نوشتن کتابخانه برای آردوینو

این مطلب قسمت هفدهم از آموزش جامع آردوینو (مبتدی و پیشرفته) است. تا اینجا نحوه استفاده از ماژول‌ها، سنسورها و عملگرهای اصلی را که برای ساخت یک محصول یا اجرای یک پروژه نیاز دارید یاد گرفتید. واقعا خسته نباشید! موضوع دیگری که در استفاده از آردوینو یا هر میکروکنترلر دیگری به شما کمک می‌کند، یاد گرفتن نحوه نوشتن کتابخانه است. در مباحث قبلی غالبا از کتابخانه‌های آماده برای راه‌اندازی ماژول‌ها استفاده کردیم. اما ممکن است همیشه نتوانید کتابخانه مناسبی برای قطعه‌تان پیدا کنید. در این صورت باید دست به کار شده و خودتان کتابخانه‌ای برای آن قطعه خاص بنویسید.

مقدمه

قبلا در مورد تعریف یک تابع در برنامه آردوینو صحبت کرده بودیم. با تعریف توابع مجزا، طول برنامه اصلی کاهش می‌یافت، تحلیل و عیب‌یابی برنامه ساده‌تر می‌شد و حالت ماژولار پیدا می‌کرد. با این وجود، تابع درون برنامه تعریف شده بود و اگر می‌خواستید همان تابع را در برنامه دیگری هم استفاده کنید، مجبور بودید متن آن را در برنامه جدید کپی کنید. کتابخانه این کار را برای شما ساده می‌کند. کتابخانه چیزی نیست جز مجموعه‌ای از توابع و تعاریف که می‌خواهید به برنامه‌تان اضافه شود. همان طور که دیده‌اید افزودن کتابخانه به کد، تنها با یک تعریف ساده در ابتدای برنامه، مثلا به صورت زیر انجام می‌شود:

```
#define <SPI.h>
```

در این آموزش قصد داریم به عنوان نمونه یک کتابخانه را قدم به قدم با هم بنویسیم. اگر خاطرتان باشد در آموزش راه‌اندازی موتور DC با آردوینو از هیچ کتابخانه‌ای استفاده نکردیم؛ این باعث شد تا مجبور شویم چند تابع را درون برنامه اصلی تعریف کنیم و حجم برنامه زیاد شد. برنامه‌ای که در آنجا نوشتیم خیلی ساده بود؛ اگر می‌خواستیم یک برنامه پیچیده بنویسیم که از ماژول‌های مختلف در آن استفاده می‌شد، حجم زیاد برنامه، سردرگم کننده و آزاردهنده می‌شد. به همین دلیل بیایید در

 |

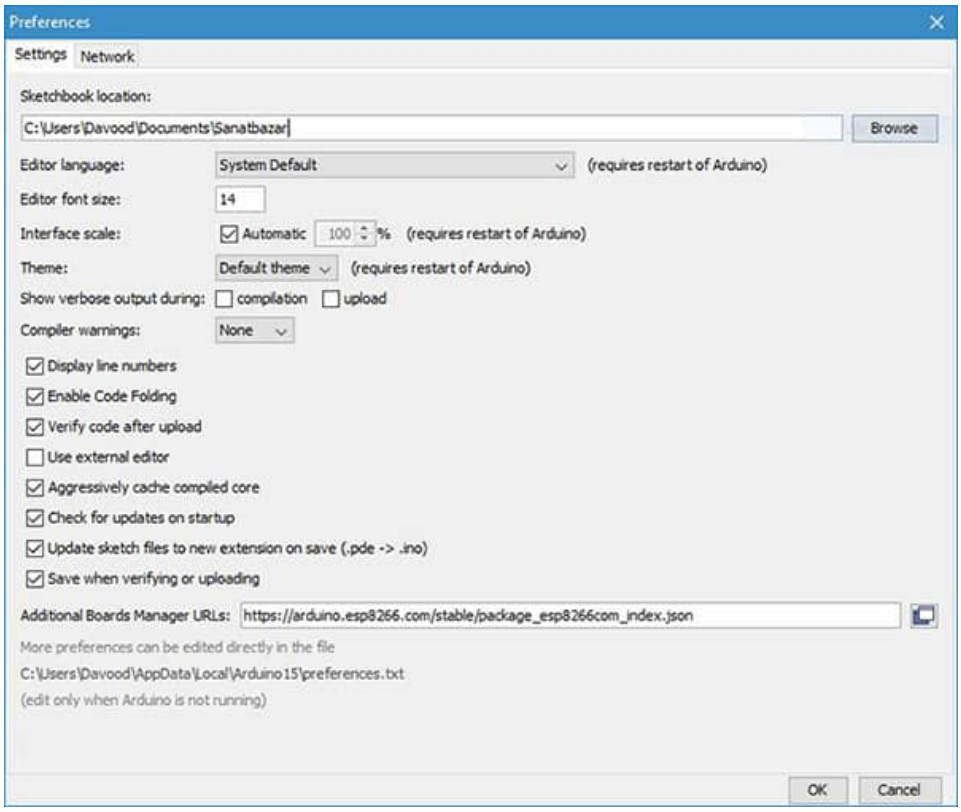
اینجا کتابخانه‌ای برای موتور DC بنویسیم. در آخر خواهید دید که این کار چقدر حجم کدتان را کم می‌کند. مستقیما به بخشی از آموزش موتور DC می‌رویم که درایور L298 را معرفی کردیم. از آنجا که کتابخانه‌های آردوینو در زبان C++ نوشته می‌شوند، حداقل به دو فایل به نامdcMotor.hوdcMotor.cppنیاز خواهیم داشت. این دو فایل به ترتیب شامل هدر (header) و بدنه (Body) کتابخانه می‌شوند. نرم‌افزار آردوینو به صورت پیش‌فرض کامپایلر زبان C++ را در خود دارد بنابراین نیاز به کار اضافه‌ای نیست. فایل دیگری نیز به نامkeywords.txtخواهیم نوشت که البته الزامی نیست ولی نوشتن آن به درد می‌خورد. در ادامه در مورد هر فایل صحبت می‌کنیم.

