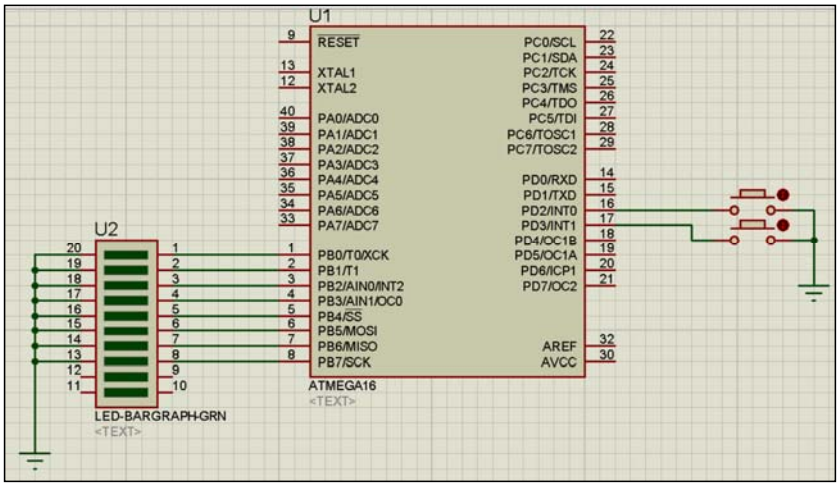
مثال دوم از پایه های ورودی/خروجی

- دو کلید به پایه های PD2 و PD3 وصل کنید با فشردن این کلیدها دو LED خاموش و روشن شوند. در تغییر بعدی برنامه یک کلید ۴ LED را خاموش و روشن کند. در تغییر بعدی عملکرد کلید ها معکوس شوند.

```
#include <io.h>
void main(void)
{
DDRB =0xFF;
while (1)
      {
            if (PIND.2==0)
                  PORTB.0=1;
            else
                  PORTB.0=0;
            PORTB.1=PIND.3;
      }
}
```



ادامه تمرین اسلاید قبل

- روشن و خاموش کردن ۴ LED با یک کلید:

```
#include <io.h>
void main(void)
{
DDRB =0xFF;
while (1)
      {
            if (PIND.2==0)
                  PORTB=0x0f;
            else
                  PORTB=0x00;
      }
}
```

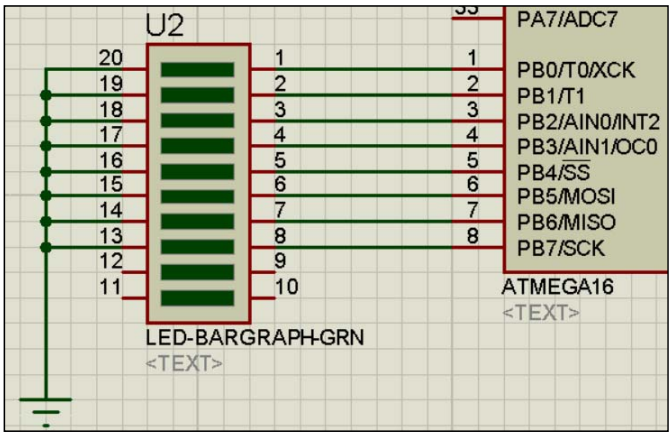
- معکوس کردن کارکرد کلیدها:

```
#include <io.h>
void main(void)
{
DDRB =0xFF;
while (1)
      {
            if (PIND.2==1)
                  PORTB.0=1;
            else
                  PORTB.0=0;
            PORTB.1=!PIND.3;
      }
}
```

مثال از ایجاد تاخیر بین دستورها

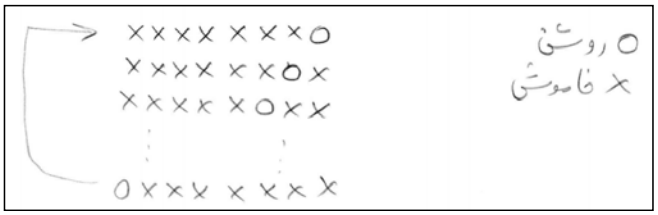
- یک LED را با یک ثانیه تاخیر روشن و خاموش کنید.

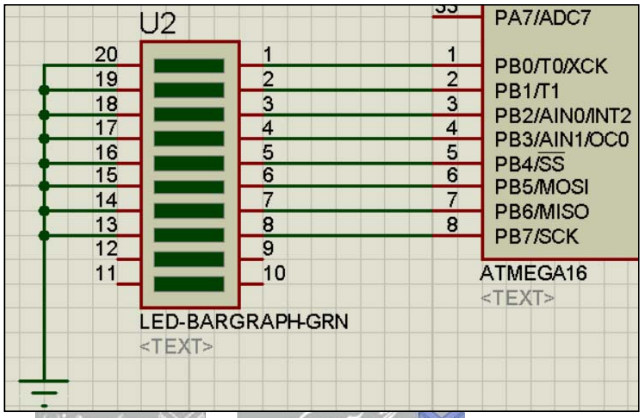
```
#include <io.h>
void main(void)
{
DDRB =0xFF;
while (1)
    {
        PORTB.0=1;
        delay_ms(1000);
        PORTB.0=0;
        delay_ms(1000);
    }
}
```



مثال دوم از ایجاد تاخیر بین دستورها

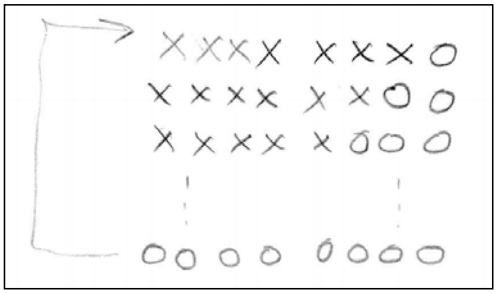
- ۸ LED را به ترتیب زیر و با تاخیر ۰.۵ ثانیه روشن کنید:



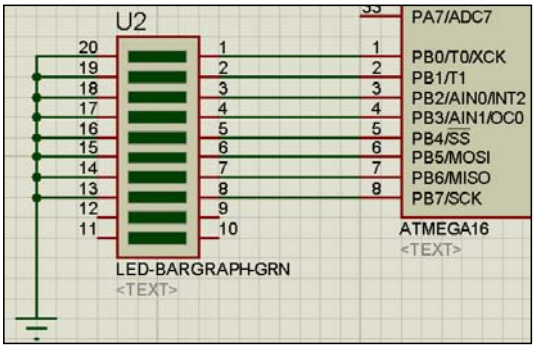



```
DDRB=0xFF;
while (1)
        {
        a=1;
        for (i=1;i<=8;i++)
        {
        delay_ms(500);
        PORTB = a;
        a=a*2;
        }
        }
}
```

مثال سوم از ایجاد تاخیر بین دستورها

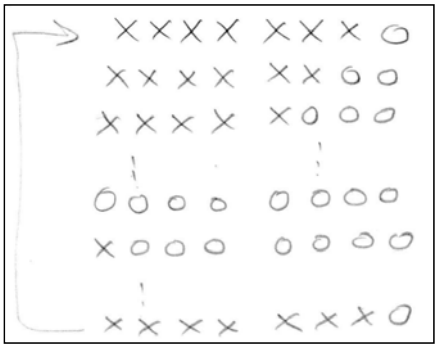


- ترکیب مقابل را روی LED ها ایجاد کنید:

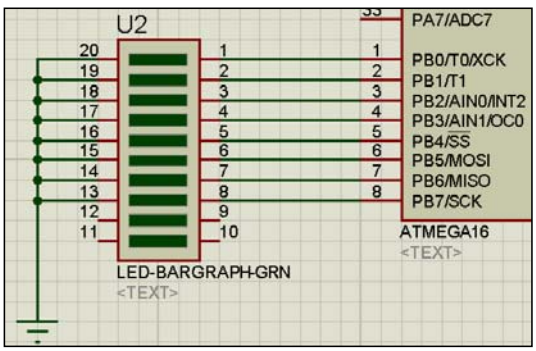



```
while (1)
   {
        a=1;
        for (i=1;i<=8;i++)
        {
        PORTB = a;
        delay_ms(500);
        a=a*2+1;
        }
   }
```


مثال چهارم از ایجاد تاخیر بین دستورها



- ترکیب مقابل را روی LED ها ایجاد کنید:



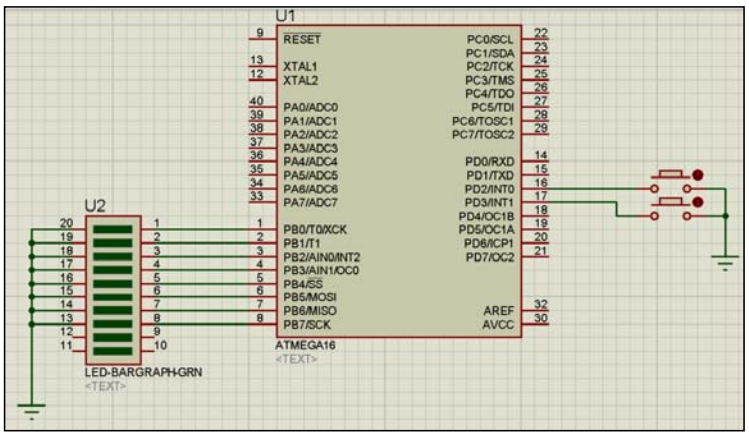

```
while (1)
   {
        a=1;
        for (i=1;i<=16;i++)
        {
        PORTB = a;
        delay_ms(500);
        if (i<9)
            a=a*2+1;
        else
            a=a/2;
        }

   }
```

مثال پنجم از ایجاد تاخیر بین دستورها

```
while (1)
      {
      a=1;
      for (i=1;i<=16;i++)
      {
      PORTB = a;
      if (PIND.2==1)
         delay_ms(500);
      else
         delay_ms(50);
      if (i<9)
         a=a*2+1;
      else
         a=a/2;
      }
      }
```

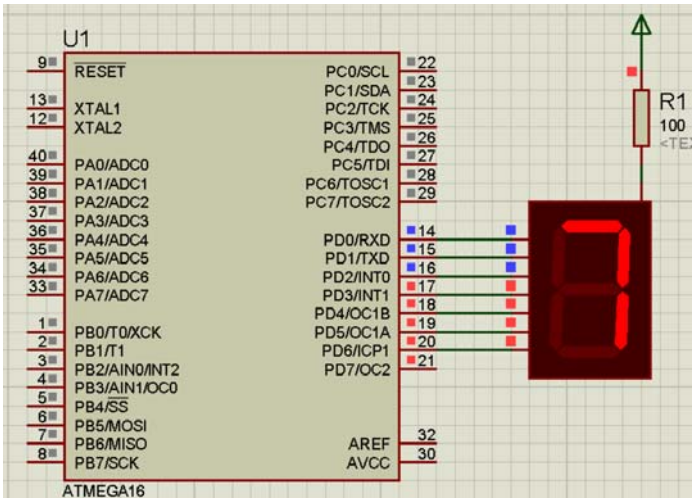
- با فشردن یک کلید سرعت حرکت LED ها ۱۰ برابر بیشتر شود.

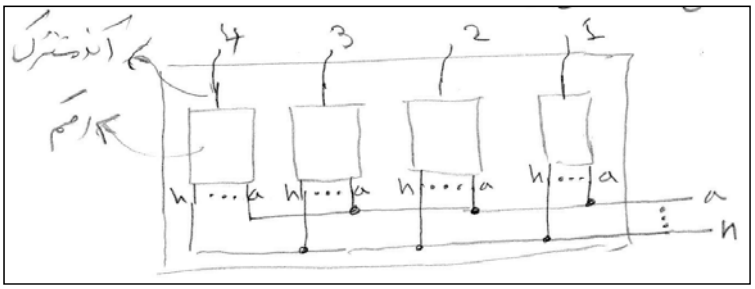


روشن کردن ابزارهای نمایش با میکروکنترلر

```
void main(void)
{
DDRD=0xFF;
PORTD=0b11111000;
while (1)
      {
      }
}
```

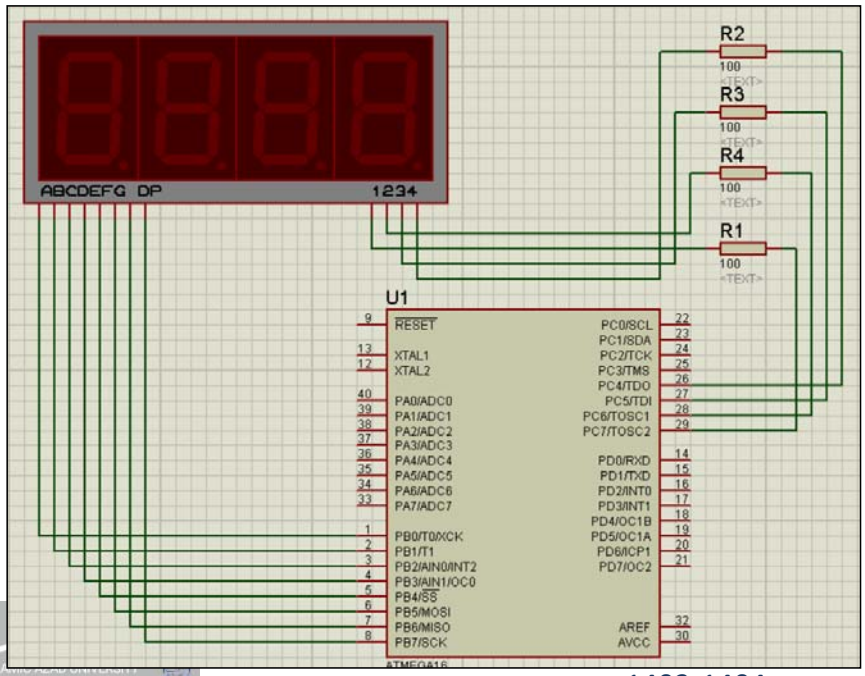
- نمایشگر 7segment برای نمایش اعداد به کار می رود:
- یک 7segment یک رقمی را با یک عدد دلخواه روشن می کنیم:





