به نام خدا

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه پایانی ریزپردازنده و زبان های اسمبلی

نام استاد : استاد ماهوش

نام دستیار استاد : آقای صامتی

انجام دهنده :

محمدحسین ملکی _4003623035

سید محمد حسین هاشمی_4022363143

خلاصه ای از پروژه :

پروژه نهایی بر روی میکروکنترلر 32ATmega طراحی شده است. هدف پروژه کنترل یک موتور DC با استفاده از پتانسیومتر و نمایش اطلاعات بر روی LCD می‌باشد. نرم‌افزارهای مورد نیاز شامل Proteus و Atmel Studio بوده و برنامه‌نویسی باید با زبان C انجام شود. امتیاز پروژه به دو بخش تقسیم شده: 200 امتیاز برای بخش اول (کنترل موتور DC) و 300 امتیاز برای بخش دوم (کنترل چراغ‌های راهنمایی).

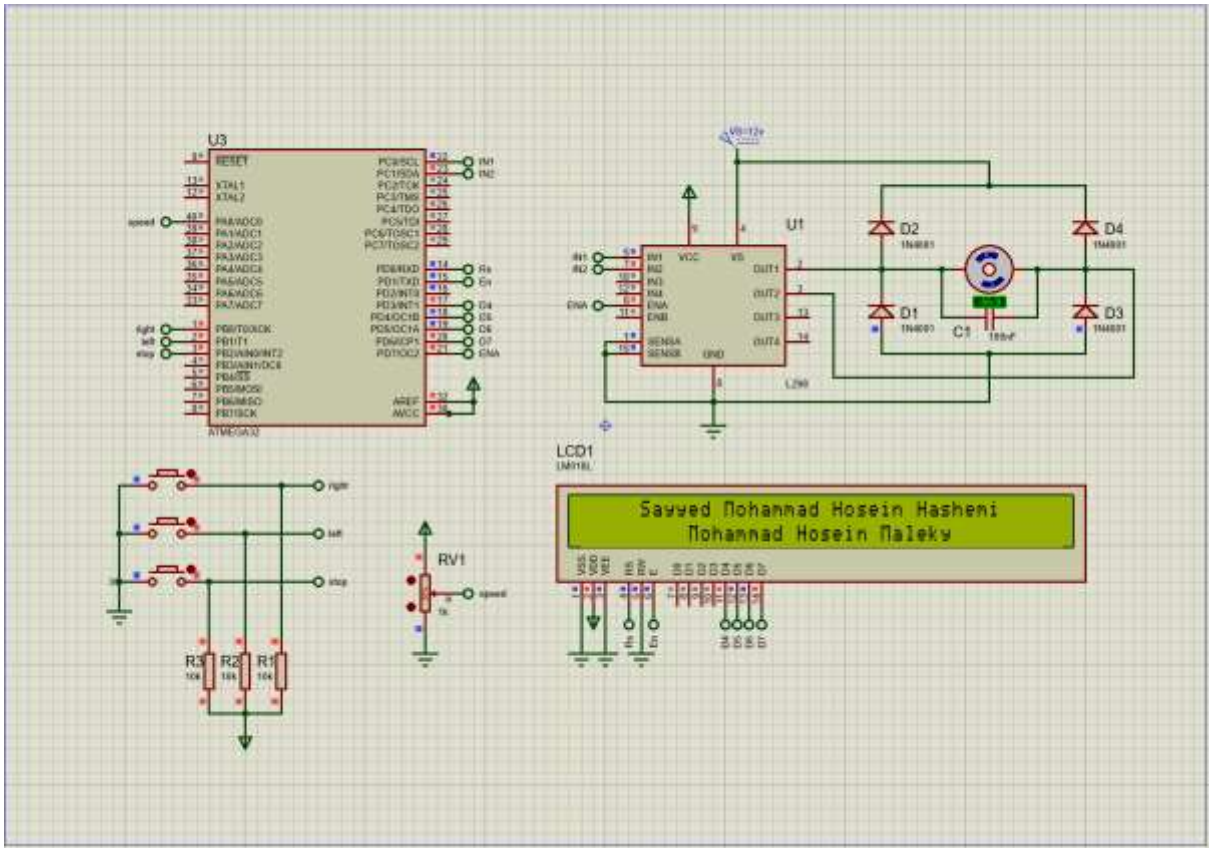
بخش اول: کنترل موتور DC

- هدف: کنترل سرعت موتور DC با استفاده از پتانسیومتر.
- ماژول‌ها:
  - موتور DC: برای حرکت موتور.
  - 298L: ماژول کنترل موتور.
  - پتانسیومتر: برای تنظیم سرعت موتور.