تنظیمات اولیه کتابخانه

پیش از نوشتن متن کتابخانه باید مسیر فولدری که در آردوینو برای نصب کتابخانه‌ها استفاده شده است را پیدا کنید. در ویندوز به صورت پیش‌فرض مسیر کتابخانه‌ها به صورت زیر است:

```
C:\Users\<user name>\Documents\Arduino\libraries
```

این فولدر را می‌توانید در پنجره Preferences در قسمت Sketchbook location پیدا کنید. این مسیر را می‌توانید به دلخواه تغییر دهید.



به مسیر نصب کتابخانه‌ها رفته و یک پوشه جدید به نام کتابخانه‌تان (در اینجا dcMotor) بسازید. تمام فایل‌های مورد نیاز را در این پوشه قرار خواهیم داد.

نوشتن فایل هدر کتابخانه

اگر با زبان C++ آشنایی زیادی ندارید کافیست در مورد هدر همین را بدانید که این فایل مثل فهرستی از تمام چیزهایی است که در کتابخانه وجود دارد. هر زمان که بخواهید از کتابخانه استفاده کنید، باید فایل هدر آن را صدا بزنید. عبارتی که با آن کتابخانه را define می‌کردید، در واقع به فایل هدر آن کتابخانه ارجاع می‌دهد. با صدا زدن کتابخانه، کامپایلر آردوینو متوجه می‌شود که چه توابعی را باید در دسترس قرار دهد. شکل کلی تمام هدرهای آردوینو چیزی شبیه این است:

```
#ifndef dcMotor_h
#define dcMotor_h
```



```
#if ARDUINO >= 100
      #include "Arduino.h"
#else
      #include "WProgram.h"
      #include "pins_arduino.h"
      #include "WConstants"
#endif

// Your code comes here ...

#endif
```

اگر دقت کنید در این هدر تمام مطالب بین این خطوط قرار دارد:

```
#ifndef dcMotor_h
#define dcMotor_h

#endif
```

هدف از نوشتن این خطوط جلوگیری از تعریف چندباره این کتابخانه است. در این دستور عبارت#ifndefمعادل if not defined است. این روش در زبان C++ بسیار متداول است. چنانچه می‌خواهید از توابع و ثوابت استاندارد آردوینو استفاده کنید (خب قطعا می‌خواهید این کار را بکنید!) نوشتن قسمت دوم کد فوق نیز الزامی است. اگر دقت کنید در فایل هدر توابع شرطی به همراه # می‌آیند. این نحوه نگارش به معنی Preprocessor بودن این دستورها است. Preprocessor ها قبل از اجرای هر فرمانی در C++ اجرا می‌شوند.

