EXPERIMENT VI

Fall - 2022/2023

MKT3811 – Microprocessors and Programming

Lab 6 Report

Submitted By: Göktuğ Can Şimay

Lab Partners: Ali Doğan, Basel Hadri

Group Number: 22

Student ID: 22067606

Date: 13.12.2022

Descriptions:

The purpose of this experiment is to realize the speed and direction control of a DC motor in C language.

The steps to follow are:

• When the "Increase" button is pressed, the % Duty Cycle of the generated PWM signal will

increase. Therefore, the rotational speed of the DC motor should also increase.

• When the "Decrease" button is pressed, the % Duty Cycle of the generated PWM signal will

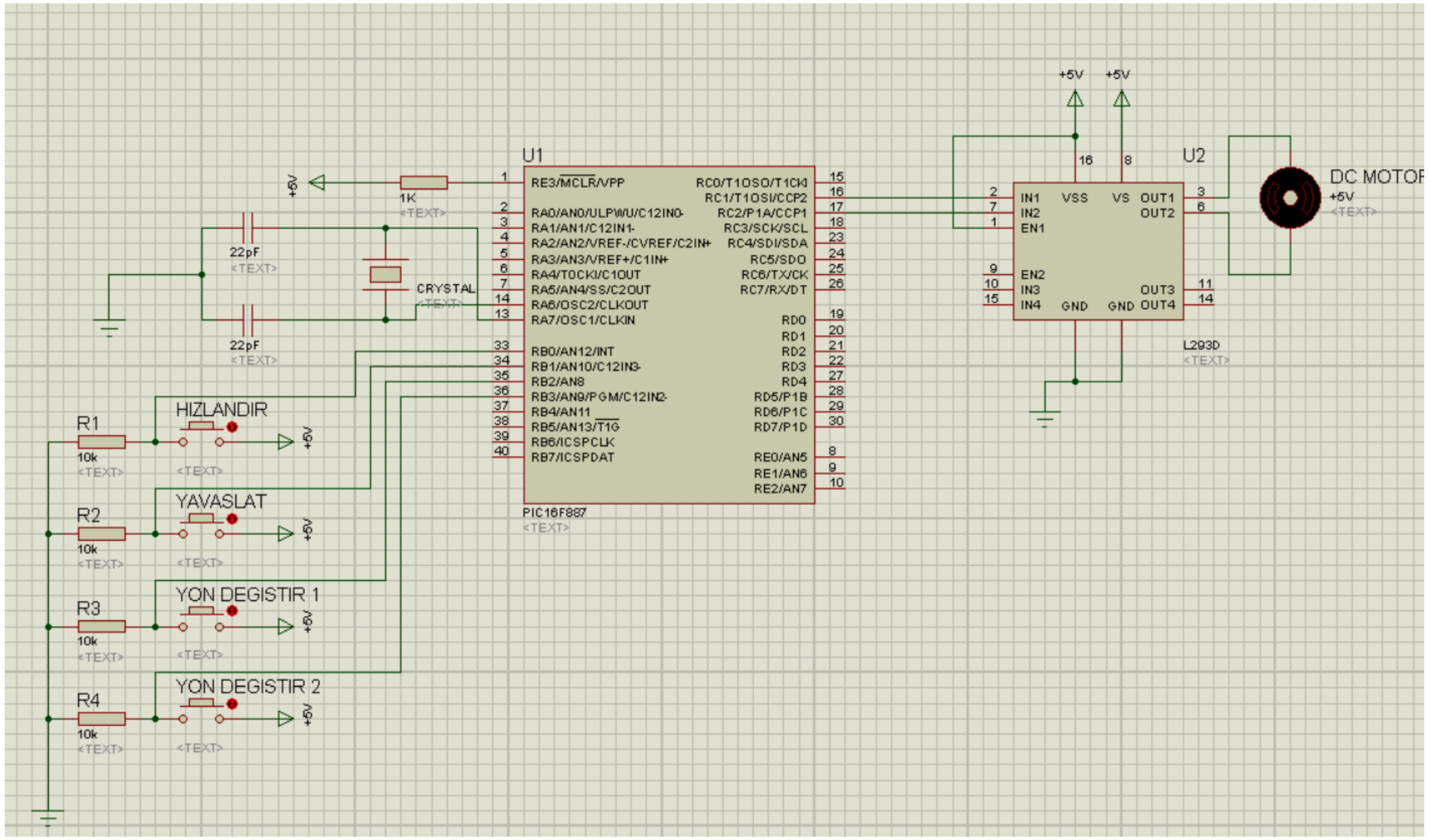
decrease. Therefore, the rotational speed of the DC motor should also decrease.