بخش دوم: کنترل چراغ‌های راهنمایی

- هدف: مدیریت چراغ‌های راهنمایی یک چهارراه.
- چراغ‌ها: 4 چراغ (2 چراغ برای مسیر عمودی و 2 چراغ برای مسیر افقی).
- الگوی عملکرد چراغ‌ها: سبز، زرد و قرمز با زمان‌بندی مشخص و استفاده از تایمر برای نمایش زمان باقی‌مانده.
- پتانسیومتر: برای تنظیم زمان سبز بودن چراغ‌ها.

بخش اول موتور **DC**



اجزا:

1. میکروکنترلر **ATmega32**: برای کنترل موتور و پردازش داده‌ها.
2. **ADC** (مبدل آنالوگ به دیجیتال): برای خواندن مقادیر آنالوگ و تبدیل آنها به دیجیتال.
3. **PWM** (مدولاسیون پهنای پالس): برای کنترل سرعت موتور.
4. **LCD**: برای نمایش پیام‌ها و وضعیت‌ها.
5. دکمه‌ها (کلیدها): برای کنترل جهت چرخش موتور.

روند کار:

1. تنظیمات اولیه:
   - تنظیم پورت‌ها: تعیین ورودی یا خروجی بودن پورت‌ها.
   - تنظیم تایمر و PWM: تنظیم تایمر 2 برای تولید سیگنال PWM.
   - تنظیم ADC: تنظیم مبدل آنالوگ به دیجیتال برای خواندن مقادیر آنالوگ.

2. نمایش پیام‌های اولیه:
   - مقداردهی اولیه و پاک کردن LCD.
   - نمایش پیام خوش‌آمدگویی و معرفی پروژه و نویسندگان آن.

3. حلقه اصلی:
   - خواندن مقدار ADC و تنظیم PWM بر اساس آن.
   - کنترل جهت چرخش موتور با استفاده از دکمه‌ها:
     - اگر دکمه متصل به PINB.0 فشار داده شود، موتور در یک جهت می‌چرخد.
     - اگر دکمه متصل به PINB.1 فشار داده شود، موتور در جهت مخالف می‌چرخد.
     - اگر دکمه متصل به PINB.2 فشار داده شود، موتور متوقف می‌شود.

کدها:

بخش کتابخانه‌ها و تعریف‌ها:

code.c

```
#include <mega32.h>
#include <delay.h>
#include <alcd.h>
```

تابع خواندن **ADC**

code.c

```
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (1<<ADLAR))
unsigned char read_adc(unsigned char adc_input)
{
  ADMUX = adc_input | ADC_VREF_TYPE;
  delay_us(10);
  ADCSRA |= (1<<ADSC);
  while ((ADCSRA & (1<<ADIF)) == 0);
  ADCSRA |= (1<<ADIF);
  return ADCH;
}
```

تنظیم **ADC**: کانال ورودی **ADC** و مرجع ولتاژ را تنظیم می‌کند.
تأخیر کوتاه: برای اطمینان از پایداری.
شروع تبدیل: بیت **ADSC** را تنظیم می‌کند.
انتظار برای اتمام تبدیل: منتظر می‌ماند تا بیت **ADIF** تنظیم شود.
پاک کردن بیت **ADIF**: با نوشتن 1 در این بیت.
بازگشت نتیجه: مقدار بالای 8 بیت داده **ADC**.