هر هدر با تعریف یک کلاس که هم‌نام کتابخانه است شروع می‌شود. درون کلاس معمولا دو بخش اصلی وجود دارد: private و public

```
class dcMotor
{
private:
      uint8_t _input1Pin;
      uint8_t _input2Pin;
      uint8_t _enablePin;

public:
      dcMotor(uint8_t input1Pin, uint8_t input2Pin, uint8_t enablePin);
      void forwardTurn(uint8_t speed);
      void backwardTurn(uint8_t speed);
      void forwardSweep(int intervals);
      void backwardSweep(int intervals);
};
```

در بخش public تعاریفی نوشته می‌شود که از بیرون کتابخانه قابل دسترسی است در حالی که تعاریف private از بیرون در دسترس نبوده و مربوط به خود کتابخانه است. در نهایت فایل هدر به صورت زیر خواهد بود:

```
/*
SanatBazar
Arduino Turorial Series
Author: Davood Dorostkar
Website: www.sanatbazar.com

*/

#ifndef dcMotor_h
#define dcMotor_h

#if ARDUINO >= 100
      #include "Arduino.h"
#else
      #include "WProgram.h"
      #include "pins_arduino.h"
      #include "WConstants"
#endif

class dcMotor
{
private:
      uint8_t _input1Pin;
      uint8_t _input2Pin;
      uint8_t _enablePin;
```

```
public:
      dcMotor(uint8_t input1Pin, uint8_t input2Pin, uint8_t enablePin);
      void forwardTurn(uint8_t speed);
      void backwardTurn(uint8_t speed);
      void forwardSweep(int intervals);
      void backwardSweep(int intervals);
};

#endif
```

بدنه کتابخانه

فایل هدر فقط شامل اسامی توابع است و توضیح آنها در فایلcpp.یا بدنه کتابخانه می‌آید. فایل هدر بخش اصلی کتابخانه است و مشخص می‌کند که چه توابعی در دسترس برنامه خواهد بود. در صورتی که تابعی در فایلh.معرفی (declaration) شده باشد، کامپایلر به دنبال تعریف (definition) آن در فایلcpp.می‌گردد. فایل بدنه معمولا با عبارت"dcMotor.h" #include#شروع می‌شود. پس از آن تمام مواردی که در فایل هدر معرفی شده است، تعریف می‌شوند. هنگام تعریف یک تابع، برای اینکه مشخص کنیم که قصد تعریف یک تابع درون فایل هدر را داریم، از اپراتور::استفاده می‌کنیم. مثلا عبارتdcMotor::forwardSweepنشان می‌دهد که می‌خواهید به تابع forwardSweep درون هدر dcMotor اشاره کنید. بدنه این کتابخانه به صورت زیر است:

```
/*
SanatBazar
Arduino Turorial Series
Author: Davood Dorostkar
Website: www.sanatbazar.com

*/

#include "dcMotor.h"

dcMotor::dcMotor(uint8_t input1Pin, uint8_t input2Pin, uint8_t enablePin)
{
      pinMode(input1Pin, OUTPUT);
      pinMode(input2Pin, OUTPUT);
      pinMode(enablePin, OUTPUT);
      _input1Pin = input1Pin;
      _input2Pin = input2Pin;
      _enablePin = enablePin;
}

void dcMotor::forwardTurn(uint8_t speed)
{
      digitalWrite(_input1Pin, HIGH);
      digitalWrite(_input2Pin, LOW);
      analogWrite(_enablePin, speed);
}

void dcMotor::backwardTurn(uint8_t speed)
{
      digitalWrite(_input1Pin, LOW);
      digitalWrite(_input2Pin, HIGH);
      analogWrite(_enablePin, speed);
}

void dcMotor::forwardSweep(int intervals)
{
      for (int i = 0; i < 255; i++)
      {
            forwardTurn(i);
            delay(intervals);
      }
      for (int i = 255; i > 0; i--)
      {
            forwardTurn(i);
            delay(intervals);
      }
}

void dcMotor::backwardSweep(int intervals)
{
      for (int i = 0; i < 255; i++)
      {
```

```
            backwardTurn(i);
            delay(intervals);
      }
      for (int i = 255; i > 0; i--)
      {
            backwardTurn(i);
            delay(intervals);
      }
}
```

نوشتن فایل keywords

تا اینجا تعریف کتابخانه تمام شده است. در برنامه‌نویسی قابلیتی وجود دارد که به آن syntax highlighting می‌گویند. Syntax highlighting انواع متغیرها و توابع موجود در برنامه شما را شناسایی کرده و هر کدام را به رنگ خاصی در می‌آورد و باعث می‌شود برنامه خواناتر شود. در حالت عادی نرم‌افزار آردوینو syntax های کتابخانه شما را شناسایی نمی‌کند و این کار را باید به صورت دستی انجام دهید. به این منظور یک فایل دیگر به نامkeywords.txtبسازید و دستورهای زیر را در آن بنویسید:

```
# SanatBazar
# Arduino Turorial Series
# Author: Davood Dorostkar
# Website: www.sanatbazar.com


# Use KEYWORD1 for constructor

dcMotor KEYWORD1

# Use KEYWORD2 for functions

forwardTurn KEYWORD2
backwardTurn KEYWORD2
forwardSweep KEYWORD2
backwardSweep KEYWORD2
```

در این کد دو نوع رنگ تعریف شده است: KEYWORD1 که برای constructor در نظر گرفتیم (در ادامه در مورد آن توضیح می‌دهم) و KEYWORD2 که مربوط به توابع است.