• The % Duty Cycle must be a maximum of 100% and a minimum of 0% (Limitation can be made

with the if (...) command).

• If the "Change Direction" button is pressed an odd number of times, the DC motor should turn

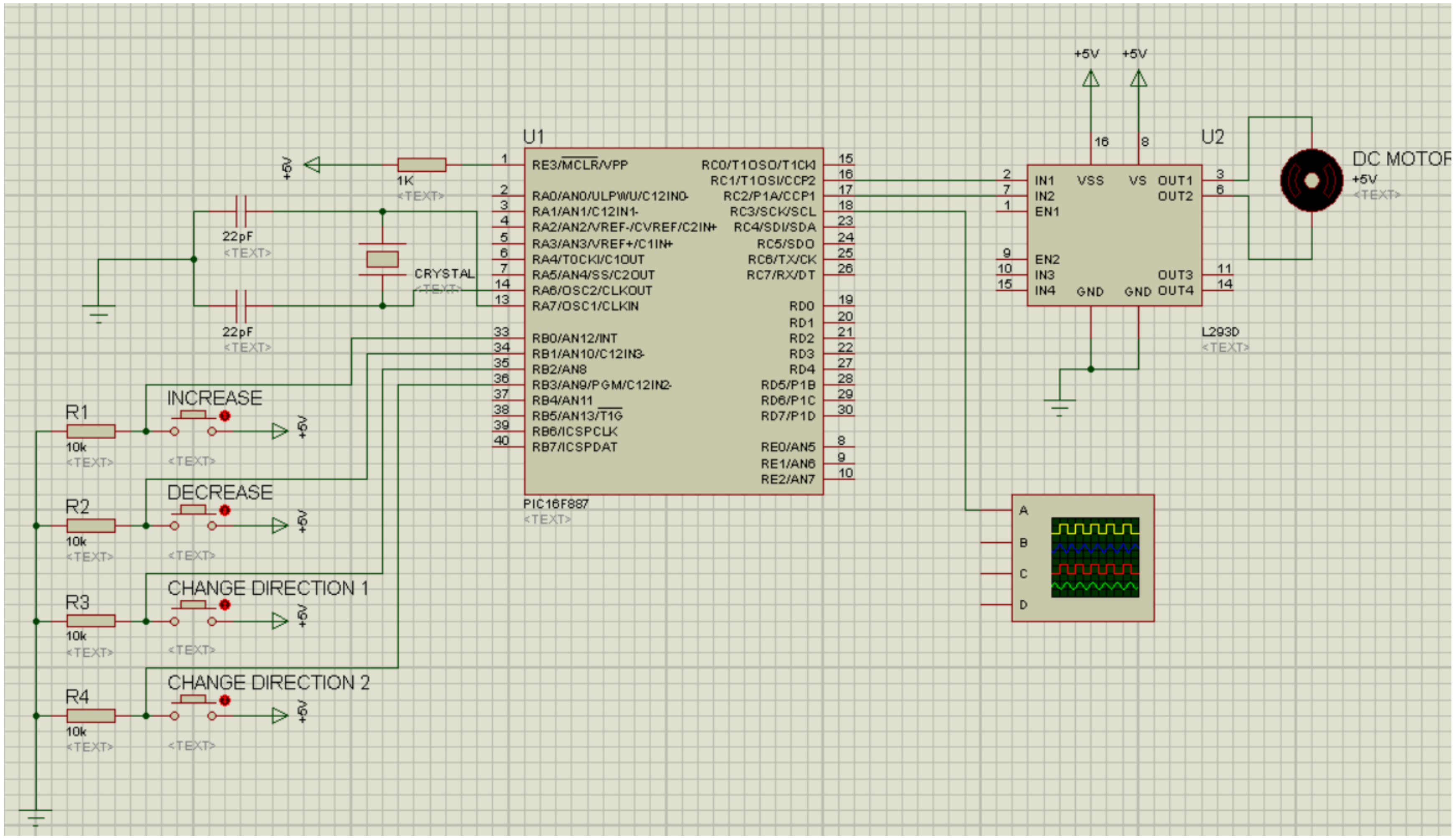
CW, and if the button is pressed an even number of times, the DC motor should rotate CCW.



