MEK303
MİKROİŞLEMCİLER

Potansiyometre ile PWM Sinyali Kullanılarak DC Motor Sürme

AHMET CEYHUN BİLİR
Sakarya Uygulumalı Bilimler Üniversitesi Mekatronik Mühendisliği
3.Sınıf

Ders Koordinatörü: Dr. Öğr. Üyesi Gökhan ATALI

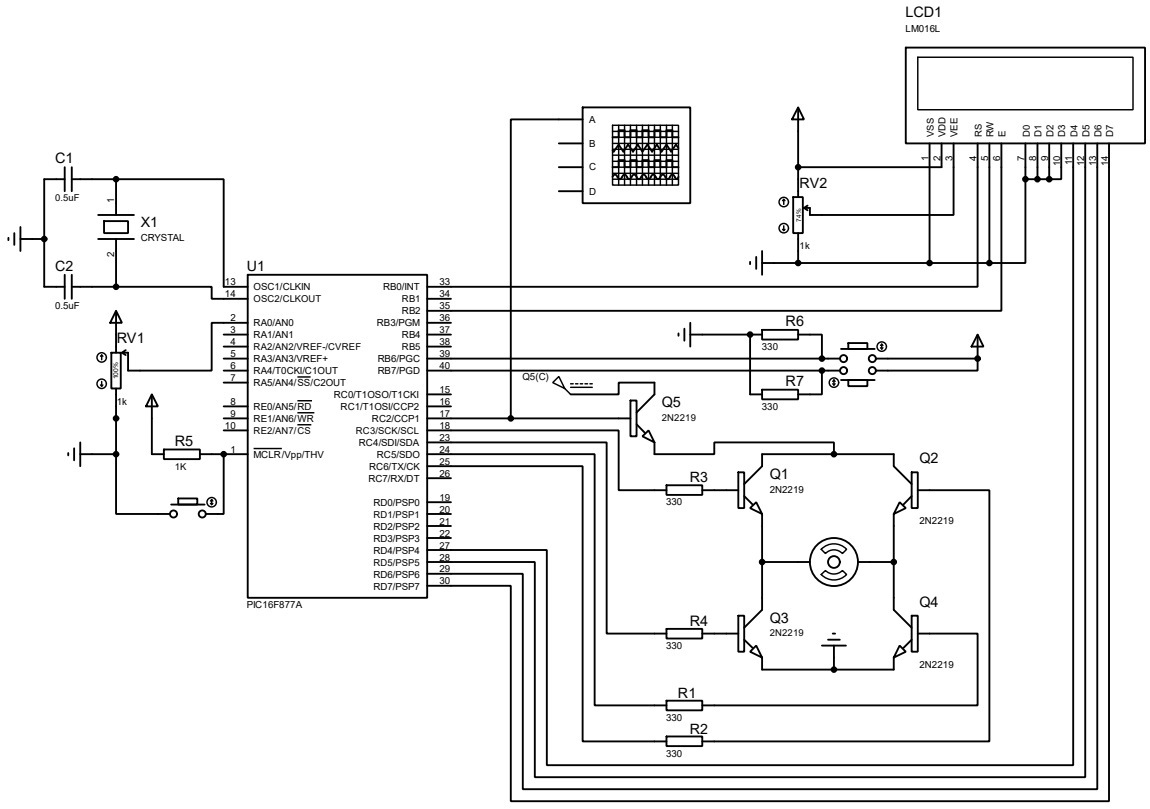
İçindekiler

1-Çalışma Sunumu

Bu çalışmada potansiyometre ile PWM sinyali kullanılarak DC motor sürme uygulaması yapılmıştır. Bu uygulama aynı zamanda DC motorun yön kontrolünü ve hız kontrolünü içermektedir. Çalışma simülasyon ortamında geliştirilmiş ve uygulamalar yine simüsyon ortamı içerisinde uygulanmıştır. Simülasyon ortamı olarak Proteus, PIC16F877A mikroişlemcisi IDE'si olarak MPLAB kullanılmıştır.