تابع اصلی:

code.c

```c
void main(void)
{
  // تنظیمات پورت ها
  DDRA = 0x00; PORTA = 0x00;
  DDRB = 0x00; PORTB = 0x00;
  DDRC = 0x03; PORTC = 0x00;
  DDRD = 0xff; PORTD = 0x00;

  // تنظیمات تایمر و PWM
  ASSR = 0 << AS2;
  TCCR2 = (1 << PWM2) | (1 << COM21) | (0 << COM20) | (0 << CTC2) | (0 <<
CS22) | (1 << CS21) | (0 << CS20);
  TCNT2 = 0x00;
  OCR2 = 0x00;

  // تنظیمات ADC
  ADMUX = ADC_VREF_TYPE;
  ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADATE) | (0 << ADIF) | (0 <<
ADIE) | (0 << ADPS2) | (0 << ADPS1) | (1 << ADPS0);
  SFIOR = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);

  OCR2 = 0;

  // تنظیمات اولیه LCD
  lcd_init(40);
  lcd_clear();
  lcd_gotoxy(16, 0);
  lcd_puts("WELCOME!");
  lcd_gotoxy(5, 0);
  lcd_puts("DC motor control with ATMEGA32");
  delay_ms(300);
  lcd_clear();
  lcd_gotoxy(5, 0);
  lcd_puts("Sayyed Mohammad Hosein Hashemi");
  lcd_gotoxy(9, 1);
  lcd_puts("Mohammad Hosein Maleky");

  // حلقه اصلی
  while (1)
  {
      OCR2 = read_adc(0);

      if (PINB.0 == 0)
      {
          PORTC.0 = 0;
          PORTC.1 = 1;
      }

      if (PINB.1 == 0)
      {
          PORTC.0 = 1;
          PORTC.1 = 0;
      }

      if (PINB.2 == 0)
      {
          PORTC.0 = 0;
          PORTC.1 = 0;
      }
  }
}
```

توضیحات تنظیمات

- پورت‌ها: **DDRB**، **DDRA** به عنوان ورودی و **DDRD**، **DDRC** به عنوان خروجی تنظیم می‌شوند.
- تایمر و **PWM**: تایمر 2 برای تولید **PWM** تنظیم شده است.
- **ADC: ADC** برای تبدیل مقادیر آنالوگ به دیجیتال تنظیم می‌شود.
- **LCD**: نمایش پیام‌های اولیه بر روی **LCD**.

حلقه اصلی

- خواندن **ADC**: مقدار **ADC** خوانده شده و به **OCR2** اختصاص می‌یابد تا **PWM** تنظیم شود.
- کنترل جهت موتور: با استفاده از دکمه‌ها، جهت چرخش موتور کنترل می‌شود:
  - **PINB**.0: موتور در جهت اول می‌چرخد.
  - **PINB**.1: موتور در جهت دوم می‌چرخد.
  - **PINB**.2: موتور متوقف می‌شود.

- **OCR2 = read_adc**(0): خواندن مقدار آنالوگ از **ADC** و تنظیم مقدار **PWM**.
- کنترل جهت موتور:
  - **PINB**.0 == 0: چرخش موتور در یک جهت.
  - **PINB**.1 == 0: چرخش موتور در جهت مخالف.
  - **PINB**.2 == 0: توقف موتور.

تنظیمات پورت‌ها:

code.c

```
DDRA = 0x00; PORTA = 0x00;
DDRB = 0x00; PORTB = 0x00;
DDRC = 0x03; PORTC = 0x00;
DDRD = 0xff; PORTD = 0x00;
```

- **DDRA** و **PORTA**: پورت A به عنوان ورودی تنظیم شده و مقدار اولیه آن صفر است.
- **DDRB** و **PORTB**: پورت B به عنوان ورودی تنظیم شده و مقدار اولیه آن صفر است.
- **DDRC** و **PORTC**: پورت C به عنوان خروجی تنظیم شده و مقدار اولیه آن صفر است (فقط دو بیت اول به عنوان خروجی تنظیم شده‌اند).
- **DDRD** و **PORTD**: پورت D به عنوان خروجی تنظیم شده و مقدار اولیه آن صفر است.

