EXPERIMENT V

Fall - 2022/2023

MKT3811 – Microprocessors and Programming

Lab 5 Report

Submitted By: Göktuğ Can Şimay

Lab Partners: Ali Doğan, Basel Hadri

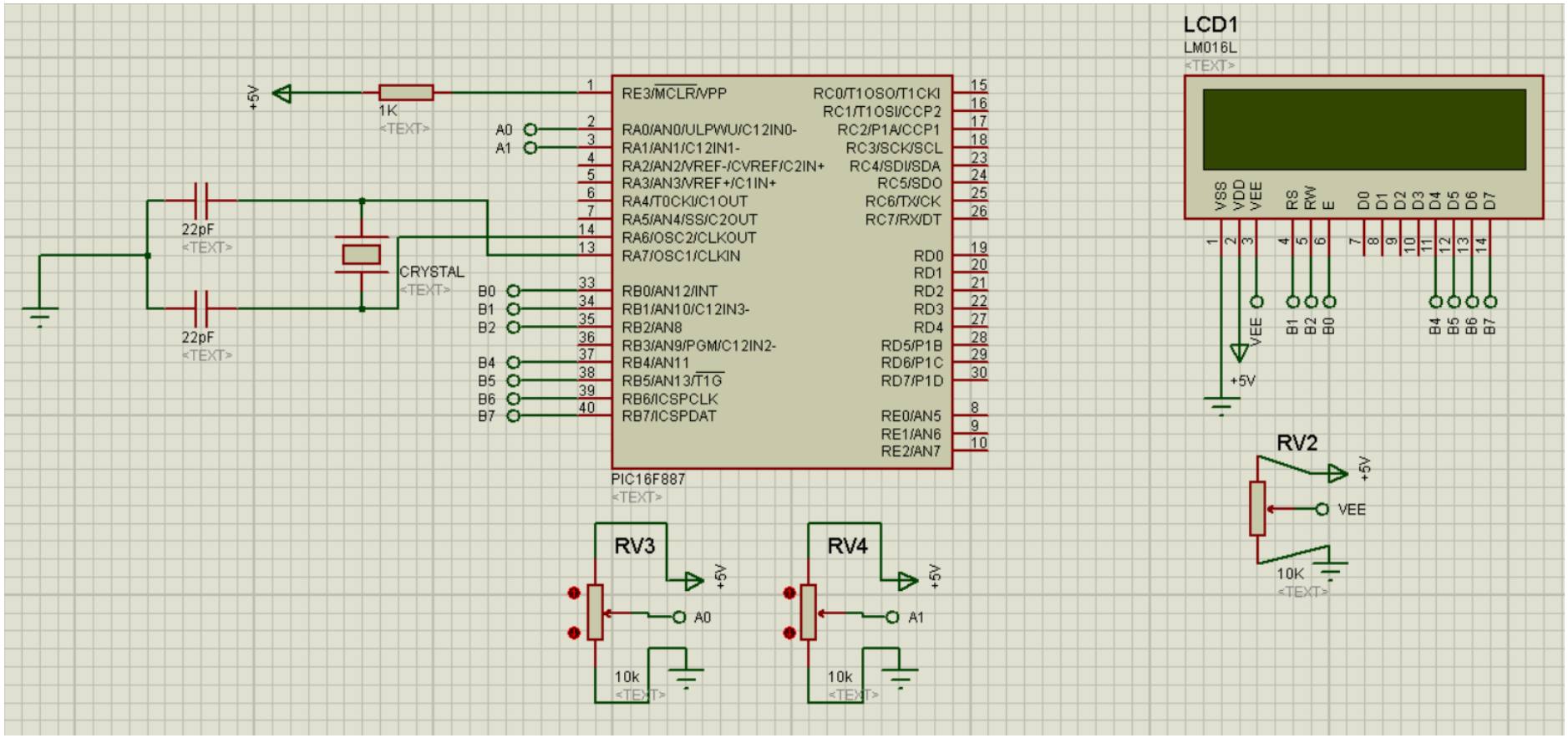
Group Number: 22

Student ID: 22067606

Date: 08.12.2022

Descriptions:

The purpose of this experiment is to implement an ADC process that occurs for each 2 seconds using a Timer in C language. ADC readings coming from 2 potentiometers should be made every 2 seconds by using a Timer and the first reading should be on the first line and the second reading should be on the second line of the LCD screen.



