**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 ATmega32 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.
  + L298: 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.

### 

### 

### 

### 

### 

### 

### 

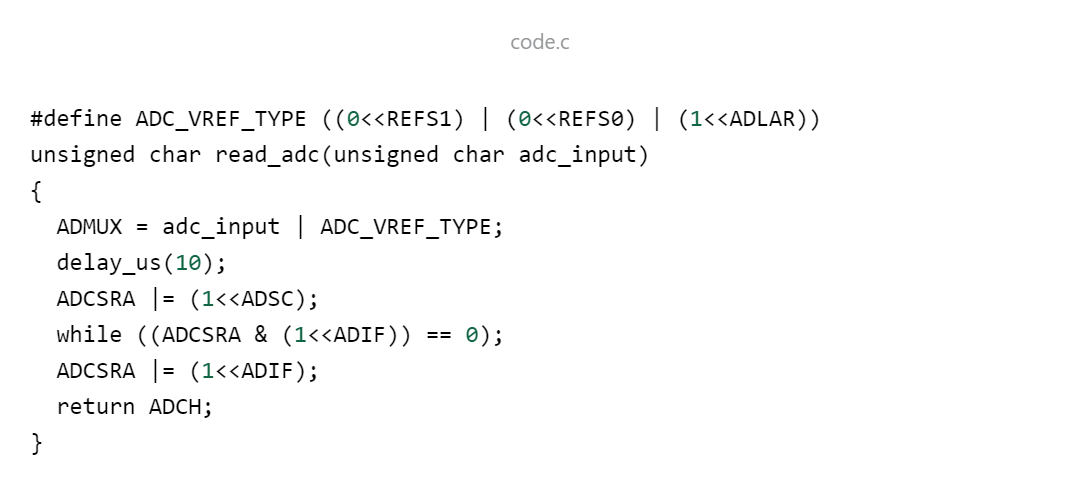
### 

### 

### **کدها:**

**بخش کتابخانه‌ها و تعریف‌ها:  
**

**تابع خواندن ADC**

****

**تنظیم ADC: کانال ورودی ADC و مرجع ولتاژ را تنظیم می‌کند.**

**تأخیر کوتاه: برای اطمینان از پایداری.**

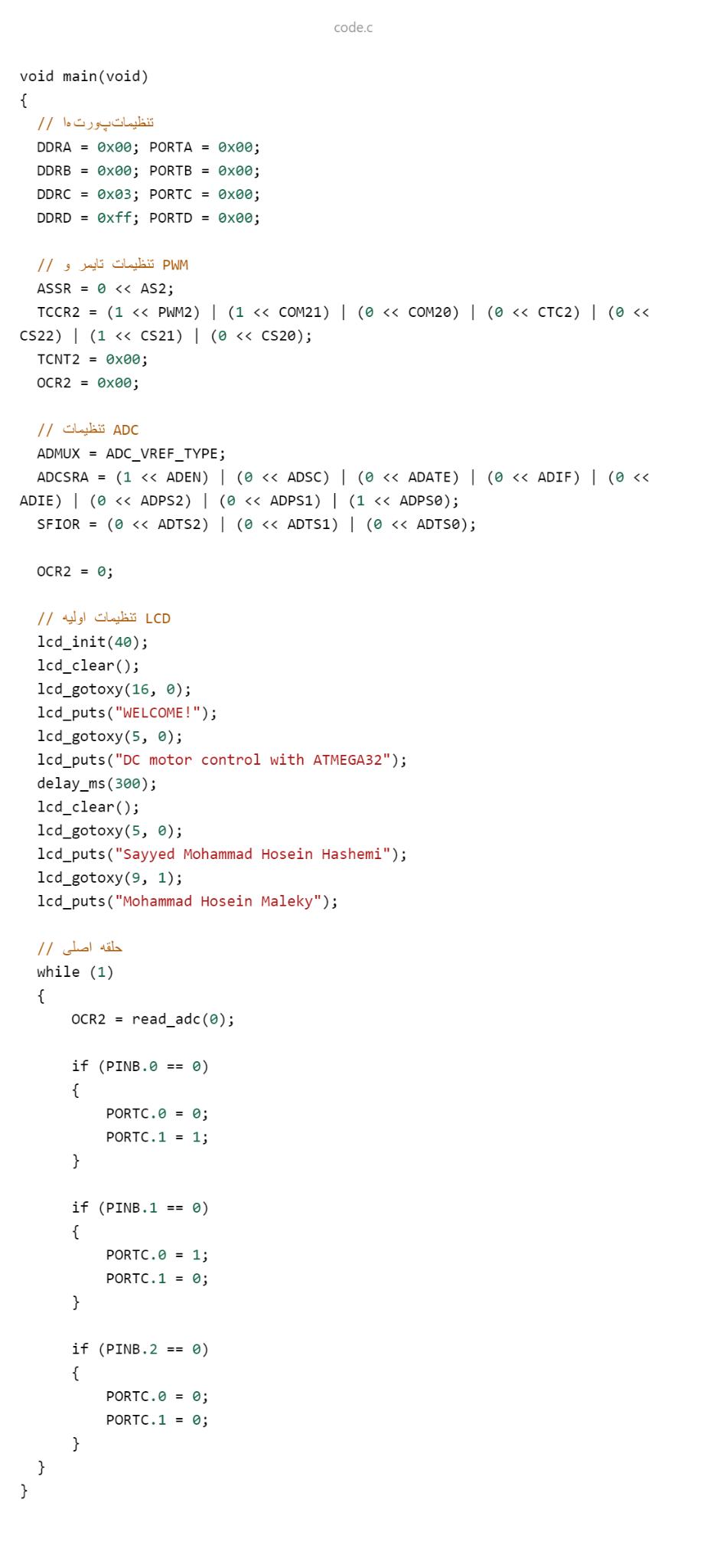
**شروع تبدیل: بیت ADSC را تنظیم می‌کند.**

**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:**



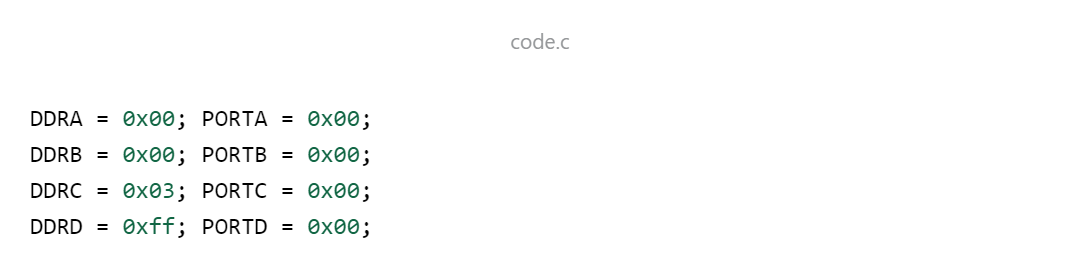
### **توضیحات تنظیمات**

* **پورت‌ها: DDRA، DDRB به عنوان ورودی و DDRC، DDRD 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**