Çalışma içerisinden kullanılan materyaller;

- 1x PIC16F877A mikrodenetleyici,
- 2x ayarlanabilir potansiyometre,
- 5x 2N2219 transistör,
- 1x LM016L LCD panel,
- 3x buton
- 6x 330 ohm direnç
- 1x 1K ohm direnç
- 2x 0.5uF kapasitör
- 1x 4MHz kristal



Şekil 1 - Proteus Şematiği

Şekil 1.1'de gözüktüğü üzere çalışmanın Proteus şematiği verilmiştir. Şematiği incelediğimizde DC motorun hız kontrolü için ayarlanabilir potansiyometre, yön kontrolü için H-Bridge devresi kullanılmıştır.

DC motorun hız kontrolü ayarlanabilir potansiyometre ile 0-5V aralığında alınan analog sinyallerin PIC16F877A içerisinde bulunan ADC sayesinde dijital sinyale dönüştürülebilmesi ile hasas bir şekilde motor hız kontrolü yapılmıştır.

DC motorun yön kontrolü ise H-Bridge devresi ile yapılmıştır. H-Bridge devresi kullanılmasının sebebi ise motor üzerinde yön kontrolü oluşturmak istenilmesidir. Bu elektronik devre sayesinde transistörlerin anahtarlanması sağlanmıştır. Bu sayede DC motor istenilen yönde (saat yönü veya saat yönü tersinde) sürülebilecektir ve butonlar vasıtasıyla dışarıdan harici bir müdahaleyle bu işlem yapılabilecektir.

LCD panel ise sistemin durumunu anlık olarak bize bildirecektir. Bu durum sırasıyla şunları içermektedir;

- Açılışta 3 saniye çalışmayı hazırlayan kullanıcının bilgileri,
- Motorun sürülmesi için verilecek gücün yüzdesel hız değeri
- Motorun döneceği yön bilgisi

6V olarak ayarlanmış DC motor dışarıdan 6V ile beslenmiştir. Çalışmanın çıktıları da simülasyon ortamındaki uygulamalarda osilaskop çıktıları gözlemlenerek sonuçlanmıştır.

Başlık 2’de çalışmanın kaynak kodu paylaşılmıştır.

Başlık 3’de ise kaynak kodu incelenmiştir.

