Course Code		**MEE306 Microcontroller**	
Term Project		**Electric Motor Velocity Controller**	
Related Learning Outcome:		**5**	
Group Number:			
Group Member ID	**Group Member Name**	**Group Member Surname**	**Group Member Signature**
200412027	**Melik Nuri**	**Erdem**	
190412063	**Metehan**	**Sarıoğlan**	
180412035	**Hasan**	**Ünlü**	
160412014	**Mehmet Berke**	**Parlat**	

Grading Policy		
Pre-Work (20%) Necessary simulation / software application / code	Lab Work (50%) Application outcomes & properly-working circuit	Post-Lab Quiz (30%) Quiz aiming to test fulfillments of Lab Work
Pre-work Term Project Results (filled by Lab assistants)		
Pre-Work (100%)		

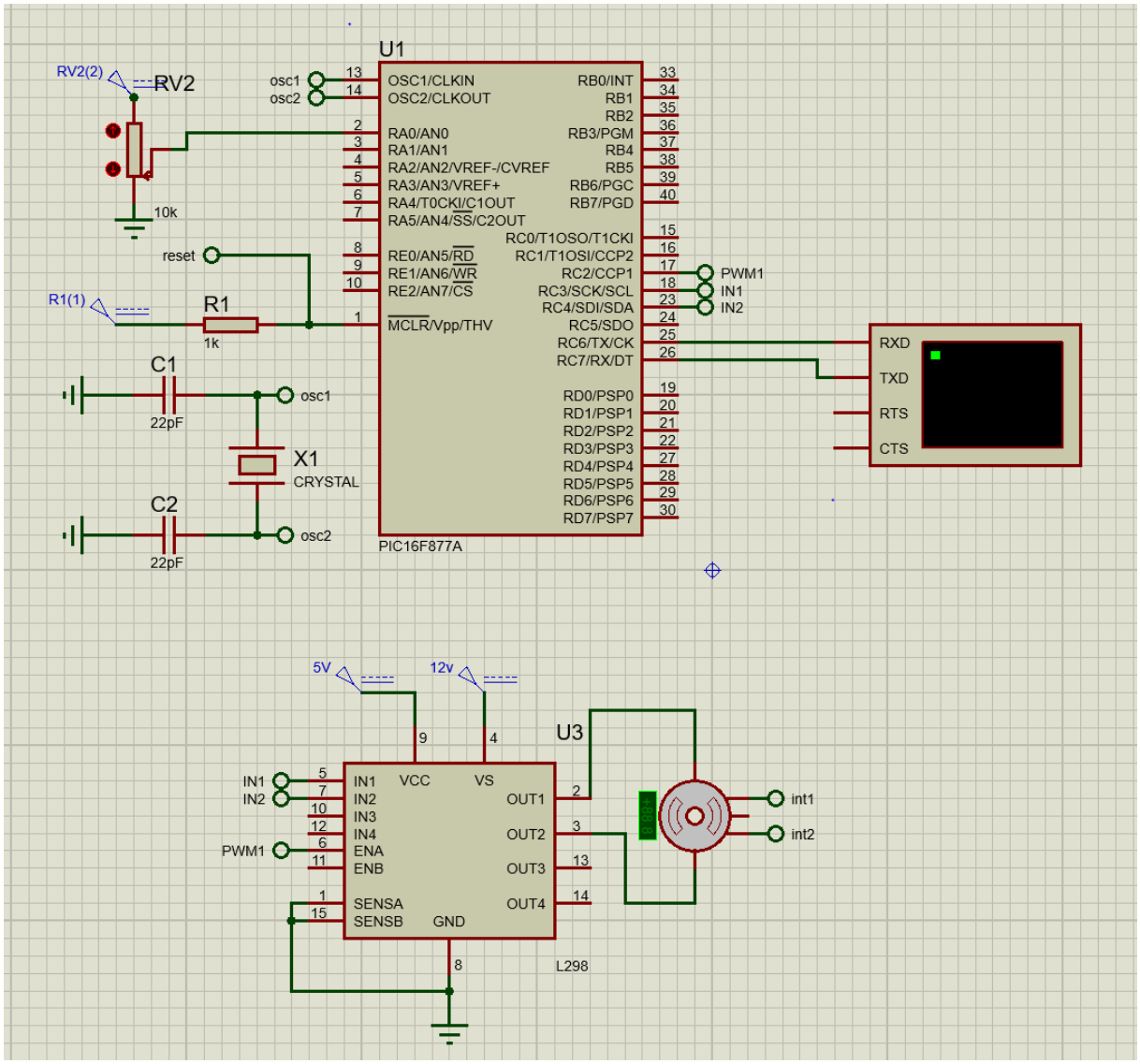
PRE- WORK TERM PROJECT - Electric Motor Velocity Controller

Introduction

Microcontrollers are essential in modern electronic control systems, enabling precise control and automation in various applications. Our project, "Electric Motor Velocity Controller," aims to implement a Proportional-Integral (PI) control system for a DC motor using a microcontroller. The primary objective is to regulate the motor's velocity with feedback from an encoder, translating theoretical concepts from the MEE306 Microcontroller course into practical applications.