تنظیمات تایمر و **PWM**:

code.c

```
ASSR = 0 << AS2;
TCCR2 = (1 << PWM2) | (1 << COM21) | (0 << COM20) | (0 << CTC2) | (0 << CS22)
| (1 << CS21) | (0 << CS20);
TCNT2 = 0x00;
OCR2 = 0x00;
```

- **ASSR**: تنظیمات ثبات کمکی تایمر/کانتر.
- **TCCR2**: تنظیمات تایمر/کانتر 2 برای تولید PWM.
  - **PWM2**: فعال کردن مد PWM.
  - **COM21**: تنظیم خروجی روی مقایسه.
  - **CS21**: تنظیم پیش‌تقسیم‌کننده به 8.
- **TCNT2**: مقدار اولیه تایمر 2.
- **OCR2**: مقدار اولیه رجیستر مقایسه خروجی (برای PWM).

تنظیمات **ADC**:

code.c

```
ADMUX = ADC_VREF_TYPE;
ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADATE) | (0 << ADIF) | (0 << ADIE)
| (0 << ADPS2) | (0 << ADPS1) | (1 << ADPS0);
SFIOR = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);
```

- **ADMUX**: تنظیم نوع مرجع ولتاژ و فعال کردن رجیستر چپ‌چین.
- **ADCSRA**: تنظیمات کنترل و وضعیت ADC.
  - **ADEN**: فعال کردن ADC.
  - **ADPS0**: تنظیم پیش‌تقسیم‌کننده به 2.

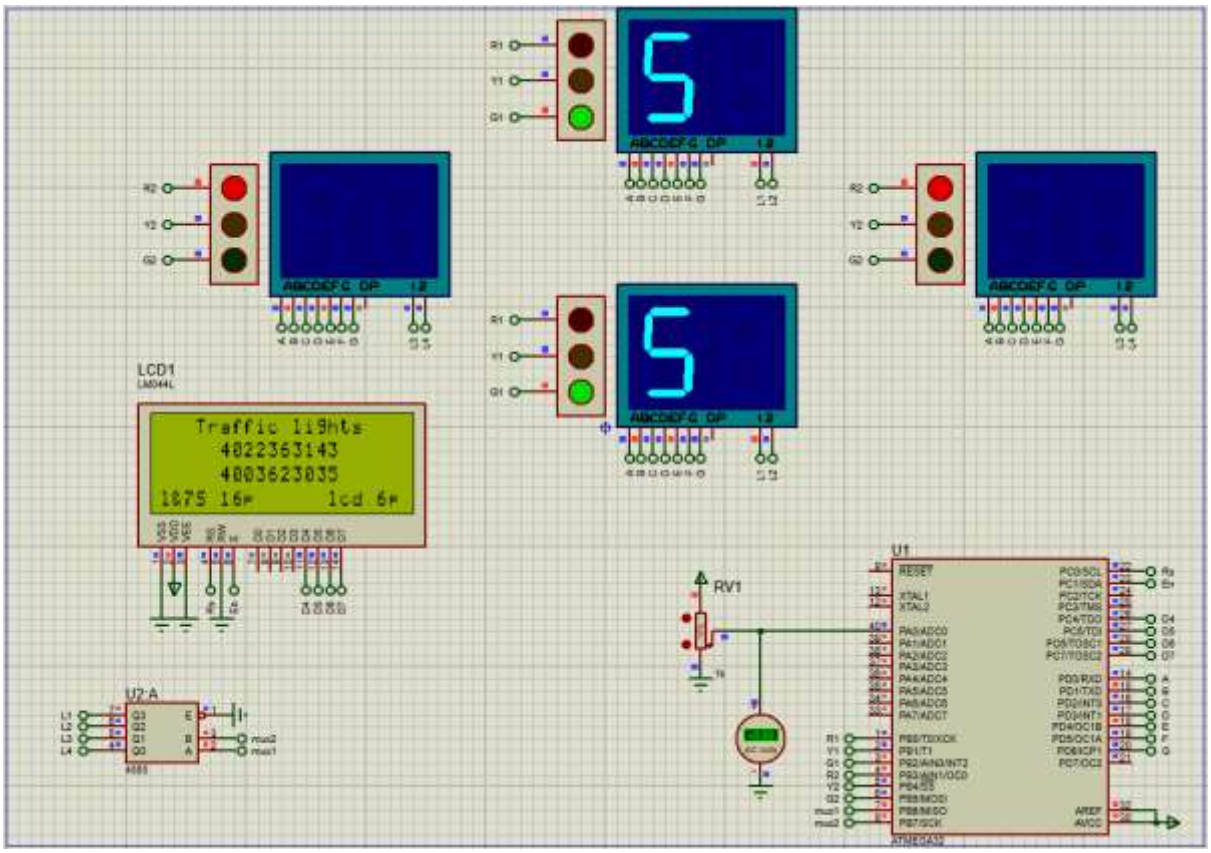
تنظیمات **LCD** :