اتصال 7segment چهار رقمی مالتی پلکس

- در این قطعه برای کاهش تعداد پایه اتصال های داخلی به شکل زیر است:



اتصال 7segment چهار رقمی مالتی پلکس

- برنامه ای بنویسید که عددی چهار رقمی را که در متغیر n ذخیره شده است را روی 7segment چهاررقمی نمایش دهد.

```
n=2081;
cn=0;
while (1)
      {
      m=n;
      a=m%10;
      m=m/10;
      b=m%10;
      m=m/10;
      c=m%10;
      d=m/10;
```

```
PORTC=0b10000000;
PORTB=digit[d];
delay_ms(5);
PORTC=0b01000000;
PORTB=digit[c];
delay_ms(5);
PORTC=0b00100000;
PORTB=digit[b];
delay_ms(5);
PORTC=0b00010000;
PORTB=digit[a];
delay_ms(5);
cn=cn+1;
if (cn==50)
    {
    n=n+1;
    cn=0;
    }
}
```

اتصال 7segment چهار رقمی مالتی پلکس (ادامه)

- در برنامه اسلاید قبل برای ذخیره ترکیب اعداد 0 تا 9 روی 7segment از آرایه ها استفاده شده است.

- تعریف آرایه در ابتدای برنامه

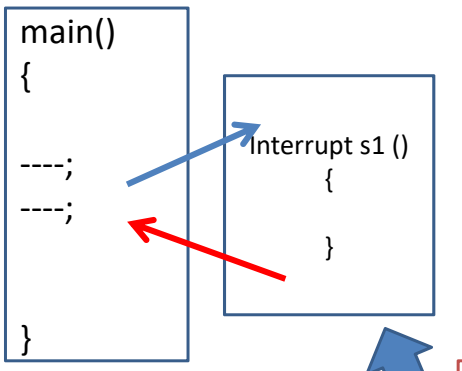
```
char digit[]={
0b11000000,
0b11111001,
0b10100100,
0b10110000,
0b10011001,
0b10010010,
0b10000010,
0b11111000,
0b10000000,
0b10010000};
```


برنامه ریزی با وقفه (Interrupt)

- دستگاه های مختلفی در میکروکنترلر وجود دارد که باید به درخواست های آنها در برنامه پاسخ داده بشود، دو راه برای پاسخ گویی وجود دارد: وقفه ، سرکشی. آنچه ما تاکنون انجام می دادیم polling بود.
- سرکشی یعنی در حلقه ای وضعیت دستگاه را بررسی کنیم.
- در وقفه، هر دستگاهی که نیاز به بررسی داشته باشد یک سیگنال وقفه به CPU می فرستد. و CPU تصمیم می گیرد (براساس برنامه نوشته شده) که به آن پاسخ دهد یا نه.
- اگر تصمیم بر این شود که پاسخ داده شود برنامه در حال اجرا موقتاً متوقف شده و یک زیر برنامه به نام روتین وقفه اجرا می شود.
- برای هر دستگاه یک یا چند عامل ایجاد تعریف می شود.


برنامه ریزی با وقفه (Interrupt)

- در این روش به دستگاه های ورودی/خروجی اجازه داده می شود در صورت اتفاق مشخصی درخواست وقفه برای CPU بفرستند. اگر درخواست پذیرفته شود، اجرای برنامه اصلی در هر خطی باشد متوقف شده، سپس زیر برنامه ای که از قبل برای آن دستگاه و آن اتفاق خاص نوشته ایم اجرا می شود. پس از آن اجرای برنامه اصلی از همان جایی که متوقف شده بود از سر گرفته می شود.



برنامه رسیدگی به وقفه:
برنامه ای است که زمانی اجرا می شود که عامل ایجاد کننده وقفه در دستگاه I/O اتفاق افتاده باشد

وقفه های خارجی سخت افزاری

- سه پایه روی پورت های ATmega32 قرار داده شده که می توان با تغییر مقدار آنها از خارج از میکرو به برنامه اصلی میکرو وقفه داد.
- با اعمال تغییر روی این پایه ها (که می تواند آمدن لبه یا تغییر سطح باشد.) میکرو فعالیت خود (اجرای برنامه main) را موقتاً قطع کرده به زیر برنامه وقفه ای که از قبل نوشته شده می رود.
- پس از اجرای زیر برنامه، به محل قبلی اجرای برنامه بر می گردد.
- این وقفه ها دارای بیت های ماسک هستند
- با بیت های ماسک می توان از فراخوانی آنها جلوگیری کرد.

وقفه های خارجی سخت افزاری (۲)

- پایه های وقفه خارجی عبارتند از:
  - PD2: INT0 (شماره وقفه: ۲، عنوان داخل [] در تعریف: EXT_INT0)
  - PD3: INT1 (شماره وقفه: ۳، عنوان داخل [] در تعریف: EXT_INT1)
  - PB2: INT2 (شماره وقفه: ۴، عنوان داخل [] در تعریف: EXT_INT2)
- رجیستر ماسک وقفه های خارجی رجیستری است با نام GICR

D7							D0
INT1	INT0	INT2	-	-	-	IVSEL	IVCE

- با نوشتن ۱ در سه بیت آخر این رجیستر می توان وقفه ها را فعال کرد.
- البته برای فعال سازی، وقفه عمومی نیز باید فعال شده باشد.
- وقفه ها را می توانیم حساس به سطح یا لبه تنظیم کنیم.

حساسیت به سطح یا لبه وقفه های خارجی

- حساسیت به سطح: هر وقت سطح پایه صفر شود وقفه صورت می گیرد. اگر این سطح همچنان پایین بماند، پس از بازگشت از روتین وقفه دوباره وقفه صورت می گیرد. (کمتر این حالت را به کار می گیریم.)
- حساسیت به لبه: تغییر سطح در پایه ورودی عامل رفتن به زیربرنامه وقفه است. (این حالت کاربردی تر است.)
- این تنظیم برای INT0 و INT1 از طریق رجیستر MCUCR انجام می شود.

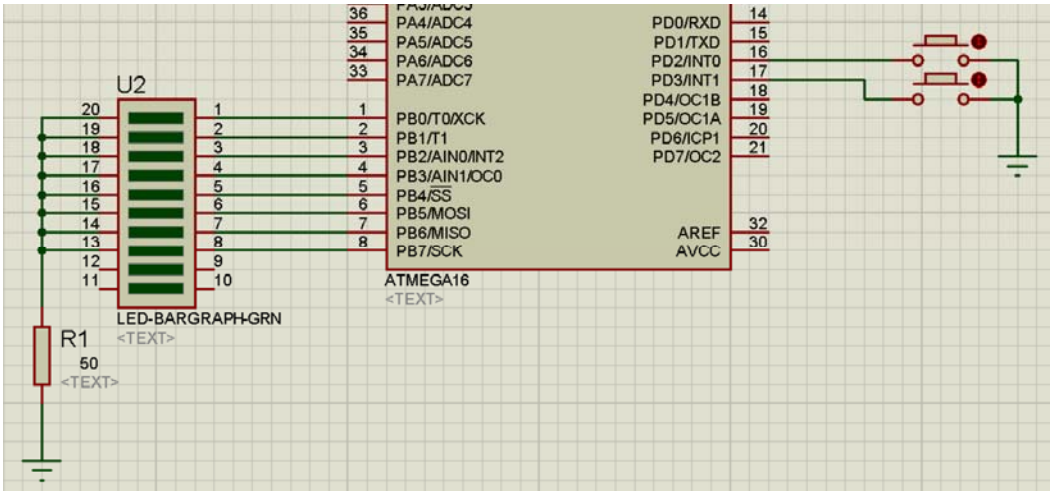
D7							D0
SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC11	ISC10	
0	0	
0	1	
1	0	
1	1	

ISC01	ISC00	
0	0	
0	1	
1	0	
1	1	

مثال از به کارگیری وقفه های خارجی

- با هر بار فشردن کلید متصل به پایه PD2 مقدار روی PORTB یکی افزایش یابد.
- و با هر بار فشردن کلید متصل به پایه PD3 مقدار روی PORTB یکی کاهش یابد.
- نمی خواهیم عدد روی PORTB منفی شود.



پاسخ مثال

- دو کلید به پایه های دارای قابلیت وقفه وصل شده اند. این دو وقفه را فعال می کنیم و حساسیت آنها را نسبت به لبه انتخاب می کنیم به این ترتیب هر بار که کلید فشار داده شود یکبار زیر برنامه رسیدگی مربوط به آن وقفه اجرا می شود. در این زیر برنامه مقدار متغیری را کاهش یا افزایش می دهیم.
- در برنامه اصلی این عدد را روی LED ها یا روی 7Segment نمایش می دهیم.

```
int a;
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
 a = a+1;
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
 a = a-1;
}
```

- زیربرنامه های رسیدگی به وقفه
- برنامه اصلی در اسلاید بعدی...

پاسخ مثال (ادامه)

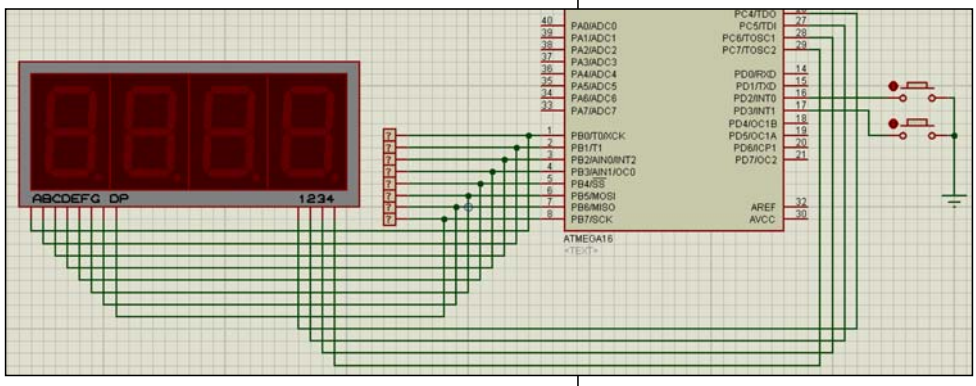
```
void main(void)
{
GICR=0b11000000;
MCUCR=0b00001010;
#asm("sei")
DDRB=0xFF;
while (1)
      {
      PORTB = a;
      }
}
```

- در برنامه اصلی ابتدا وقفه را برای دو وقفه خارجی فعال کرده ایم. (بیت های ماسک در بیت های ششم و هفتم رجیستر GICR)
- سپس حساسیت به لبه به هر دو وقفه را روی حساس به لبه پایین رونده قرار داده ایم (با نوشتن 10 در بیت های صفرم تا سوم رجیستر MCUCR)
- خط سوم برای فعال سازی عمومی کل وقفه ها است.
- در حلقه while هم متغیر a را (که در زیر برنامه های وقفه مقدار می گیرد) را روی LED ها نمایش داده ایم.

پاسخ مثال قبل با نمایش روی 7Segment

```
void main(void)
{
int x,y;
char digit[]={
0b11000000,0b11111001,0b10100100,0b10110000,0b10011001,
0b10010010,0b10000010,0b11111000,0b10000000,0b10010000};
GICR=0b11000000;
MCUCR=0b00001010;
DDRB=0XFF;
DDRC=0XFF;
#asm("sei")
while (1)
      {
      x=a%10;
      y=a/10;
      PORTC=0b10000000;
      PORTB=digit[x];
      delay_ms(5);
      PORTC=0b01000000;
      PORTB=digit[y];
      delay_ms(5);
      }
}
```

- زیربرنامه های رسیدگی وقفه بدون تغییر هستند.



