FACULTY OF ENGINEERING AND ARCHITECTURE DEPARTMENT OF MECHATRONICS ENGINEERING

MICROCONTROLLER

TERM PROJECT

Burak Berke Çanaklı

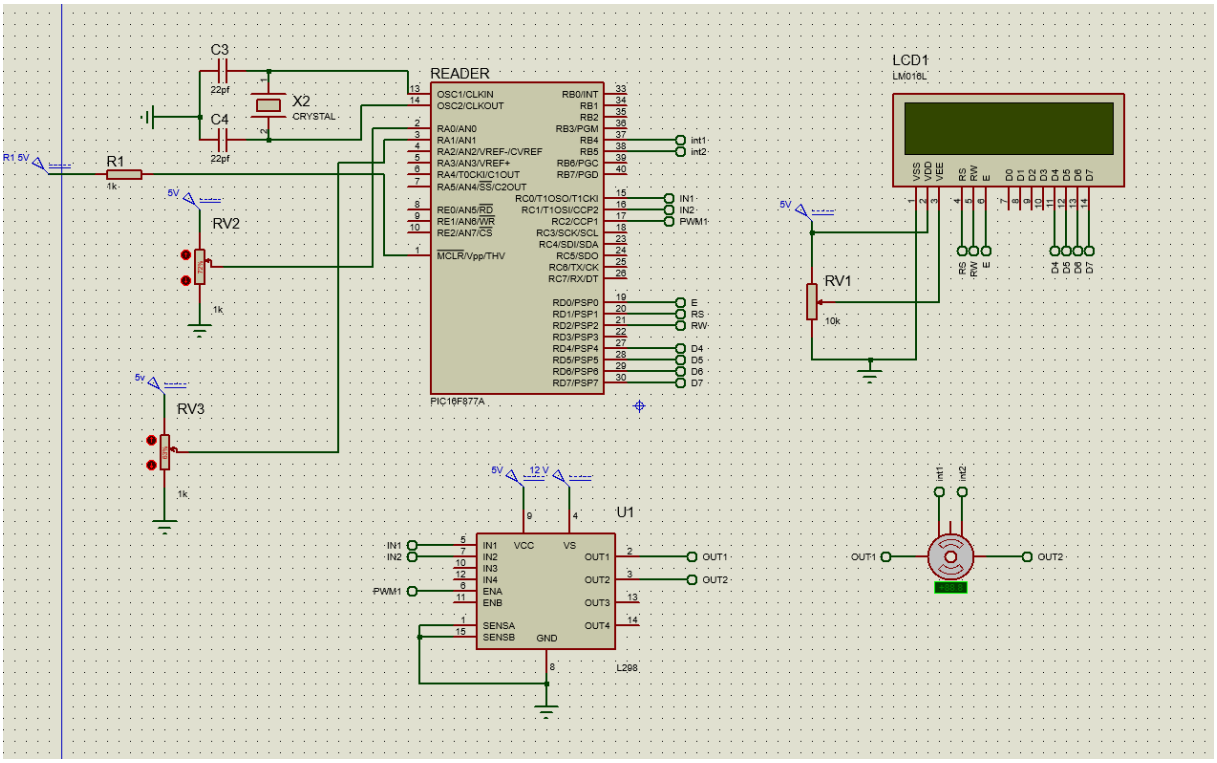
160412023

OVERVIEW OF THE PROJECT

The purpose of the project is to control the position of a DC motor with inputs given from the user via the PIC microcontroller.The inputs are a reference angle and the proportional gain parameter Kp.Also, the controller does not execute another code and process until the input value Kp is received. Reference angle of the second input is obtained from an analog voltage divider reading.It is converted using the ADC converter module of the microcontroller. Another input then used to calculate the reference angle and instantaneous position is based on the motor encoder input.The position error is calculated and the starting position of the motor is always 0 degrees. Finally, the shorter direction to be turned is calculated. After all calculations and readings.We output a pwm signal to the motor enable pin to set its speed with a certain duty cycle based on a function of Kp and error. We set the motor driver's input pins to high or low according to the shorter turning direction calculated.When there are no more errors, the motor has reached the desired position and the motor is stopped.

