دیود نورانی چیست؟

دیود قطعه‌ای الکتریکی اســـت که جریان را در یک جهت از خود عبور می‌دهد و در جهت دیگر در برابر عبور جریان از خود مقاومت بالایی نشان می‌دهد. برای اطلاعات بیشتر درباره‌ی دیود به بخش پیش‌نیازهای الکتریکی مراجعه کنید. دیود نورانی نوعی دیود اســت که اگر در جهت درســت از آن جریان الکتریکی عبور داده شــود، نور تولید می‌کند.

تاریخچه‌ی دیود نورانی

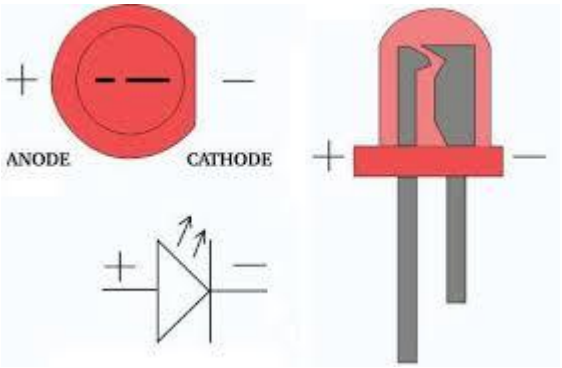
اولین دیود های نورانی در ســال ۱۹۶۲ میلادی و تنها با رنگ قرمز به صــورت صــنعتی تولید و وارد بازار شــدند. دیودهای نورانی ســبز، آبی، زرد و نارنجی در دهه ۷۰ میلادی تولید شــدند. بهره نوری دیودهای نورانی رفته رفته افزایش یافتند تا اینکه در دهه ۸۰ و اوایل دهه ۹۰ میلادی، به صـــورت گروهی و با کارایی بســـیار بالا وارد بازار شــدند. دیودهای نورانی اولیه به علت بهره پایینشــان، تنها در مدارات الکترونیکی اســتفاده می‌شــدند، اما در حال حاضر همان طور که می‌دانید وارد مصارف خانگی شده‌اند و جای لامپ‌های کم مصرف را گرفته‌اند.

طیف نوری و انواع دیود نورانی

طیف نوری دیودهای نوری تقریبا تمامی طیف نور را در بر می‌گیرد. این طیف شـــامل تمامی نور مرئی، مادون قرمز و فرابنفش ا ست. شدت نور تولیدی دیود نورانی به جریان آن بستگی دارد، برای همین در بعضی از موارد برای راه‌اندازی دیودهای نوری از منبع جریان استفاده می‌کنند. معمولا دیودهای نوری از توان پایینی برخوردارند و بسیار کم مصرف هستند. این خواص باعث شده تا کاربرد این قطعه الکتریکی بسیار بالا باشد. در چراغ‌های راهنمایی رانندگی، علائم سطح شهر، چراغ‌های خودرو، روشنایی در موزه‌ها (به دلیل نداشتن پرتو ماورای بنفش برای اشیاء داخل موزه مضر نیستند) و در بسیاری از موارد دیگر کاربرد دارند.