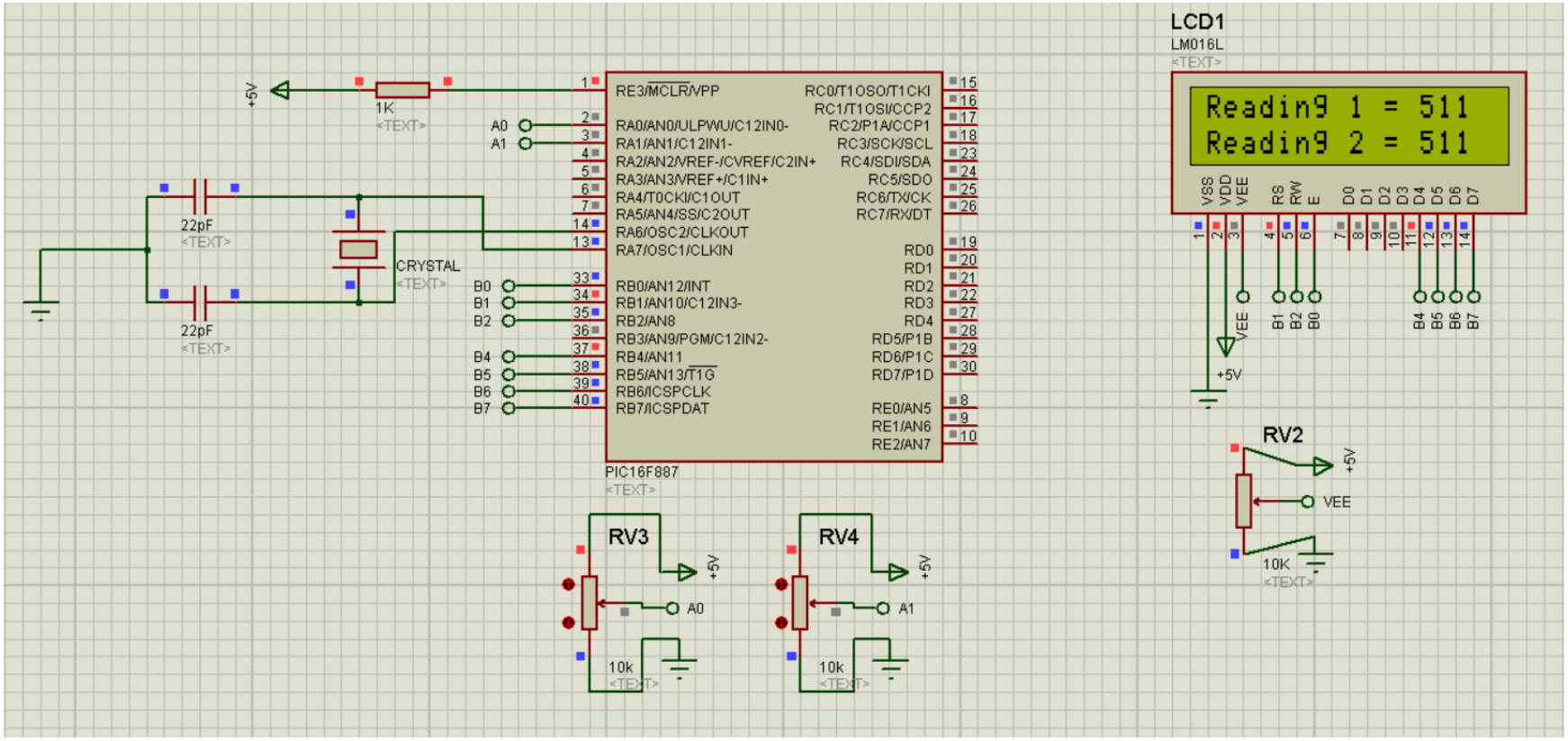
Proteus Schematic Design

As you can see, I used PIC16F887 microcontroller. I supplied +5V power to the Vpp terminal. Since I want to use an external oscillator, I connected my oscillator to the 13th and 14th inputs via capacitors.
Görüldüğü üzere LED ve potansiyometre pinlerini belirledim.

```
//lab05 ADC Reading
/*   fout = fclk/(4*prescaler*(256-TMR0)*count)
     for prescaler = 1 : 256 and TMR0 = 0
     if need is 2 sec delay
     Tout = 1/fout = 1/0.5Hz = 2sec
     count  =  4Mhz/(4*256*(256-0)*0.5Hz)=30.51=30  counts(approxi)
with mini delays I assumed it is 30 */
#include <16f887.h>
#device adc=10 //10 bitlik ADC kullanımı
#fuses XT,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,PUT,NOWRT,NODEBUG,NOCPD
/*Konfigürasyon ayarları */
#use delay (clock=4M) /*4 Mhz harici osilatör kullanacağız. */
#define use_portb_lcd TRUE
#include <lcd.c>
unsigned int16 result_1=0; //İlk sonuç
unsigned int16 result_2=0; //İkinci sonuç
unsigned int16 sayac=0; //İkinci sonuç
//TIMER0 Interrupt Service Routine Kodu
#INT_TIMER0
void  TIMER0_isr(void) {
   sayac++;
   if(sayac>30) {
      sayac=0;
      lcd_init(); /* LCD fonksiyonlarını çağırmak için. */
      delay_ms(5);
      setup_adc_ports(sAN0|sAN1);  /*  AN0  ve  AN1  pinleri  analog
input oldu. */
      setup_adc(ADC_CLOCK_DIV_32);   /*   ADC   clock   frekansı
Fosilator/32 oldu. */
```