2- Kaynak Kodu

```
// PIC16F877A Configuration Bit Settings


// 'C' source line config statements


// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = ON         // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; low-voltage programming enabled)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)


// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.


#include <xc.h>
#include <stdio.h>
#define _XTAL_FREQ 4000000
#define Sil 1
#define BasaDon 2
#define SolaYaz 4
#define SagaYaz 5
#define ImlecGizle 12
#define ImlecYanSon 15
#define ImlecGeri 16
#define KaydirSaga 24
#define KaydirSola 28
#define EkraniKapat 8
#define BirinciSatir 128
#define IkinciSatir 192
#define KarakterUretAdres 64
```

```
#define CiftSatir8Bit 56
#define CiftSatir4Bit 48
#define TekSatir8Bit 40
#define TekSatir4Bit 32
#define RS RB0
#define RW RB1
#define E RB2
#define Data PORTD

void Lcd_Port(char a)
{
   if(a & 1)
      RD4 = 1;
   else
      RD4 = 0;

   if(a & 2)
      RD5 = 1;
   else
      RD5 = 0;

   if(a & 4)
      RD6 = 1;
   else
      RD6 = 0;

   if(a & 8)
      RD7 = 1;
   else
      RD7 = 0;
}

void Lcd_Cmd(char a) // LCD komut fonksiyonu
{
   RS = 0;
   Lcd_Port(a);
   E = 1;
```

```
    __delay_ms(4);
    E = 0;
}

void Lcd_Clear() // LCD temizleme fonksiyonu
{
    Lcd_Cmd(0);
    Lcd_Cmd(1);
}

    void Lcd_Set_Cursor(char a, char b) // LCD satir sutün seçme fonksiyonu
{
    char temp,z,y;
    if(a == 1)
    {
        temp = 0x80 + b - 1;
        z = temp>>4;
        y = temp & 0x0F;
        Lcd_Cmd(z);
        Lcd_Cmd(y);
    }
    else if(a == 2)
    {
        temp = 0xC0 + b - 1;
        z = temp>>4;
        y = temp & 0x0F;
        Lcd_Cmd(z);
        Lcd_Cmd(y);
    }
}

void Lcd_Init()        // LCD baslatma fonksiyonu
{
    Lcd_Port(0x00);
    __delay_ms(20);
    Lcd_Cmd(0x03);
    __delay_ms(5);
```

```
    Lcd_Cmd(0x03);
    __delay_ms(11);
    Lcd_Cmd(0x03);
    Lcd_Cmd(0x02);
    Lcd_Cmd(0x02);
    Lcd_Cmd(0x08);
    Lcd_Cmd(0x00);
    Lcd_Cmd(0x0C);
    Lcd_Cmd(0x00);
    Lcd_Cmd(0x06);
}

void Lcd_Write_Char(char a) // LCD ye karakter yazd?rma fonksiyonu
{
    char temp,y;
    temp = a&0x0F;
    y = a&0xF0;
    RS = 1;
    Lcd_Port(y>>4);
    E = 1;
    __delay_us(40);
    E = 0;
    Lcd_Port(temp);
    E = 1;
    __delay_us(40);
    E = 0;
}

void Lcd_Write_String(char *a) // LCD ye string yazdirma fonksiyonu
{
    int i;
    for(i=0;a[i]!='\0';i++)
    Lcd_Write_Char(a[i]);
}

int map(float x, float in_min, float in_max, float out_min, float out_max)
{
```

```
  return (int)((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min);
}

void duty_send( int deger)
{
   CCP1X= deger & 2 ;
   CCP1Y = deger & 1;
   CCPR1L= deger >>2 ;
}

void main(void)
{
   float digital = 0.0;
   float voltage_coefficent = 0.0048;    // 5V / 1024
   float voltage = 0.0;
   char char_digital[15];
   int speed = 0.0;
   int counter = 0;
   int rotate = 0;
   TRISA=0XFF;
   TRISB=0xC0;
   TRISC=0X00;
   TRISD=0X00;

   PORTB=0X00;
   PORTC=0X00;
   PORTD=0X00;

   Lcd_Init();

   ADCON0bits.ADON  = 1;
   ADCON1bits.PCFG0 = 0;
   ADCON1bits.PCFG1 = 0;

   ADCON1bits.PCFG2 = 0;
   ADCON1bits.PCFG3 = 0;
   ADCON1bits.ADFM  = 1;
```

```
while(1)
{
   // giris
   if (counter == 0)
   {
      Lcd_Set_Cursor(1,2);
      Lcd_Write_String("AHMET C. BILIR");
      Lcd_Set_Cursor(2,4);
      Lcd_Write_String("B200104029");
      __delay_ms(3000);
      Lcd_Clear();
      counter++;
   }
   //AN0
   ADCON0bits.CHS2=0;
   ADCON0bits.CHS1=0;
   ADCON0bits.CHS0=0;
   ADCON0bits.GO =1;
   while(ADCON0bits.GO_nDONE);

   //Voltage
   digital = ADRESH*256 + ADRESL;
   voltage = digital * voltage_coefficent;
   //sprintf(char_digital,"%0.2f",voltage);

   //HIZ
   Lcd_Set_Cursor(1,5);
   speed = map(voltage,0,4.9,0,100);
   if (speed < 1)
   {
      speed=0;
      digital=0;
   }
   sprintf(char_digital,"%d",speed);
   Lcd_Write_String("HIZ: %");
   Lcd_Write_String(char_digital);
```

```
//PWM               // Tosc = 1/4MHz => 0,25uS
int dutty = 0;      //PWM PERIYODU = [(PR2)+1] x 4 x Tosc x TMR2Prescale
                    //PWM PERIYODU = 256 x 4 x 0,25uS x 4 = 1024uS => l.024mS
                    //PWM Dutty Cycle = [CCPR1L:CCP1CON<5:4>] x Tosc x TMR2Prescale
                    //PWM Dutty Cycle = ? x 1uS
CCP1CON=0B00001111; // PWM MODE ACTIVE
T2CON = 0B00000101; // T2ON + 1:4 Prescale
TMR2=0;
PR2=255;

//Yön Tesip Etme
if (PORTBbits.RB6 == 1)
{
   while(PORTBbits.RB6 == 1);
   rotate = 1;
}
else if (PORTBbits.RB7 == 1)
{
   while(PORTBbits.RB7 == 1);
   rotate = 2;
}

// DC Motor Durumu
if (speed > 0)
{
   // CCW
   if (rotate == 1)
   {
      PORTCbits.RC3 = 1;
      PORTCbits.RC5 = 1;
      PORTCbits.RC4 = 0;
      PORTCbits.RC6 = 0;
      duty_send(digital);
      Lcd_Set_Cursor(2,5);
      Lcd_Write_String("YON: ---");
   }
```

```
            //CW
            if (rotate == 2)
            {
                PORTCbits.RC3 = 0;
                PORTCbits.RC5 = 0;
                PORTCbits.RC4 = 1;
                PORTCbits.RC6 = 1;
                duty_send(digital);
                Lcd_Set_Cursor(2,5);
                Lcd_Write_String("YON: ---");
            }
        }
        else
        {
            Lcd_Clear();
            PORTCbits.RC3 = 0;
            PORTCbits.RC5 = 0;
            PORTCbits.RC4 = 0;
            PORTCbits.RC6 = 0;
            sprintf(char_digital,"%d",speed);
            Lcd_Set_Cursor(1,5);
            Lcd_Write_String("HIZ: %");
            Lcd_Write_String(char_digital);
            Lcd_Set_Cursor(2,5);
            Lcd_Write_String("YON: ---");
        }
    }
    return;
}
```

