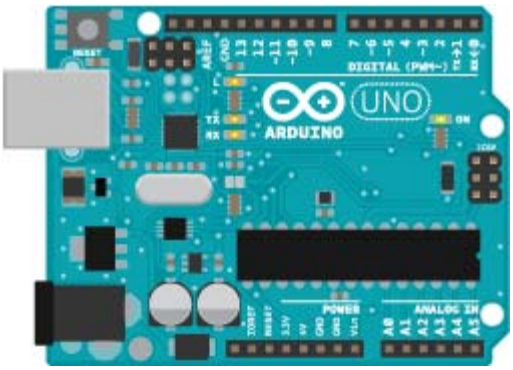
برنامه نویسی سخت افزاری



سرفصل بخش چهارم :

- ارتباط سریال چیست ؟
- بررسی ارتباط سریال در آردوینو های مختلف
- ارتباط سریال در آردوینو
- ارتباط اردوینو با کامپیوتر
- دستورات پایه ای ارتباط سریال آردوینو
- دستورات ارسال دیتای سریال در آردوینو
- دستورات دریافت دیتای سریال در آردوینو
- دستورات کار با دیتای سریال در آردوینو

❖ ارتباط سریال چیست ؟

برد های اردوینو برای ارتباط با دنیای بیرون مجموعه پروتکل های ارتباطی دارند که از طریق این پروتکل ها می توانند با کامپیوتر ، موبایل ، ماژول و سنسور های مختلف ارتباط برقرار کنند ، در میکرو های AVR و برد های اردوینو پروتکل هایی مانند I2C ، SPI ، ONE - WIRE و ... وجود دارند کا از بین این پروتکل ها ارتباط سریال از سادگی خاصی برخوردار است که به راحتی و با کمترین هزینه می توان از این پروتکل برای کار های مد نظر استفاده کرد ، در ارتباط سریال بیت ها بصورت سری و پشت سر هم از طریق یک سیم ارسال می شوند ، ارتباط سریال دارای استاندارد های مختلفی است که در اردینو و میکروکنترل های AVR از پروتل سطح منطقی TTL استفاده می کند یعنی داده‌ها با دامنه 0 و 5+ ولت انتقال می یابند .

در ارتباط سریال چند مشخصه مهم داریم که باید به آنها توجه کنیم :

در روش انتقال سریال, داده ها چون فقط از یه سری ۰ و ۱ تشکیل میشه درکشون سخته واسه همین فرستنده و گیرنده از قوانین مشترکی برای ارسال و دریافت داده ها استفاده میکنن.
این قوانین شامل:

۱ – سرعت انتقال داده ها(baud rate) : سرعت انتقال داده می تواند 300 Baud تا 250000 Baud باشد ، که معمولا از 9600 استفاده می کنند

۲ – طول بیت داده(bit length) : طول بیت هم میتونه بین ۵ تا ۹ بیت باشه و ابن بیتها بین بیت آغاز و پایان فرستاده میشن

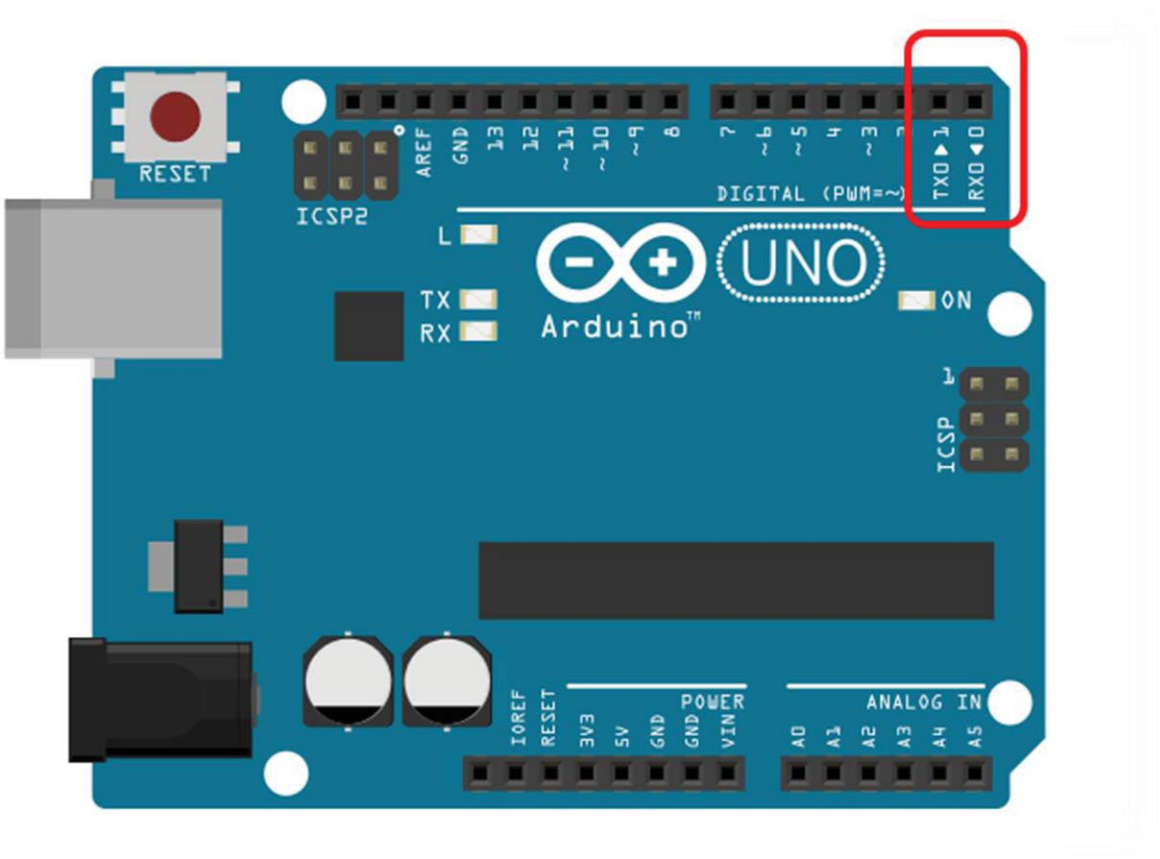
۳ –بیت های آغاز و پایان(Start - Stop bit) : بیت آغاز همیشه یک بیته و اونم همیشه صفره ولی بیت پایان همیشه یکه و میتونه از یک یا دو بیت تشکیل بشه. از دو بیت واسه سیستمهای قدیمی که کند بودن استفاده میشد که تا اومدن بایت جدید زمان داشته باشه خودشو جمع و جور کنه که الانه همه از یه بیت پایان استفاده میکنن.

۴ – بیت توازن(parity bit) : بیت توازن هم تو بعضی از سیستم ها واسه حفظ درستی داده استفاده میشه. این بیت توازن مدلهای مختلف داره که عبارتند از توازن–فرد, توازن–زوج ,

بدون-توازن و ... . روش کارش هم اینه که اگه توازن توازن-فرد (odd-parity) رو انتخاب کردیم تعداد کل یک های بیت ارسالیمون به اضافه بیت توازن, فرده .یعنی اگه تو دادمون دو تا یک وجود داشته باشه بیت توازنمون میشه ۱ ولی اگه سه تا یک وجود داشته باشه میشه ۰. تا تعادل برقرار بشه .