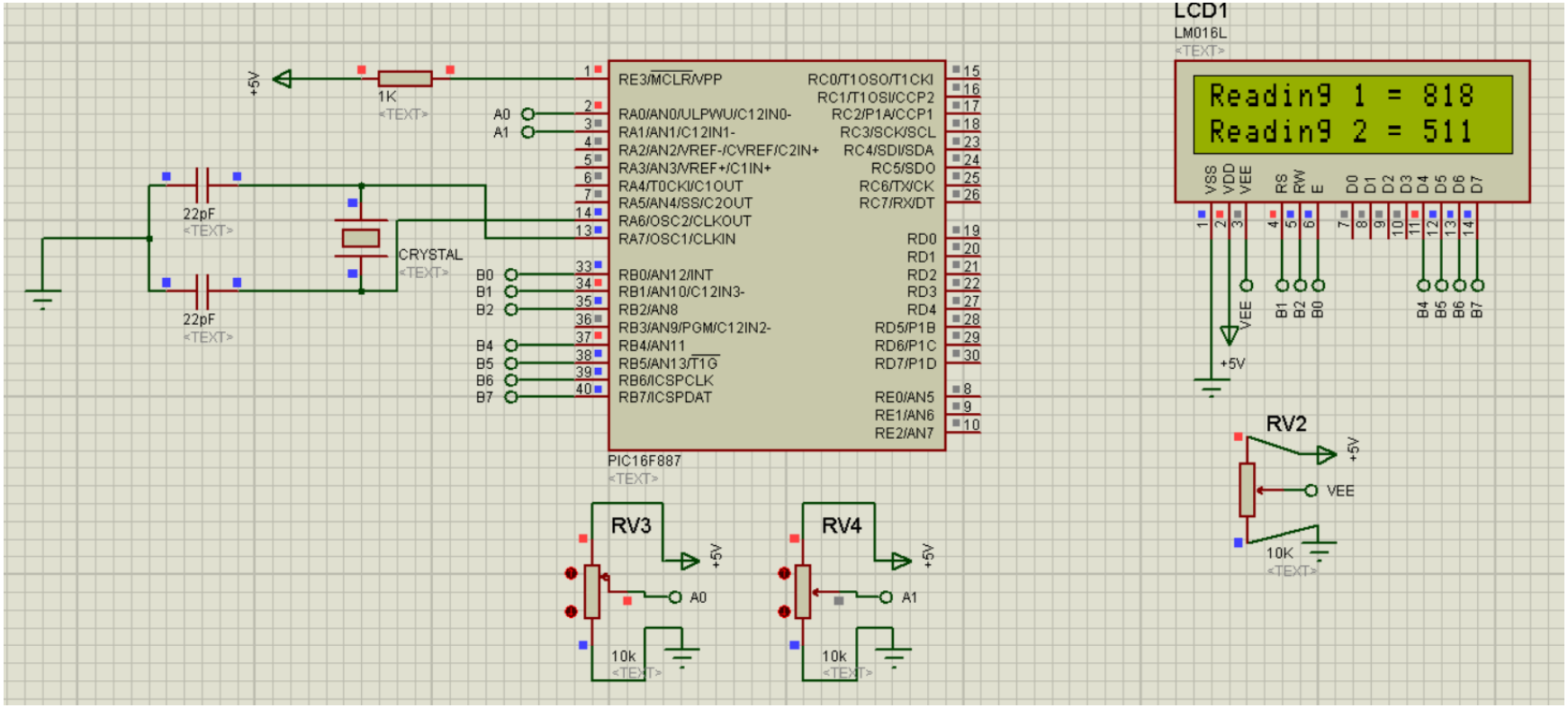
```
            set_adc_channel(0); /* AN0 kanalındaki sinyale ADC uygula.
*/
            delay_ms(5);
            result_1=read_adc(); /* Okunan ADC ilk sonuca atandı. */
            printf(lcd_putc,"\fReading  1  =  %lu",result_1);  /*  İlk
sonucumu birinci satıra yazdırdım. */
            set_adc_channel(1); /* AN1 kanalındaki sinyale ADC uygula.
*/
            delay_ms(5);
            result_2=read_adc(); /* Okunan ADC ikinci sonuca atandı. */
            printf(lcd_putc,"\nReading  2  =  %lu",result_2);  /*  İkinci
sonucumu alt satıra yazdırdım. */
      }
         set_timer0(61); // Timer0 preload value
         clear_interrupt(INT_TIMER0); // Clear Timer0 overflow bit
      }


      void main () {
         setup_oscillator(OSC_4MHZ); // Set the internal oscillator to
4MHz
         clear_interrupt(INT_TIMER0); // Clear Timer0 overflow bit
         enable_interrupts(INT_TIMER0); // Enable Timer0 interrupt
         enable_interrupts(GLOBAL); // Enable global interrupts
         setup_timer_0(T0_INTERNAL    |    T0_DIV_256);    //    Timer0
configuration: internal clock source + 256 prescaler
         set_timer0(61); // Timer0 preload value


         lcd_init(); /* LCD fonksiyonlarını çağırmak için. */
         setup_oscillator(OSC_4MHZ);
         setup_adc_ports(sAN0|sAN1); /* AN0 ve AN1 pinleri analog input
oldu. */
```

```
         setup_adc(ADC_CLOCK_DIV_32); /* ADC clock frekansı Fosilator/32
oldu. */
         while(TRUE);
      }
```