code.c

```
lcd_init(40);
lcd_clear();
lcd_gotoxy(16, 0);
lcd_puts("WELCOME!");
lcd_gotoxy(5, 0);
lcd_puts("DC motor control with ATMEGA32");
delay_ms(300);
lcd_clear();
lcd_gotoxy(5, 0);
lcd_puts("Sayyed Mohammad Hosein Hashemi");
lcd_gotoxy(9, 1);
lcd_puts("Mohammad Hosein Maleky");
```

- **lcd_init**: مقداردهی اولیه LCD.
- **lcd_clear**: پاک کردن صفحه LCD.
- **lcd_gotoxy**: تنظیم مکان‌نما روی LCD.
- **lcd_puts**: نمایش پیام روی LCD.
- **delay_ms**: ایجاد تأخیر.

کنترل چراغ‌های راهنمایی رانندگی



خلاصه پروژه کنترل چراغ راهنمایی

مقدمه

پروژه نهایی بر روی 32ATmega تعریف شده است و هدف کنترل چراغ‌های راهنمایی با استفاده از این میکروکنترلر است. در این پروژه، شما باید یک مدار را در نرم‌افزار Proteus طراحی کرده و کدی بنویسید که 32ATmega بر اساس آن عمل کند. نرم‌افزارهای مورد نیاز شامل Proteus و Atmel Studio هستند و زبان برنامه‌نویسی C است. پروژه به دو بخش تقسیم شده و هر بخش شامل امتیازهای جداگانه است. تیم‌ها باید از دو نفر تشکیل شوند.

بخش اول

در این بخش، هدف کنترل یک موتور DC با استفاده از 32ATmega است. با استفاده از یک پتانسیومتر، باید بتوان سرعت چرخش موتور را تنظیم کرد. جهت چرخش موتور مهم نیست و تنها تغییر سرعت چرخش مورد نظر است. ماژول‌های مورد نیاز شامل موتور DC، ماژول 298L برای کنترل موتور و پتانسیومتر است.

بخش دوم

در این بخش، هدف مدیریت چراغ‌های راهنمایی یک چهارراه با استفاده از 32ATmega است. در این چهارراه، چهار چراغ راهنمایی وجود دارد که باید به ترتیب سبز، زرد و قرمز شوند. زمان سبز بودن چراغ‌ها باید حداکثر 60 ثانیه باشد و زمان قرمز بودن کمتر از زمان سبز بودن و به دلخواه تعیین شود. زمان زرد بودن چراغ‌ها حداکثر 2 ثانیه است. چراغ‌های روبروی هم رنگ مشابه دارند و چراغ‌های عمودی و افقی باید رنگ‌های متفاوت داشته باشند. از پتانسیومتر برای تنظیم زمان سبز بودن چراغ استفاده می‌شود.

کتابخانه‌ها و تعاریف اولیه:

code.c

```c
#include <mega32.h>
#include <delay.h>
#include <alcd.h>

#define no_0 0b10111111 // 0
#define no_1 0b10000110 // 1
#define no_2 0b11011011 // 2
#define no_3 0b11001111 // 3
#define no_4 0b11100110 // 4
#define no_5 0b11101101 // 5
#define no_6 0b11111101 // 6
#define no_7 0b10000111 // 7
#define no_8 0b11111111 // 8
#define no_9 0b11101111 // 9

char numbers[10] = {~no_0, ~no_1, ~no_2, ~no_3, ~no_4, ~no_5, ~no_6, ~no_7,
~no_8, ~no_9};

#define counter_delay 30
```

این بخش شامل کتابخانه‌های مورد نیاز و تعاریف اولیه برای نمایش اعداد بر روی سون سگمنت است..