تشخیص و نحوه عملکرد دیود

دیود نورانی همان طور در قبل گفته شـد دیودی اسـت که عبور جریان الکتریکی در جهت درسـت از داخل آن، باعث تولید نور می‌شـود. پس در ابتدا باید بتوان جهت درسـت دیود نورانی یا همان پایه مثبت و منفی آن را به درستی تشخیص دهیم. برای این کار ۲ راه وجود دارد. وقتی شما دیود را در ابتدا خریداری می‌کنید، یکی از پایه های آن بلندتر از دیگری است. پایه بلندتر پایه مثبت است. این راه فقط مناسب برای دیودهایی است که قبل از آن طول پایه‌های آنها تغییر نکرده باشد. به بیان دیگر عبور جریان الکتریکی فقط از این پایه به پایه منفی ممکن است، اگر به صورت معکوس این کار انجام شود باعث شکسته شدن دیود می‌شود که درباره این موضوع به صورت کامل در بخش پیش نیازهای الکتریکی توضیح داده شده است. راه حل دوم نگاه کردن به داخل دیود است. اگر به شـکل زیر توجه کنید می‌بینید که در داخل یک دیود نورانی ۲ صـفحه وجود دارد که یکی از دیگری کوچکتر اسـت. صـفحه‌ی کوچکتر نمایانگر پایه مثبت بوده و صـفحه بزرگتر نمایانگر پایه منفی اسـت. این روش برای پیدا کردن جهت دیود هایی مناسب است پایه های آنها در گذشته بریده شده باشند. (مثلا برای لحیم کردن در داخل یک برد چاپی)

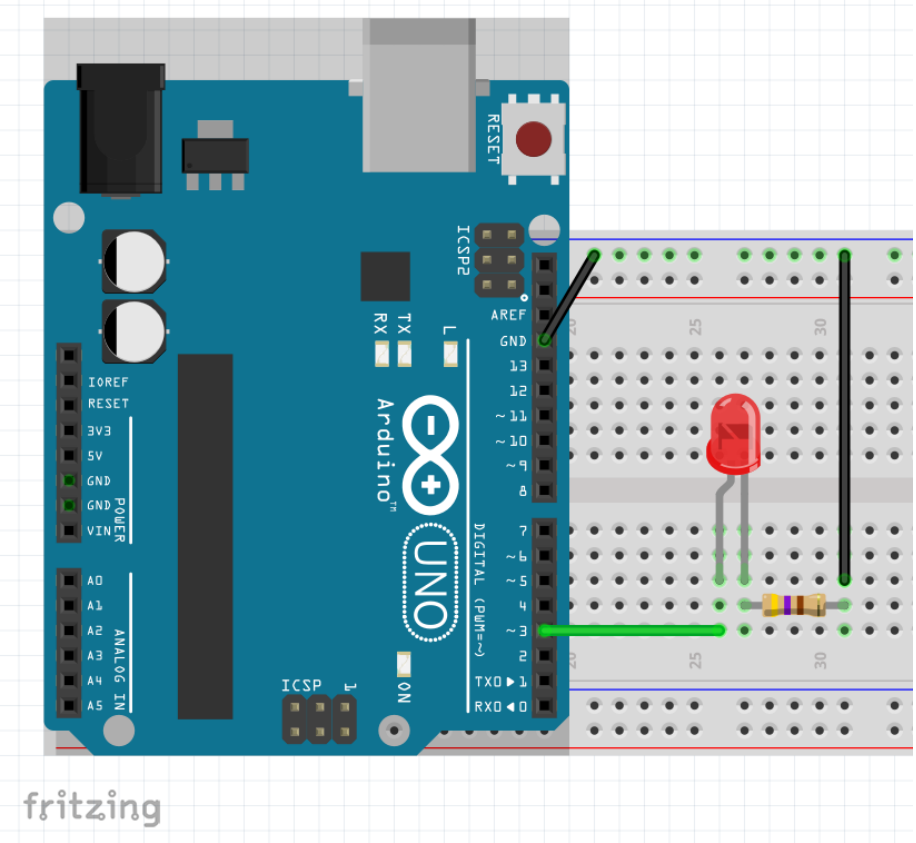


مدار دیود نورانی

همان طور که در بخش معرفی آردینو گفته شـد، بردهای آردینو به هنگام نمایش مقدار ۱ باینری ولتاژ ۵ ولت را در خروجی خود تولید می‌کنند. حال فرض کنید که می‌خواهیم توسـط یکی از پین‌های آردینو یک ال‌ای‌دی را خاموش و روشن کنیم. این کار را می‌توانیم با صفر و یک کردن مقدار باینری آن پایه انجام بدهیم.