❖ بررسی ارتباط سریال در آردوینو های مختلف

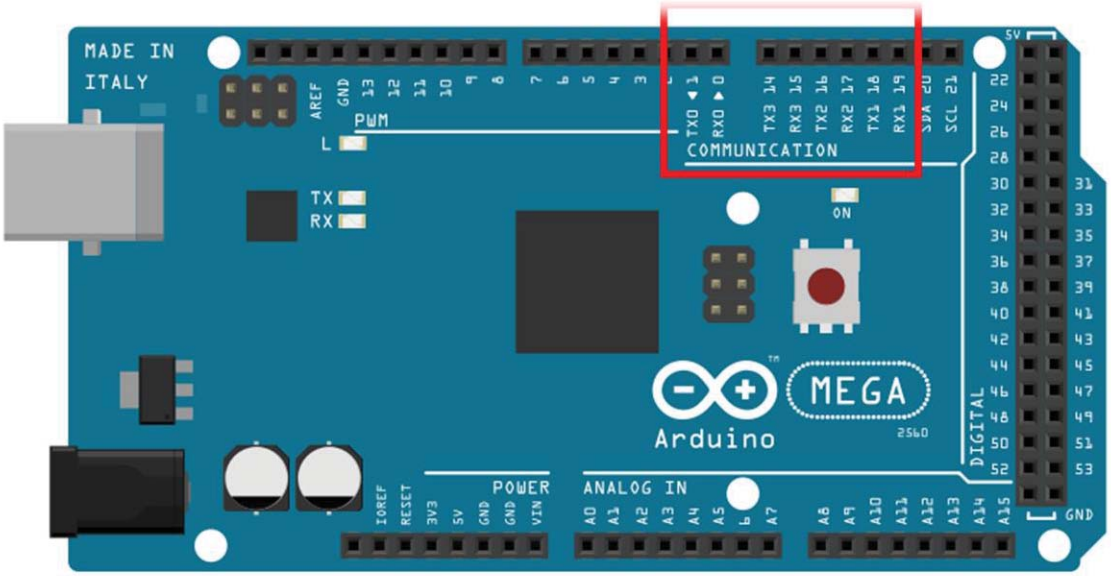
آردوینو دارای دو نوع ارتباط سریال است یکی ارتباط سریال سخت افزاری و یکی هم ارتباط سریال نرم افزاری که فعلا به ارتباط سریال سخت افزاری می پردازیم ، در اردوینو هایی که با atmega328 ، atmega32u4 و میکرو های مشابه طراحی شده اند مانند Arduino nano ، Arduino uno و ... دارای یک پل ارتباطی سریال (یک پایه برای ارسال اطلاعات TX و یک پایه برای دریافت اطلاعات RX) هستند ، پایه های 0 و 1 در این اردوینو ها برای ارتباط سریال هستند



در تصویر بالا همان طور که می بینید پایه 0 اردوینو برای دریافت اطلاعات (Rx) و پایه 1 اردوینو برای ارسال اطلاعات (Tx) است

در مدل هایی که از میکرو mega و arm استفاده می شود ، اردوینو دارای تعداد پل های ارتباطی سریال بیشتری است

بعنوان مثال در زیر Arduino MEGA 250 رو می بینید:



در این آردوینو چهار پل ارتباط سریال یا بهتره بگیم چهار درگاه ارتباط می بنید که عبارتند از

درگاه سریال 0 :

پایه 0 بعنوان Rx و پایه 1 اردوینو بعنوان Tx عمل می کند (مانند اردوینو هایی که معرفی کردیم)

درگاه سریال 1 :

پایه 19 بعنوان Rx و پایه 18 اردوینو بعنوان Tx عمل می کند

درگاه سریال 2 :

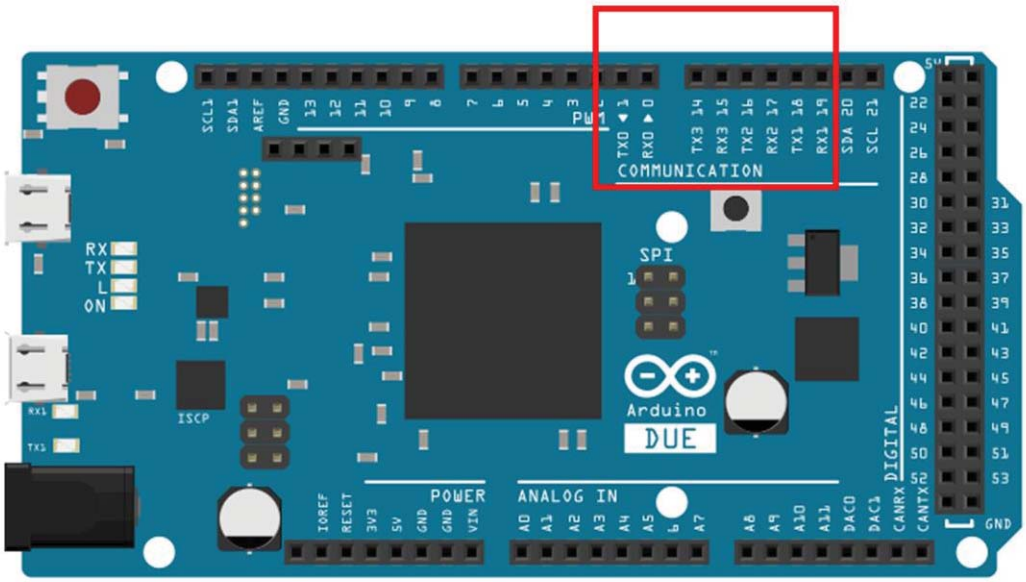
پایه 17 بعنوان Rx و پایه 16 اردوینو بعنوان Tx عمل می کند

درگاه سریال 3 :

پایه 15 بعنوان Rx و پایه 14 اردوینو بعنوان Tx عمل می کند

توجه : سایر اردوینو هایی هم که با میکروی مگا طراحی می شوند به همین شکل می باشند

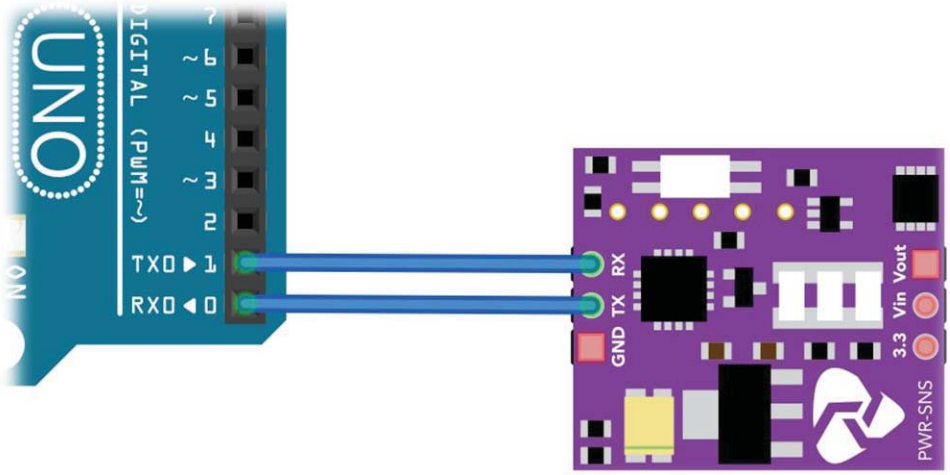
همچنین در آردوینو های سری ARM مانند ARDUINO DUE نیز مشخصات درگاه سریال دقیقا مانند اردوینو سری مگا می باشد