مقدمه ای بر Arduino

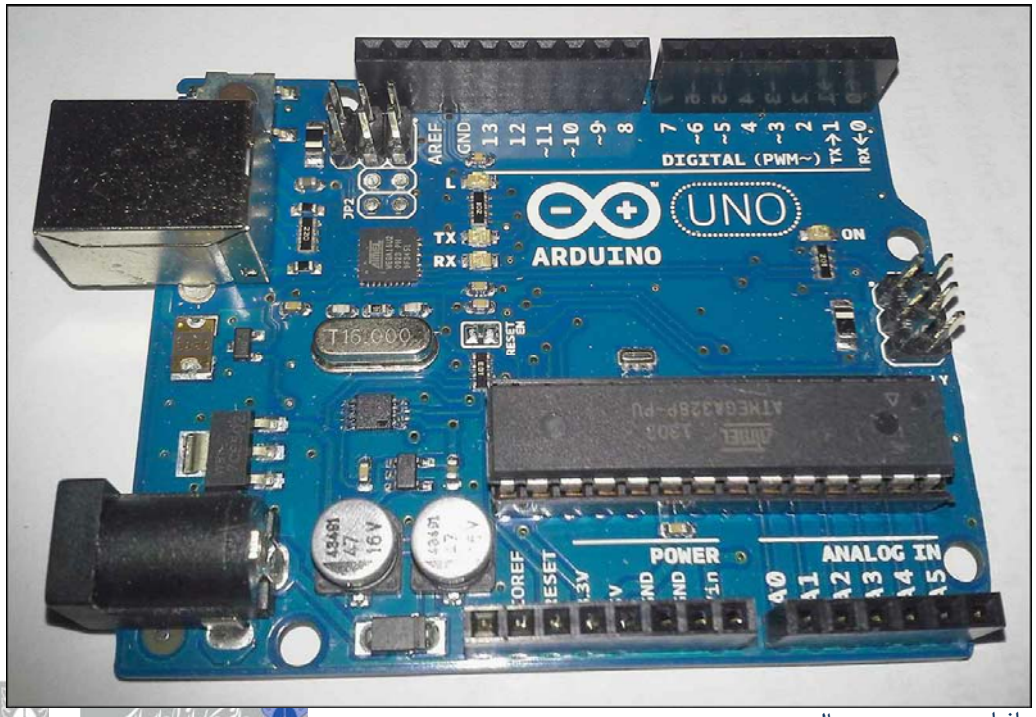
- آردوینو یک ساختار سخت افزاری مبتنی بر میکروکنترلرها است.
- به کمک آردوینو می توان فرمان های دیجیتال و آنالوگ را به دستگاه های دیگر ارسال و دریافت کرد. (درست همانند میکروکنترلر)
- ولی ابزار طراحی شده روی برد (board) آردوینو به کاربر این امکان را می دهد که کاربر، کاربرد دلخواه را سریعتر و ساده تر پیاده سازی کند.
- آردوینو برای برنامه نویسی میکروکنترلر روی برد، نرم افزار خود را ارائه کرده است.
- این نرم افزار شامل کامپایلر، و پروگرامر است.
- برنامه نویسی به زبان C انجام می شود. ولی کتابخانه ها و توابع اختصاصی برای آردوینو تعریف شده است.
- اتصال برد آردینو با کامپیوتر شخصی برنامه نویس از طریق پورت USB انجام می شود. و برای پرگرام کردن میکروکنترلر، مدار اضافی مورد نیاز نیست. (ابزار پروگرام روی برد قرار داده شده است.)

انواع بردهای Arduino

- شرکت آردوینو برای برآورده کردن نیازهای مختلف مشتریان گروه های مختلفی از برد های آردوینو را طراحی کرده و ساخته است.
- اکثر این برد ها بر مبنای مدل های مختلفی از میکروکنترلرهای AVR طراحی شده اند.
- یعنی روی برد یک میکروکنترلر AVR قرار دارد، اما برنامه نویسی برای آن از طریق نرم افزار اختصاصی آردوینو انجام می شود.
- در اینجا چند خانواده مختلف آردوینو به همراه چند تصویر از آنها را معرفی می کنیم:
  - Arduino Uno:
  - پایه ای ترین و محبوب ترین خانواده آردوینو که امکانات پایه میکروکنترلر را در اختیار کاربر قرار می دهد.
  - میکروکنترلر به کار گرفته شده: ATmega328
  - امکانات: ۱۴ پایه دیجیتال I/O، ۶ پایه آنالوگ خروجی و ...
  - یک مدل از این خانواده را مفصل توضیح خواهیم داد.

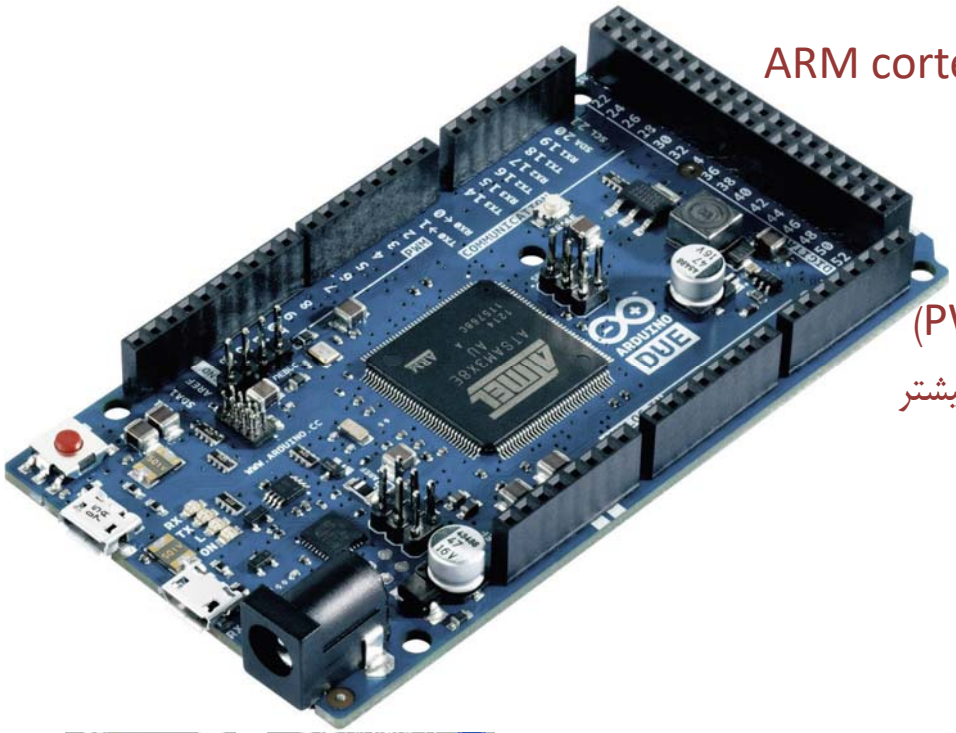
Arduino Uno

- در شکل زیر یک مدل از این گروه دیده می شود.

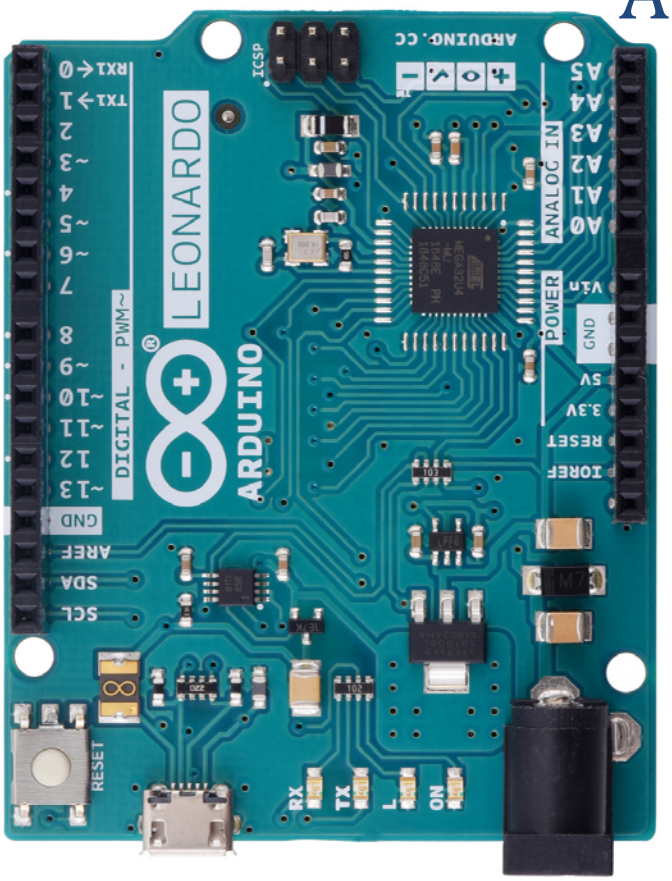


خانواده های دیگر Arduino

- Arduino Due:
  - برمبنای پردازنده ARM cortex-M3
  - قدرت پردازش بالاتر
  - ۵۴ پایه I/O
  - ۱۲ ورودی آنالوگ
  - ۱۲ خروجی آنالوگ (PWM)
  - حافظه Ram و Flash بیشتر
  - پورت های سریال بیشتر



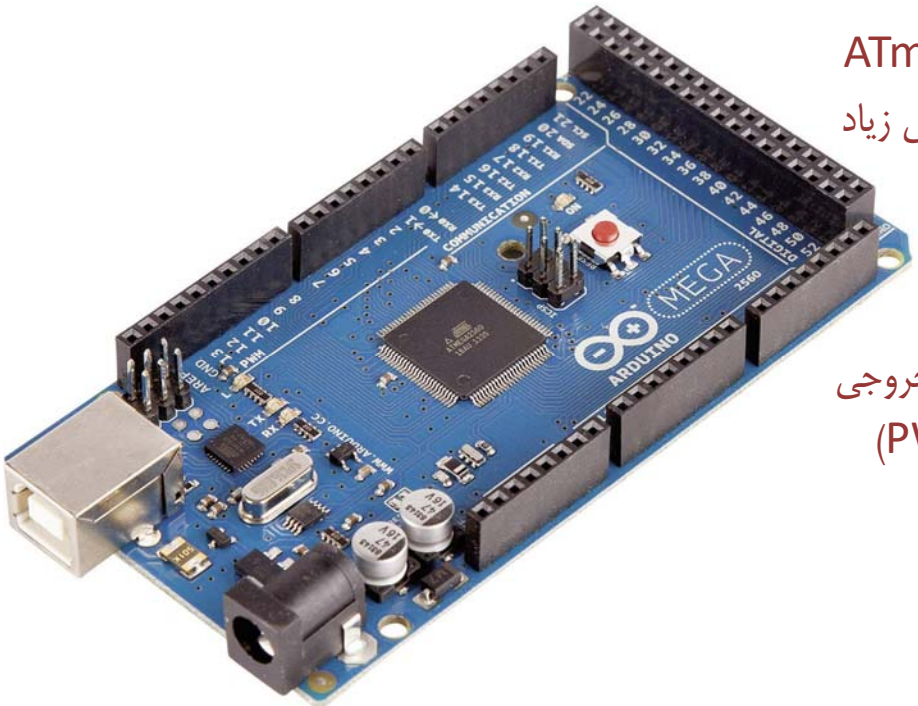
خانواده های دیگر Arduino



- Arduino Leonardo
  - میکروکنترلر ATmegau4
  - ۲۰پایه دیجیتال
  - ۱۲ پایه آنالوگ
  - ۷ پایه PWM
  - اندازه برد مشابه با Uno
  - امکان ارتباط ساده با تجهیزات USB
  - شبیه ساز کیبرد و ماوس USB

خانواده های دیگر Arduino

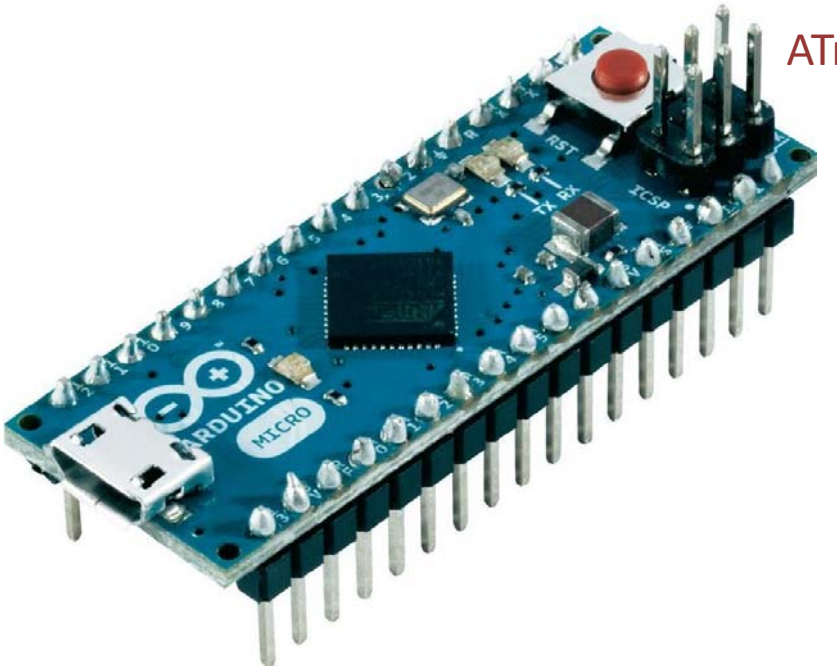
- Arduino Mega
  - میکروکنترلر ATmega2560
  - هدف: تعداد ورودی خروجی زیاد



