به نام خدا

مستندات آموزشی روز اول
رویداد هاردوار

بخش معماری کامپیوتر
سطح مقدماتی- آردوئینو

مقدمه

- به بخش معماری کامپیوتر (سطح ساده) خوش آمدید.
- در صورتی که هرگونه سوال یا ابهامی داشتید از استف رویداد (ما!) بپرسید.
- امیدواریم تجربه خوبی را از شرکت در این رویداد به دست آورید.

بخش ابتدایی رویداد شامل آموزش های مفاهیم مرتبط است که شامل بخش های زیر است:

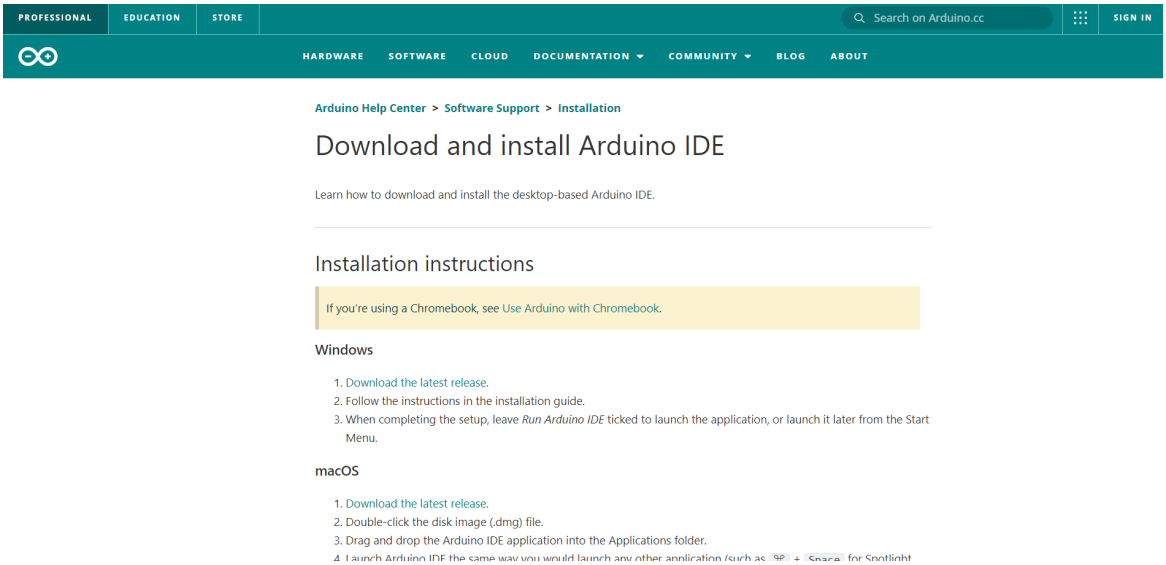
- **نرم افزار:** آموزش نصب نرم افزار مربوط به کدزنی
- **سخت افزار:** آشنایی ابتدایی با آدوئینو + سخت افزارهای مورد نیاز
- **مثال:** یک برنامه ساده با آردوئینو

- هشدار

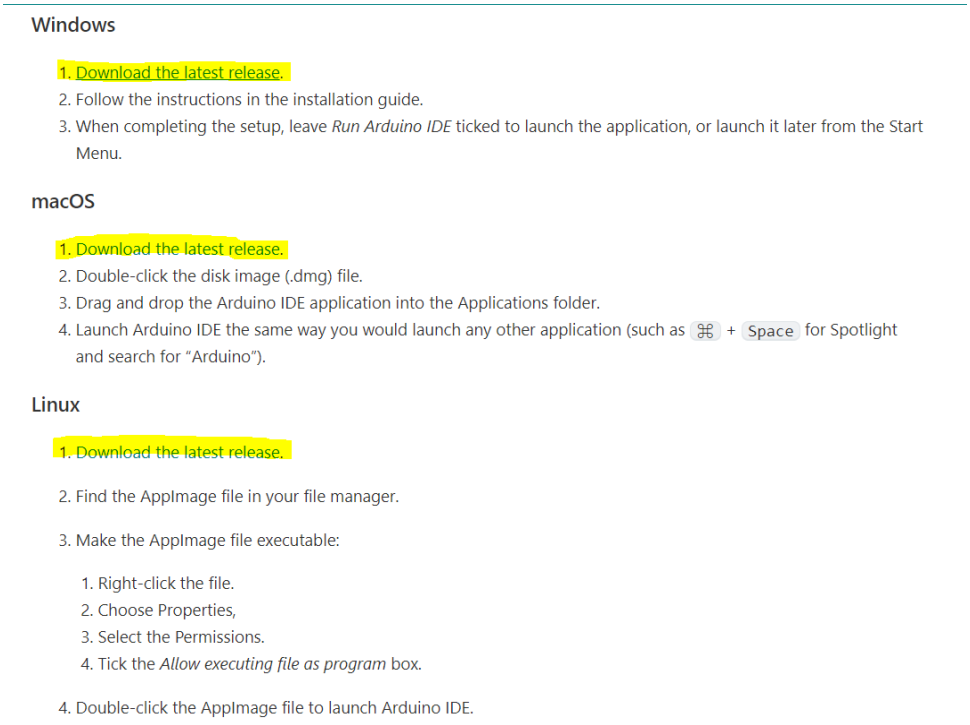
منبع تغذیه سخت افزار شما از طریق USB لپ تاپ تامین میشود، در صورت اتصال کوتاه و یا هرگونه جریان کشی در سیم بندی اشتباه باعث آسیب به پورت USB لپ تاپ میشود.
لطفا قبل از اتصال منبع تغذیه، از صحت اتصالات مدار خود دقت فرمایید.
هرگونه آسیب با مسئولیت شرکت کننده است.