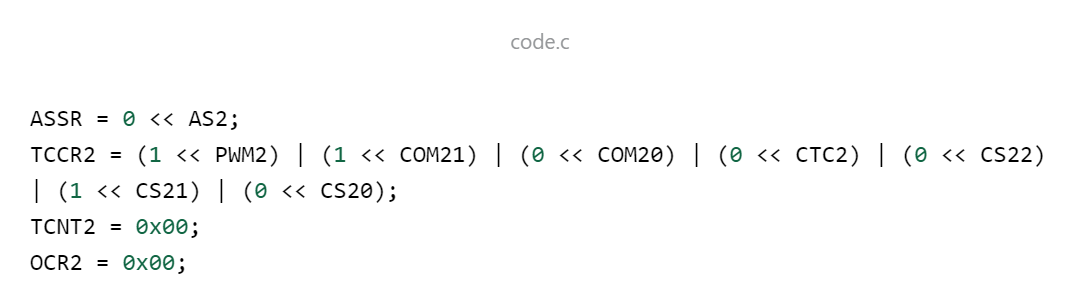
* **خواندن ADC: مقدار ADC خوانده شده و به OCR2 اختصاص می‌یابد تا PWM تنظیم شود.**
* **کنترل جهت موتور: با استفاده از دکمه‌ها، جهت چرخش موتور کنترل می‌شود:**
  + **PINB.0: موتور در جهت اول می‌چرخد.**
  + **PINB.1: موتور در جهت دوم می‌چرخد.**
  + **PINB.2: موتور متوقف می‌شود.**
* **OCR2 = read\_adc(0):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

****

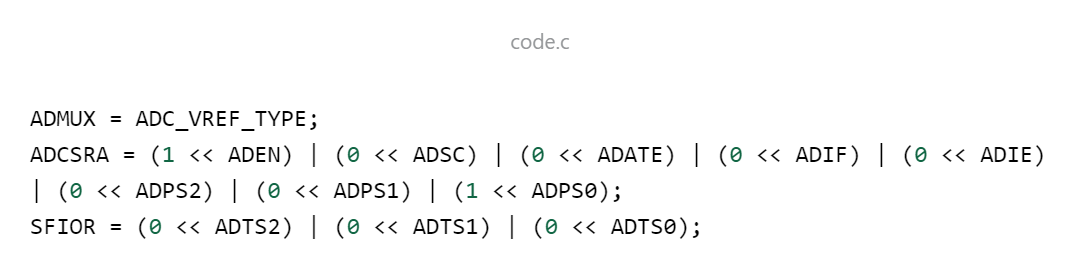
* **DDRA و PORTA**: پورت A به عنوان ورودی تنظیم شده و مقدار اولیه آن صفر است.
* **DDRB و PORTB**: پورت B به عنوان ورودی تنظیم شده و مقدار اولیه آن صفر است.
* **DDRC و PORTC**: پورت C به عنوان خروجی تنظیم شده و مقدار اولیه آن صفر است (فقط دو بیت اول به عنوان خروجی تنظیم شده‌اند).
* **DDRD و PORTD**: پورت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