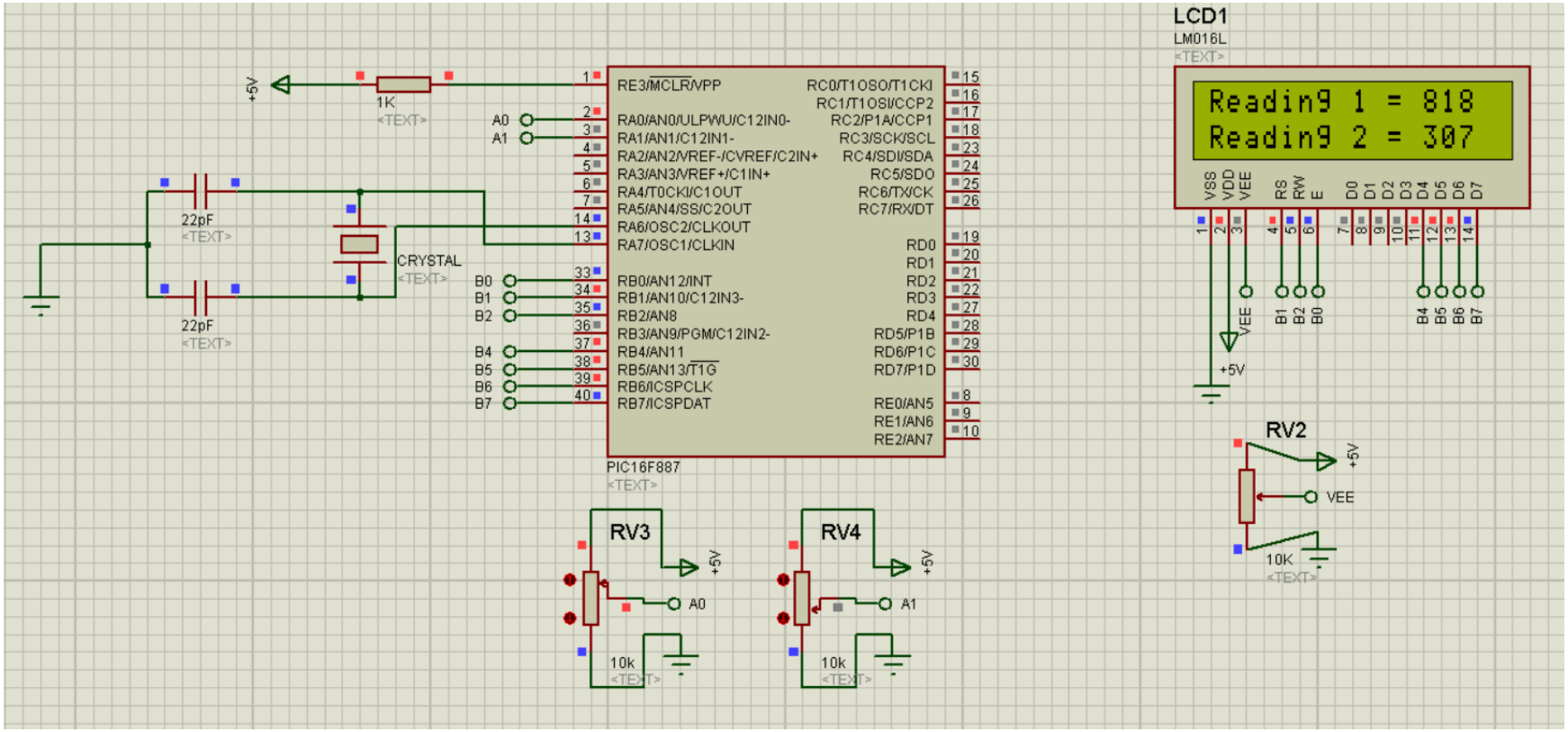
Proteus Simulation Part 1

Kullandığım Timer modülü ile her 2 saniyede bir 2 potansiyometreden ayrı ayrı analog okuma yapıyorum. Değerleri LCD üzerine yazdırdım.