نصب Arduino IDE

1. به اینترنت متصل شوید !
2. وارد این لینک شوید. با چنین صفحه ای روبه رو میشوید:



3. بر اساس نوع سیستم عاملی که دارید، یکی از 3 نسخه نرم افزار (Windows - Linux - macOS) را دانلود کنید (بر روی یکی از 3 بخش هایلایت شده کلیک کنید).



4. پس از انتخاب محل مناسب برای دانلود، فرایند دانلود نرم افزار شروع میشود. حجم فایلی که نیاز است دانلود شود بر حسب سیستم عامل های مختلف بدین شرح است:

Windows	162 مگابایت
macOS	197 مگابایت
Linux	196 مگابایت

5. پس از اتمام دانلود فایل، آن را باز کنید.

فرایند نصب در Windows

بر روی فایل دانلودشده با نام arduino-ide_2.1.0_Windows_64bit.exe راست کلیک کرده و Run as Administrator را بزنید (یا صرفا بر روی آن دو بار کلیک کنید).

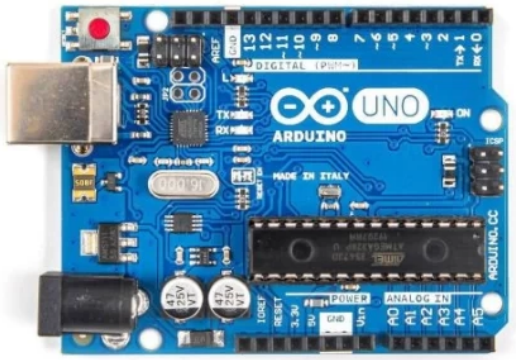
سپس مراحل نصب را بر اساس پیش فرض تعیین شده پیش روید.

6. حالا دیگه Arduino IDE نصب شده، حالا باید Driver و Library های مربوط به Arduino Uno رو روی IDE مون نصب کنیم...

آشنایی با وسایل مورد نیاز

آشنایی با آردوینو

آردوئینو یک میکروکنترلر است که میتوانیم یک سری کارهای سخت افزاری را با آن انجام دهیم.
چیزهای مورد نیاز مرتبط با آردوینو را در اینجا میبینیم.
آردوینو همون مغز متفکری که ما توی یه سیستم سخت افزاری داریم و همه فکرها و این تصمیم ها رو بر اساس برنامه ای که ما نوشتیم انجام میده:



آردوینو یه سری چیز میز داره که ما اینجا باهاش کار داریم.

- ☐ مهمترین بخش آردوییِنو Pin های اون هست که به دو دسته کلی تقسیم میشه:
  - پین های دیجیتال: این پین ها می تونن فقط مقادیر صفر و یک داشته باشن. توی بورد ما عدد O به معنی O ولته و عدده 1 به معنی 5 ولت هست. این پین ها رو میشه هم به عنوان خروجی تعریف کرد و هم ورودی. وقتی پین به صورت خروجی تعریف بشه، میشه روی اون مقادیر دلخواه خودمون رو ریخت و باهاش چیزهایی مثل LED یا وسایل دیگه رو کنترل کرد. وقتی پین به عنوان ورودی تعریف بشه، میشه مقداری که روی اون قرار گرفته رو خوند. مثلاً میشه اون رو به یک کلید متصل کرد و با خوندن مقدار اون متوجه شد که کلید وصله یا نه.
  - پین های آنالوگ: این پین ها دیگه به صورت باینری نیستن و میتونن مقادیر پیوسته رو هم نشون بدن. مثلاً با استفاده از این پیشن میشه ولتاژ دو سر یک پتانسیومتر رو خوند.
- ☐ دو تا چیز داره به اسم TX و RX که برای ارسال و دریافت **سریال** (Serial) ازش استفاده میشه.
- ☐ دو تا جا هم داره به اسم GND و 5V که به ترتیب ولتاژهای **O و 5 ولت** ما هستند. این دو تا نیازند چون منبع تغذیه و انرژی ما همین دو جا هستند.

☐ یه بخش PWM داره که برای تنظیم میزان نور LED میشه از اون استفاده کرد.

ساختار کلی برنامه هایی که برای آردوییو میشه نوشت به صورت زیره:

BareMinimum.ino

```
1  void setup() {
2    // put your setup code here, to run once:
3  }
4
5  void loop() {
6    // put your main code here, to run repeatedly:
7  }
```

همون طور که می بینید برنامه هایی که برای آردوییو مینویسن به زبان C و C++ هست. این برنامه ها دو بخش اصلی setup و loop رو شامل میشن. کد هایی که در بخش setup نوشته میشن فقط یک بار و اون هم موقع روشن شدن بورد اجرا میشن. اما کدهایی که در بخش loop نوشته میشن تا زمانی که بورد روشن هست پشت سر هم اجرا میشن. در واقع برنامه ی اصلی شما در همین بخش loop نوشته میشه.

برنامه های آردوییو با فرمت ino ذخیره میشن و حتماً حواس تون باشه که اسم پوشه با اسم فایل برنامه تون باید یکی باشه.