**تنظیمات تایمر و PWM:**



* **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:**



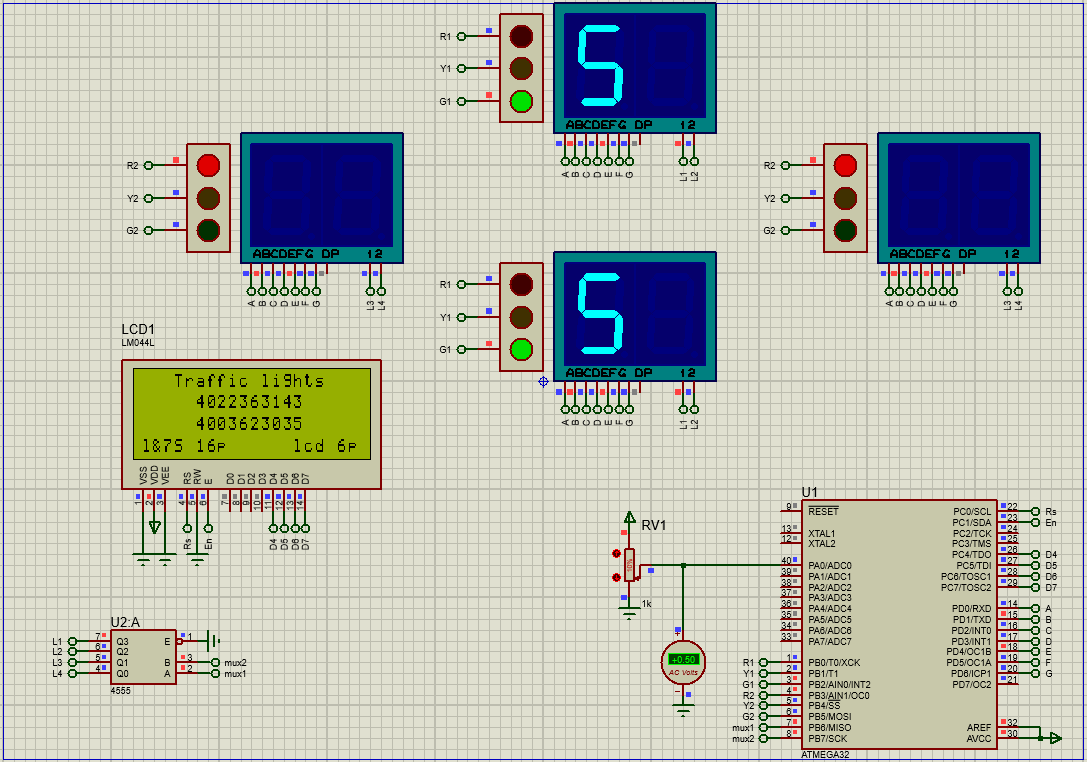
* **ADMUX**: تنظیم نوع مرجع ولتاژ و فعال کردن رجیستر چپ‌چین.
* **ADCSRA**: تنظیمات کنترل و وضعیت ADC.
  + **ADEN**: فعال کردن ADC.
  + **ADPS0**: تنظیم پیش‌تقسیم‌کننده به 2.

**تنظیمات LCD :**



* **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 ATmega32 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 ATmega32 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.