توابع

code.c

```
void counter_up(int status) {
  if (status == 0) {
      PORTB.6 = 0;
      PORTB.7 = 0;
  } else if (status == 1) {
      PORTB.6 = 1;
      PORTB.7 = 0;
  } else if (status == 2) {
      PORTB.6 = 0;
      PORTB.7 = 1;
  } else {
      PORTB.6 = 1;
      PORTB.7 = 1;
  }
}

void light_status(int light, int status) {
  if (light == 0) {
      if (status == 0) {
          PORTB.0 = 1;
          PORTB.1 = 0;
          PORTB.2 = 0;
      } else if (status == 1) {
          PORTB.0 = 0;
          PORTB.1 = 1;
          PORTB.2 = 0;
      } else {
          PORTB.0 = 0;
          PORTB.1 = 0;
          PORTB.2 = 1;
      }
  } else {
      if (status == 0) {
          PORTB.3 = 1;
          PORTB.4 = 0;
          PORTB.5 = 0;
      } else if (status == 1) {
          PORTB.3 = 0;
          PORTB.4 = 1;
          PORTB.5 = 0;
      } else {
          PORTB.3 = 0;
          PORTB.4 = 0;
          PORTB.5 = 1;
      }
  }
}
```

توابع `counter_up` و `light_status` برای تنظیم وضعیت چراغ‌ها و شمارنده‌ها استفاده می‌شوند.

خواندن **ADC**

code.c

```
#define ADC_VREF_TYPE ((0<<REFS1) | (0<<REFS0) | (0<<ADLAR))
unsigned int read_adc(unsigned char adc_input) {
  ADMUX = adc_input | ADC_VREF_TYPE;
  delay_us(10);
  ADCSRA |= (1 << ADSC);
  while ((ADCSRA & (1 << ADIF)) == 0);
  ADCSRA |= (1 << ADIF);
  return ADCW;
}
```

این تابع مقدار ADC را از کانال مشخص شده می‌خواند.

تابع اصلی

code.c

```
void main(void) {
  DDRA = 0b00000000;
  PORTA = 0b00000001;

  DDRB = 0xff;
  PORTB = 0x00;

  DDRC = 0xff;
  PORTC = 0x00;

  DDRD = 0xff;
  PORTD = 0xff;

  ADMUX = ADC_VREF_TYPE;
  ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADATE) | (0 << ADIF) | (0 <<
ADIE) | (0 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
  SFIOR = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0);
```

code.c

```
lcd_init(20);
lcd_gotoxy(6, 1);
lcd_puts("WELCOME!");
lcd_gotoxy(3, 2);
lcd_puts("Traffic lights");
delay_ms(200);

lcd_clear();
lcd_gotoxy(3, 0);
lcd_puts("Traffic lights");
lcd_gotoxy(5, 1);
lcd_puts("4022363143");
lcd_gotoxy(5, 2);
lcd_puts("4003623035");
lcd_gotoxy(0, 3);
lcd_puts("l&7S 16p    lcd 6p");

l1s = 2, l2s = 0;
l1 = maxTime;
l2 = maxTime * 3 / 4;
```

code.c

```
  while (1) {
      l1--;
      l2--;
      if (l1 == 0) {
          if (l1s == 0) {
              l1 = adc_value;
              l1s = 2;
          } else {
              l1 = adc_value * 3 / 4;
              l1s = 0;
          }
      }

      if (l1 <= 2 && l1s == 2) {
          l1s = 1;
      }

      if (l2 == 0) {
          if (l2s == 0) {
              l2 = adc_value;
              l2s = 2;
          } else {
              l2 = adc_value * 3 / 4;
              l2s = 0;
          }
      }

      if (l2 <= 2 && l2s == 2) {
          l2s = 1;
      }

      light_status(0, l1s);
      light_status(1, l2s);

      h1h = l1 / 10;
      l1l = l1 % 10;
      h2h = l2 / 10;
      l2l = l2 % 10;

      for(i = 0; i < 1; i++) {
          PORTD = numbers[h1h];
          counter_up(3);
          delay_ms(counter_delay);

          PORTD = numbers[l1l];
          counter_up(2);
          delay_ms(counter_delay);

          PORTD = numbers[h2h];
          counter_up(1);
          delay_ms(counter_delay);

          PORTD = numbers[l2l];
          counter_up(0);
          delay_ms(counter_delay);
      }
      adc_value = read_adc(0) * maxTime / 1023;
  }
}
```

در تابع اصلی، پورت‌ها و ADC تنظیم می‌شوند. چراغ‌های راهنمایی و شمارنده‌ها بر اساس مقدار ADC کنترل می‌شوند. نمایش اعداد بر روی سون سگمنت نیز انجام می‌شود.