3- Kaynak Kodun İncelenmesi

İlk olarak kaynak kodunda gerekli kütüphaneler ve tanımlar şekil 2'de gözüküğü üzere yapılmıştır.

```
// PIC16F877A Configuration Bit Settings

// 'C' source line config statements

// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = ON         // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; low-voltage programming enabled)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdio.h>
#define _XTAL_FREQ 4000000
#define Sil 1
#define BasaDon 2
#define SolaYaz 4
#define SagaYaz 5
#define ImlecGizle 12
#define ImlecYanSon 15
#define ImlecGeri 16
#define KaydirSaga 24
#define KaydirSola 28
#define EkraniKapat 8
#define BirinciSatir 128
#define IkinciSatir 192
#define KarakterUretAdres 64
#define CiftSatir8Bit 56
#define CiftSatir4Bit 48
#define TekSatir8Bit 40
#define TekSatir4Bit 32
#define RS RB0
#define RW RB1
#define E RB2
#define Data PORTD
```

Şekil 2 - Kütüphaneler ve Sabit Tanımlamalar

Akabinde LCD panel ile gerekli işlemler (silme, satıra gitme gibi işlemler) yapmak için Lcd_Port(), Lcd_Cmd(), Lcd_Clear(), Lcd_Set_Cursor(), Lcd_Init(), Lcd_Write_Char(), Lcd_Write_String() fonksiyonları tanımlanmıştır.

Daha sonra map() fonskiyonu ve duty_send() fonksiyonu tanımlanmıştır. Bu fonksiyonlardan map() fonksiyonu 5 adet parametreyle, duty_send() fonskiyonu 1 adet parametreyle çalışmaktadır.

Sırasıyla:

map(A,B,C,D,E): Fonksiyon istenilen A parametresini istenilen değerler arasında tekrar ölçekleme işlemini yapmaktadır. B ve C parametreleri o an alabileceği minimum ve maksimum değerleri temsil etmektedir. D ve E parametreleri ise hangi değerler arasında ölçekleneceğini temsil eder. Potansiyometreden gelen analog değer ADC sayesinde 10 bitlik değere dönüşmektedir ve bu da 0-1023 arasında dijital olarak bize ulaşmaktadır. A değeri bu durumda 0-1023 arasında bir değer olacaktır. B ve C parametreleri minimu ve maksimumu temsil ettiğinden sırasıyla 0 ve 1023 olarak verilmelidir. D ve E parametreleri, bu çalışmada 0 ile 5V arasında giriş alındığı için girişin voltaj karşılığını hız bilgisine dönüştürmek istendiğinden dolayı, D ve E sırasıyla 0 ve 100 seçilmiştir. Bu bize anlık voltajı 0-100 arasında oranlayarak güncel motor hızını hesaplamamıza olanak sağlamıştır. Şekil 3'de ölçekleme işleminin matematiksel karşılığı verilmiştir.

```
int map(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (int)((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min);
}
```