آشنایی با بردبورد (Bread Board)

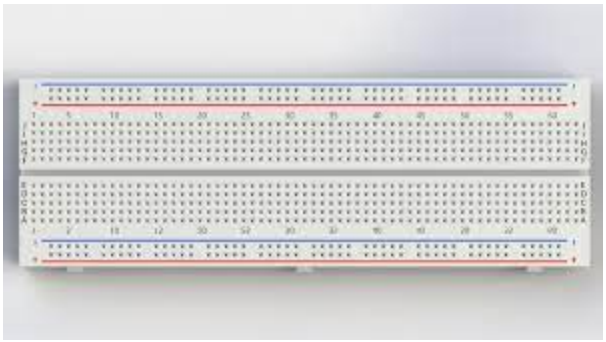
برد برد یه واسطه یا همون رابطه که بین مغز متفکر (آردوئینو) و نیروهای اجرایی مون (سنسور و ماژول های دیگه) کار میکنه. در واقع mother board ما هست که همه چیز روش سوار میشه. یعنی ما بخوایم اطلاعات رو به آردوئینومون منتقل کنیم یا از اون اطلاعاتی رو به سنسورها و لامپ ها و چیزهای بیرون بدیم، توی جفت این حالت ها نیاز به یه واسط داریم به اسم بردبورد.

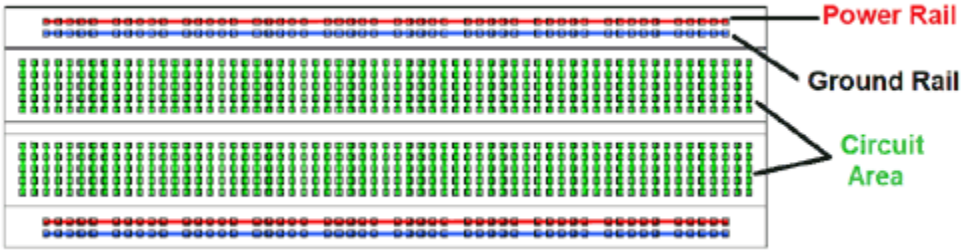
کار خاصی انجام نمیده جز اینکه یه سری اتصالات رو برای ما ساده میکنه.

برد برد شامل دو قسمت میشه که با یک شیار از هم جدا میشن که توی شکل میتونین ببینین.

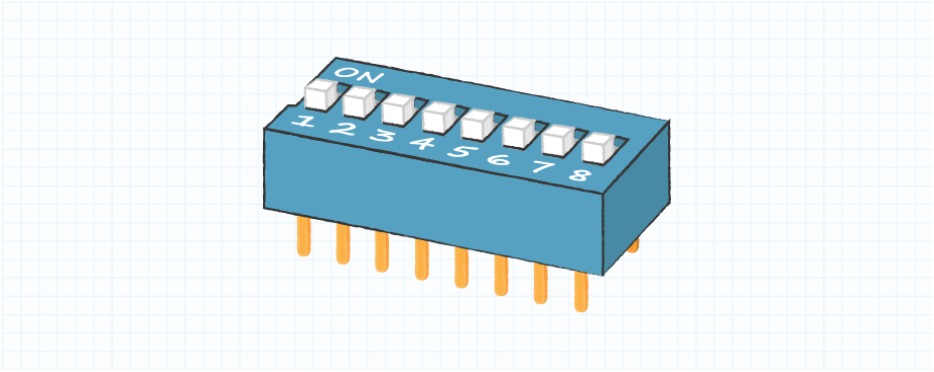
برد بورد به دسته پین های پنج تایی تقسیم شده که هر ستون پنج تایی از پین ها عمودی به هم متصل هستن.

پین های افقی که در لبه ها قرار دارن به صورت سرتاسری به هم متصل شدن.





آشنایی با dip switch



این قطعه در واقع مجموعه ای از چند سوییچ قرار گرفته در کنار میباشد. میتوانید از این سوییچ برای روشن و خاموش کردن مجموعه ای از چراغ ها استفاده کنید.

دقت کنیم که این پایه های این سوییچ باید در دو سمت شیار موجود در میانه برد بورد قرار بگیرد در غیر این صورت اتصال کوتاه بین پایه های سوییچ رخ داده و عملا سوییچ کاربرد خود را از دست میدهد.

مقاومت و سیم

سیم ها برای انتقال جریان بین قطعات رو برد بورد استفاده میشوند که کافی است برای برقراری ارتباط، پین موجود در سر های سیم را در محل مورد نظر قرار دهید.

مقاومت ها هم برای کنترل جریان استفاده میشوند. در طول چالش دو نوع مقاومت در اختیار شما قرار میگیرد که میتوانید طبق نقشه از آنها استفاده کنید.

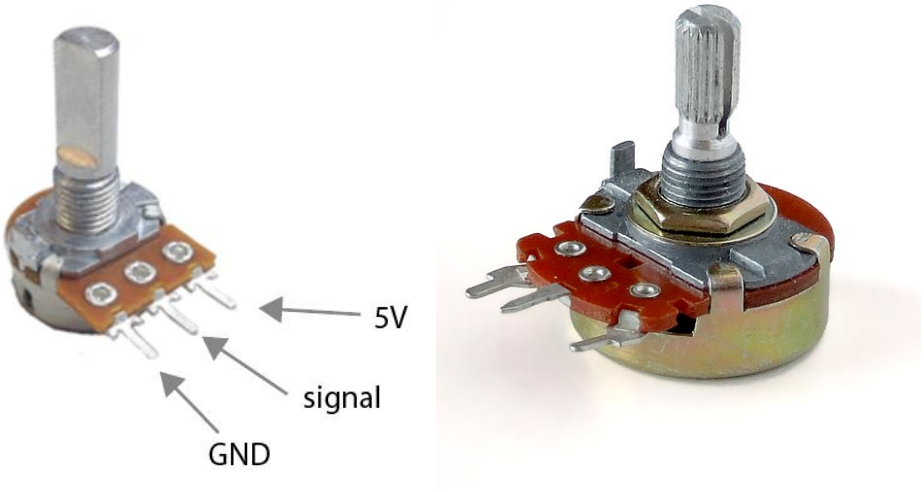
خازن

خازن جزو عناصر مداری هست که باعث ایجاد تأخیر در مدار میشه. اما خبر خوب اینجاست که این تأخیر میتونه به دردمون بخوره! وقتی که در ولتاژ منبع تغذیه به دلیل هرگونه نویزی یک نوسان ایجاد بشه خازن می تونه جلوی این نوسانات رو بگیره و باعث بهبود عملکرد سیستم بشه.

potentiometer (پتانسیومتر)

این قطعه یک مقاومت سه پایانه با یک ولوم چرخشی یا دکمه لغزنده قابل تغییر برای ولتاژ است.