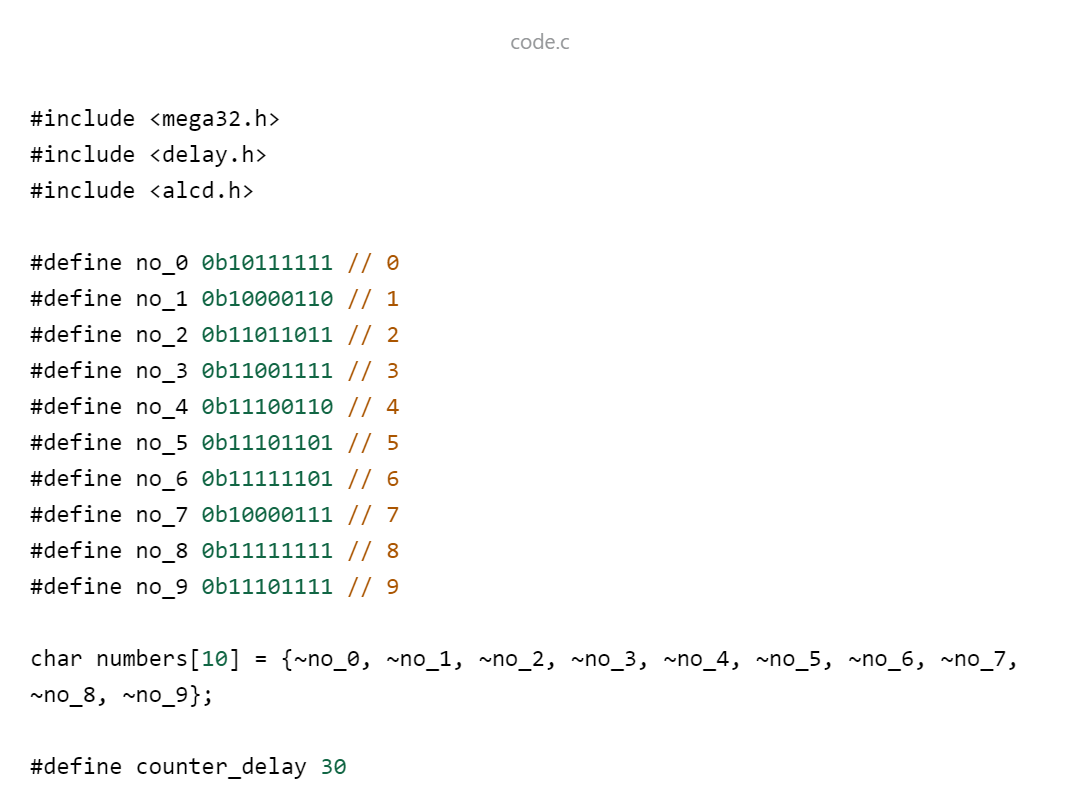
#### **بخش اول**

در این بخش، هدف کنترل یک موتور DC با استفاده از ATmega32 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 L298 برای کنترل موتور و پتانسیومتر است.

#### **بخش دوم**

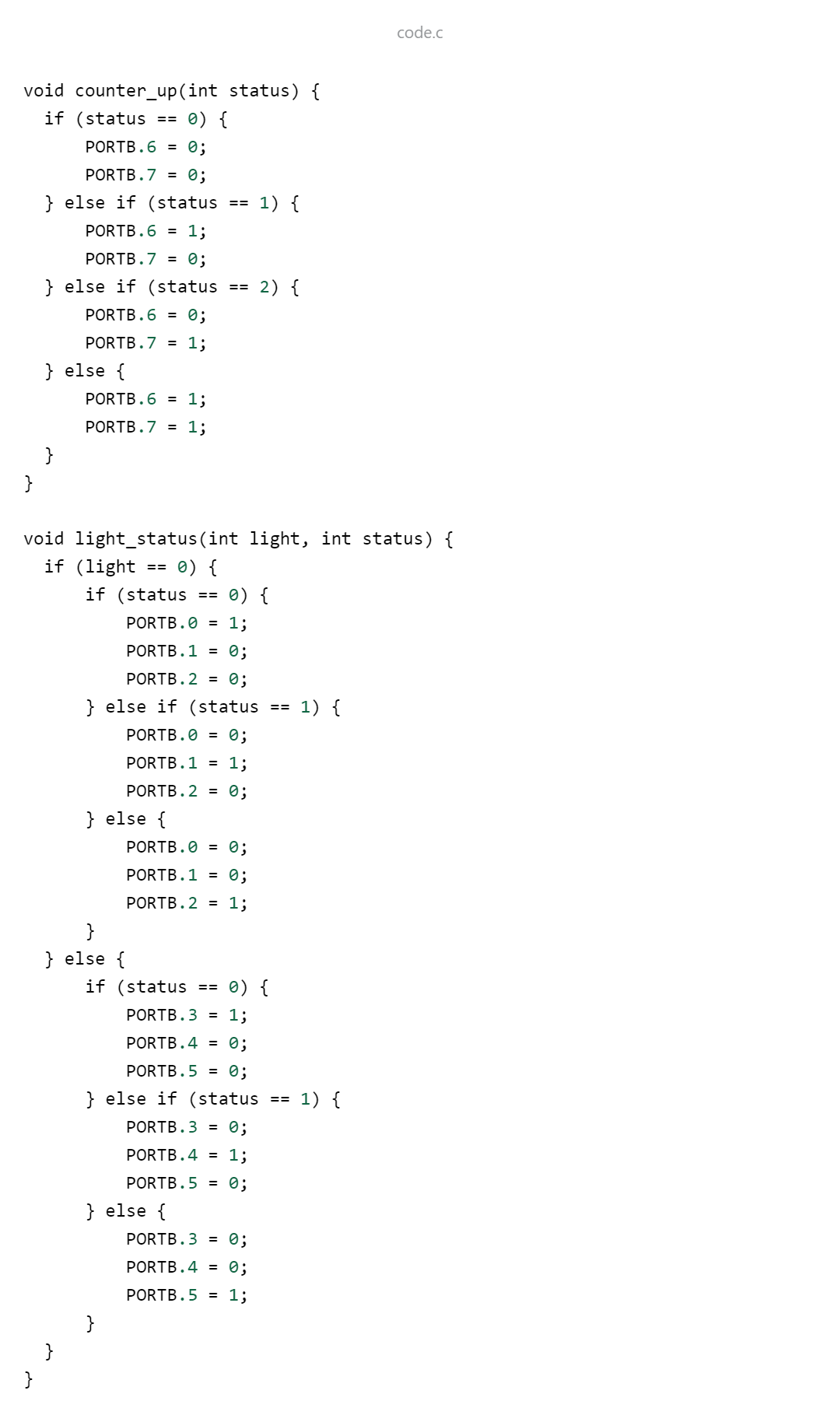
در این بخش، هدف مدیریت چراغ‌های راهنمایی یک چهارراه با استفاده از ATmega32 است. در این چهارراه، چهار چراغ راهنمایی وجود دارد که باید به ترتیب سبز، زرد و قرمز شوند. زمان سبز بودن چراغ‌ها باید حداکثر 60 ثانیه باشد و زمان قرمز بودن کمتر از زمان سبز بودن و به دلخواه تعیین شود. زمان زرد بودن چراغ‌ها حداکثر 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:**