Şekil 3 - Ölçekleme Fonksiyonu

duty_send(A): Fonksiyon içindeki işlemler, bu parametre üzerinde bazı bit işlemleri gerçekleştirir. CCP1X" değişkenine, "deger" parametresinin 2'ye bölümünden kalanı atar, CCP1Y" değişkenine, "deger" parametresinin 2'ye bölümünden kalanını alır. CCPR1L değişkenine, A parametresini sağa kaydırma işlemi (bit düzeyinde kaydırma) yaparak 2'ye böler ve sonucu atar. Şekil 4'te bu işlem gösterilmiştir.

```
void duty_send( int deger)
{
    CCP1X= deger & 2 ;
    CCP1Y = deger & 1;
    CCPR1L= deger >>2 ;
}
```

Şekil 4 - Duty Fonksiyonu

Buraya kadar yapılan tanımlamalar, uygulamada kullanılacak işlemler için gerekli olan kodlamalardır. Ana fonksiyon akabinde çağrılır. Ana fonksiyonda tanımlanan değişkenler Şekil 5'de verilmiştir. Gerekli port giriş ve çıkışları TRIS ile tanımlanmıştır. PORT ile kullanılacak portlar temizlenmiştir. ADCON ile ADC için tanımlamalar yapılmıştır ve akabinde çevrim başlatılacaktır.

```
void main(void)
{
    float digital = 0.0;
    float voltage_coefficent = 0.0048;     // 5V / 1024
    float voltage = 0.0;
    char char_digital[15];
    int speed = 0.0;
    int counter = 0;
    int rotate = 0;
    TRISA=0XFF;
    TRISB=0xC0;
    TRISC=0X00;
    TRISD=0X00;

    PORTB=0X00;
    PORTC=0X00;
    PORTD=0X00;

    Lcd_Init();

    ADCON0bits.ADON  = 1;
    ADCON1bits.PCFG0 = 0;
    ADCON1bits.PCFG1 = 0;

    ADCON1bits.PCFG2 = 0;
    ADCON1bits.PCFG3 = 0;
    ADCON1bits.ADFM  = 1;
```

Şekil 5 - Main Fonksiyonu Tanımlamalar

Program şekil 6'da gözüktüğü üzere while(1) fonskiyonu ile sonsuz döngüye sokulmuştur. Counter değişkenine bağlı olarak 3 saniye boyunca kullanıcının isim ve numara bilgisi LCD panel üzerine yazdılırmış ve counter değişkeni arttırılarak program kapatılana kadar bu döngüye bir daha girilmesinin önüne geçilmiştir. ADCON ile çevrim başlatılmış ve çevrim bitene kadar while() döngüsüne sıkıştırılmıştır. Çevrim bittikten sonra 10 bitlik değer uygun işlem yapılarak digital değişkenine aktarılmıştır. Digital değeri voltaj katsayısıyla çarpılarak 0-1023 digital değeri aralığında hangi voltaja tekabül ettiği hesaplanmıştır. Bu işlemden sonra map fonksiyonu sayesinde voltaj değişkeni yüzdesel olarak hangi hıza tekabül ettiği ölçeklenmiştir. Hızın 1'den küçük olması durumunda dahi mV elektrik akımı olmasından dolayı motor dönmektedir. Bunun önüne geçmek için hızın 1'den küçük olduğu takdirde speed ve digital 0 olarak tanımlanmıştır. Bu değerler de LCD panele yazdırılmıştır.

```
while(1)
{
    // giris
    if (counter == 0)
    {
        Lcd_Set_Cursor(1,2);
        Lcd_Write_String("AHMET C. BILIR");
        Lcd_Set_Cursor(2,4);
        Lcd_Write_String("B200104029");
        __delay_ms(3000);
        Lcd_Clear();
        counter++;
    }
    //AN0
    ADCON0bits.CHS2=0;
    ADCON0bits.CHS1=0;
    ADCON0bits.CHS0=0;
    ADCON0bits.GO =1;
    while(ADCON0bits.GO_nDONE);

    //Voltage
    digital = ADRESH*256 + ADRESL;
    voltage = digital * voltage_coefficent;
    //sprintf(char_digital,"%0.2f",voltage);

    //HIZ
    Lcd_Set_Cursor(1,5);
    speed = map(voltage,0,4.9,0,100);
    if (speed < 1)
    {
        speed=0;
        digital=0;
    }
    sprintf(char_digital,"%d",speed);
    Lcd_Write_String("HIZ: %");
    Lcd_Write_String(char_digital);
```