PROCEDURE OF THE PROJECT

- **ELECTRICAL CIRCUIT**



- **Code**

```
#include <16F877A.h>
#device adc=10 //setting 10 bit converter resolution
#FUSES XT, NOWDT, NOPROTECT, NOBROWNOUT, NOLVP, NOPUT, NODEBUG, NOCPD
#use delay(crystal=4000000)
#include <lcd.c>
int8 encoderRead; //variable that stores encoder reading
int8 last_read; //variable to check changing encoder value
signed int8 quad = 0; //variable to change position variable
#INT_RB // RB port interrupt on change


void RB_IOC_ISR(void)              //interrupt function for every encoder read
{
 clear_interrupt(INT_RB);        // clearing the interrupt to get out of the function
 encoderRead = input_b() & 0x30; //reading encoder if there is a input
 if(encoderRead == last_read)    // check if the position is changed
  return;                        //if there is no change return nothing

 if(bit_test(encoderRead, 4) == bit_test(last_read, 5)) //to determine the rotation
  quad -= 1;         //if the rotation is ccw add one to quad variable
 else
  quad += 1;      //if the rotation is cw subtract one from quad variable
 last_read = encoderRead; //for using less memory, only previous values saved
}

int16 EncoderGet(void) // function to find the change in pulse number or angle
{
  signed int16 value = 0;  //reset the change to zero
```

