicas anteriores usando un controlador PID
Desarrollo: Para cada uno de los siguientes apartados, realizar los programas que se piden.
Duración: 2 semanas.

1. Tomando como base el programa en C que se encuentra en el apéndice 1 varíe las constantes Kp, Ki y Kd del controlador PID y modifique el código, si es necesario, para controlar las velocidades de los motores de su robot.

Apéndice 1

```
/************************************************
*            robot_pid.c                       *
*                                              *
*            v. 1.0                            *
*                                              *
*            Este programa controla la         *
*            velocidad de dos motores          *
*            usando un control PID             *
*                                              *
*            Jesus Savage                      *
*            Ruben Anaya                       *
*            Carlos Munive                     *
*                                              *
*            FI-UNAM, Nov-2007          *
*                                              *
************************************************/

// Definiciones PIC
#include <16f877.h>
#device ADC=8
#include <stdlib.h>
#fuses HS,NOPROTECT
#use delay(clock=20000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1FFF, 0x1FFF void loader16F877(void){}

// Definicion de Constantes
#define CNT_T2 3
#define NUM_TIMES_T2 64*CNT_T2
#define NUM_SECOND 80/CNT_T2
#define SIZE_DATA 50
#define ADELANTE 0x0a

// Constantes de acondicionamiento motor izquierdo
#define Kci 0.2
#define Kci1 Kci*2.7
#define Kci2 Kci*2.2
#define Kci3 Kci*2.0
#define Kci4 Kci*1.6
#define Kci5 Kci*1.2

// Constantes de acondicionamiento motor derecho
#define Kcd             0.18
#define Kcd1 Kcd*2.7
#define Kcd2 Kcd*2.2
#define Kcd3 Kcd*2.0
#define Kcd4 Kcd*1.6
#define Kcd5 Kcd*1.2
```

```
// Constantes del control PID
#define Kp 0.7 //1.7
#define Ki 0.0 //0.35
#define Kd 0.000 //0.005

//Variables Globales
long CntRight=0,CntLeft=0;
int flag=0;
long wi=0,wd=0;
float fCntRight,fCntLeft;

// Control PID
mv_pid(){
        long  rni=0,rnd=0;
        float frni=0.0,frnd=0.0;
        float ewi=0,ewd=0;
        static float prev_rni=0,prev_rnd=0,prev_ewi=0,prev_ewd=0;
        static float prev2_ewi=0,prev2_ewd=0;
        float cntder,cntizq;

        //Acondiciona los valores de las entradas (encoders)
        if(wd<=62)cntder=wd*Kcd1;
        else if(wd<125)cntder=wd*Kcd2;
        else if(wd<250)cntder=wd*Kcd3;
        else if(wd<350)cntder=wd*Kcd4;
        else cntder=wd*Kcd5;

        if(wi<=62)cntizq=wi*Kci1;
        else if(wi<125)cntizq=wi*Kci2;
        else if(wi<250)cntizq=wi*Kci3;
        else if(wi<350)cntizq=wi*Kci4;
        else cntizq=wi*Kci5;

        cntizq=cntizq/1024.0;
        cntder=cntder/1024.0;

        // Calcula el error
        ewd=cntder-fCntRight;
        ewi=cntizq-fCntLeft;

        // Calcula el control para cada motor
        frni= prev_rni + Kp*(ewi-prev_ewi)+ Ki*(ewi-prev_ewi)+
                        Kd*(ewi-2*prev_ewi+prev2_ewi);
        frnd= prev_rnd + Kp*(ewd-prev_ewd)+ Ki*(ewd-prev_ewd)+
             Kd*(ewd-2*prev_ewd+prev2_ewd);

        // Guarda valores para la siguiente iteracion
        prev_rni=frni;
        prev_rnd=frnd;
        prev_ewi=ewi;
        prev_ewd=ewd;

        //Acondiciona los valores de las salidas
        if(frni<0)frni=0;
        else if(frni>1.0)frni=1.0;
        if(frnd<0)frnd=0;
        else if(frnd>1.0)rnd=1.0;
        rni=2048.0*frni;
        rnd=2048.0*frnd;

        // Coloca los valores para generar los PWM
        set_pwm1_duty(rni);
        set_pwm2_duty(rnd);

}

// Rutina para la interrupcion del TIMER2.
#INT_TIMER2
void wave_timer2() {
        static long k=0;
```

```
		static int i=0;

		k++;
		if(k>NUM_TIMES_T2){
			k=0;
			CntLeft=get_timer0();
			CntRight=get_timer1();
			fCntRight=CntRight/1024.0;
			fCntLeft=CntLeft/1024.0;

			// Muestra los datos
			if(flag==1){
				i++;
				if(i > SIZE_DATA)flag=0;
				printf("%f %f\n\r",fCntLeft,fCntRight);
			}
			else i=0;
			// Ejecuta el algoritmo de control PID
			mv_pid();
			// Inicializan los contadores de los temporizadores de nuevo
			set_timer0(0);
			set_timer1(0);
		}
}

/* Inicializa puertos */
inicializa_puertos(){
		// Se inicializan los pines para generar los pwms
		setup_ccp1(CCP_PWM);
		setup_ccp2(CCP_PWM);
		// Se indica la dirrecion de los motores
		output_d(ADELANTE);
}

// Inicializa interrupciones y temporizadores
inicializa_interrupciones(){
		// Se inicializan los temporizadores 0 y 1 como encoders
		set_timer1(0);
		setup_timer_1(T1_EXTERNAL|T1_DIV_BY_1);
		set_timer0(0);
		setup_timer_0(RTCC_DIV_1|RTCC_EXT_L_TO_H);
		// Se inicializa el temporizador 2 para generar el PWM
		setup_timer_2(T2_DIV_BY_1, 127, 10);
		// Se habilitan las salidas
		enable_interrupts(INT_TIMER2);
		enable_interrupts(GLOBAL);
}

/* Programa principal */
void main(){

  char linea[5];

  inicializa_interrupciones();
  inicializa_puertos();

  while(TRUE){
			// lee las velocidades de los motores
			gets(linea);
			wi=atol(linea);
			flag=1;
			gets(linea);
			wd=atol(linea);
			flag=1;
		}
}
```