Proteus Schematic For Real Life Application

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.

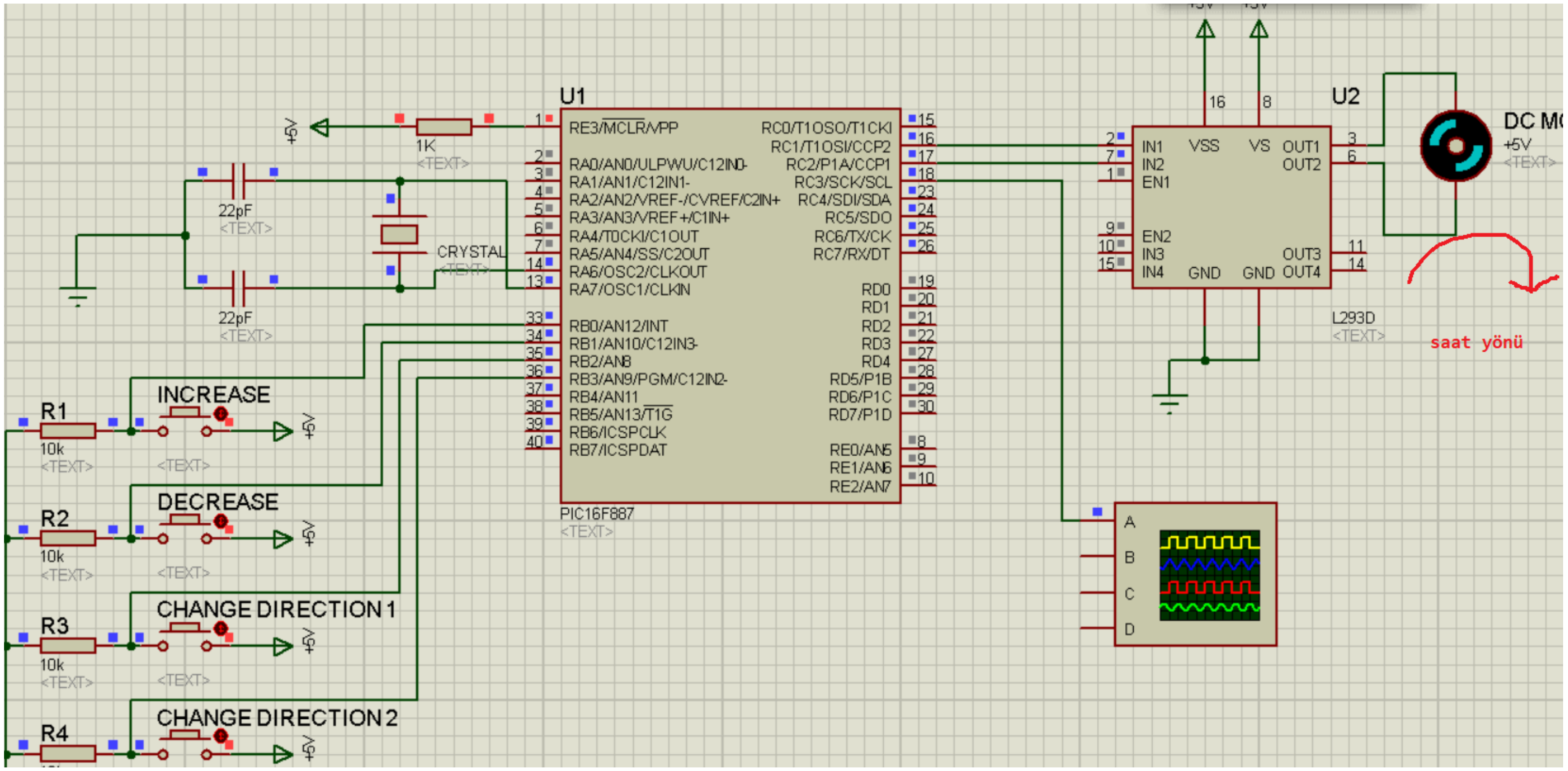
Motor sürücümün ve butonlarımın bağlantılarını Görüldüğü gibi yaptım. Gerçek hayatta osiloskop ile inceleme yapmayacağımız için onu diğer şematikte çizdim.