توجه مهم : سطرح ولتاژ منطقی در پایه های آردوینو هایی که با ARM طراحی می شوند 3.3 ولت است و ما نباید به هیچ یک از پایه ها بیشتر از 3.3 ولت بدهیم وگرنا آردوینو آسیب می بیند

❖ ارتباط سریال در آردوینو

در ارتباط سریالی که ما در آردوینو از آن استفاده می کنیم ، از دو پایه RX وTX برای تبادل اطلاعات استفاده می کنیم ، ماژول ، سنسور و یا هر چیز دیگه ای که ما قصد برقراری ارتباط سریال با آن را داریم باید دارای این دو پین و یا یکی از این دو پین باشد ، از پایه TX برای ارسال اطلاعات و از پایه RX برای دریافت اطلاعات استفاده می کنیم ، وقتی میخواهیم ماژولی که دارای درگاه سریال است را به اردوینو متصل کنیم ، باید پایه TX ماژول را به پایه RX اردوینو وصل کنیم (اگر قصد دریافت اطلاعات را داریم) در این صورت ماژول اطلاعات را از طریق پایه TX ارسال می کند و ما از طریق پایه RX اردوینو آن اطلاعات را دریافت و پردازش می کنیم ، همچنین اگر قصد داشته باشیم اطلاعات را از طریق اردوینو به ماژول بفرستیم باید پایه RX ماژول را به پایه TX اردوینو متصل کنیم تا بتوانیم اطلاعات آن را از ازدوینو برای ماژول بفرستیم ، در زیر یک نمونه از این نوع اتصال را می بینید



بسته به این که باید اطلاعات را ارسال و دریافت کنیم و یا فقط دریافت کنیم و یا اینکه فقط باید ارسال کنیم می توانیم از این پایه ها استفاده کنیم ، اگر ما به ارتباط دو طرفه بین ماژول و اردوینو نیاز داشته باشیم باید باید از هر دو پایه RX و TX اردوینو و ماژول استفاده کنیم ولی اگر فقط به ارسال و یا فقط به دریافت اطلاعات نیاز داریم بسته به نیازمان می توانیم از یک پایه استفاده کنیم ، اکر می خواهیم فقط اطلاعات را از اردوینو به ماژول بفرستیم باید پایه TX اردوینو را به RX ماژول وصل کنیم و اگر فقط میخواهیم اطلاعات را از ماژول دریافت کنیم باید پایه RX اردوینو را به TX ماژول وصل کنیم

❖ ارتباط اردوینو با کامپیوتر

همه اردوینو ها (به جز تعداد محدودی) دارای یک رابط USB هستند که از طریق آن اردوینو را به کامپیوتر وصل می کنیم و به وسیله همان درگاه ، آردوینو را پروگرام می کنیم . درگاه USB از طریق یک مبدل (usb to ttl) به سریال تبدیل می شود و اردوینو از طریق همان پایه های RX و TX پروگرام می شود . در واقع آردوینو یک AVR یا ARM هست که روی یک فیبر سوار شده و از طریق یک مبدل usb to ttl با کامپیوتر ارتباط برقرار می کند . پس ما از طریق همین usb هم اردوینو را پروگرام می کنیم و هم یک ارتباط سریال بین کامپیوتر و اردوینو برقرار می کنیم که به راحتی می توانیم اطلاعات را بصورت دو طرفه بین اردوینو و کامپیوتر رد و بدل کنیم .

در کامپایلر آردوینو یک سریال مانیتور (serial monitor) تعبیه شده است که از طریق آن می توان هم اطلاعات را ارسال کرد و هم اطلاعات دریافت شده را نمایش داد

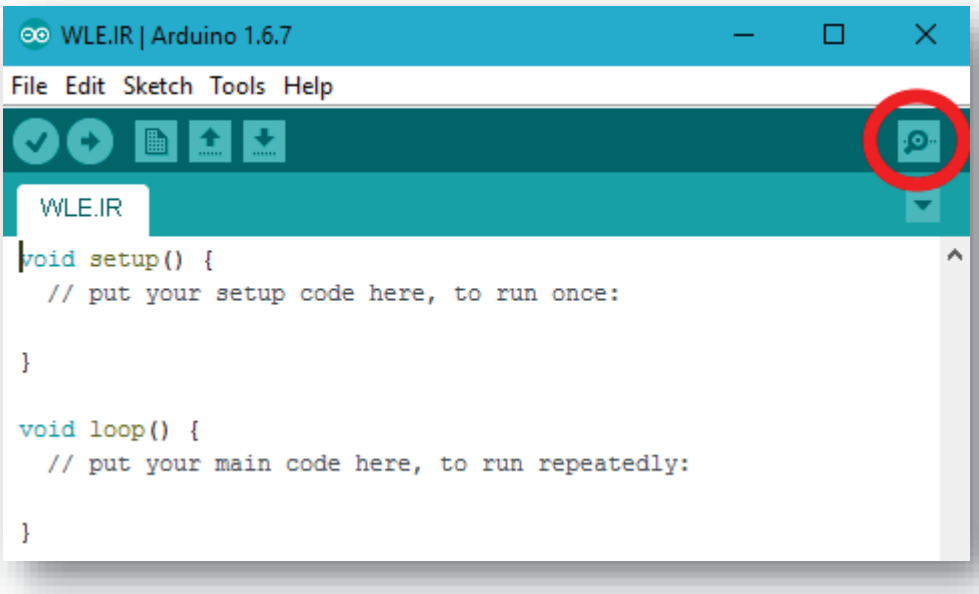
برای باز کردن سریال مانیتور می توانید به صورت های زیر عمل کنید