استفاده از کتابخانه

ساده‌ترین روش استفاده از کتابخانه این است که پوشه شامل تعاریف کتابخانه را درون پوشه کتابخانه‌های آردوینو قرار دهید (در این مثال از ابتدا پوشه را در همان محل ایجاد کردیم). می‌توانید کتابخانه را به صورت فایلzip.فشرده کنید و در دسترس افراد دیگر هم قرار بدهید. فایل فشرده را می‌توانید از مسیر ...sketch□Import Library نصب کنید. روش‌های مختلف نصب کتابخانه در آموزش استفاده از نرم‌افزار آردوینو توضیح داده شده است. یادتان باشد که قبل از استفاده از کتابخانه باید آن را در ابتدای برنامه فراخوانی کنید. آخرین نکته در مورد کتابخانه استفاده از constructor است. اگر به خاطر داشته باشید زمانی که از کتابخانه‌ها استفاده می‌کردیم، معمولا در ابتدا یک شیء نمونه از کلاس آن کتابخانه ایجاد می‌کردیم. مثلا در آموزش استفاده از سروو موتور پس از فراخوانی کتابخانه یک سروو موتور ایجاد کردیم:

```
Servo servoMotor(10);
```

هدف از این کار فراخواندن constructor است. Constructor تابعی است که درون بخش public هدر تعریف می‌شود و کتابخانه را آماده استفاده می‌کند. Constructor می‌تواند مقادیری را به عنوان ورود دریافت کند:

```
dcMotor(uint8_t input1Pin, uint8_t input2Pin, uint8_t enablePin);
```

یا اینکه ورودی آن را خالی بگذارید:

```
dcMotor();
```

این موضوع بستگی به استفاده‌ای که از آن می‌کنید دارد. ویژگی‌های constructor به صورت زیر است:

- نام constructor دقیقا با نام کلاس یکی است
- قبل از نام constructor هیچ نوعی نوشته نمی‌شود
- زمانی که یک نمونه از کلاس مورد نظر ساخته می‌شود، به طور خودکار constructor فراخوانی می‌شود
- اگر در کلاس constructor تعریف نکنید، خود کامپایلر یک constructor خالی فرض می‌کند

و اما آخرین نکته در مورد کتابخانه؛ برای صدا زدن یک تابع درون کتابخانه از.استفاده می‌شود (مانند کد زیر). توجه کنید که تمام توابع کتابخانه را باید به همراه ورودی‌های مناسب هر کدام صدا بزنید؛ در غیر این صورت با خطا روبرو می‌شوید. با استفاده از کتابخانه‌ای که نوشتیم، برنامه طولانی راه‌اندازی موتور DC به صورت زیر در خواهد آمد:

```
/*
SanatBazar
Arduino Turorial Series
Author: Davood Dorostkar
Website: www.sanatbazar.com

*/

#include <dcMotor.h>
#define enablePin 10
#define input1Pin 9
#define input2Pin 8
dcMotor dc(input1Pin, input2Pin, enablePin);

void setup()
{
}

void loop()
{
   dc.forwardSweep(30);
   dc.backwardSweep(30);
   delay(2000);
}
```

کتابخانه dcMotor را می‌توانید در زیر دانلود کنید و اگر خواستید امکانات دیگری به آن اضافه کنید:

دانلود کتابخانه موتور DC

نتیجه‌گیری

در این آموزش با یکی از بخش‌های بسیار مهم در برنامه‌نویسی یعنی نوشتن کتابخانه آشنا شدید. اگر آموزش‌های قبلی را دنبال کرده‌اید، زمان آن فرا رسیده که وارد دنیای برنامه‌نویسان حرفه‌ای شوید!

در آموزش بعدی، نحوه استفاده از نرم‌افزار MATLAB و سیمولینک به همراه آردوینو را یاد خواهید گرفت.

نظرات شما باعث بهبود محتوای آموزشی ما می‌شود. اگر این آموزش را دوست داشتید، همین‌طور اگر سوالی در مورد آن دارید، از شنیدن نظراتتان خوشحال خواهیم شد.