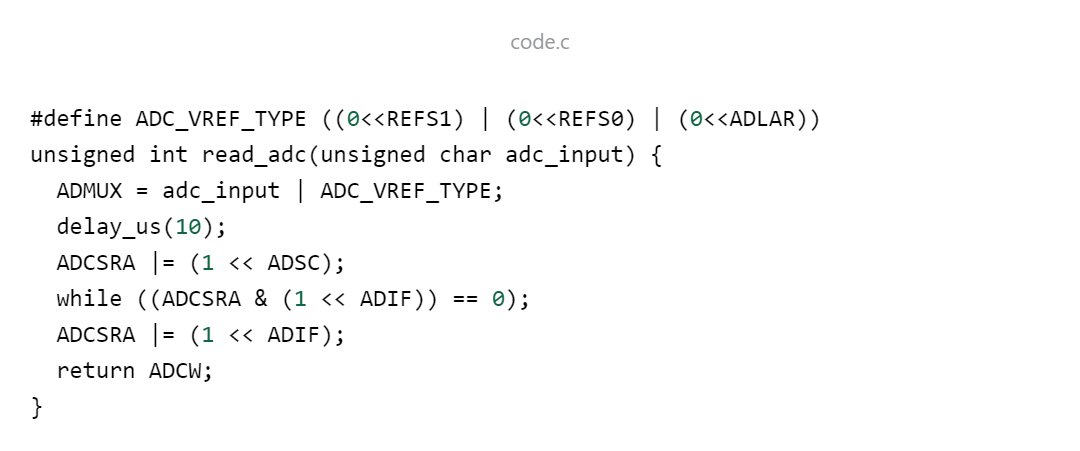
این بخش شامل کتابخانه‌های مورد نیاز و تعاریف اولیه برای نمایش اعداد بر روی سون سگمنت است..

**توابع**



توابع counter\_up و light\_status برای تنظیم وضعیت چراغ‌ها و شمارنده‌ها استفاده می‌شوند.