مدار راه اندازی ال‌ای‌دی بسـیار سـاده اسـت. برای راه اندازی یک ال‌ای‌دی فقط نیاز اسـت که بعد از پایه منفی ال‌ای‌دی، مقاوتی قرار بدهیم تا جریان الکتریکی را محدود کند. برای این کار از مقاومت ۴۷۰ اهمی اسـتفاده

می‌کنیم. این مقاومت با کد رنگی زرد-بنفش-قهوه‌ای کدگذاری می‌شـــود. برای اطلاعات بیشـــتر از کد رنگی به بخش پیش‌نیازهای الکتریکی مراجعه کنید. بقیه مدار نیز مانند شکل بسته می‌شود. بعد از انجام بخش مربوط به مدارات الکتریکی، حال نوبت به آن رســـیده که بخش مربوط به برنامه‌نویســـی را انجام بدهیم که در این بخش یعنی قطع و وصل کردن پین شماره ۳ برد آردینو. در این مدار اگر ولتاژ ۵ ولت (۱ باینری) به پین شماره ۳ داده شود، ال‌ای‌دی روشن خواهد شد و اگر ۰ ولت (۰ باینری)، ال‌ای‌دی خاموش خواهد شد.



چشمک زدن دیود نوری بدون توقف

در این بخش دوباره می‌خواهیم دیود نوری را خاموش و روشن کنیم. در بخش معرفی فضای برنامه نویسی دیود نوری داخلی آردینو را که به صـورت پیش فرض روی تمامی برد های آردینو هسـت خاموش-روشـن کردیم. در اینجا می‌خواهیم همان کار را انجام دهیم با این تفاوت که دیگر دیود نوری داخلی برد نیست بلکه مداری خارجی است و دوم اینکه نمی‌خواهیم از تابع delay() استفاده کنیم. همان طور که گفته شد تابع delay تمامی برنامه را برای مدت زمان مشخصی متوقف می‌کند. این مشکل بزرگی است. در حالت واقعی شما همیشه از میکروکنترلر

خود انتظار دارید که پارامترهایی را بخواند و دستورهایی بدهد. اگر از تابع delay استفاده کنید در واقع باعث شدید که میکروکنترلر برای مدتی هیچ کاری انجام ندهد که این امر مشکل ساز است. به طور مثال شما سیستم اطفاء حریقی ساخته‌اید. اگر این سیستم در لحظه‌ای که باید، آب را وصل نکند ممکن است مشکلات بسیار زیادی به بار بیاید. برای همین استفاده از این تابع به جز در شرایط خاص مناسب نیست و توصیه می‌شود از تابع دیگری به نام millis استفاده کنید.

millis ( )

فراخوانی این تابع مدت زمانی را که میکروکنترلر در حال اجرا کردن برنامه فعلی بوده است در واحد میلی ثانیه بر می‌گرداند. عدد این تابع نیز بعد از ۵۰ روز صفر می‌شود. به بیان ساده‌تر از ابتدای شروع برنامه شمارنده‌ای در حال کار است که هر شمارش آن زمان به خصوصی طول می‌کشد. آخرین عدد این شمارنده در جایی ذخیره شده که توسط این تابع، شما می‌توانید زمان مورد نیاز برای آن تعداد شمارش را دریافت کنید.

```
time = millis()
long time_1 = millis();              // خروجی تابع در متغیر اول ریخته میشود
long time_2 = millis();              // خروجی تابع در متغیر دوم ریخته میشود
long delta_time = time_2 - time_1    // تفاضل زمان ها در متغیر جدیدی ذخیره میشود
```

abs ( )

این تابع مقدار قدر مطلق را بر می‌گرداند. به بیان ساده‌تر اگر عدد داده شده به تابع بزرگ‌تر مساوی صفر باشد، خود آن عدد را بر می‌گرداند و اگر کمتر از صفر باشد، قرینه آن را بر می‌گرداند. به همین دلیل خروجی این تابع همواره مثبت است.

```
x_absolute = abs(x)
int a = abs(-1);   >> a : 1
int b = abs(1);    >> b : 1
```