  - ۵۴ پایه دیجیتال ورودی/خروجی
  - ۱۵ خروجی آنالوگ (PWM)
  - ۱۶ ورودی آنالوگ
  - ابزارهای جانبی بیشتر
  - برد بزرگتر

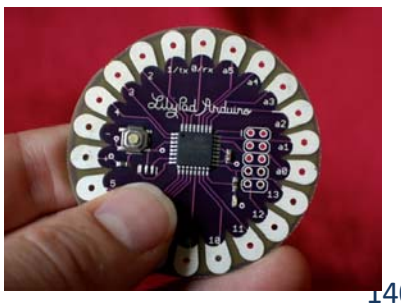
خانواده های دیگر Arduino

- Arduino Micro
  - میکروکنترلر: ATmega32u4
  - هدف: اندازه برد کوچکتر
  - کاملاً دارای کاربرد مشابه
  - با Leonardo
  - با تعداد پایه مشابه
  - ولی برد کوچکتر



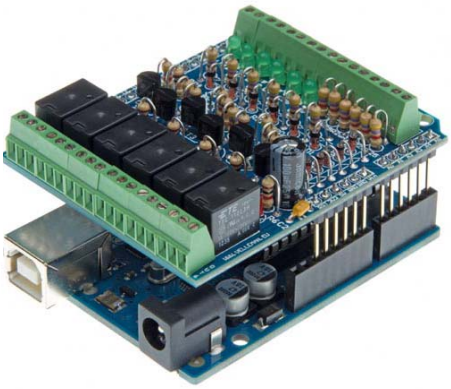
برخی دیگر از خانواده های Arduino (برای مطالعه)

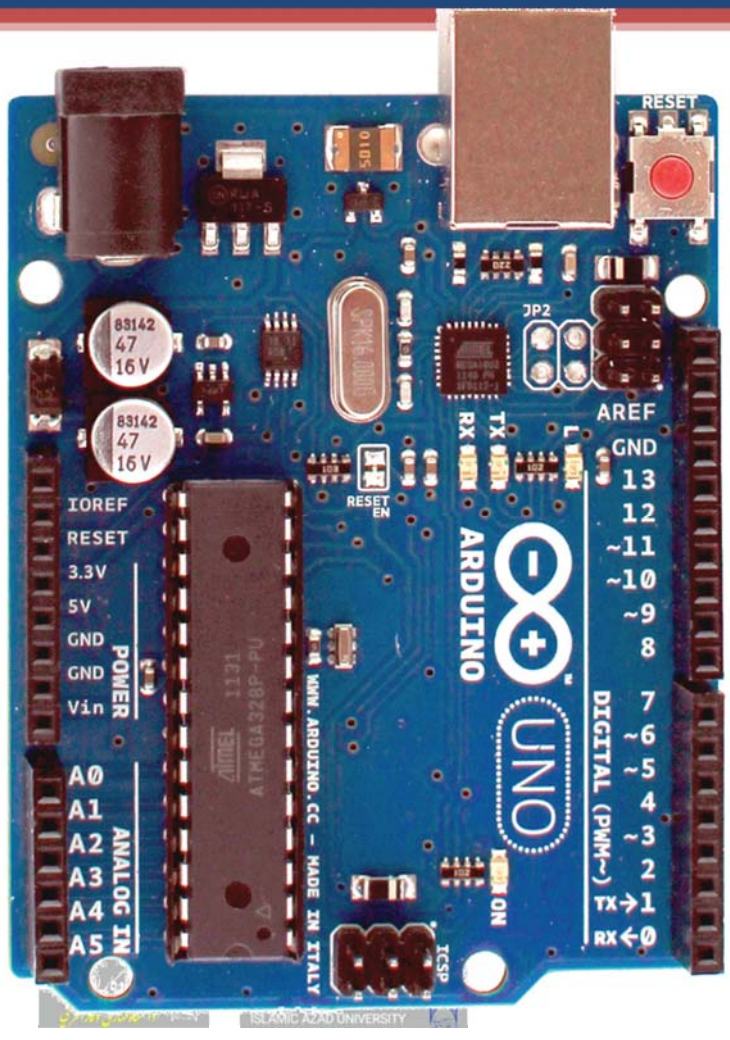
- Arduino Esplora
  - برای ارتباط با gameها controller
- Arduino Yun
  - دارای دو پردازنده (یکی دارای سیستم عامل Linux)
- Arduino Ethernet
  - برای برقراری ارتباط سیمی با شبکه و اینترنت
- LilyPad Arduino
  - اندازه بسیار کوچک مخصوص تعبیه در لوازم پوشیدنی: مثل ساعت و لباس



به کارگیری Sheild ها در آردوینو

- Sheild ها بردی اضافی در ابعاد برد اصلی Arduino هستند و روی برد اصلی سوار می شوند و امکانات جدیدی را به برد اضافه می کنند.
- برای کاربردهای مختلف انواع مختلفی از این sheildها معرفی شده اند.
- مثال: برای کنترل موتور (driver)
- مثال: ورودی خروجی ولتاژ بالا
- مثال: برای اتصال به شبکه

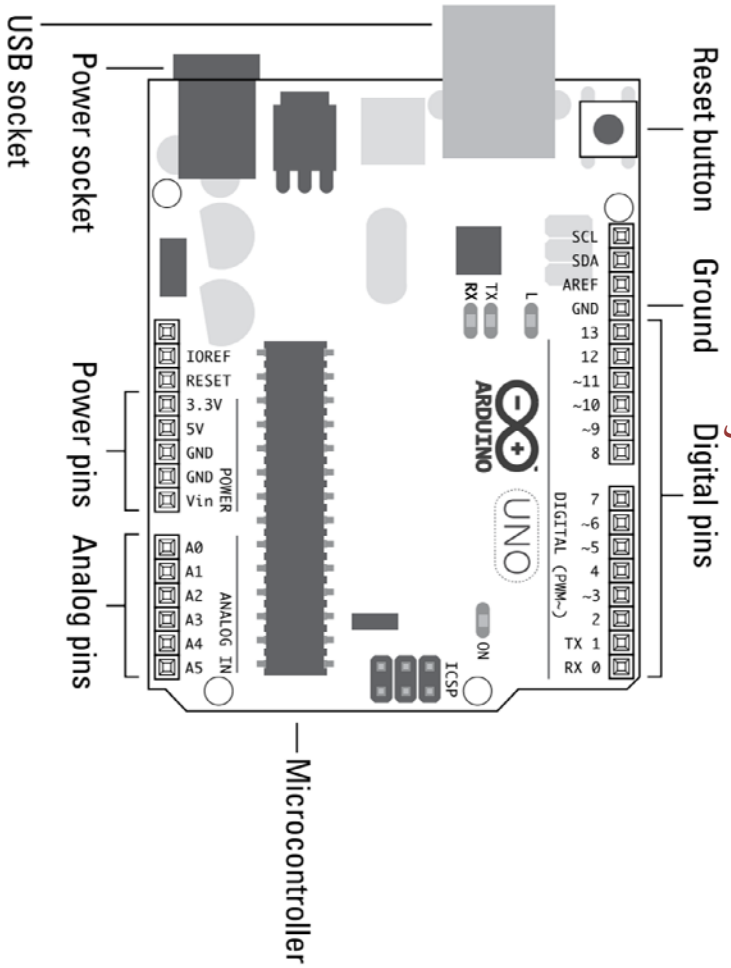




معرفی Arduino Uno

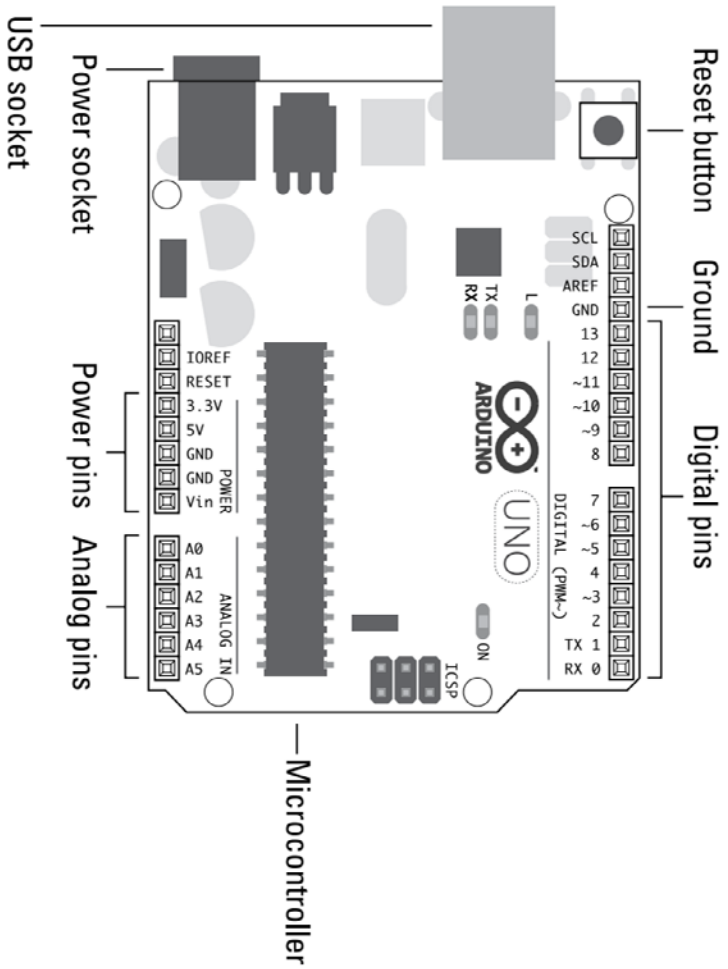
- ۱۴ ورودی و خروجی دیجیتال (۰و ۱)
  - از شماره 0 تا 13
- ۶ ورودی آنالوگ (A0 تا A5)
- ۶ خروجی آنالوگ (PWM)
  - نمایش با علامت ~ روی پایه های دیجیتال
- یک پورت سریال I2C
- یک پورت سریال SPI
- یک پورت سریال UART (متصل به پورت USB) و پایه های TX و RX
- تغذیه مجزا (بالا سمت چپ)
- تغذیه از پورت USB بالا سمت راست

معرفی Arduino Uno



- چهار LED ریز روی برد Uno تعبیه شده است.
  - ON: روشن بودن تغذیه بورد را نمایش می دهد.
  - TX و RX: هنگام ارسال و دریافت پورت UART روشن می شوند.
  - L: یک LED مستقل است که به پایه 13 دیجیتال متصل شده است. و قابل کنترل از درون برنامه است.
- با فشردن کلید Resetبرنامه از ابتدا اجرا می شود.
- دقت کنید: نام پایه های آردوینو با نام پایه های میکروکنترلر AVR متفاوت است.

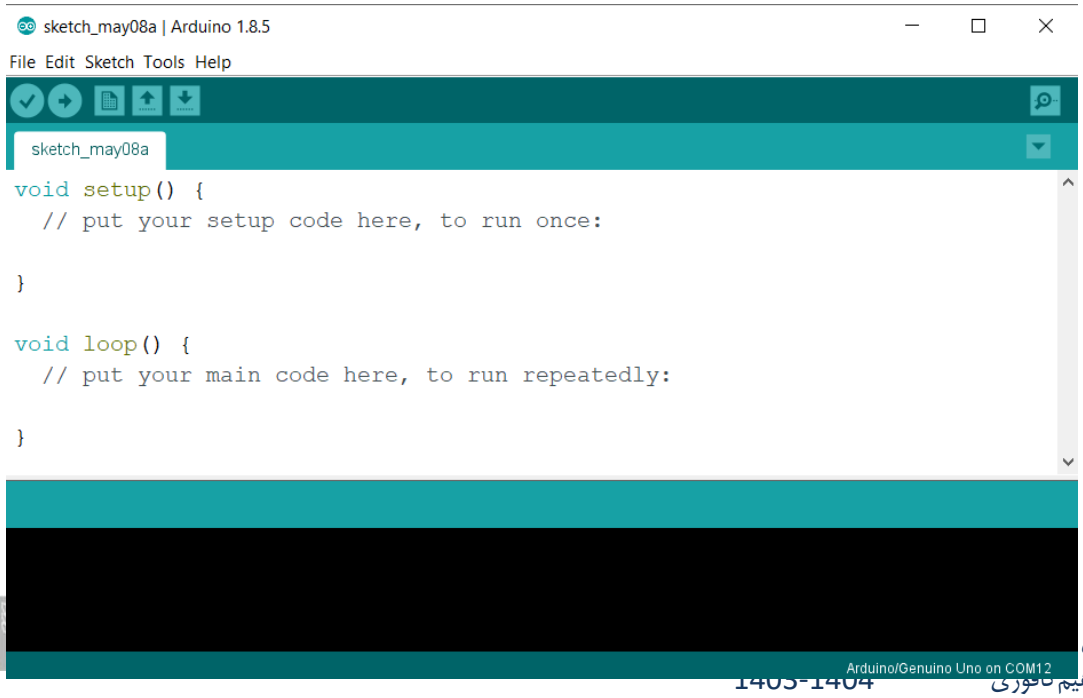
معرفی Arduino Uno (ادامه)