**خواندن ADC**

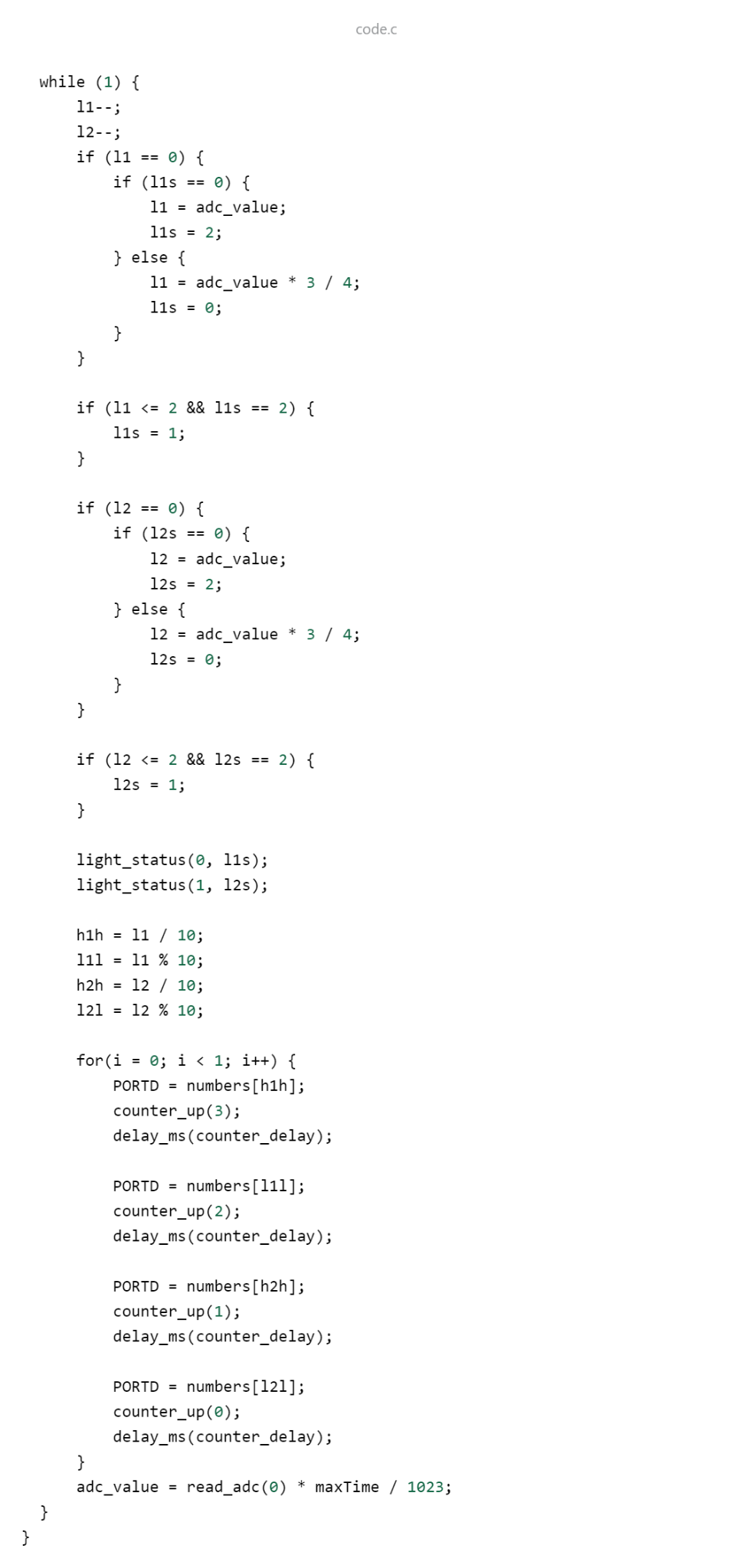


این تابع مقدار ADC را از کانال مشخص شده می‌خواند.

**تابع اصلی**







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 L298**

برای کنترل موتور DC از ماژول L298 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.