کد چشمک زن بدون توقف

باید توجه کرد که پایه مثبت دیود به پین شماره ۳ وصل شده پس باید وضعیت این پین را تغییر دهیم تا دیود شروع به چشمک زدن کند. برای استفاده از تابع millis باید در دو زمان مختلف این تابع را فراخوانی کنیم و اگر اختلاف این ۲ زمان بیشتر از مکث مورد نظر ما بود وضعیت دیود را تغییر دهیم.

توجه: به هنگام آپلود کردن کد بر روی برد حتما برد را از مدار خارج کنید و بعد از آپلود کردن، برد را در مدار بگزارید و حتما تغذیه برد را از یک منبع تأمین کنید نه از دستگاهتان.

```
int led = 3;      // شماره پین دیود
int time_1 = 0;   // زمان اولیه در این متغیر ذخیره میشود
int time_2 = 0;   // زمان ثانویه در این متغیر ذخیره میشود
int stat = 0;     // وضعیت قبلی دیود نوری در این متغیر ذخیره میشود 0 به معنای خاموش و 1 به معنای روشن
void setup() {
  pinMode(led, OUTPUT);      // مشخص نمودن حالت پین
  digitalWrite(led, LOW);    // اطمینان از خاموش بودن دیود نوری
  time_1 = millis();
}
void loop() {
  time_2 = millis();
  if (time_2 - time_1 >= 1000){    // چک کردن مدت زمان مکث
    time_1 = time_2;
    if (stat == 0){
      digitalWrite(led, HIGH);    // روشن کردن ال ای دی چون قبلا خاموش بوده
      stat = 1;
    }
    else {
      digitalWrite(led, LOW);    // خاموش کردن ال ای دی چون قبلا روشن بوده
      stat = 0;
    }
  }
  // تکرار این حلقه تا چشمک زن ایجاد شود
}
```

کد اصلاحی چشمک زن بدون توقف

توجه: کد قبلی در زمانی ممکن است دچار یک باگ شود. همان طور که در تعریف تابع millis گفته شد، این تابع بعد از ۵۰ روز دوباره ۰ می‌شود. اول اینکه به هنگام صفر شدن ممکن است متغیری که زمان اولیه را در خود ذخیره می‌کند قبل از صفر شدن زمان را ذخیره کند و سپس تابع صفر شود و بعد متغیر زمان ثانویه مقداردهی شود. اینگونه متغیر زمان اولیه بزرگتر از متغیر زمان ثانویه است و تفاضل این دو منفی خواهد بود. این در حالی است که شرط ما بر روی "تفاضل بزرگتر مساوی صفر" گذاشته شده است. دوم اینکه یک متغیر int فقط می‌تواند تا عدد ۳۲۷۶۸ را در خود ذخیره کند ولی تابع millis تا ۵۰ روز که معادل ۴,۳ میلیارد میلی ثانیه است صفر نمی‌شود. به همین دلیل ساختار داده‌ای int نمی‌تواند این عدد را بعد از یک مدت در خود ذخیره کند (تابع millis خروجی‌اش یک متغیر unsigned long است).

برای مشکل اول ما باید به جای استفاده از تفاضل دو عدد از اختلاف این ۲ عدد یا همان قدر مطلق تفاضل آنها استفاده کنیم. برای این کار از تابع abs استفاده خواهیم کرد.