از این قطعه در چالش برای تنظیم میزان روشنایی نور LED ها و فاصله زمانی بین روشن و خاموش شدن چراغ ها استفاده میشود.



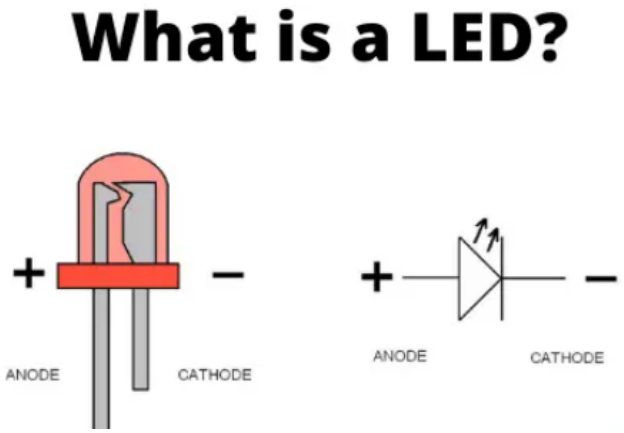
برای محاسبه ی ولتاژ خروجی پتانسیومتر می توان از قطعه کد زیر استفاده نمود.

```
uint16_t analogValue = analogRead(A4);
float voltage = map(analogValue, 0, 1023, 0, 5);
```

طبق این کد در ابتدا ولتاژ پین مربوطه با تابع analogRead خوانده می شود. خروجی این تابع عددی بین 0 تا 1023 است، هنگامی که ولتاژ خروجی صفر ولت باشد عدد آنالوگ صفر شده و هنگامی که ولتاژ خروجی 5 ولت باشد عدد آنالوگ 1023 می شود. سپس با استفاده از تابع map این مقدار آنالوگ به ولتاژ 0 تا 5 ولت نگاشت می شود.

LED

از این قطعه برای یک طرفه کردن جریان و ایجاد نور استفاده میشود. عملکرد پروژه شما با استفاده از نحوه روشن و خاموش شدن LED ها بررسی میشود.

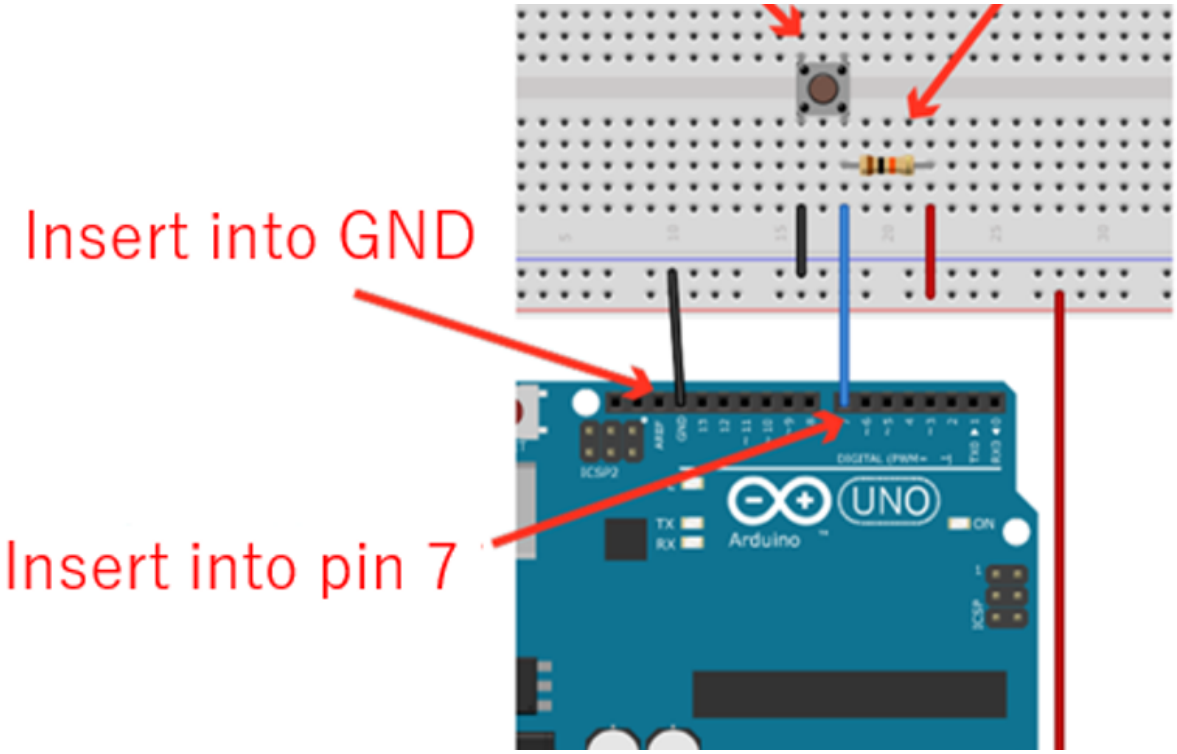


برای کار با LED اول از همه باید پینی که به اون وصله رو به عنوان خروجی تعریف کرد. بعد از اون میشه با مقدار HIGH یا LOW اون رو روشن و خاموش کرد.