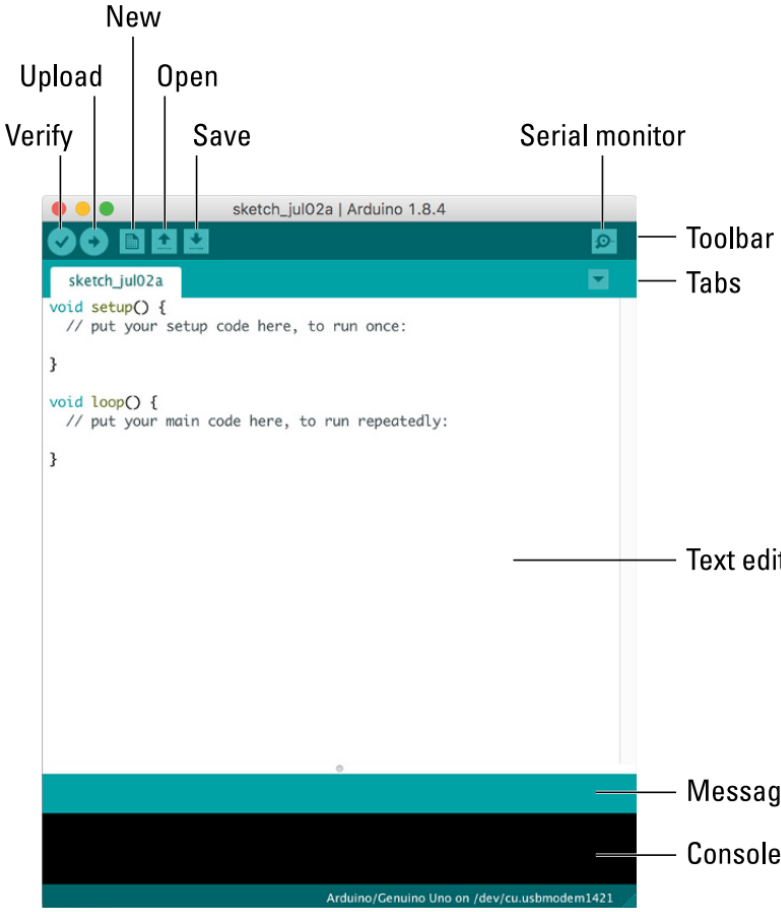
- پایه های Power:
  - ولتاژ مورد نیاز برای مدارهای دیگر و پایه زمین مشترک را فراهم می کنند.
  - ولتاژهای ۳.۳ و ۵ ولت در اختیار کاربر است تا در مدارهای دیگر استفاده شود.
  - پایه Vin پایه منبع تغذیه ورودی برد است. یعنی می توان به جای تغذیه از پورت USB یا ورودی تغذیه مستقل بالا سمت چپ از طریق این پایه تغذیه برد را تامین کرد.

محیط نرم افزار برنامه نویسی برای Arduino (Arduino IDE)

- نرم افزار ارائه شده امکان برنامه نویسی و پرگرام کردن برد را از طریق پورت USB به کاربر می دهد. نرم افزار مجانی است و قابل دانلود از سایت سازنده است.



محیط نرم افزار برنامه نویسی برای Arduino



- برنامه دارای دو تابع اصلی است:
- Setup()
  - که یکبار در ابتدای برنامه راه اندازی می شود.
  - برنامه که در آن نوشته می شود فقط یکبار اجرا می شود
- Loop()
  - برنامه داخل آن برای همیشه تکرار می شود.
  - مشابه while (1) است.

اولین برنامه: خاموش و روشن کردن LED روی برد

- برنامه:

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```


اولین برنامه: خاموش و روشن کردن LED روی برد

توضیح برنامه

- در برنامه ریزی برای arduino برای انجام هر کار ساده یا مشکل یک تابع از پیش تعریف شده است. به عنوان مثال تابع زیر یک پایه را خروجی یا ورودی می کند. مشابه آنچه پیش از این به عنوان با مقدار دادن به رجیستر DDRA انجام می شد.
- تابع pinMode
- **pinMode(LED_BUILTIN, OUTPUT);**
  - تابع بالا پایه LED_BUILTIN (که همان پایه ۱۳ است) را خروجی می کند.
  - کلمه LED_BUILTIN از پیش تعریف شده است و برای برد Uno برابر ۱۳ است.
  - کلمه OUTPUT از پیش تعریف شده و به معنای خروجی کردن پایه است.
  - بنابرین تابع بالا را برای Uno به شکل زیر هم می توان نوشت:
- **pinMode(13, OUTPUT);**
  - نکته: حروف کوچک و بزرگ در نوشتن نام تابع مهم است.

اولین برنامه: خاموش و روشن کردن LED روی برد

توضیح برنامه (ادامه)

- تابع digitalWrite():
- **digitalWrite(LED_BUILTIN, HIGH);**
  - تابع بالا مقدار پایه LED_BUILTIN (که همان پایه ۱۳ است) را یک (5V) می کند.
  - چون پایه خروجی شده است، LED روشن می شود.
  - این تابع معادل مقدار دادن به رجیستر PORT در AVR است
  - کلمه HIGH از پیش تعریف شده و برابر ۱ است.
  - بنابرین تابع بالا را برای Uno به شکل زیر هم می توان نوشت:
- **digitalWrite(13, 1);**

- نکته LOW کلمه از پیش تعریف شده و برابر 0 است.

اولین برنامه: خاموش و روشن کردن LED روی برد

توضیح برنامه (ادامه)

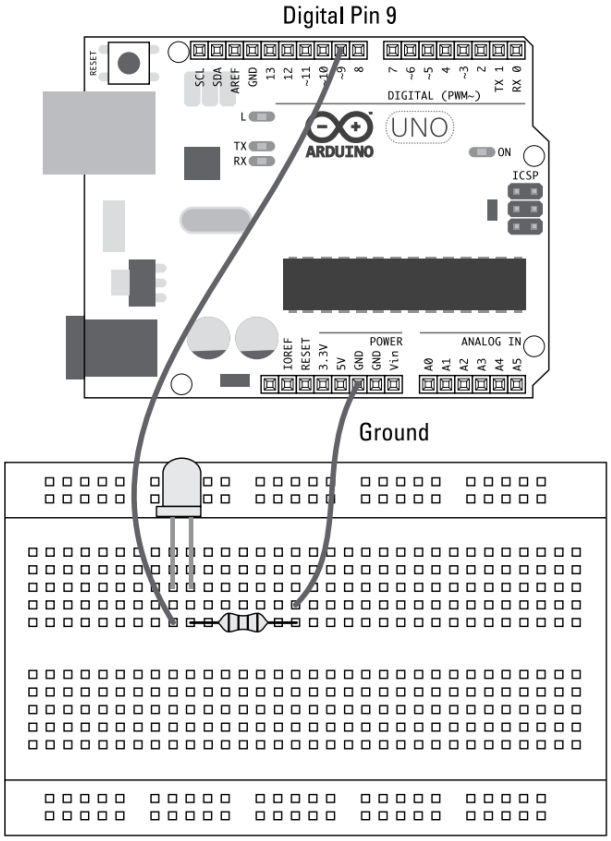
```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

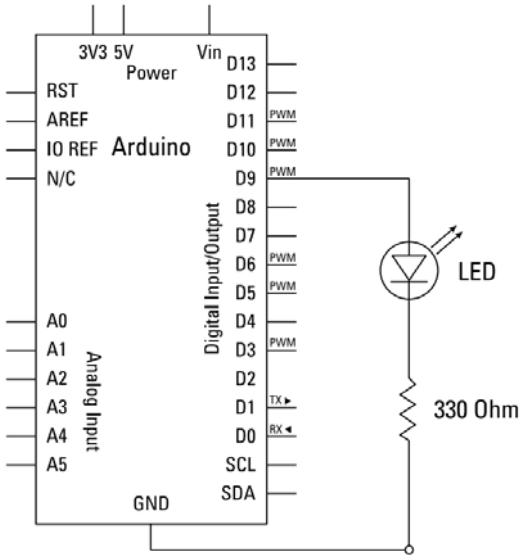
```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

- بنابرین برنامه نوشته شده (که در این صفحه تکرار شده است.) به این ترتیب عمل می کند:
- ابتدا در setup() که یکبار در ابتدای برنامه اجرا می شود. پایه ۱۳ که به LED روی برد متصل است، خروجی می شود.
- سپس برنامه داخل loop() برای همیشه تکرار می شود.
- در این برنامه: مقدار پایه ۱۳ یک و سپس صفر می شود.
- بین دستور ها تاخیر ۱۰۰۰ میلی ثانیه برابر با یک ثانیه قرار داده شده است:
- delay(1000)

مدار لازم برای اتصال یک LED خارجی برای برنامه اسلاید قبل

- نکته در این شکل از پایه 9دیجیتال برای فرمان به LED استفاده شده است.