توجه : باید اردوینو به سیستم وصل باشد و پورت ایجاد شده انتخاب شده باشد

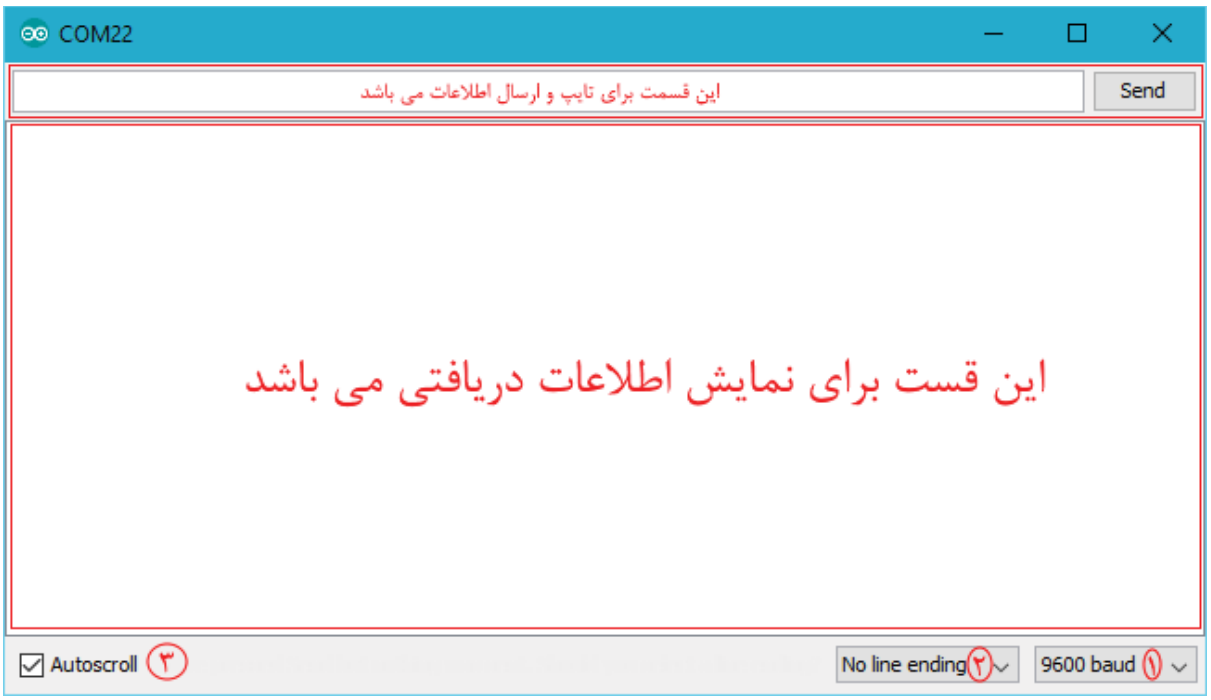
۱ – می توانید از کلید ترکیبی ctrl+shift+M استفاده کرد

۲ – از منوی Tools گزینه serial monitor انتخاب کرد

۳ – از داخل برنامه کلید میانبر زیر را فشار دهیم



با این کار منو سریال مانیتور به شکل زیر باز می شود



در تصویر بالا قسمت ۱ برای تنظیم نرخ ارسال داده و ۲ و ۳ برای سفارشی کردن محیط دریافت اطلاعات

❖ دستورات پایه ای در ارتباط سریال اردوینو

۱ – دستور شروع ارتباط سریال – begin()

کاربرد دستور : تعیین سرعت انتقال داده ها در ارتباط سریال (baud)

شکل کلی دستور :

----------دستور مشترک بین همه اردوینو ها :----------

```
Serial.begin(speed)
Serial.begin(speed, config)
```

----------دستورات مخصوص اردوینو های سری مگا :----------

```
Serial1.begin(speed)
Serial2.begin(speed)
Serial3.begin(speed)
Serial1.begin(speed, config)
Serial2.begin(speed, config)
Serial3.begin(speed, config)
```

پارامترهای دستور :

speed : سرعت انتقال داده ها (baud rate) می باشد ، می تواند یکی از مقادیر زیر باشد :

300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200

config : (اختیاری) در صورتی که لازم باشد می توانید با استفاده از این قسمت طول بیت داده (bit length) ، بیت توازن (parity bit) ، بیت پایان (Stop bit) را تنظیم کنیم ، در صورتی که این قسمت را استفاده نکنیم مقادیر پیش فرض طول 8 بیت داده ، بیت پایان 1 و بدون بیت توازن در نظر گرفته می شود

می توانید یکی از مقادیر زیر را انتخاب کنید :

- SERIAL_5N1
- SERIAL_6N1

- SERIAL_7N1
- SERIAL_8N1 (the default)
- SERIAL_5N2
- SERIAL_6N2
- SERIAL_7N2
- SERIAL_8N2
- SERIAL_5E1
- SERIAL_6E1
- SERIAL_7E1
- SERIAL_8E1
- SERIAL_5E2
- SERIAL_6E2
- SERIAL_7E2
- SERIAL_8E2
- SERIAL_5O1
- SERIAL_6O1
- SERIAL_7O1
- SERIAL_8O1
- SERIAL_5O2
- SERIAL_6O2
- SERIAL_7O2
- SERIAL_8O2

فرمت استفاده از دستور : باید این دستور را در داخل حلقه void setup بنویسیم با اجرای این دستور پایه های 0 و 1 اردوینو بعنوان درگاه سریال تنظیم شده و دیگر از آنها نمی توان بعنوان ورودی و خروجی استفاده کرد

به شکل زیر باید از دستور استفاده کنیم (البته می توانید از این دستور در هر جایی از بدنه برنامه استفاده کنید)

```
void setup() {

Serial.begin(9600); //مثال

}

void loop() {

}
```

در سری مگا نیز باید به شکل زیر تنظیم بشه

```
void setup(){

  Serial.begin(9600);     //این که بین همه اردوینو ها مشترک هست
  Serial1.begin(38400);   // تنظیم سرعت داده درگاه سریال اول
  Serial2.begin(19200);   //تنظیم سرعت داده درگاه دوم
  Serial3.begin(4800);    // تنظیم سرعت داده درگاه سوم
}

void loop() {


}
```

۲ - دستور پایان ارتباط سریال - end()

کاربرد دستور : همان طول که دیدید با استفاده از دستور قبل که بیان کردیم پایه 0 و 1 میکرو از کار می افتادند و دیگر نمی توانستیم بعنوان ورودی و خروجی از آن ها استفاده کنید ، با استفاده از این دستور می توانیم به این محدودیت پایان دهیم و مجددا این پایه ها را بعنوان ورودی و خروجی مورد استفاده قرار بدیم

شکل کلی دستور :

----------دستور مشترک بین همه اردوینو ها :----------

```
Serial.end()
```

----------دستورات مخصوص اردوینو های سری مگا :----------

```
Serial1.end()
Serial2.end()
Serial3.end()
```

پارامترهای دستور :

هیچ پارامتری وجود ندارد

مثال :

```
void setup() {
Serial.begin(9600);
}
void loop() {
برنامه مد نظر
Serial.end() //در این خط ارتباط سریال پایان می یابد
}
```

❖ دستورات ارسال دیتای سریال در آردوینو

این دستورات برای ارسال اطلاعات از طریق پورت سریال به کار می روند

۱ - ارسال اطلاعات پشت سر هم - print()