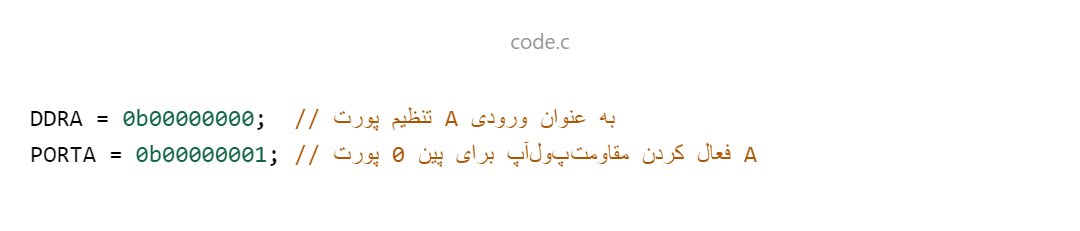
#### **نمایشگر LCD**

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**

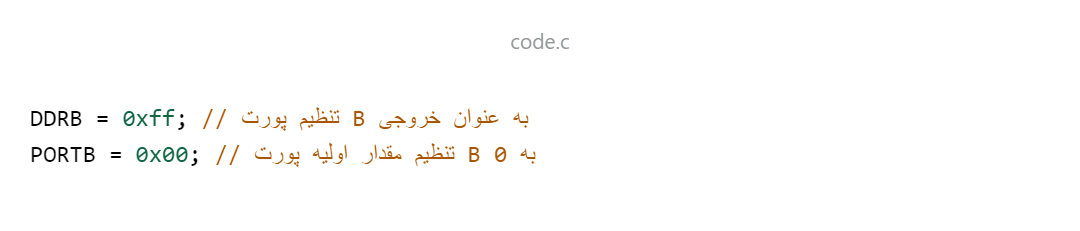
* **پیکربندی پورت‌ها:** تنظیم پورت‌های توضیحات پروژه کنترل چراغ راهنمایی

**A پورت (DDRA و PORTA) :**



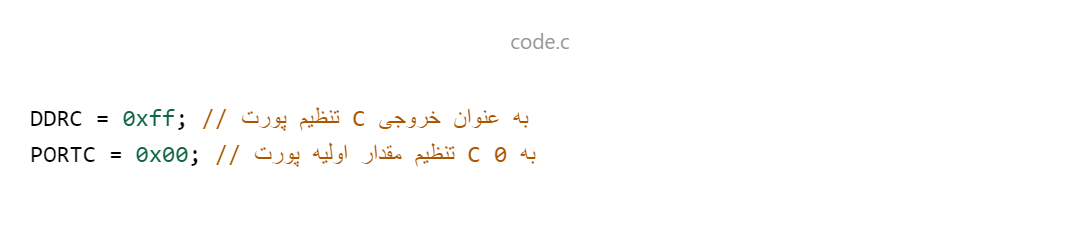
پورت A برای خواندن ورودی‌های آنالوگ از پتانسیومتر استفاده می‌شود. پین 0 از پورت A با فعال کردن مقاومت پول‌آپ داخلی به عنوان ورودی تنظیم شده است.

**B پورت (DDRB و PORTB) :**



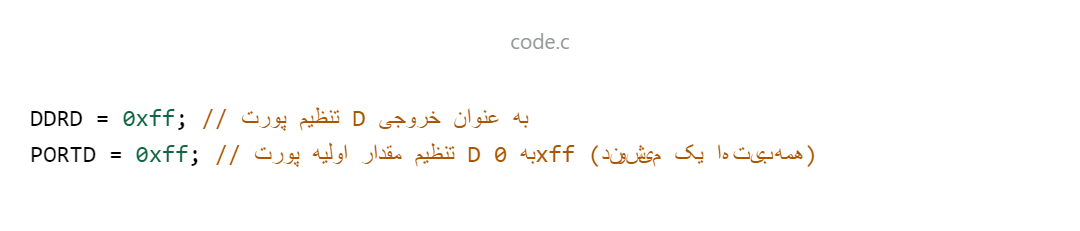
پورت B به عنوان خروجی برای کنترل LED های چراغ راهنمایی و سگمنت‌های شمارنده تنظیم شده است.

**C پورت (DDRB و PORTB) :**



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

**D پورت (DDRB و PORTB) :**



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:



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