مثال دوم: کنترل LED با کلید

- می خواهیم با فشردن کلیدی LED را خاموش و روشن کنیم.
- یک کلید به یک پایه آردوینو وصل می شود (پایه را ورودی می کنیم).
- و LED به پایه دیگر وصل می شود (پایه خروجی می کنیم).

```
const int buttonPin = 2;
const int ledPin =  13;
int buttonState = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

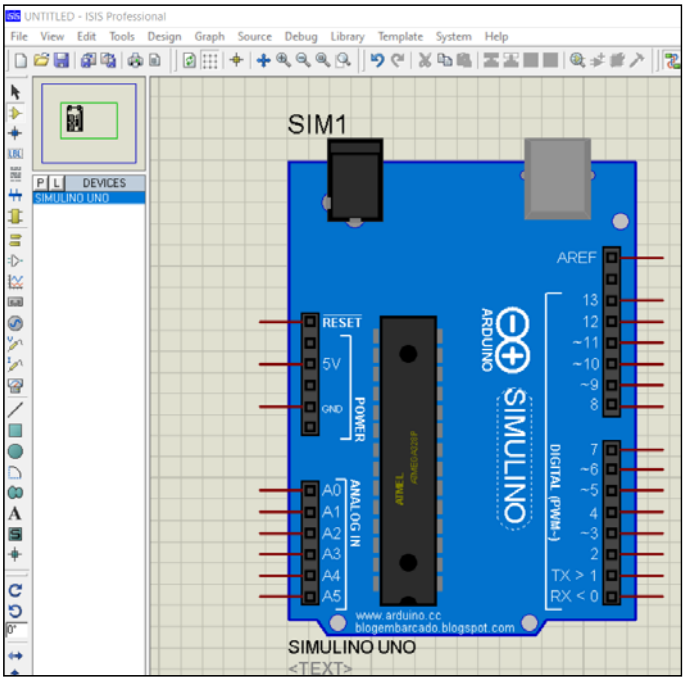
توضیح برنامه اسلاید قبل

- تابع pinMode
- **pinMode(buttonPin, INPUT_PULLUP);**
- تابع بالا پایه **buttonPin**(که همان پایه 2 است که بالا تعریف شده) را خروجی می کند.
- کلمه **INPUT_PULLUP** از پیش تعریف شده و به معنای ورودی کردن پایه و وصل مقاومت pull-up داخلی است.
- بنابرین تابع بالا را برای Uno به شکل زیر هم می توان نوشت:
- **pinMode(2, INPUT_PULLUP);**

توضیح برنامه اسلاید قبل (ادامه)

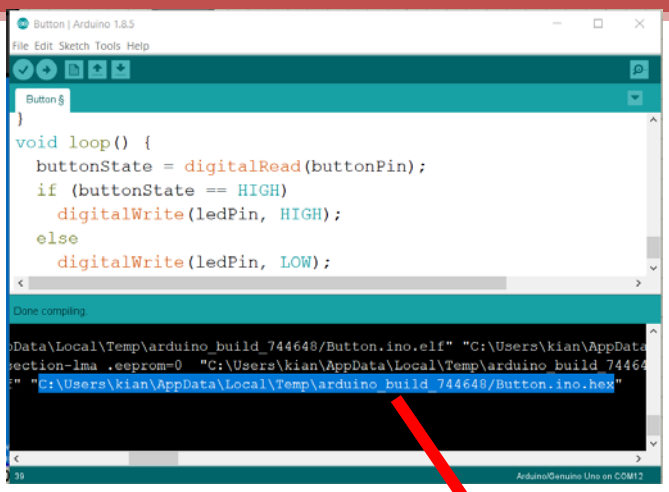
- تابع digitalRead():
- **digitaRead**(buttonPin);
- تابع بالا مقدار پایه buttonPin (که در برنامه برابر با 2 است) را می خواند.
- چون پایه ورودی و pull-up شده است، مقدار خوانده شده آن برابر با وضعیت کلید است.
- این تابع معادل خواندن رجیستر PIN در AVR است
- بنابرین تابع بالا را برای Uno به شکل زیر هم می توان نوشت:
- **digitalRead(2);**
- برنامه در loop به طور همیشگی وضعیت کلید را خوانده و برمبنای آن LED را روشن می کند.
- برای تمرین: عملکرد کلید را معکوس کنید و همزمان دو LED را فرمان دهی کنید.

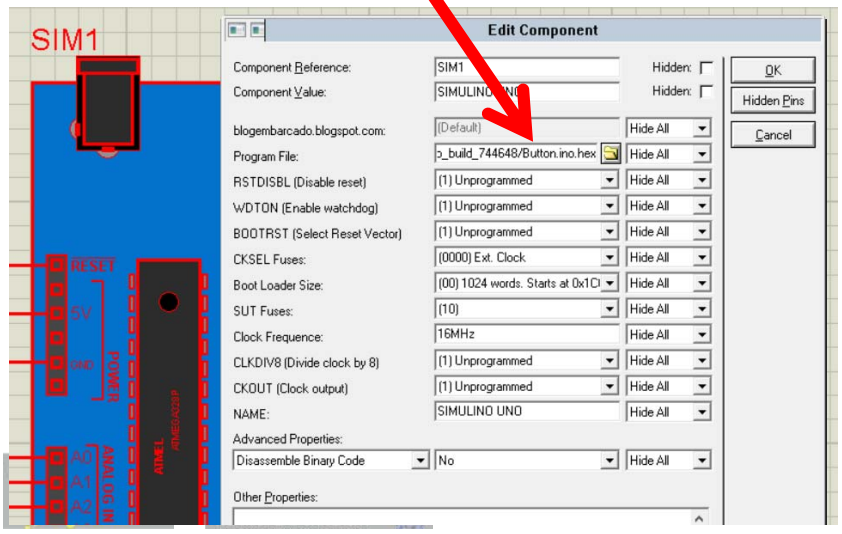
شبیه سازی Arduino در Proteus



- با توجه به کتابخانه های موجود امکان شبیه سازی برد در Proteus هم فراهم شده است.
- برای اینکار باید ابتدا کتابخانه لازم را دانلود و در فولدر کتابخانه های Proteus اضافه کنیم. (فایل ها روی وبلاگ در دسترس هستند.)
- سپس پس از قرار دادن برد روی صفحه شبیه سازی، برنامه کامپایل شده در Arduino IDE را به برد معرفی می کنیم.

نحوه اضافه کردن برنامه کامپایل شده برای شبیه سازی

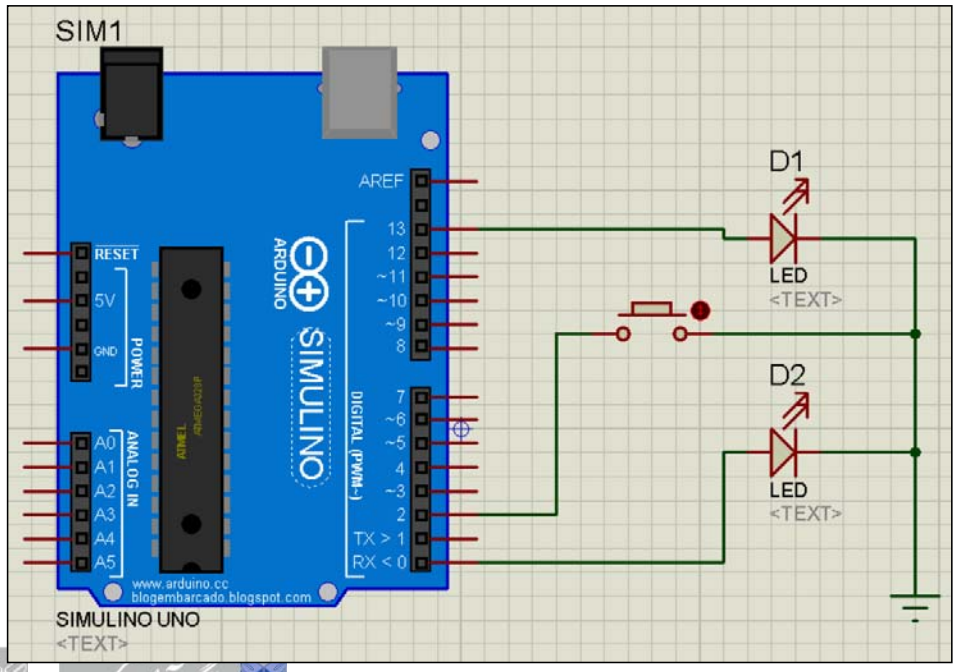




- پس از کامپایل در Arduino IDE صفحه پیامی در پایین صفحه دیده می شود. از متن پیام آدرس فایل hex را کپی کرده در قسمت Program File در مشخصات برد شبیه سازی در Proteus وارد می کنیم.

مثال: دو LED را خاموش و روشن کنید، سرعت خاموش و روشن شدن را با یک کلید کم و زیاد کنید.

- مدار مثال:



مثال: دو LED را خاموش و روشن کنید، سرعت خاموش و روشن شدن را با یک کلید کم و زیاد کنید.

- برنامه:

```
int buttonState = 0;
void setup() {
  pinMode(13, OUTPUT);pinMode(0, OUTPUT);
  pinMode(2,INPUT_PULLUP);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH)
    d=100;
  else
    d=1000;
  digitalWrite(13,1); digitalWrite(0,0);
  delay(d);
  digitalWrite(13,0); digitalWrite(0,1);
  delay(d);
}
```

مثال: دو LED را خاموش و روشن کنید، سرعت خاموش و روشن شدن را با یک کلید کم و زیاد کنید.

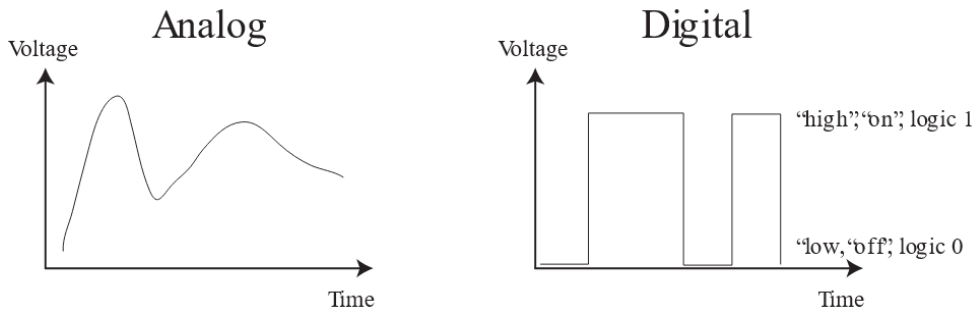
- توضیح برنامه:
- در بخش setup دو پایه 0و13 خروجی شده اند. و پایه دو ورودی با pull-up شده است.
- در بخش loop (که همیشه اجرا می شود.)
  - در حلقه وضعیت کلید بررسی می شود.
  - در دو حالت کلید مقدار متغیر d یکبار برابر با ۱۰۰ و در حالت دیگر برابر با ۱۰۰۰ قرار داده می شود.
  - مقدار تاخیر بین خاموش و روشن کردن LEDها برابر با d قرار داده شده است.
  - یعنی در یک حالت کلید ۱۰۰ میلی ثانیه تاخیر بین دستورها قرار داده می شود.
  - و در حالت دیگر ۱۰۰۰ میلی ثانیه برابر با ۱ ثانیه تاخیر بین دستورها قرار می گیرد.

ورودی و خروجی آنالوگ

- دیجیتال
  - پایه دیجیتال فقط می تواند دو مقدار 0 و 1 داشته باشد. (صفر ولت و پنج ولت)
  - چه در خروجی که برنامه ریز فرمان می دهد.
  - و چه در ورودی که برناکه ریز مقدار پایه را می خواند (بالاتر از ۲.۵ ولت یک و پایین تر از ۲.۵ ولت صفر در ورودی محسوب می شود.)
  - پایه های میکروکنترلر دیجیتال هستند.
- آنالوگ
  - می توان مقداری پیوسته بین حداکثر و حداقل مجاز به عنوان مقدار آنالوگ داشت.
  - یعنی در خروجی بتوانیم ولتاژی دلخواه میان صفر و ۵ ولت تولید کنیم.
  - و در ورودی بتوانیم مقداری بین صفر تا ۵ ولت را دریافت کنیم.
  - در آردوینو تولید خروجی آنالوگ به کمک PWM و دریافت ورودی آنالوگ به کمک ADC انجام می شود.

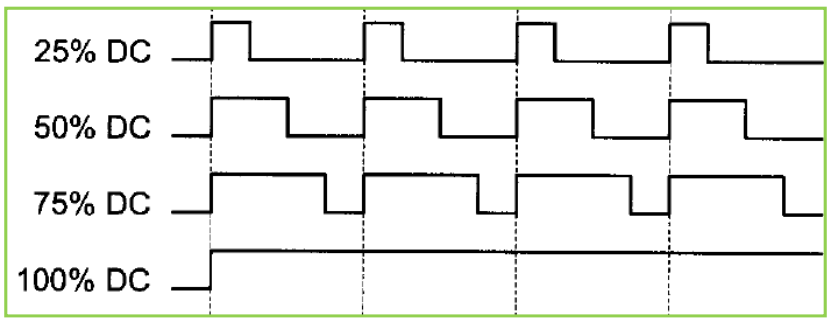
مقدار دیجیتال و آنالوگ در طول زمان

- مقدار دیجیتال فقط دو سطح ۰ یا یک دارد.
- مقدار آنالوگ می تواند مقداری پیوسته میان حداکثر و حداقل داشته باشد.



تولید ولتاژ آنالوگ روی پایه های Arduino

- روی شش پایه دیجیتال آردوینو می توان ولتاژی آنالوگ بین ۰ تا ۵ ولت تولید کرد.
- این کار با ایجاد موج PWM روی پایه های دیجیتال انجام می شود.
- شکل موج PWM، موجی مربعی است که پهنای پالس آن قابل تنظیم است (با فرکانس ثابت)
- با ساخت شکل موج PWM می توان مقدار DC (آنالوگ) یا میانگین سیگنال تولیدی را تغییر داد.
- کاربرد:
  1. کنترل سرعت موتور DC.
  2. کنترل شدت نور.
  3. تبدیل دیجیتال به آنالوگ در فرکانس پایین.



- می توان این موج را روی پایه های ۳، ۵، ۶، ۹، ۱۰، و ۱۱ برد آردوینو تولید کرد.

تابع کاربردی آردوینو برای ایجاد ولتاژ آنالوگ در خروجی (تولید موج PWM)

- تابع analogWrite():
- **anlalogWrite(مقدار خروجی ,شماره پایه);**
- تابع بالا مقدار ولتاژ پایه تعیین شده را برابر با مقدار خروجی قرار می دهد.
- مقدار خروجی (N) باید عددی بین ۰ تا ۲۵۵ باشد.
- ولتاژ خروجی متناسب با N ایجاد می شود.
- اگر N=0 باشد ولتاژ خروجی صفر و اگر N=255 باشد ولتاژ خروجی ۵ ولتاژ می شود.
- می توان بین ولتاژ خروجی (V) و N را بطه زیر را نوشت.

$$V = \frac{N}{255} \times 5$$

- به عنوان مثال تابع زیر مقدار ۲.۵ ولت را روی پایه ۶ ایجاد می کند. (N=128)
- **analogWrite(6,128);**

مثال از به کارگیری خروجی آنالوگ

- با تغییر مقدار ولتاژ خروجی آنالوگ می توان شدت نور یک LED را تغییر داد.
- برنامه ای بنویسید که در آن شدت نور LED به تدریج افزایش یابد تا به حداکثر خود برسد و سپس این روند دوباره تکرار شود.
- ابتدا پایه ۶ را خروجی می کنیم.
- سپس متغیری با نام light تعریف می کنیم.
- هر ۲۰۰ ms ده تا light را اضافه می کنیم.
- مقدار light را به شکل آنالوگ در پایه ۶ می نویسیم.
- زمانی که light به ۲۵۰ رسید (حداکثر نور)، light را صفر می کنیم (حداقل نور) و این روند تکرار می شود.

```
void setup() {
  pinMode(6,OUTPUT);
}
int light=0;
void loop() {
  analogWrite(6,light);
  light=light+10;
  delay(200);
  if (light==250)
     light=0;
}
```

```
void setup() {
  pinMode(9,OUTPUT);
}
int light=0,i;
void loop() {
  for (i=0;i<=25;i++)
  {
    light=i*10;
    analogWrite(9,light);
    delay(200);}
  for (i=25;i>=0;i--)
  {
    light=i*10;
    analogWrite(9,light);
    delay(200);}
}
```

مثال دوم

- برنامه ای بنویسید که در آن LED ابتدا به تدریج روشن و سپس به تدریج خاموش شود. و این روند تکرار شود.
- برای اینکار باید دو مرحله افزایشی و کاهشی برای light داشته باشیم.
- دو حلقه for به کار می بریم.
- در حلقه اول i افزایشی است و مقدار آن پس از ضرب در ۱۰ به light و سپس به پایه آنالوگ انتقال می یابد.
- اما در حلقه دوم i از ۲۵ تا صفر کاهش
- می یابد.

مثال سوم

- سوال قبل را برای دو LED انجام دهید با این که روند تغییر نور LED دوم برعکس LED اول باشد.