Şekil 6 - Sonsuz Döngüye Giriş

Ardından dutty değişkeni tanımlanmıştır. Bu değişken bizim PWM periyodumuz için kritik bir role sahiptir ve uygun hesaplamalar ve TMR2 için tanımlamalar şekil 7'de gözülüğü üzere kod bloğunda gösterilmektedir.

```
//PWM                 // Tosc = 1/4MHz => 0,25uS
int dutty = 0;        //PWM PERIYODU = [(PR2)+1] x 4 x Tosc x TMR2Prescale
                      //PWM PERIYODU = 256 x 4 x 0,25uS x 4 = 1024uS => 1.024mS
                      //PWM Dutty Cycle = [CCPR1L:CCP1CON<5:4>] x Tosc x TMR2Prescale
                      //PWM Dutty Cycle = ? x 1uS
CCP1CON=0B00001111; // PWM MODE ACTIVE
T2CON = 0B00000101; // T2ON + 1:4 Prescale
TMR2=0;
PR2=255;
```

Şekil 7 – PWM, Dutty Hesaplama ve Tanımlamalar

Butonlardan gelen sinyale göre motorun hangi yöne döneceği şekil 8'de hesaplanmış ve rotate değişkenine atılmıştır. Bu sayede uygun anahtarlar bir sonraki adımda tetiklenecektir.

```
//Yön Tesip Etme
if (PORTBbits.RB6 == 1)
{
    while(PORTBbits.RB6 == 1);
    rotate = 1;
}
else if (PORTBbits.RB7 == 1)
{
    while(PORTBbits.RB7 == 1);
    rotate = 2;
}
```

Şekil 8 - Yön Karar Verme

Şekil 9'da hız bilgisine göre motor sürme işlemi için yön koşullarına bağlı olarak uygun tetiklemeler ve tanımlamalar yapılmıştır. Bu sayede anahtarlamalar yön bilgisine göre çalışmış ve motorun yönü saat yönünde(CW-ClockWise) veya saat yönü tersinde (CCW- Counter ClockWise) seçilmiştir. Aksi bir durumda ekrana "STOPPED" yazdırılmıştır.

```
    // DC Motor Durumu
    if (speed > 0)
    {
        // CCW
        if (rotate == 1)
        {
            PORTCbits.RC3 = 1;
            PORTCbits.RC5 = 1;
            PORTCbits.RC4 = 0;
            PORTCbits.RC6 = 0;
            duty_send(digital);
            Lcd_Set_Cursor(2,5);
            Lcd_Write_String("YON: ---");
        }
        //CW
        if (rotate == 2)
        {
            PORTCbits.RC3 = 0;
            PORTCbits.RC5 = 0;
            PORTCbits.RC4 = 1;
            PORTCbits.RC6 = 1;
            duty_send(digital);
            Lcd_Set_Cursor(2,5);
            Lcd_Write_String("YON: ---");
        }
    }
    else
    {
        Lcd_Clear();
        PORTCbits.RC3 = 0;
        PORTCbits.RC5 = 0;
        PORTCbits.RC4 = 0;
        PORTCbits.RC6 = 0;
        sprintf(char_digital,"%d",speed);
        Lcd_Set_Cursor(1,5);
        Lcd_Write_String("HIZ: %");
        Lcd_Write_String(char_digital);
        Lcd_Set_Cursor(2,5);
        Lcd_Write_String("YON: ---");
    }
}
return;
```

Şekil 9 - Motor Sürme

Bu şekilde uygulama başarılı bir şekilde gerçekleştirilmiş ve kaynak kod açıklanmıştır.