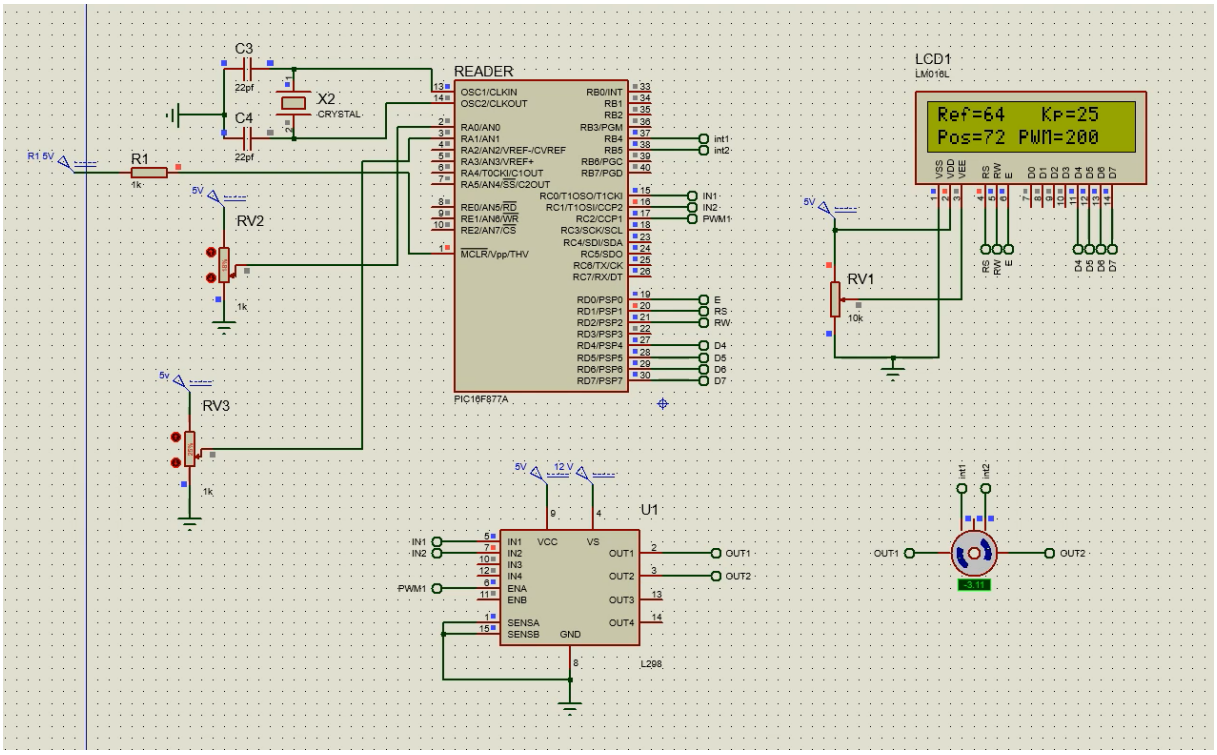
```
  while(quad >= 4){       //if quad is 4, 1 pulse period is done in ccw direction
    value += 1;           //add 1 to value to store change in the number of pulses or angles
    quad -= 4;            //reset quad to zero

  }
  while(quad <= -4){      //if quad is -4, 1 pulse period is done in cw direction

    value -= 1;           //subtract 1 from value to store change in the number of pulses of angles
    quad += 4;            // reset quad to zero
  }
  return value;           // function returns the value of change in the angle
}


int16 PWM(int16 a,int16 b) //function for calculating PWM value
{
 signed int16 value;
 value=a*b; //will get error and Kp values as inputs
 //bounding the PWM value
 if (value>1023)
  return 1023;
 else if (value<0)
  return 0;
 else
  return value;
}


void main()
{
```

```
//defining variables

 int16 analog_proportional_controller;
 int16 analog_refangle;
 int16 error = 0;
 int16 realPosition = 0;
 int kp = 10;
 int V = 60;

 lcd_init();
 delay_ms(10);
 set_tris_c(0x00000000); //setting the outputs
 output_c(0x00);

 setup_ccp1(CCP_PWM); //setuping the PWM
 setup_timer_2(T2_DIV_BY_16, 255, 1);     //timer2 setup
 setup_adc_ports(AN0_AN1_AN3); //setuping the analog reading
 setup_adc(ADC_CLOCK_DIV_32);    //enable ADC and set clock for ADC
 enable_interrupts(INT_RB);      //enable all interrupts

 enable_interrupts(GLOBAL);
 clear_interrupt(INT_RB);



 while(True)          //infinite loop
 {
  //analog reading
  set_adc_channel(0);        // next analog reading will be from channel 0
  delay_us(10);              //allow time after channel selection and reading
 analog_refangle = read_adc();    //start and read A/D then convert it to the reference angle
value
```