```
void setup() {
  pinMode(9,OUTPUT); pinMode(10,OUTPUT);
}
int light=0,i;
void loop() {
  for (i=0;i<=25;i++)
  {
    light=i*10;
    analogWrite(9,light); analogWrite(10,250-light);
    delay(200);}
  for (i=25;i>=0;i--)
  {
    light=i*10;
    analogWrite(9,light); analogWrite(10,250-light);
    delay(200);}
}
```

ورودی آنالوگ

- در برد آردوینو این امکان قرار داده شده است تا مقدار آنالوگ هم قابل خواندن توسط برنامه باشد. این امکان با ابزاری با نام ADC (مبدل آنالوگ به دیجیتال) انجام می شود.
- ADC در ورودی ولتاژ آنالوگی را دریافت کرده و در خروجی عددی دیجیتال تحویل می دهد.
- محدوده عدد خروجی بستگی به رزلوشن (دقت) ADC دارد.
- مثلاً برای ADC ۱۰ بیتی عدد خروجی بین ۰ تا ۱۰۲۳ است.
- و برای ADC ۸ بیتی عدد خروجی بین ۰ تا ۲۵۵ است.
- اگر ولتاژ آنالوگ ورودی یک ADC n بیتی بین ۰ تا ۵ ولت باشد، عدد تولید شده آن (D) با ولتاژ آنالوگ ورودی دارای رابطه زیر است.

$$D = \frac{V}{5} \times (2^n - 1)\ ,\ (\text{برای آردوینو}) D = \frac{V}{5} \times 1023$$

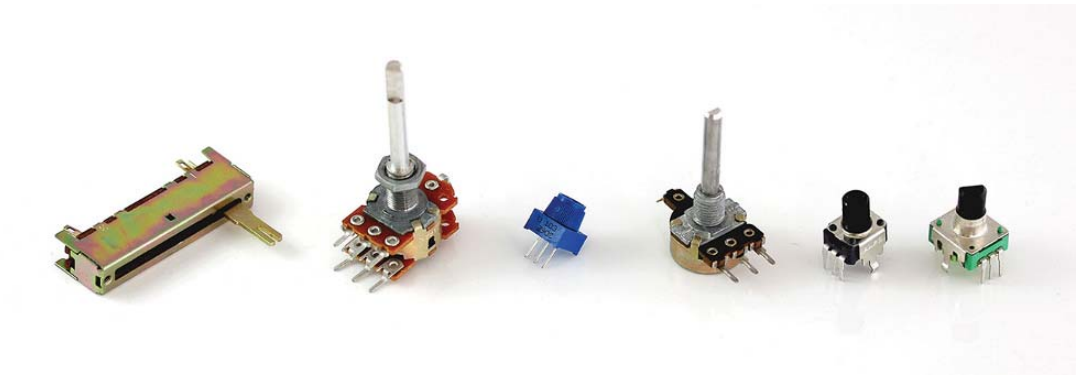
- شش ورودی مجزا برای ورودی آنالوگ روی برد آردوینو تعبیه شده است.

تابع کاربردی آردوینو برای خواندن ولتاژ آنالوگ ورودی (به کمک ADC)

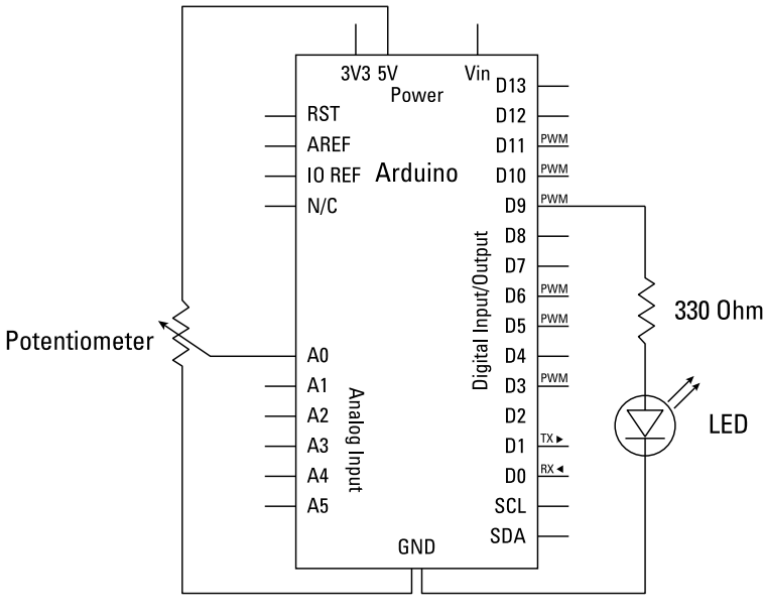
- تابع analogRead()
- **D=analogRead(شماره پایه آنالوگ ورودی);**
- تابع بالا مقدار ولتاژ آنالوگ پایه داده شده را خوانده و تبدیل به دیجیتال می کند. عدد ایجاد شده در خروجی تابع تحویل داده می شود.
- مقدار (D) عددی بین ۰ تا ۱۰۲۳ است (ADC میکروکنترلر روی برد ۱۰ بیتی است).
- عدد D متناسب با ولتاژ ورودی پایه است
- اگر D=0 باشد ولتاژ ورودی صفر و اگر N=1023 باشد ولتاژ ورودی ۵ ولت بوده است.

مثال از خواندن مقدار آنالوگ

- برنامه بنویسید که مقدار آنالوگ پایه A0 را بخواند و بر طبق آن سرعت خاموش و روشن شدن LED را کم و زیاد کند. با ولتاژ نزدیک به ۵ سرعت کم و با ولتاژ نزدیک به سرعت زیاد شود.
- می توان با پتانسیومتر (مقاومت متغیر) ولتاژ آنالوگ تولید کرد.
- نمونه ای از پتانسیومتر ها:



مدار سوال اسلاید قبلی



- پتانسیومتر سه سر دارد.
- اگر سر بالایی و پایینی به منبع تغذیه و زمین وصل شود، سر میانی آن ولتاژی آنالوگ بین منبع تغذیه و زمین خواهد داشت.
- پایه ۹ برای روشن کردن LED به کار رفته است.
- در این مثال پایه ۹ به شکل دیجیتال به کار گرفته شده است.

پاسخ مثال

- ابتدا پایه ۹ را خروجی تعریف می کنیم.
- سپس در حلقه مقدار آنالوگ را می خوانیم.
- برای اینکه محدوده D بین ۰ تا ۲۵۵ شود، مقدار خوانده شده از ADC را به چهار تقسیم کرده ایم.
- پایه ۹ را خاموش و روشن کرده ایم.
- تاخیر بین دستور ها را برابر با مقدار D میلی ثانیه قرار داده ایم.
- بنابرین اگر ورودی ۵ ولت باشد، D ۲۵۶ می شود. و ۲۵۶ میلی ثانیه تاخیر بین دستورها داریم.
- اگر ولتاژ کم باشد، D کوچک و تاخیر کم و سرعت چشمک زدن LED بیشتر می شود.

```
void setup() {
pinMode(9,OUTPUT);
}
int D;
void loop() {
  D=analogRead(0)/4;
  digitalWrite(9,HIGH);
  delay(D);
  digitalWrite(9,LOW);
  delay(D);
}
```

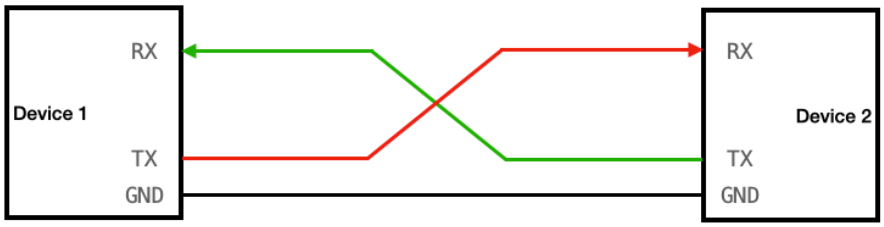

مثال دوم: شدت نور LED را با پتانسیومتر تنظیم کنید.
LED دوم را با عملکرد معکوس به مدار اضافه کنید.

- در این مثال نیز مقدار آنالوگ ورودی خوانده شده و محدوده آن با تقسیم به چهار به ۰ تا ۲۵۵ تغییر داده شده است.
- سپس همین مقدار برای ایجاد ولتاژ آنالوگ روی پایه ۹ استفاده شده است.
- روی پایه ۱۰ هم با اعمال 255-D تغییر شدت نور برعکس شده است.
- یعنی زمانی که به عنوان مثال ولتاژ تولیدی برای LED اول 4 ولت است و LED پر نور است، ولتاژ تولیدی برای LED دوم 1 ولت است و LED کم نور است.