توضیحات تکمیلی

ماژول **298L**

برای کنترل موتور DC از ماژول 298L استفاده می‌شود که به شما امکان کنترل سرعت و جهت موتور را می‌دهد. این ماژول می‌تواند دو موتور را کنترل کند و برای پروژه‌هایی که نیاز به کنترل دقیق موتور دارند، بسیار مناسب است.

پتانسیومتر

پتانسیومتر به عنوان یک سنسور آنالوگ استفاده می‌شود که مقدار ولتاژ خروجی آن با تغییر مقاومت تغییر می‌کند. این ولتاژ به ADC میکروکنترلر وارد می‌شود و به یک مقدار دیجیتال تبدیل می‌شود که برای تنظیم زمان چراغ‌ها استفاده می‌شود.

نمایشگر **LCD**

نمایشگر LCD برای نمایش پیام‌های خوش‌آمدگویی و اطلاعات اولیه پروژه استفاده می‌شود. این نمایشگر به پورت‌های میکروکنترلر متصل شده و با استفاده از توابع کتابخانه `alcd.h` کنترل می‌شود.

نکات پایانی

- پیکربندی پورت‌ها: تنظیم پورت‌های توضیحات پروژه کنترل چراغ راهنمایی

A پورت (**DDRA** و **PORTA**) :

code.c

```
DDRA = 0b00000000;  // تنظیم پورت A به عنوان ورودی
PORTA = 0b00000001; // فعال کردن مقاومت‌پول‌آپ برای پین 0 پورت A
```

پورت A برای خواندن ورودی‌های آنالوگ از پتانسیومتر استفاده می‌شود. پین 0 از پورت A با فعال کردن مقاومت پول‌آپ داخلی به عنوان ورودی تنظیم شده است.

B پورت (**DDRB** و **PORTB**) :

code.c

```
DDRB = 0xff; // تنظیم پورت B به عنوان خروجی
PORTB = 0x00; // تنظیم مقدار اولیه پورت B 0 به
```

پورت B به عنوان خروجی برای کنترل LED های چراغ راهنمایی و سگمنت‌های شمارنده تنظیم شده است.

C پورت (**DDRB** و **: (PORTB**

code.c

```
DDRC = 0xff; // تنظیم پورت C به عنوان خروجی
PORTC = 0x00; // تنظیم مقدار اولیه پورت C 0 به
```

پورت C به عنوان خروجی برای کنترل نمایشگر LCD استفاده می‌شود.

D پورت (**DDRB** و **PORTB**) :

code.c

```
DDRD = 0xff; // تنظیم پورت D به عنوان خروجی
PORTD = 0xff; // تنظیم مقدار اولیه پورت D 0 به0xff (همه‌بیت‌ها یک می‌شوند)
```

پورت D به عنوان خروجی برای نمایش مقادیر بر روی سون سگمنت استفاده می‌شود.

تنظیمات **ADC**

برای خواندن مقادیر آنالوگ از پتانسیومتر، ADC (مبدل آنالوگ به دیجیتال) پیکربندی می‌شود:

code.c

```
ADMUX = ADC_VREF_TYPE; // تنظیم منبع ولتاژ مرجع و انتخاب کانال ADC
ADCSRA = (1 << ADEN) | (0 << ADSC) | (0 << ADATE) | (0 << ADIF) | (0 << ADIE)
| (0 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // تنظیمات ADC
SFIOR = (0 << ADTS2) | (0 << ADTS1) | (0 << ADTS0); // تنظیمات اضافی ADC
```

- **ADMUX**: انتخاب کانال ADC و تنظیم ولتاژ مرجع.
- **ADCSRA**: فعال‌سازی ADC، تنظیمات پیش‌تقسیم‌کننده برای سرعت نمونه‌برداری.
- **SFIOR**: تنظیمات اضافی برای ADC.