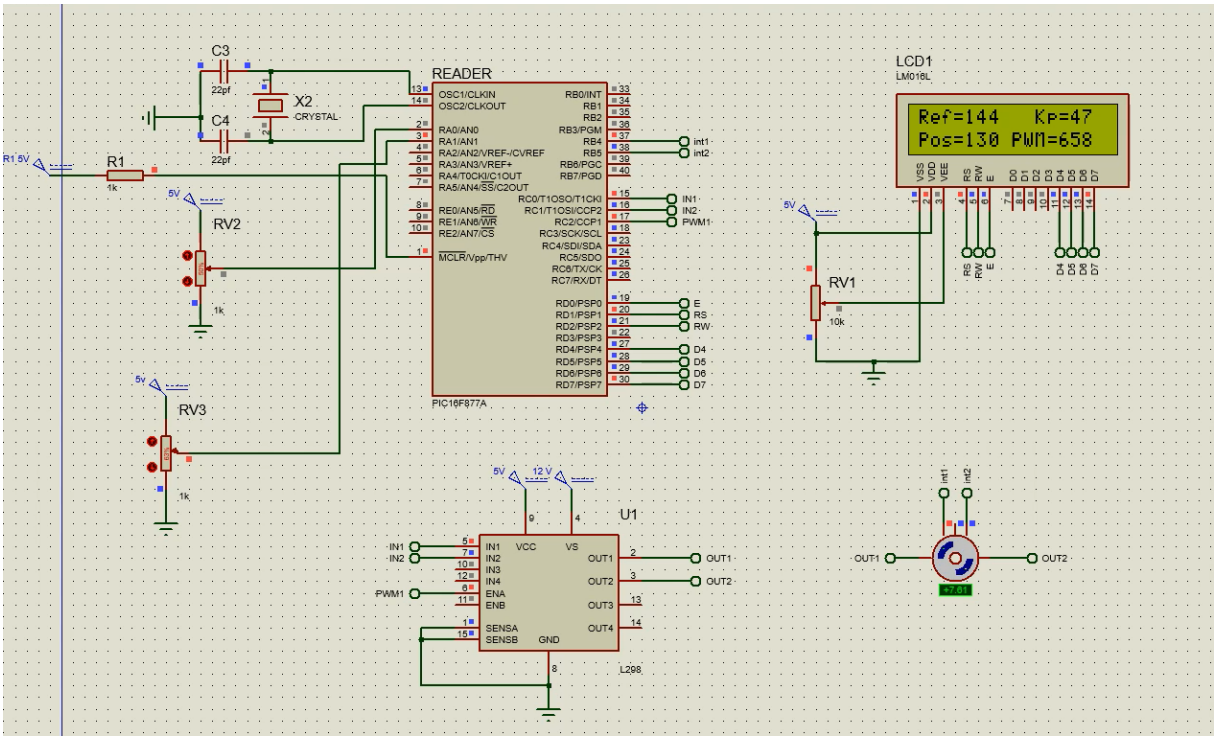
```
 int16 reference_angle =((float)analog_refangle*250/1023)+20;
 set_adc_channel(1);
 delay_us(10);
 analog_proportional_controller = read_adc();
 int16 control_gain = kp + ((float)analog_proportional_controller/1023)*V;

 long change = EncoderGet(); //update change in position
 if(change)
 {
  realPosition += change; //update the position information
 }
 float rev = realPosition/360.0f;
 int16 angle = ((int16)(rev*360))%360;

 if(angle<=reference_angle)     //if 1 rev. in ccw direction is done reset position to 0
  {
   error = reference_angle-angle;
   set_pwm1_duty(PWM(control_gain,error)); //set duty cyle value to change the motor speed
   output_high(PIN_C0);
   output_low(PIN_C1);
  }
 if(angle>reference_angle)        //if 1 rev. in cw direction is done reset position to 0
  {
   error = angle-reference_angle;     //update the error
   set_pwm1_duty(PWM(control_gain,error));   //set duty cyle value to change the motor speed
   output_low(PIN_C0);
   output_high(PIN_C1);
  }
```

```
  //printing  on LCD
  printf(lcd_putc, "\fRef=%lu   ", reference_angle);
  printf(lcd_putc, "Kp=%lu", control_gain);
  printf(lcd_putc, "\nPos=%lu ", realPosition);
  printf(lcd_putc, "PWM=%lu",PWM(control_gain,error));
  delay_ms(10);
  }
}
```