```
void setup() {
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
}
int D;
void loop() {
  D=analogRead(0)/4;
  analogWrite(9,D);
  analogWrite(10,255-D);
}
```

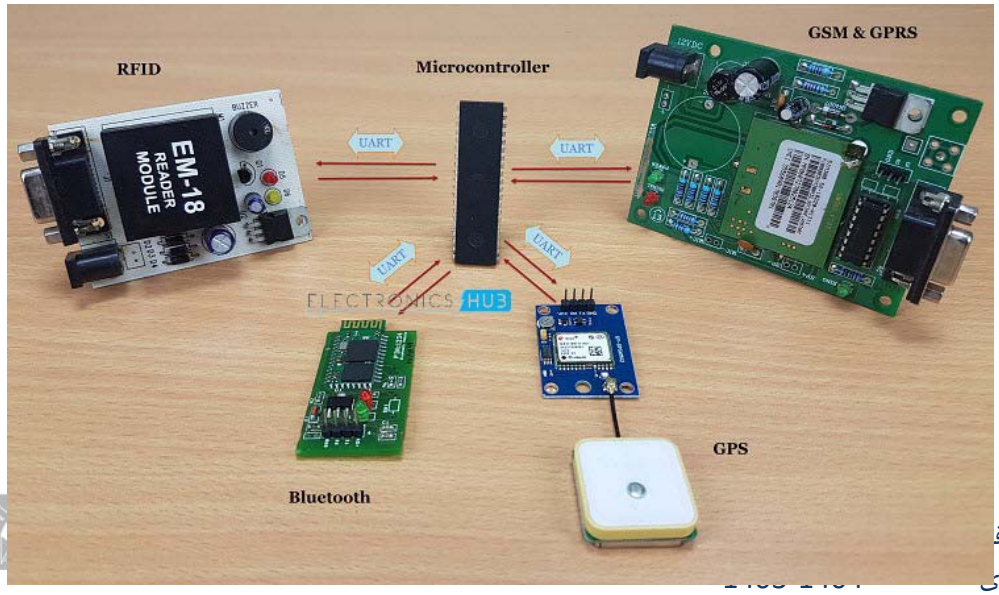
ارتباط با استاندارد سری UART

- در ارتباط سری، دو دستگاه می توانند از طریق یک یا دو سیم و یک سیم زمین با یکدیگر ارتباط برقرار کنند.
- ارسال اطلاعات به شکل یک بیت در هر واحد زمان انجام می شود.
- چندین بیت کنار هم قرار می گیرند و یک بسته را ایجاد می کنند (مثلاً به طول ۸ بیت)
- شروع و پایان بسته با علائم خاصی مشخص می شود.
- یکی از استانداردهای ارتباط سری (universal asynchronous receiver transmitter)UART است که در آن نحوه اتصال دو دستگاه به شکل زیر است.
- اگر فقط یک دستگاه فرستنده و دیگری گیرنده باشد، یکی از سیم های TX به RX کافی است.
- سیم GND اجباری است.



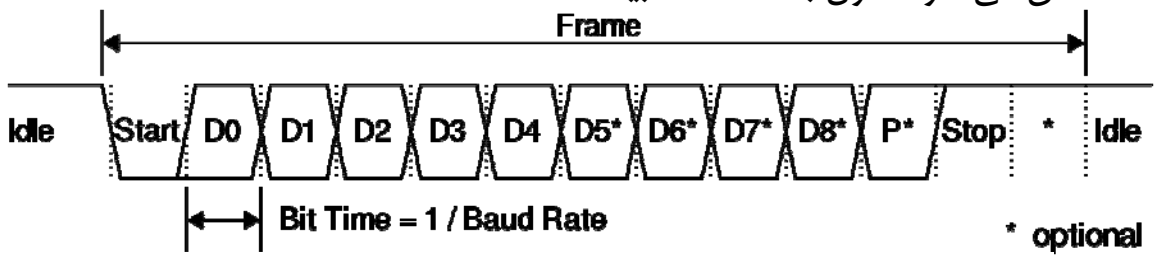
کاربرد ارتباط سری

- به کمک ارتباط سری می توان ارتباط میکروکنترلر (مجزا یا در برد آردوینو) را با سایر دستگاه ها مانند سنسورها و دستگاه های جانبی ارتباطی (بلوتوث و برد GSM و...) برقرار کرد.
- همچنین می توان ارتباط دو میکرو کنترلر یا دو برد آردینو را با هم برقرار کرد.
- به غیر از UART، دو استاندارد سری متداول دیگر SPI و I2C هستند.

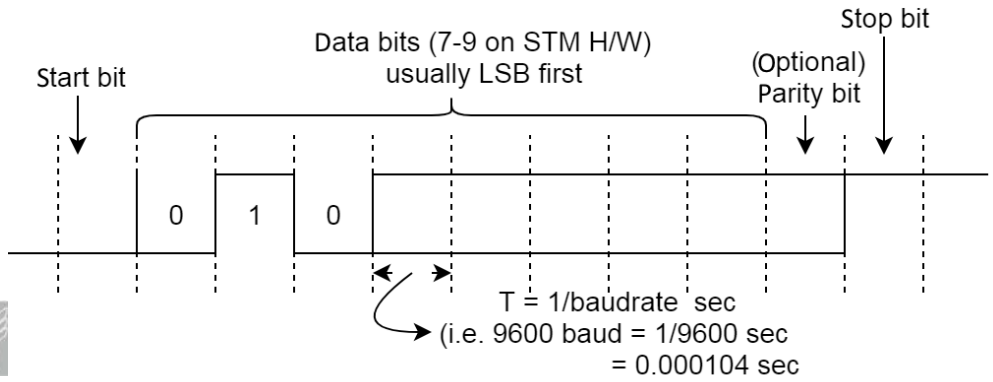


شکل بسته ارسالی در استاندارد سری UART

- مقدار صفر در ابتدا و مقدار یک در انتها قرار دارد. فاصله زمانی بیتها با baud rate مشخص می شود. طول بسته ۵ تا ۹ بیت است.



UART Framing



برنامه نویسی در آردوینو برای برقراری ارتباط سری UART

- در آردوینو این امکان برقرار شده است که اگر برد با پورت USB به کامپیوتر شخصی متصل باشد بتوان از طریق UART بین برد آردینو و کامپیوتر شخص پیام رد و بدل کرد.

- برای استفاده از UART توابع زیر موجود است:

- Serial.begin(9600); راه اندازی اولیه با بود ریت ۹۶۰۰
- Serial.write('a'); یک کاراکتر را روی پورت آردوینو ارسال می کند.
- Serial.print ("abcd"); مجموعه ای از کاراکترها را روی پورت آردوینو ارسال می کند.
- Serial.println ("abcd"); مشابه تابع قبل اما در پایان ارسال انتقال به خط بعدی
- در هر دو تابع می توان عدد را هم به جای کاراکترها ارسال کرد.

مثال برای ارسال داده با UART

- مثال: برنامه ای بنویسید که هر ۰.۵ ثانیه مقدار آنالوگ پایه A0 را بخواند و مقدار خوانده شده را (که عددی بین ۰ تا ۱۰۲۳ است.) ابتدا به عددی بین ۰ تا ۲۵۵ تبدیل کند و آن را از طریق UART برای کامپیوتر شخصی ارسال کند.

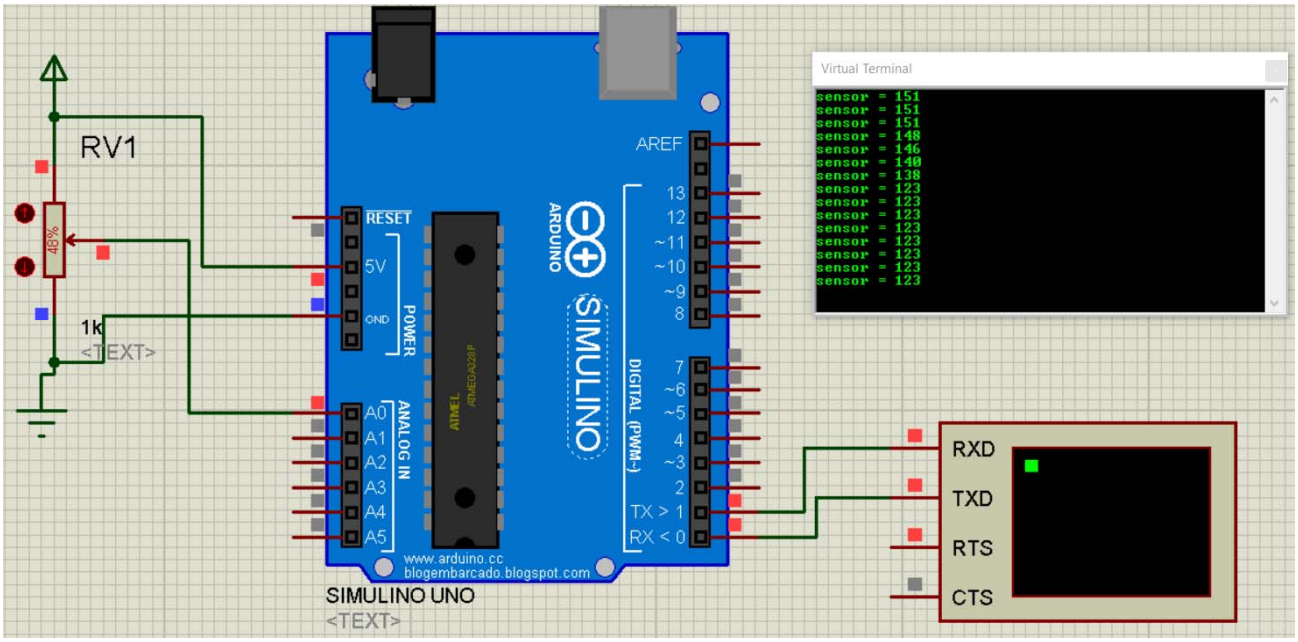
```
int D = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  D = analogRead(A0);
  D = D/4;
  Serial.print("sensor = ");
  Serial.println(D);
  delay(500);
}
```


بررسی پاسخ مثال قبل

- ابتدا در setup (که یکبار اجرا می شود.) پورت UART را راه اندازی اولیه می کنیم.
- سپس در حلقه loop (که همیشه اجرا می شود.)
  - ابتدا مقدار آنالوگ را از پایه A0 می خوانیم. عدد خوانده شده در D ذخیره می شود که مقدار آن بین ۰ تا ۱۰۲۳ است. سپس با تقسیم به چهار محدوده آن را به ۰ تا ۲۵۵ تغییر داده می شود.
  - سپس در بخش ارسال ابتدا یک رشته از کاراکتر ها به عنوان راهنما و سپس مقدار دریافت شده از ورودی آنالوگ فرستاده می شود.
  - در انتها ۰.۵ ثانیه تاخیر در حلقه گذاشته شده است.

شبیه سازی مثال

- می توان با استفاده از ترمینال مجازی پروتئوس پیام ارسالی از آردینو را نمایش داد.



مثال دوم از ارسال با UART

- برنامه ای بنویسد که با هر بار فشردن کلیدی مقدار آنالوگ پایه را ارسال کند.
- تفاوت با مثال قبل: درمثال قبلی هر ۰.۵ ثانیه ارسال انجام می شد. حالا با فشردن کلید باید همان ارسال انجام شود.

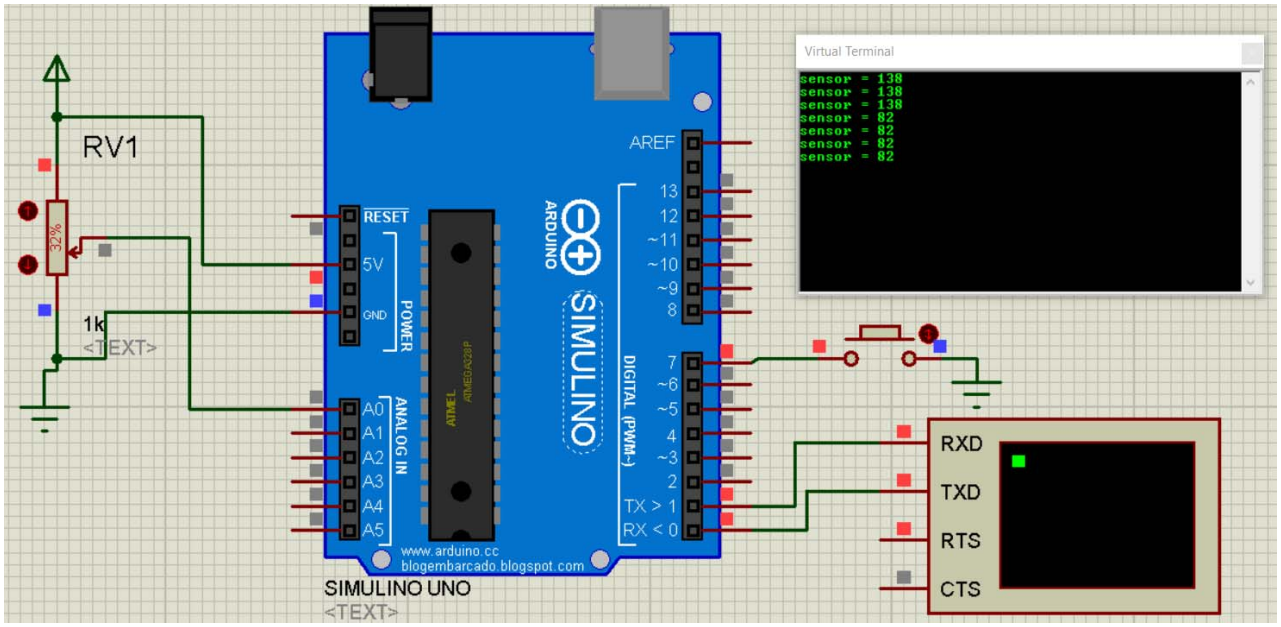
```
int D = 0;
void setup() {
  Serial.begin(9600);
  pinMode(7,INPUT_PULLUP);
}
void loop() {
  while (digitalRead(7)==1) {};انتظار برای فشردن کلید
  while (digitalRead(7)==0) {};انتظار برای رها کردن کلید
  D = analogRead(A0);
  D = D/4;
  Serial.print("sensor = ");
  Serial.println(D);
}
```

توضیح مثال

- در این مثال ابتدا پایه ۷ با Pull-up ورودی شده است.
- سپس در حلقه loop دو حلقه while گذاشته ایم.
- حلقه اول تا زمانی که کلید فشار داده نشده صبر می کند. (مقدار پایه ۱ است.)
- حلقه دوم تا زمانی که کلید فشار داده شده صبر می کند. (مقدار پایه ۰است.)
- یعنی کاربر باید کلید را فشار دهد و رها کند تا ادامه برنامه که ارسال است انجام شود.
- این دو حلقه while برای تشخیص لبه رو کلید قرار داده شده است.
- اگر یکی ار این حلقه ها وجود نداشته باشد. با یک فشار کلید تعداد زیادی مقدار ارسال می شود. چون برنامه در حلقه loop چندین بار دور خواهد زد.

شبیه سازی مثال دوم

- با هر بار فشار کلید متصل به هفت یک خط به خروجی ارسال می شود.



دریافت از پورت UART

- برای دریافت پیام ها روی برد آردوینو توابع زیر تعریف شده است.
- Serial.available();
- این تابع که مقدار خروجی آن صفر یا یک است نشان می دهد. که داده ای از پورت UART دریافت شده است یا خیر.
- Serial.read();
- اگر خروجی تابع availableیک باشد می توان با این تابع یک پیام (۸ بیتی) را دریافت کرد.

مثال از دریافت با UART

- برنامه ای بنویسید که یک عدد یک رقمی از UART دریافت کند (۰ تا ۹) و با توجه به آن شدت نور یک LED را تنظیم کند. (مقدار آنالوگ یک پایه خروجی را تنظیم کند.)

```
void setup() {
  Serial.begin(9600);
  pinMode(3,OUTPUT);
}
int light;
void loop() {
  if (Serial.available())
     {
       light=Serial.read()-'0';
       light=light *28; // 9*28=252 almost full light
       analogWrite(3,light);
     }
}
```

توضیح پاسخ مثال

- در ابتدا پایه سه خروجی و پورت UART راه اندازی می شود.
- اگر بایتی دریافت شود، (Available)
  - بایت خوانده می شود با صفر مقایسه می شود
  - مقدار مقایسه برای تعیین ولتاژ آنالوگ خروجی به کار می رود.
  - اگر ورودی ۹ باشد با ضرب در ۲۸ به ۲۵۲ تبدیل می شود که تقریبا LED را به طور کامل روشن می کند.

مثال دوم دریافت با UART

- می خواهیم با سه فرمان g، r، و b سه کار مختلف روی LED ها انجام شود.
  - g: LED سبز روشن شود.
  - r: LED قرمز روشن شود.
  - b: به شکل چشمک زن یکی در میان هر دو روشن شوند.

```
void setup() {
  Serial.begin(9600);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
}
int m;
void loop() {
  if (Serial.available())
     {
      m=Serial.read();
      if (m=='r')
         digitalWrite(2,HIGH);
      else
         digitalWrite(2,LOW);
      if (m=='g'||m=='b')
         digitalWrite(3,HIGH);
      else
         digitalWrite(3,LOW);
     }
  if (m=='b')
    {
      digitalWrite(2,!digitalRead(2));
      digitalWrite(3,!digitalRead(3));
    }
  delay(100);
}
```