برای حل مشکل دوم ۲ راه حل داریم. اول اینکه از داده‌هایی استفاده کنیم که فضای بیشتری از حافظه را اشغال می‌کند و توانایی ذخیره کردن اعداد بزرگتری را دارد. به طور مثال می‌توانیم از unsigned long استفاده کنیم. این راه حل راه حل مناسبی نیست چون خیلی راحت است. در واقع ما با این کار فضای حافظه‌مان را بی دلیل اشغال می‌کنیم و این کار بسیار نسنجیده‌ای است. راه بهتری وجود دارد. در بخش پیش نیاز برنامه نویسی به عملگری اشاره شد که خروجی آن باقی مانده ۲ عدد بود (%). ما می‌توانیم باقی مانده تابع millis را بر ۱۰۰۰۰ پیدا کنیم و هر وقت این باقیمانده به اندازه ۱۰۰۰ واحد تغییر کرد عملیات تغییر وضعیت را انجام دهیم. البته این راه حل نیز مشکلی بزرگ دارد آن هم اینکه توان پردازشی قابل توجهی را به خاطر عملگر باقی مانده اشغال می‌کند (البته در مثالهای ما تفاوت این ۲ راه حل دیده نمی‌شود ولی در پروژه‌های بخصوصی شاید با این مشکلات روبه‌رو شوید). در آینده با عملگرهای بیتی آشنا می‌شویم و می‌توانیم همین مثال را بدون توان پردازشی بالا حل کنیم.

```
int led = 3;      // شماره پین دیود
int time_1 = 0;   // زمان اولیه در این متغیر ذخیره میشود
int stat = 0;     // وضعیت قبلی دیود نوری در این متغیر ذخیره میشود 0 به معنای خاموش و 1 به معنای روشن
void setup() {
  pinMode(led, OUTPUT);      // مشخص نمودن حالت پین
  digitalWrite(led, LOW);    // اطمینان از خاموش بودن دیود نوری
  time_1 = millis();
}
void loop() {
  if ((millis() % 10000) - time_1 >= 1000){    // چک کردن مدت زمان مکث
    time_1 = millis() % 10000;
    if (stat == 0){
      digitalWrite(led, HIGH);    // روشن کردن ال ای دی چون قبلا خاموش بوده
      stat = 1;
    }
    else {
      digitalWrite(led, LOW);    // خاموش کردن ال ای دی چون قبلا روشن بوده
      stat = 0;
    }
  }
  // تکرار این حلقه تا چشمک زن ایجاد شود
}
```

کوتاه ترین کد چشمک زن تا به حال

این کوتاه ترین کدی اســت که با معلوماتتان تا به اینجای کار می‌توانید برای یک چشــمک زن ســاده بنویســید (حداقل ما تا به الان توانســته‌ایم). توضــیحات این بخش با خودتان فقط این را بدانید که تابع digitalWrite هر متغیر باینری را قبول می‌کند. یعنی هر عدد بزرگتر از صفر HIGH خواهد بود و صفر LOW خواهد بود.

```
int led = 3;      //  شماره پین دیود
int time_1 = 0;   // زمان اولیه در این متغیر ذخیره میشود
int stat = 0;     // وضعیت قبلی دیود نوری در این متغیر ذخیره میشود 0 به معنای خاموش و 1 به معنای روشن
void setup() {
  pinMode(led, OUTPUT);      // مشخص نمودن حالت پین
  digitalWrite(led, LOW);    // اطمینان از خاموش بودن دیود نوری
  time_1 = millis();
}
void loop() {
  if ((millis() % 10000) - time_1 >= 1000){    // چک کردن مدت زمان مکث
    time_1 = millis() % 10000;
      digitalWrite(led, stat);    // نوشتن وضعیت دیود نوری
      stat = abs (stat - 1);      // عوض کردن وضعیت
  }
  // تکرار این حلقه تا چشمک زن ایجاد شود
}
```

توجه: هیچ وقت کدتان را این قدر ساده نکنید. این کار باعث پیچیده شدن کد می‌شود و تغییر و اصلاح آن بسیار سخت می‌شود. همیشه سعی کنید کدی خوانا و ساده بنویسید که دنبال کردن آن کار ساده‌ای باشد.