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
}
```

سوئیچ قطع و وصل جریان

همانطور که از اسم این قطعه پیداست، از این قطعه برای قطع و وصل کردن جریان استفاده میشود.
با استفاده از سوییچ در پروژه میتوانید بین حالت های متفاوت اجرا جابجا شوید.



برای خوندن مقدار یک پین باید اول اون رو به صورت ورودی تعریف کرد و بعد با استفاده از تابع digitalRead میشه مقدار اون رو خوند. در مدار بالا ما پین آردوینو رو میخونیم. وقتی که کلید فشرده نشده، پین ما با یک مقاومت به 5 ولت وصل شده و ما 1 رو میخونیم و وقتی کلید فشرده میشه پین ما به O ولت وصل میشه و ما هم O رو میخونیم. این طوری می تونیم متوجه بشیم کلید فشرده شده یا نه.

Interrupt v.s Polling نمونه برداری در برابر وقفه

فرض کنید شما منتظر دوست تون هستید تا با همدیگه در چالش هاردوار شرکت کنید. چطوری می تونید متوجه بشید که دوست تون رسیده؟ یه راه اینه که در بازه زمانی کوتاه برید دم در خونه تون ببینید دوست تون اومده یا نه. راه دیگه اینه که کار خودتون رو انجام بدید و هر وقت دوست تون زنگ خونه رو زد برید پیشش. به راه اول Polling و به راه دوم Interrupt میگن.

برای متوجه شدن تغییر وضعیت کلید هم همچین موضوعی وجود داره. شما میتونید طبق روش نمونه برداری در فواصل کوتاه با استفاده از تابع digitalRead وضعیت کلید رو بخونید با استفاده از کد زیر:

```
void setup() {
  pinMode(7, INPUT);
}
void loop() {
  bool keyValue = digitalRead(7);
  delay(100);
}
```

یا اینکه می تونید از روش وقفه استفاده کنید و هنگامی که کلید تغییر وضعیت داد بره تابع isrFunction رو اجرا کنه:

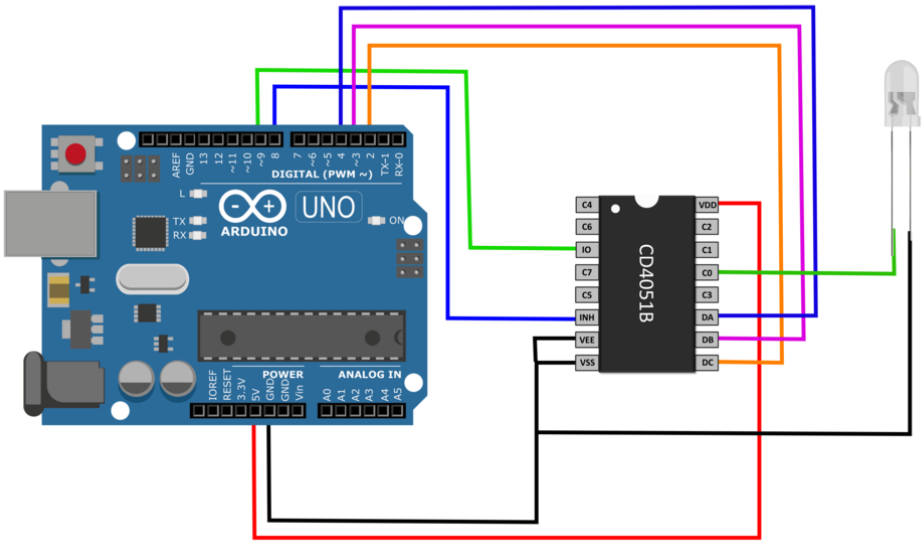
```
void setup() {
  pinMode(7, INPUT);
  attachInterrupt(digitalPinToInterrupt(7), isrFunction, FALLING);
}
void loop() {
}
void isrFunction(){
  //do something when key pressed
}
```

خوبی روش interrupt اینه که دیگه شما بخشی از برنامه اصلی تون رو صرف کاری که ممکنه زیاد رخ نده نمی کنید. حواس تون به تابع isrFunction هم باشه که از نوع void هست و هیچ آرگومان ورودی هم نداره اگر هم متغیری داخلش تغییر میکنه باید از نوع volatile باشه.

مالتی‌پلکسر (Multiplexer)

نوعی مدار منطقی ترکیبی است که به منظور تخصیص یکی از چندین خط ورودی به تنها یک خط خروجی مشترک طراحی شده است. اینکه کدام ورودی در خروجی قرار بگیرد، توسط یک منطق کنترلی مشخص می‌شود.

در اینجا شما با یک مالتی‌پلکسر هشت به یک کار میکنید که توسط سه پین ورودی کنترل میشود. ورودی به صورت یک عدد باینری داده میشود و ورودی مربوط به خروجی متصل میشود.



طبق کد زیر از بین 8 ورودی میخوایم ورودی دلخواهی که در جایگاه bitPos قرار داره رو انتخاب کنیم و در خروجی قرار بدیم. برای این کار سیگنال های select در مالتی پلکسر رو به درستی انتخاب می کنیم و خروجی مالتی پلکسر رو میخونیم:

```
pinMode(s0Pin, OUTPUT);
pinMode(s1Pin, OUTPUT);
pinMode(s2Pin, OUTPUT);
pinMode(outPin, INPUT);
```

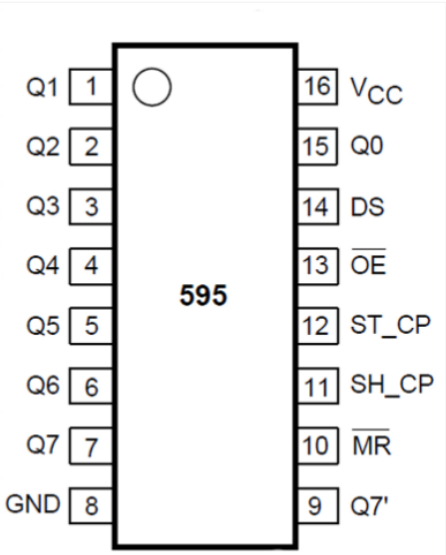
```
bool Multiplexer::getBit(uint8_t bitPos) {
  digitalWrite(s0Pin, bitPos & 1);
  digitalWrite(s1Pin, bitPos & 1 << 1);
  digitalWrite(s2Pin, bitPos & 1 << 2);
  return digitalRead(outPin);
}
```

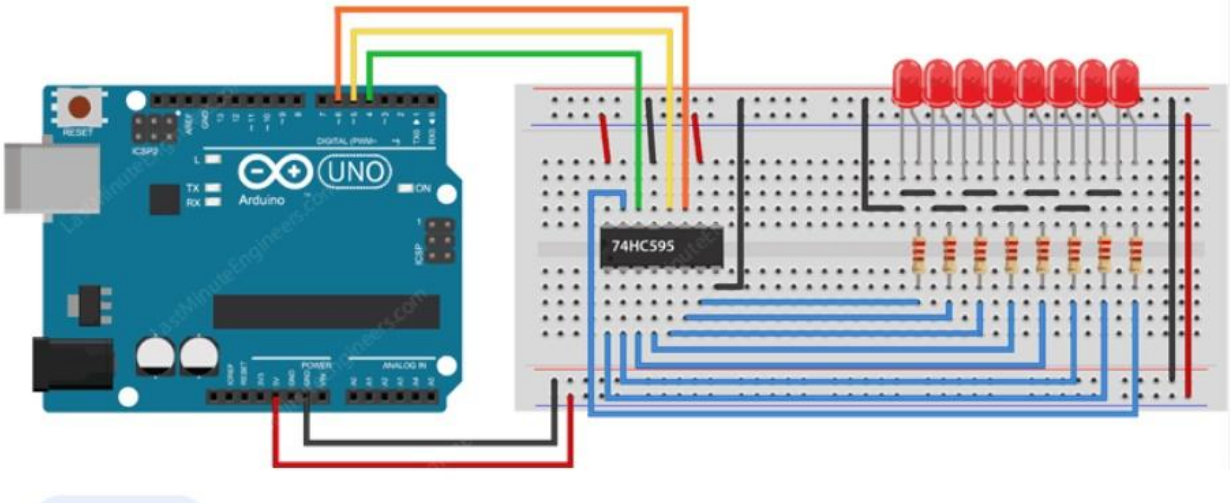
شیفت رجیستر

زنجیره‌ای از فلیپ‌فلاپ‌ها است که یک پالس ساعت مشترک دارند و خروجی هر فلیپ‌فلاپ، به ورودی فلیپ‌فلاپ بعدی در زنجیره متصل شده‌است در نتیجه مداری حاصل می‌شود که می‌تواند در هر پالس ساعت، آرایه بیتی که در آن ذخیره شده را یک واحد به سمت چپ یا راست شیفت (انتقال) دهد. در عمل شیفت، بیتی که در ورودی قرار دارد به داخل آرایه آورده می‌شود (شیفت به داخل) و آخرین بیت از آرایه خارج می‌شود و از بین می‌رود (شیفت به بیرون).

در شکل زیر می توانید شیت پین های شیفت رجیستر مورد استفاده در چالش را مشاهده کنید:

PIN	SYMBOL	DESCRIPTION
1	Q1	parallel data output
2	Q2	parallel data output
3	Q3	parallel data output
4	Q4	parallel data output
5	Q5	parallel data output
6	Q6	parallel data output
7	Q7	parallel data output
8	GND	ground (0 V)
9	Q7'	serial data output
10	$\overline{MR}$	master reset (active LOW)
11	SH_CP	shift register clock input
12	ST_CP	storage register clock input
13	$\overline{OE}$	output enable (active LOW)
14	DS	serial data input
15	Q0	parallel data output
16	V_{CC}	positive supply voltage





در شکل بالا می توانید یک مدار شیفت رجیستر را مشاهده کنید.

این آیسی دارای یک شیفت رجیستر و یک رجیستر در کنار اون هست. خوبی این کار اینه که میشه عملیات شیفت رو انجام داد بدون اینکه خروجی ما دچار تغییر بشه. پس از اینکه عملیات شیفت انجام شد میشه با لودکردن رجیستر جانبی خروجی آیسی رو به روزرسانی کرد. اینطوری فقط همون بیتی که تغییر کرده روی خورجی آیسی تغییر میکنه و عملیات شیفت تأثیری روی بیت های دیگه نداره.

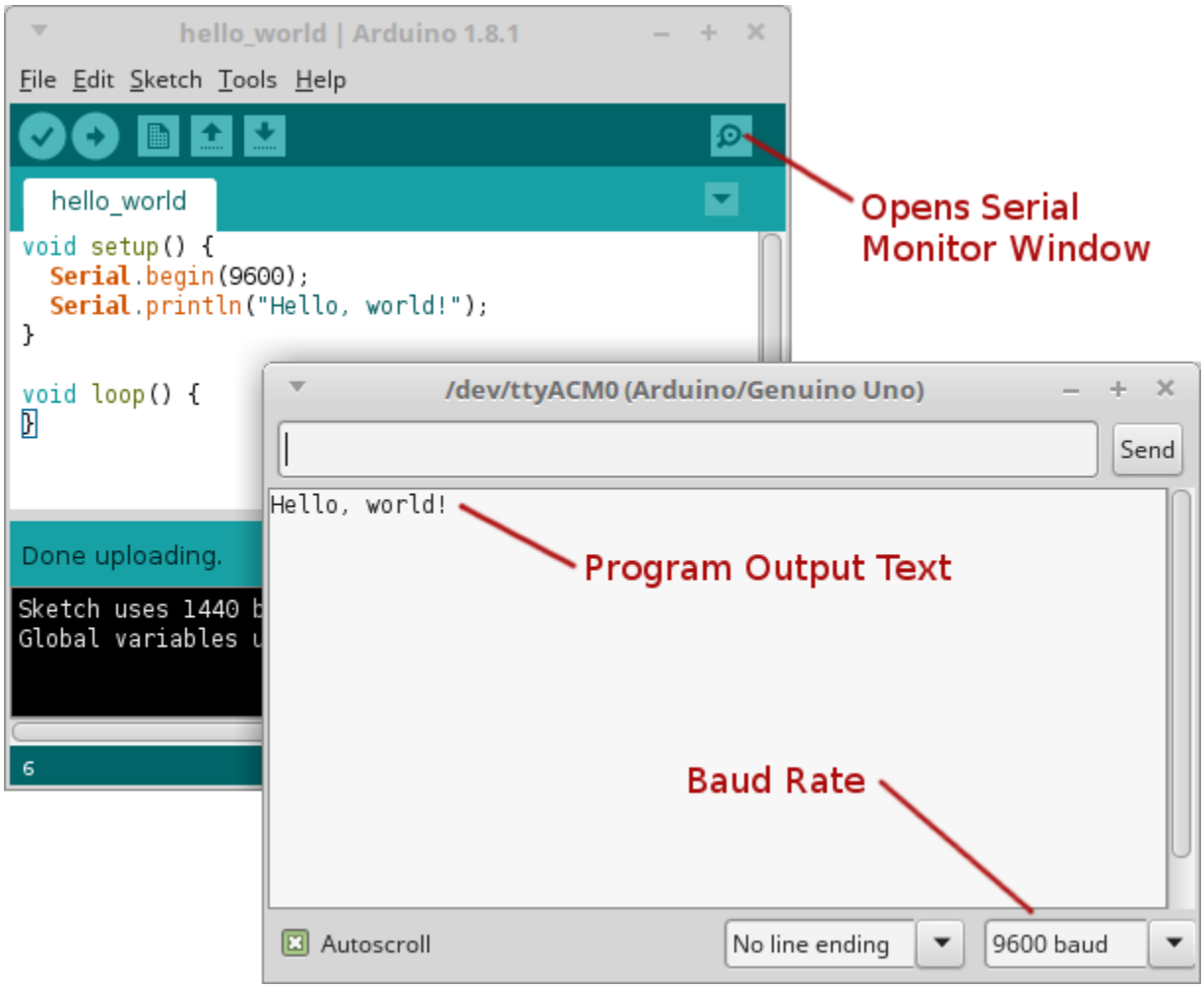
برای قرار دادن یک بایت data روی خروجی آیسی به نحوی که بیت پرارزش MSB اول وارد بشه، میشه از قطعه کد زیر استفاده کرد:

```
digitalWrite(latchPin, LOW);
shiftOut(dataPin, clockPin, MSBFIRST, data);
digitalWrite(latchPin, HIGH);
```

در این کد ابتدا پین کلاک رجیستر جانبی low شده و عملیات شیفت روی شیفت رجیستر انجام میشه. در انتها پین کلاک رجیستر جانبی high شده و در این لبه ی بالارونده حاصل عملیات شیفت روی خروجی آیسی قرار می گیرد.

ارتباط سریال

با استفاده از ارتباط سریال UART روی بورد آردوینو می تونیم با هر دستگاه دیجیتال دیگه ای از جمله کامپیوتر و لپتاپ ارتباط برقرار کنیم. برای این کار در محیط برنامه آردوینو بایستی گزینه Serial Monitor رو انتخاب کنید.



اول از همه بایستی baudrate یا سرعت انتقال داده بین دوتا دستگاه رو روی مقدار یکسانی تنظیم کنیم.
عددهای 9600 و 115200 از اعداد رایج هستن.
برای فرستادن یک پیام روی کامپیوتر می تونیم از کد زیر استفاده کنیم:

```
void setup() {
  Serial.begin(115200);
}
void loop() {
  Serial.println("Hello World!");
  delay(1000);
}
```

از اونجایی که ارتباط سریال یک ارتباط دوطرفه هست، میتونیم از کامپیوتر هم پیغامی رو روی آردوییِنو دریافت کنیم. کد زیر یک روش برای کنترل LED از طریق کامپیوتر رو نشون میده.
این کد اولین عدد صحیح وارد شده در سریال مانیتور کامپیوتر رو میخونه و اون رو به عنوان مقدار خروجی LED قرار میده:

```
void setup() {
  Serial.begin(115200);
  pinMode(7, OUTPUT);
}
void loop() {
  if (Serial.available()) {
    int ledValue = Serial.parseInt();
    Serial.readStringUntil('\n');
    digitalWrite(7, ledValue);
  }
}
```

یک برنامه ساده (!) با آردوئینو

حالا که با سخت افزار و نرم افزار آردوئینو آشنا شدیم، یه مثال رو میخوایم پیاده سازی کنیم تا ارتباط بین سخت افزار نرم افزار رو متوجه بشیم.

میخوایم دو تا برنامه بزنیم:

1. یه لامپ به صورت چشمک زن، روشن و خاموش بشه، که این دستورات روشن و خاموش شدن رو بورد آردوئینو میده.
2. مقابله با delay !

این دو تا کار برای انجام قسمت اصلی رویداد (مسابقه) - که چند ساعت دیگه‌ست - مفیده.

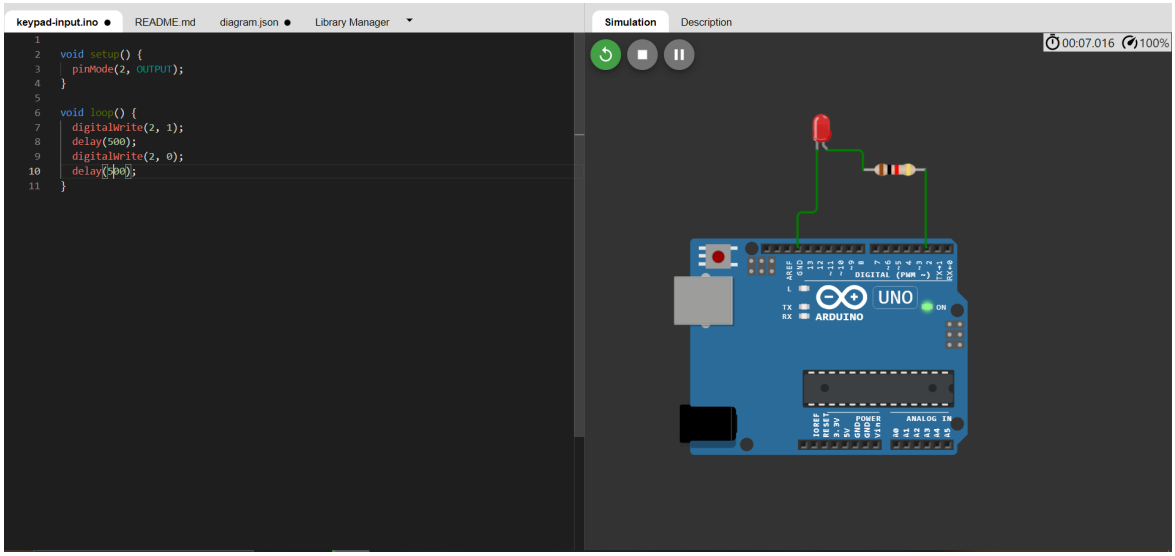
چالش منفی یک - لامپ چشمک زن

میخوایم یه برنامه ساده توی Arduino IDE بنویسیم که بیاد و لامپ ما رو خاموش و روشن کنه به صورتی که یه مدت روشن باشه و یه مدت خاموش، برای این کار، نرم افزار Arduino IDE رو که در بخش آموزشی نصب کردید باز کنید و این قطعه کد رو در اون بنویسید:

```
1
2     const int emergency_light = 10;
3     void setup() {
4       Serial.begin(115200);
5       pinMode(emergency_light, OUTPUT);
6     }
7
8     void loop() {
9       digitalWrite(emergency_light, HIGH);
10      delay(2000);
11      digitalWrite(emergency_light, LOW);
12      delay(2000);
13    }
```

حالا آردوئینو و لامپ رو مطابق با شکل زیر متصل کنید. برای شبیه سازی می تونید از سایت زیر استفاده کنید:

https://wokwi.com/



حالا قطعه کد رو اجرا کنید.

با تغییر فرکانس مربوط به روشن/خاموش شدن، سرعت خاموش/روشن شدن لامپ رو تغییر بدین.

البته توجه کنید که در مدار بالا، خروجی با پین 2 رسم شده اما در کدی که در ابتدا دیدیم پین شماره 10 مشخص شده بود. خلاصه یعنی هر پینی رو که در واقعیت متصل کردید در کد هم همون رو معلوم کنید.

نکته آموزشی : PWM یا گول زدن چشم! در برنامه بالا تأخیر بین روشن/خاموش شدن رو خیلی کم درنظر بگیرید طوری که نتونید با چشم روشن/خاموش شدن رو تشخیص بدید. یکی از تأخیرها رو 5 میلی ثانیه ثابت بذارید و اون یکی رو از 1 تا 20 میلی ثانیه تغییر بدید. چه مشاهده ای می کنید؟!

چالش صفر - دست گرمی

همان طور که در کد بالا مشاهده می کنید، از تابع delay به منظور ایجاد تأخیر بین عملیات روشن و خاموش کردن LED استفاده شده. استفاده از این تابع چندان خوشایند نیست زیرا تمامی فعالیت های پردازنده رو برای مدتی متوقف می کنه و این طوری پردازنده نمیتونه به کار های دیگه ش برسه. برای مثال اگر این وسط دکمه ای زده بشه پردازنده نمیتونه متوجه بشه که دکمه زده شده. در این بخش شما بایستی یک روش ارائه کنید تا بدون تابع delay بشه لامپ چشمک زن رو روشن/خاموش کرد.

ایده ی کلی به این صورته که بایستی از یک counter استفاده کنید. همچنین در آردوینو از تابع millis برای این کار میشه استفاده کرد. در مورد این تابع تحقیق کنید و با استفاده از هر دوی این روش ها لامپ چشمک زن رو پیاده سازی کنید.