با استفاده از این دستور می توانیم اطلاعات را از طریق درگاه سریال ارسال کنیم ، با این دستور از این دستور اطلاعات از طریق پایه TX آردوینو فرستاده میشه ، می توانیم با این دستور اعداد ، کارکتر ، متن و به طور کلی کد استاندارد اسکی را بفرستیم ، اطلاعاتی که ارسال می شود پشت سر هم ارسال می شوند ، یعنی بدون فاصله پشت سر هم چاپ می شوند ، هم چنین در صورت نیاز می توانیم تعیین کنیم اعداد با چه فرمتی ارسال شوند مثلا بصورت دودویی (یعنی صفر و یک) ؛ دهدهی (همان اعداد خودمان) ، هگزا دسیمال و .... که در زیر توضیح می دهیم

کاربرد دستور : ارسال اطلاعات از طریق درگاه سریال (پشت سر هم)

شکل کلی دستور :

```
Serial.print(val)
Serial.print(val, format)
```

پارامترهای دستور :

val : اطلاعاتی که میخواهیم بفرستیم (هر نوع دیتایی مجاز است)

format : در صورتی که اعداد را ارسال کنیم ، می توانیم با این بخش مبنای عدد ارسالی را تغییر دهیم ، می توانیم تعیین کنیم عدد به **دهدهی** (DEC) ، **دودویی** (BIN) ، **اکتال** (OCT) ، **هگزا دسیمال** (HEX) تبدیل شود و سپس ارسال شود .(اعداد دهدهی همان اعدادی هستند که بصورت روزمره ازشون استفاده می کنیم ، اعداد دودویی همان صفر و یک هستند) **هچنین در صورتی که عدد اعشاری باشند می توانیم رقم های اعشاری را تعیین کنیم**

برای این بخش می تواند یکی از فرمت های HEX ، OCT ، BIN ، DEC و **یا یک عدد صحیح** (0، 1 ، 2 ، ...) **را برای تعیین عدد رقم اعشاری جایگزین کنیم**

به مثال زیر دقت کنید:

```
Serial.print(78, BIN)              نتیجه ----> "1001110"
Serial.print(78, OCT)              نتیجه ----> "116"
Serial.print(78, DEC)              نتیجه ----> "78"
Serial.print(78, HEX)              نتیجه ----> "4E"
Serial.println(1.23456, 0)         نتیجه ----> "1"
Serial.println(1.23456, 2)         نتیجه ----> "1.23"
Serial.println(1.23456, 4)         نتیجه ----> "1.2346"
```

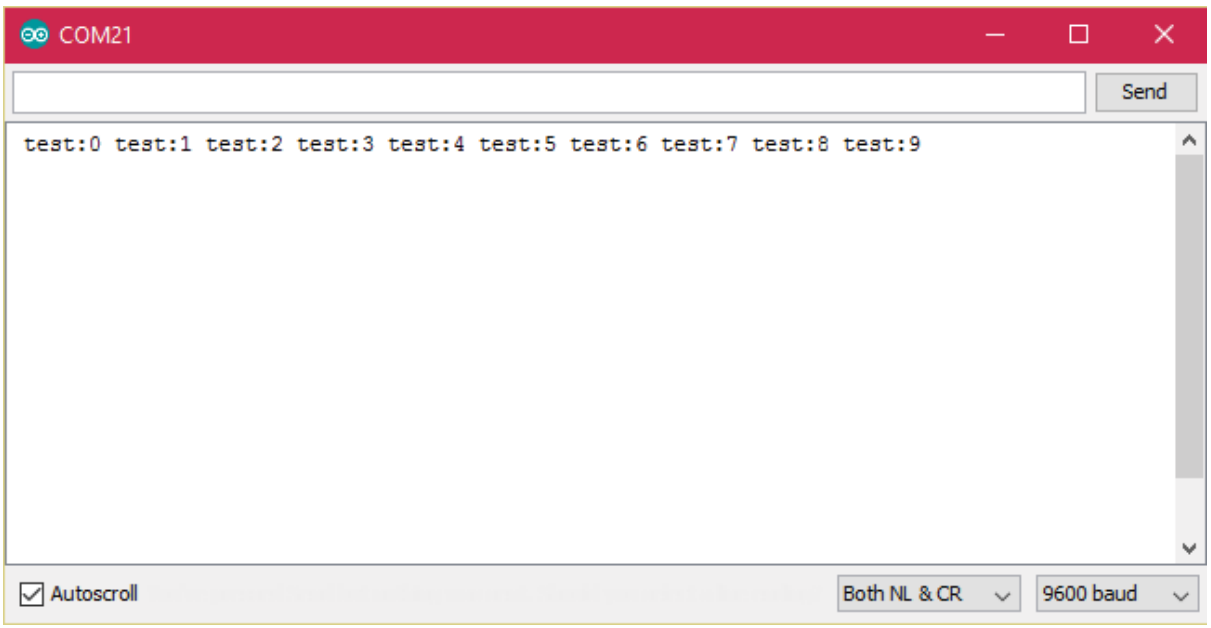
مثال ۱ : عبارت test و شمارنده افزایشی را از طریق درگاه سریال ارسال کنید بصورتی که خروجی به شکل test:1 , test:2 , test:3 , test:4 و .... چاپ شوند

```
int x = 0 ;                          یک متغییر عددی تعریف می کنیم
void setup() {
  Serial.begin(9600);    سرعت ارسال اطلاعات را انتخاب می کنیم
}
void loop() {
 Serial.print(" test:");                    یک رشته می فرستیم
 Serial.print(x);              یک عدد را داخل متغییر می فرستیم
 delay(1000);                          یک ثانیه تاخیر می کنیم
x++;          متغییر را از مقدار اولیه یک واحد افزایش می دهیم
}
```

توجه : برای دیدن نتیجه، بعداز اینکه برنامه را روی اردوینو آپلود کردید ، سریال مانیتور (serial monitor) اردوینو را باز کنید و نتیجه را مشاهده کنید

این برنامه هر ثانیه عبارت test:x را پشت سر هم با x افزایشی ارسال می کند

نکته : اگر دقت کرده باشید در این برنامه مقادیر ارسالی پشت سر هم چاپ می شوند به شکل زیر



این فاصله ای هم که در بالا می بینید بخاطر کارکتر خالیه که قبل کلمه به شکل "test:x" ایجاد کردیم برای ایجاد فاصله بین مقادیر ارسالی می توانیم از دستور زیر استفاده کنیم

```
Serial.print("\t")
```

این دستور را بعداز سایر دستورات قرار دهید در مثال بالا به شکل زیر می شود

```
Serial.print("test:");
Serial.print(x);
Serial.print("\t")
```

همچنین اگر بخواهیم فاصله زیاد باشد می توانیم دو بار (و یا بیشتر) عبارت \t را بنویسیم به شکل زیر

```
Serial.print("test:");
Serial.print(x);
Serial.print("\t\t")
```