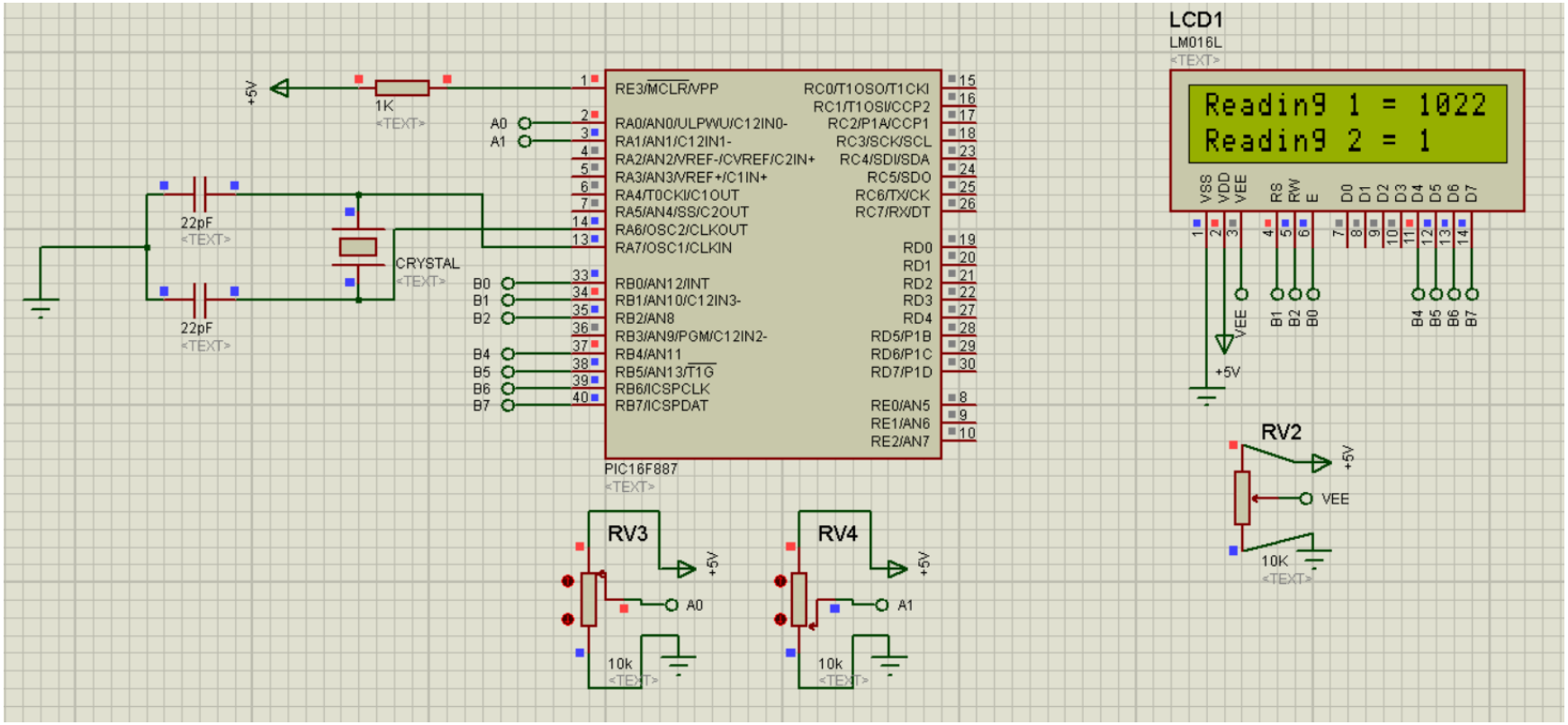
Proteus Simulation Part 2

İlk okuma için potansiyometre ayarını değiştirdiğim için 2 saniye sonraki görselde yeni değeri yazdırdı.



Proteus Simulation Part 3

İkinci okuma için potansiyometre ayarını değiştirdiğim için 2 saniye sonraki görselde yeni değeri yazdırdı.



Proteus Simulation Part 4

En sonunda iki potansiyometreyi de birbirine zıt şekilde ayarladım. (max ve min) 2 saniye sonraki okumada analog değerleri LCD üzerine yazdırıldı.