This project integrates key learning outcomes from the course, such as hardware interfacing, control algorithm development, and serial communication between the microcontroller and a PC. By adjusting control parameters like controller gains (Kp and Ki) and sampling time in real-time, we observed the system's performance changes. We generated a reference velocity using a potentiometer, and the PI control algorithm minimized the error between the reference and actual velocity. This project not only tested our technical skills but also deepened our understanding of control system dynamics.

Proteus Simulation



Code

```
#include <16F877A.h>
#include <stdlib.h>
#FUSES XT, NOWDT, NOPROTECT, NOBROWNOUT, NOLVP, NOPUT, NODEBUG, NOCPD
#use delay(crystal=20000000)

#use rs232 (baud=9600,xmit=PIN_C6, rcv=PIN_C7, parity=N, stop=1)
#DEFINE IN1 PIN_C3
#DEFINE IN2 PIN_C4
int counter = 0;
char strInput[16];
unsigned long inputString;
unsigned long int revAngle = 0.0f;
unsigned long int prevAngle = 0.0f;
signed long dx_dt = 0;
int i = 0;
int x = 0;
int y = 20;
signed long int result_1;
signed long int wref;
signed long int wact;
signed long long int error;
```

```
signed long int controlout;
signed long long int total_error;
float kp = 0.1;
float ki = 0.001;
float dt;

#int_ext
void external_interrupt()
{
  revAngle++;
}

#int_timer0
void tmr_int()
{
  set_timer0(60);
  x++;
  if (x >= y)
  {
    dx_dt = (revAngle - prevAngle) * (15.79 / y);
    prevAngle = revAngle;
    total_error = ((error * dt) + (total_error));
    x = 0;
  }
}

void main()
{
  setup_psp(PSP_DISABLED);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T1_DISABLED, 0, 1);
  setup_CCP1(CCP_OFF);
  setup_CCP2(CCP_OFF);

  port_b_pullups(TRUE);
  enable_interrupts(GLOBAL);
  clear_interrupt(int_ext);
  setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);
  set_timer0(60);
  enable_interrupts(int_timer0);
  enable_interrupts(int_ext);
  setup_adc_ports(AN0_AN1_AN3);
  setup_adc(ADC_CLOCK_DIV_32);

  setup_ccp1(CCP_PWM);
  setup_timer_2(T2_DIV_BY_16, 255, 1);
  set_pwm2_duty(0);
  output_low(IN1);
  output_high(IN2);
  dt = 0.01 * y;

  while (1)
  {
    if (kbhit())
    {
      char i = getc();

      if (i == 'p' || i == 'i' || i == 's')
      {
        inputString = atol(strInput);
        if (i == 'p')
        {
          kp = inputString * 0.1;
        }
        else if (i == 'i')
        {
          ki = inputString * 0.001;
        }
        else if (i == 's')
        {
          y = inputString;
```

```
            dt = 0.01 * y;
          }
          memset(strInput, 0, sizeof(strInput));
          counter = 0;
        }
        else
        {
          strInput[counter] = i;
          counter++;
          if (counter >= sizeof(strInput))
          {
            counter = sizeof(strInput) - 1;
          }
        }
      }

      set_adc_channel(0);
      delay_us(10);
      result_1 = read_adc();
      wref = (result_1) * (9.78);
      wact = dx_dt;
      error = (wref - wact);
      controlout = (error * kp) + (total_error * ki);

      if (controlout > 1023) {
        controlout = 1023;
      }
      else if (controlout < 0) {
        controlout = 0;
      }
      set_pwm1_duty(controlout);
      delay_ms(100);
      printf("\n motor pwm :%ld", controlout);
      printf("\n actual velocity :%ld", wact);
      printf("\n reference velocity :%ld", wref);
      printf("\n kp:%f", kp);
      printf("\n ki:%.3f", ki);
      printf("\n y=%d", y);
      printf("\n sampling time:%f", dt);
      printf("\n wait for 2sec");
      delay_ms(2000);
    }
}
```

Conclusion

The Electric Motor Velocity Controller project successfully demonstrated the application of PI control theory in a practical setting. Using a microcontroller to regulate a DC motor's speed with encoder feedback, we achieved precise control and real-time parameter adjustment. This hands-on experience highlighted the challenges of implementing control systems, such as tuning control gains and managing sampling times.

We overcame technical challenges like establishing stable serial communication, ensuring accurate encoder feedback, and fine-tuning the PI control algorithm. Observing the motor's behavior under different control settings enhanced our understanding of theoretical concepts and their practical implications.

Overall, this project reinforced the importance of integrating theoretical knowledge with practical skills. It provided us with a solid foundation in microcontroller-based control applications, preparing us for future endeavors in electronic control systems. The successful completion of this project met the course objectives and equipped us with the confidence and competence to tackle similar challenges in our professional careers.