حالا مثال اخری را تست می کنیم نتیجه به شکل زیر خواهد بود

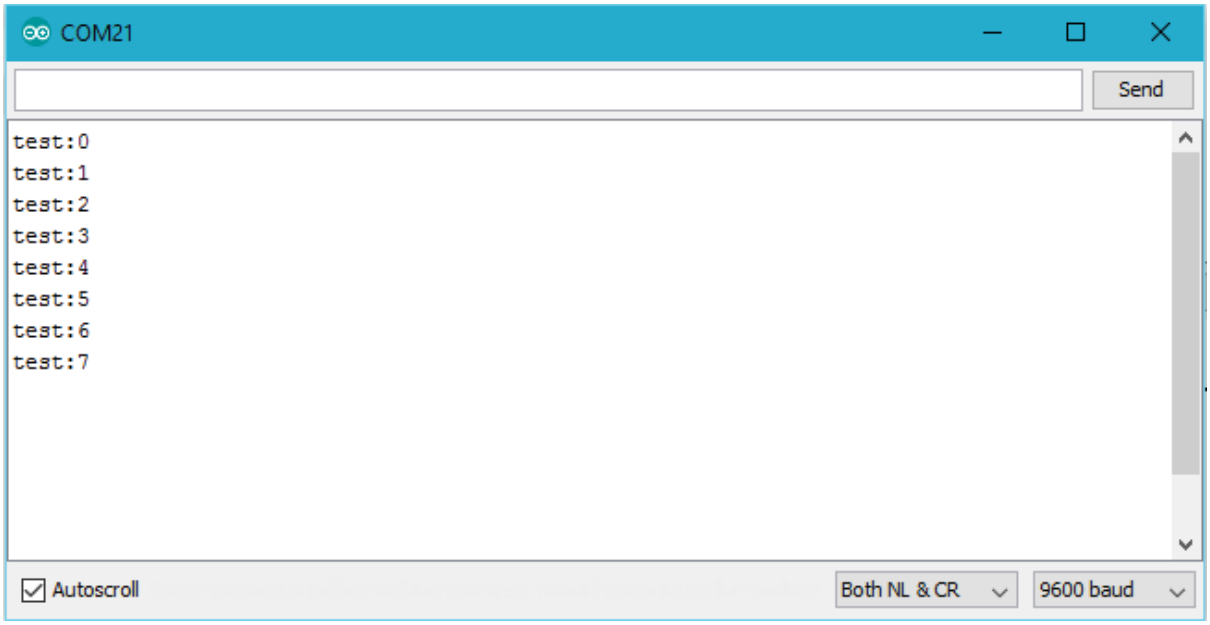
همچنین در صورتی که بخواهید عبارات پشت سر هم نوشته نشوند بلکه هر دیتا که ارسال می شود در خطی جدید نوشته شود می توانید از دستور زیر کمک بگیرید

```
Serial.print("\n");
```

با نوشتن این دستور بعداز سایر دستورات ، هر بار که دیتا ارسال می شود در یک خط جدید به نمایش در می آید برای مثال :

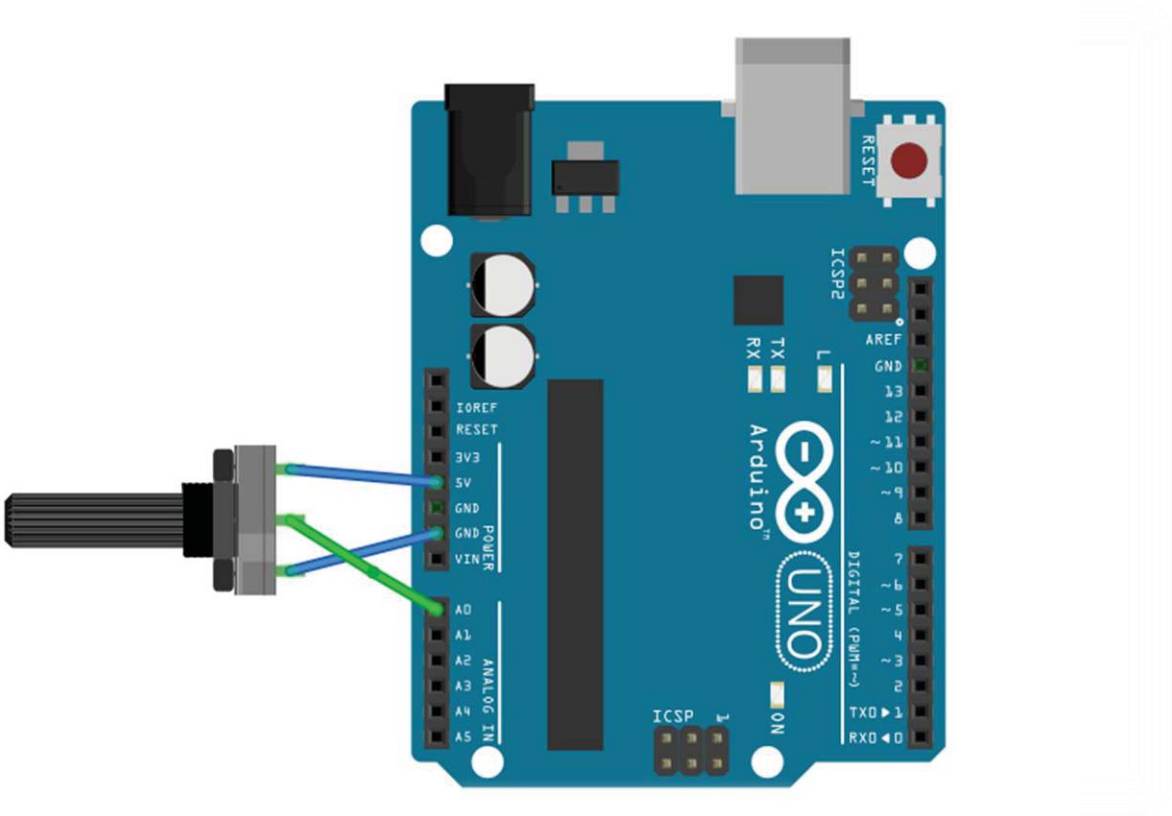
```
int x = 0 ;
void setup() {
  Serial.begin(9600);
}
void loop() {
 Serial.print("test:");
 Serial.print(x);
  Serial.print("\n");
 delay(1000);
x++;
}
```

این همان مثال قبلی است که تنها `Serial.print("\n");` را جایگزین کرده ایم ، نتیجه مثال بالا به شکل زیر خواهد بود



مثال 2 : **برنامه ای بنوسید که اطلاعات را هر نیم ثانیه از مبدل انالوگ صفر بگیرد و از طریق درگاه سریال ارسال کند**