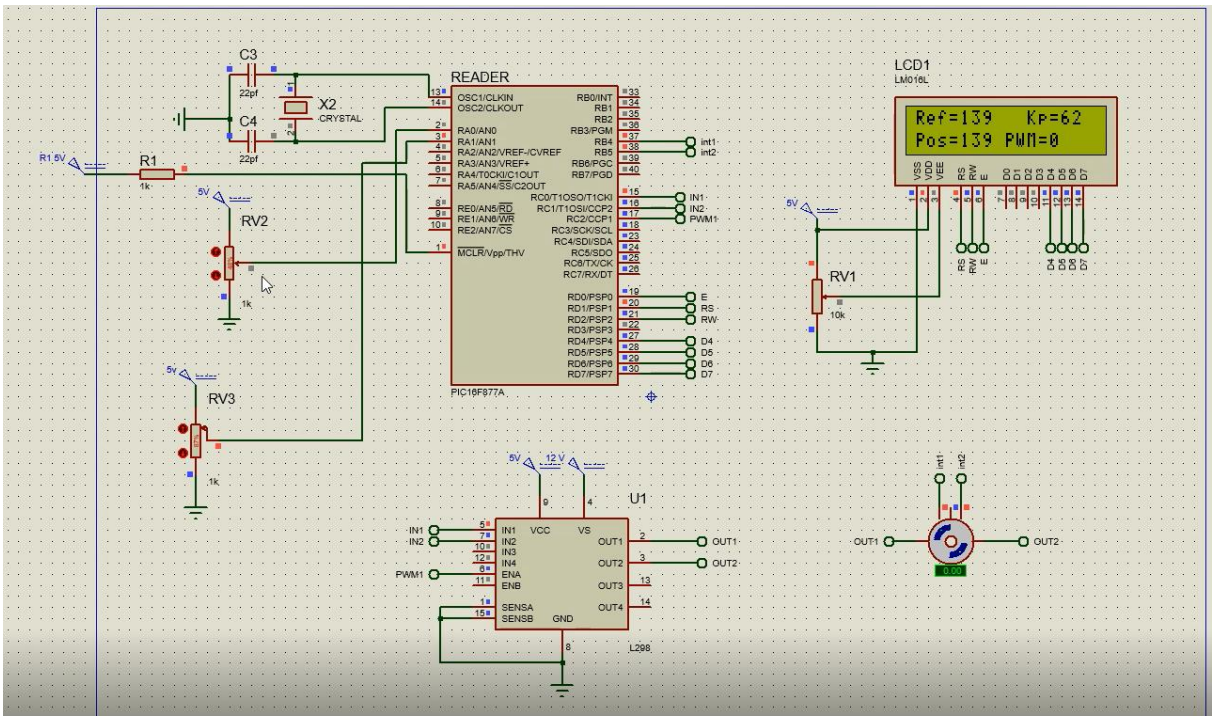
- **Simulation Snapshot-1**



- **Simulation Snapshot-2**



- **Simulation Snapshot-3**

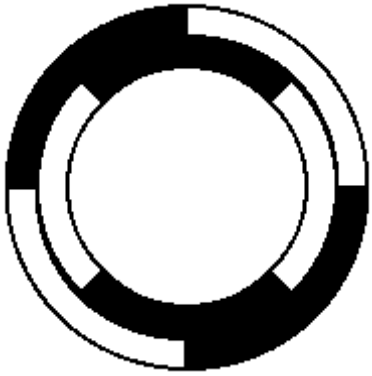


CONCLUSION

Using PIC, ı can change the position of the motor with the basic voltage input in real time. I can do this thanks to various modules in the microcontroller.I learned and used the ADC In addition, information on how to use the applications of coding and tried to learn. After carrying out the serial communication with other PIC only once, the main codes starts to execute. Firstly ADC module reads the analog input which is converted into reference angle value between 20-270 degrees with this function:

$$Q_{ref} = analogValue * \frac{250}{1023} + 20$$

Then I get the encoder readings and with appropriate codes convert it into position information :



I can use the help of the Proteus. In the picture ,there are 2 pulses for each sensor. I have 360 degree and that means each pulse change is 1 degree change. Also, it is clear that at every one fourth of a pulse , encoder information changes. I store these changes in the quad variable. Every four quad means one pulse and one degree angle. For the rotation detection, I used characterisitc changes in encoder value for each rotation with an elegant function.According to this encoder value the error is calculated and based on the error I set the duty cycle of the PWM value with this function :

$pwm\_duty\_cycle = Kp * error$

As the error gets smaller, it means that the motor turns slower and leads to a more stable system.In addition, the motor will rotate more rapidly increase Kp.Since the motor is almost always moving at maximum speed, when the time error is zero and the speed is set to zero, it would be rotated to pass the reference value.

I used Second active potentiometer in order to alter the value of the P-controller gain. I defined A default value of Kp and according to potentiometer value I changed as below;

$$K_p = K_p{}^{default} + analogValue * \frac{V}{1023}$$