Proteus Schematic Design For Booklet

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. ***Burada butonları istendiği gibi isimlendirdim ve osiloskop bağlantısını hazırladım.***

```
      //lab06_code -- Her kodu satır satır yorum ile açıklamaya çalıştım.
      // DC motor control using PIC16F887 and L293D CCS C code
      #include <16F887.h>
      #fuses HS,NOWDT,PUT,NOPROTECT
      #use delay(clock = 8000000)
      #use fast_io(C)
      #use fast_io(B)
      #use pwm(output = PIN_C1, output = PIN_C2, timer = 2, frequency =
500Hz, duty = 0)
      unsigned int8 i = 1;
      void main() {
        port_b_pullups(TRUE);                 // Enable PORTB pull-ups
        output_c(0);                          // PORTC initial state
        set_tris_c(0);                    // All PORTC pins are configured
as outputs
        output_b(0);                          // PORTB initial state
        set_tris_b(0x0F);                     // Configure RB0 to RB3 as
inputs
        setup_timer_2(T2_DIV_BY_16, 255, 1);
        pwm_off();                            // Turn off all pwm outputs
        while(TRUE) {
          if(input(PIN_B0) == 0) {            // If RB0 button pressed
            i++;                             // Increment i by 1 (i = i +
1)
            if(i > 99) {
              i = 100;
      }
            pwm_set_duty_percent(i * 10UL);   // Duty cycle change in
tenths %
            delay_ms(100);                    // Wait 100ms
      }
          if(input(PIN_B1) == 0) {           // If RB1 button pressed
            i--;                              // Decrement i by 1 (i = i
- 1)
               if(i < 1) {
                  i = 1;
      }
            pwm_set_duty_percent(i  *  10UL);   //  Duty  cycle  change  in
tenths %
            delay_ms(100);                   // Wait 100ms
      }
          if(input(PIN_B2) == 0) {           // If RB2 button pressed
              pwm_off();                  // Turn off pwm for both outputs
              output_c(0);                   // PORTC pins low
              delay_ms(100);                 // Wait 100ms
              pwm_on(PIN_C1);                // Turn pwm on at RC1
              delay_ms(100);
      }
          if(input(PIN_B3) == 0) {           // If RB3 button pressed
              pwm_off();                        // Turn off pwm for both
outputs
              output_c(0);                   // PORTC pins low
              delay_ms(100);                 // Wait 100ms
              pwm_on(PIN_C2);                // Turn PWM on at RC2
              delay_ms(100);
      }
      }
      }
```



Proteus Simulation

Simülasyonlarda fotoğraflarında motorun hareketini gözlemek mümkün olmadığı için yalnızca bir görsel ekledim.