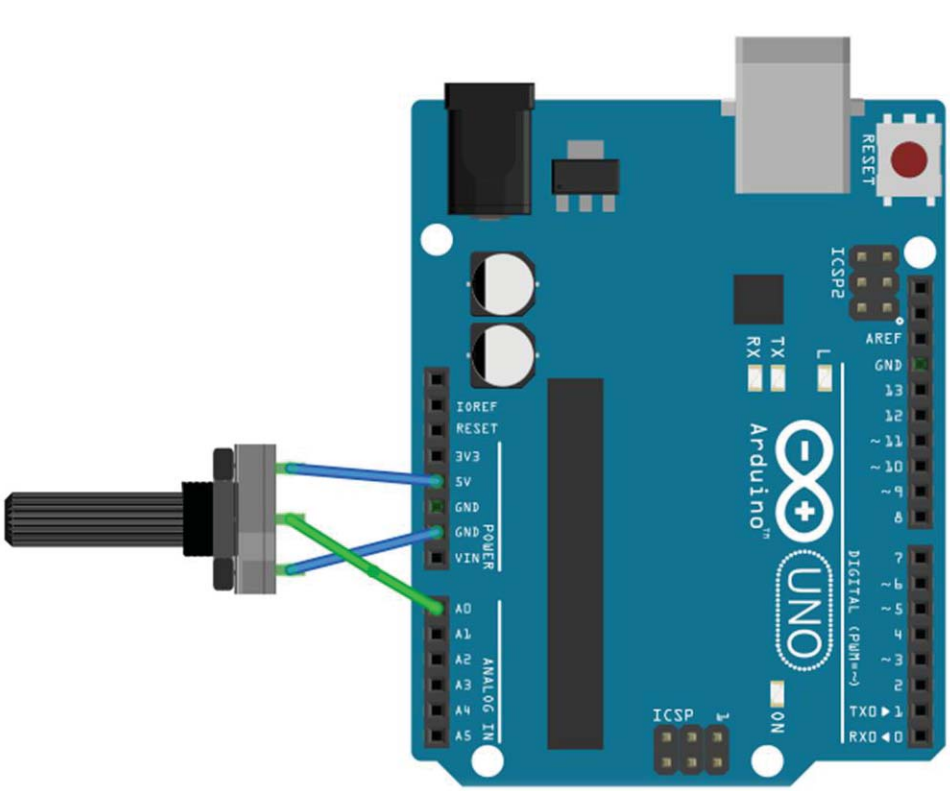
مدار : پتانسیومتر ۱۰ کیلو اهم را به شکل زیر به اردوینو وصل کنید



برنامه آردوینو:

```
int x = 0 ;                                          تعریف متغییر
void setup() {
Serial.begin(9600);                                  تعیین سرعت ارسال اطلاعات
}
void loop() {
 x = analogRead(0);                                  خواندن اطلاعات از پایه انالوگ صفر
 Serial.print("analogRead:");                        ارسال رشته به درگاه سریال
 Serial.print(x);                                    ارسال متغییر به درگاه سریال
 Serial.print("\n");                                 رفتن به خط جدید
 delay(500);                                         تاخییر نیم ثانیه ای
```

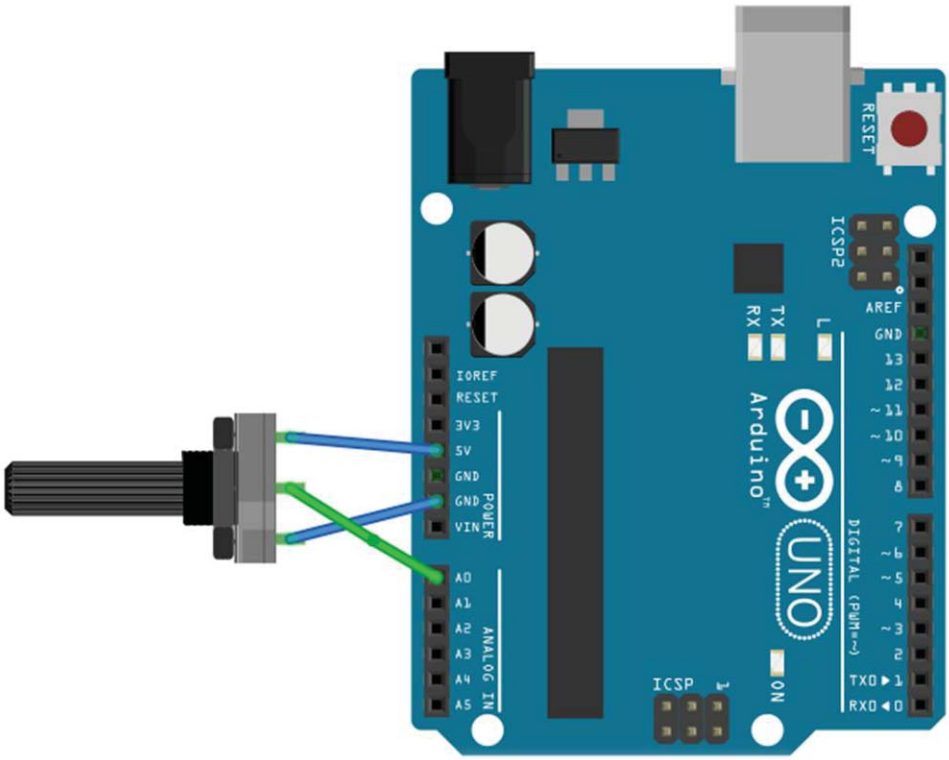
مثال ۳ : برنامه ای بنوسید که اطلاعات را هر نیم ثانیه از مبدل انالوگ صفر بگیرد و سپس به اعداد باینری (صفر و یک) تبدیل کند و از طریق درگاه سریال ارسال کند



برنامه آردوینو

```
int x = 0 ;
void setup() {
Serial.begin(9600);

}

void loop() {

 x = analogRead(0);

 Serial.print("analogRead:");

 Serial.print(x , BIN);

 Serial.print("\n");

 delay(500);

}
```

مثال ۴ : برنامه ای بنوسید که اطلاعات را هر نیم ثانیه از مبدل انالوگ صفر بگیرد و اعداد را با چهار رقم اعشار قطع کند ، سپس از طریق درگاه سریال ارسال کند




```
float x = 0 ;                                   تعیین متغییر از نوع اعشاری
void setup() {
Serial.begin(9600);                             تعیین سرعت ارسال اطلاعات
}
void loop() {
 x = analogRead(0);                             خواندن مقادیر انالوگ و ریختن در متغییر ایکس
 x = x/0.3;          چون مقادیر خوانده شده رند هستند ما تقسم بر یک عدد اعشاری کردیم تا مقادیر کلی اعشاری شوند
 Serial.print("analogRead:");                   متن را ارسال می کنیم
 Serial.print(x , 4);                           متغییر ایکس را تا چهار رقم اعشار ادامه می دهیم
 Serial.print("\n");                           به خط جدید می رویم
 delay(500);                                    تاخیر نیم ثانیه ای
}
```

توضیح : متغییر از نوع `float` عدد را تا ۲ رقم اعشاری ادامه می دهد ما این رقم را به ۴ رقم افزایش دادیم

۲ - ارسال اطلاعات در خط جدید println()

دستور println() شبیه دستور print() است و برای ارسال هر نوع اطلاعاتی می باشد با این تفاوت که دستور print() اطلاعات را پشت سر هم ارسال می کرد ولی دستور println() دستورات را در خط جدید ارسال می کند ، تمامی توضیحاتی که در دستور قبلی ارائه شد در مورد این دستور نیز صادق است . دستور println() در واقع ترکیب دستور print() با print("\n") است که برای مختصر نویسی می توان از آن استفاده کرد

کاربرد دستور : ارسال اطلاعات از طریق درگاه سریال (در خط جدید)

شکل کلی دستور :

```
Serial.println(val)
Serial.println(val, format)
```

پارامترهای دستور :

val : اطلاعاتی که میخواهیم بفرستیم (هر نوع دیتایی مجاز است)

format : در صورتی که اعداد را ارسال کنیم ، می توانیم با این بخش مبنای عدد ارسالی را تغییر دهیم ، می توانیم تعیین کنیم عدد به دهدهی (DEC) ، دودویی (BIN) ، اکتال (OCT) ، هگزا دسیمال (HEX) تبدیل شود و سپس ارسال شود .(اعداد دهدهی همان اعدادی هستند که بصورت روزمره ازشون استفاده می کنیم ، اعداد دودویی همان صفر و یک هستند) هچنین در صورتی که عدد اعشاری باشند می توانیم رقم های اعشاری را تعیین کنیم

برای این بخش می تواند یکی از فرمت های HEX ، OCT ، BIN ، DEC و یا یک عدد صحیح (0، 1 ، 2 ، ...) را برای تعیین عدد رقم اعشاری جایگزین کنیم

همه مثال های قبلی را می توانید با این دستور جایگزین و تست کنید ما برای جلو گیری از اتلاف وقت فقط یک مثال می آوریم تا با عملکرد کلی آن آشنا شوید

مثال ۵ : عبارت MOHAJER را از طریق درگاه سریال ارسال کنید بصورتی که خروجی ها در خط جدید به نمایش در آیند

```
void setup() {
  Serial.begin(9600);
}
void loop() {
Serial.println("Mohajer")
; delay(500);
}
```

۳ - ارسال دیتا - write()

برای ارسال دیتا در فرم های مختلف به کار می رود ، در زیر سه تابع را می بینید که در تابع 1 یک عدد دسیمال)DEC(بین 0 تا 255 را به تابع می دهیم و تابع علامت اسکی عدد را بر میگرداند ، بعنوان مثال اگر عدد 65 را به آن بدهیم حرف A را باز می گرداند و یا اگر عدد 194 را به آن بدهیم علامت آ فارسی را برمیگرداند. در تابع 2 یک رشته متن ارسال می شه و در تابع سوم یک ارایه با طول مشکل ارسال خواهد شد

کاربرد دستور : ارسال اطلاعات در فرم های مختلف

شکل کلی دستور :

----------مشترک بین همه اردوینو ها----------

```
Serial.write(val)                    1
Serial.write(str)                    2
Serial.write(buf, len)               3
```

----------مخصوص سری مگا و arm ----------

```
Serial1.write(val)
Serial1.write(str)
Serial1.write(buf, len)
Serial2.write(val)
Serial2.write(str)
```

```
Serial2.write(buf, len)
Serial3.write(val)
Serial3.write(str)
Serial3.write(buf, len)
```

پارامترهای دستور :

val : یک عدد بین 0 تا 255 (علامت اسکی آن برگردانده می شود)

str : می توان یک رشته (متن) را بفرستیم

buf: آرایه ای برای ارسال اطلاعات

len: طول ارایه ای که قرار است اطلاعات را درون آن بفرستیم

مثال ۶ (برای تابع اول) : برنامه ای بنویسید که حروف و علامت فارسی و انگلیسی را ارسال کند

```
int a = 32;                          تعریف متغییر با مقدار اولیه ۳۲
void setup(){
  Serial.begin(9600);                تعیین سرعت ارسال دیتا
}
void loop(){
Serial.print("val(");                نوشتن عبارت مد نظر و باز گذاشتن پرانتز
Serial.print(a);                     نمایش متغییر مورد نظر
Serial.print("): ");                 بستن پرانتز
Serial.write(a);                     برگرداندن علامت اسکی اعداد
Serial.print("\n");                  رفتن به خط جدید
delay(500);                          توقف نیم ثانیه ای
a++;                                 افزایش متغییر به مقددار یک واحد
}
```

در مثال بالا یک شمارنده ایجاد کرده ایم که از عدد ۳۲ که اولین عدد دسیمال استاندارد اسکی (یعنی اسپیس) شروع به شمارش می کنیم و هم عدد شمارش شده و هم علامت اسکی را با استفاده از تابع برمیگردانیم و سپس ارسال می کنیم

مثال ۷ (برای تابع دوم) یک برنامه بنویسید که عبارت MOHAJER را با استفاده از پورت سریال ارسال کند

```
void setup(){
  Serial.begin(9600);
}
void loop(){
Serial.write("MOHAJER")
; Serial.write("\n");
delay(500);
}
```

این تابع شبیه به print () عمل می کند و متن (رشته) را از طریق پورت سریال ارسال می کند

مثال ۸ (برای تابع سوم) : یک متغییر آرایه ای با 6 عضو بنویسید و سپس اعضای ان را کنار هم چیده و سپس نمایش دهید

```
  char buf[6]={'M','O','H','A','J','R'};        یک ارایه ۶ عضوی تعریف می کنیم با اعضای مشخص

 void setup()
{
  Serial.begin(9600);                          تعیین سرعت ارسال اطلاعات
}
void loop(){
Serial.write(buf,6);                           باز خوانی ارایه از خانه صفر تا پنج و نمایش پشت سر هم
Serial.write("\n");                            رفتن به خط جدید
delay(500);                                    یک ایست موقت نیم ثانیه ای

  }
```

۴ - خالی کردن بافر - flush()

خب اول ببینیم بافر چیه ، شما در میکرو بافر رو به این شکل در نظر بگیرید تا براتون قابل درک باشه ، یک حافظه موقت است که اطلاعات ارسالی و دریافتی قبلا از استفاده در آن ذخیره می شوند.

حالا وقتی با استفاده از دستورات ارسال println( )و print() و یا هر دستور دیگری اطلاعات را ارسال کردیم ، اطلاعات وارد بافر سریال میشه و به محض اینکه طرف مقابل اطلاعات را دریافت کرد بافر شروع به خالی شدن می کند ، دستور flush()کارش این هست بافر را بصورت کلی خالی می کند تا همه اطلاعات به درستی ارسال شوند .

این دستور بعداز دستورات println()و print() و قرار میگیرد

کاربرد دستور : خالی کردن بافر و تکمیل ارسال اطلاعات

شکل کلی دستور :

----------مشترک بین همه اردوینو ها----------

```
Serial.flush()
```

---------- مخصوص سری مگا و arm ----------

```
Serial1.flush()
Serial2.flush()
Serial3.flush()
```

پارامترهای دستور :

هیچ پارامتری ندارد

مثال ۹ : در مثال زیر بعداز ارسال اطلاعات با دستور flush() بافر را خالی می کنیم

```
void setup() {
  Serial.begin(9600);
}
void loop() {
Serial.println("Mohajer");
Serial.flush();
delay(500);

}
```