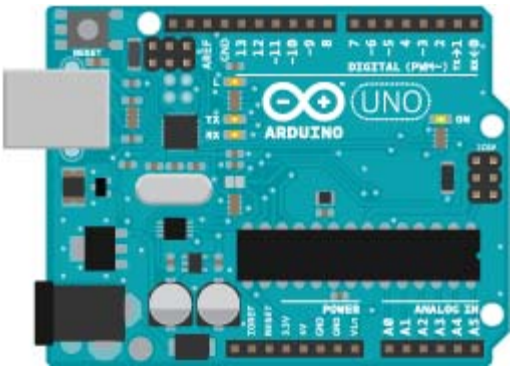
برنامه نویسی سخت افزاری



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

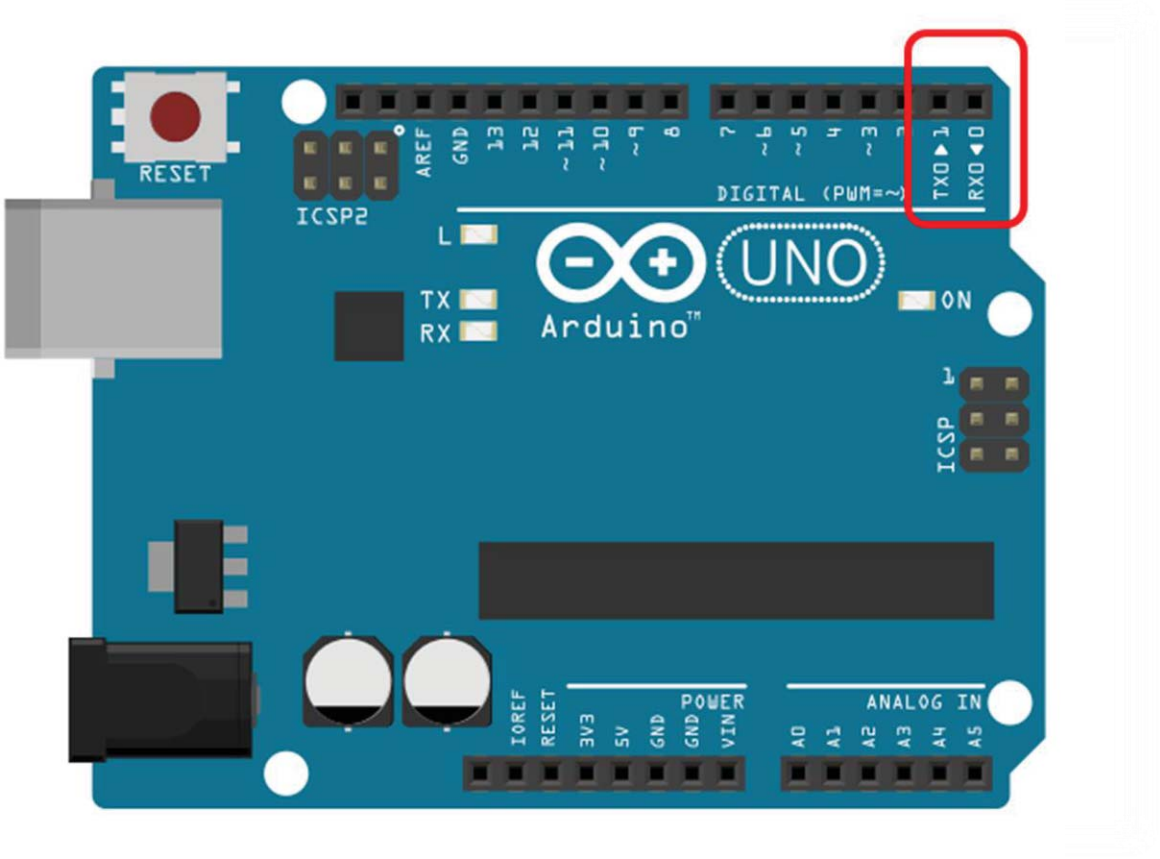
۳ –بیت های آغاز و پایان(Start - Stop bit) : بیت آغاز همیشه یک بیته و اونم همیشه صفره ولی بیت پایان همیشه یکه و میتونه از یک یا دو بیت تشکیل بشه. از دو بیت واسه سیستمهای قدیمی که کند بودن استفاده میشد که تا اومدن بایت جدید زمان داشته باشه خودشو جمع و جور کنه که الانه همه از یه بیت پایان استفاده میکنن.

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

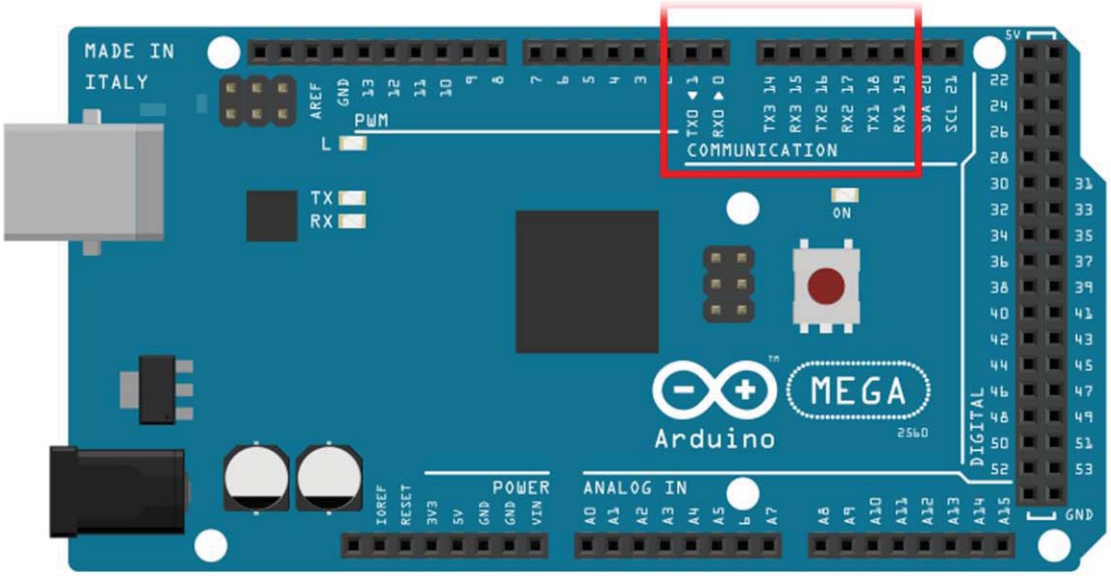
آردوینو دارای دو نوع ارتباط سریال است یکی ارتباط سریال سخت افزاری و یکی هم ارتباط سریال نرم افزاری که فعلا به ارتباط سریال سخت افزاری می پردازیم ، در اردوینو هایی که با atmega328 ، atmega32u4 و میکرو های مشابه طراحی شده اند مانند Arduino nano ، Arduino uno و ... دارای یک پل ارتباطی سریال (یک پایه برای 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 :

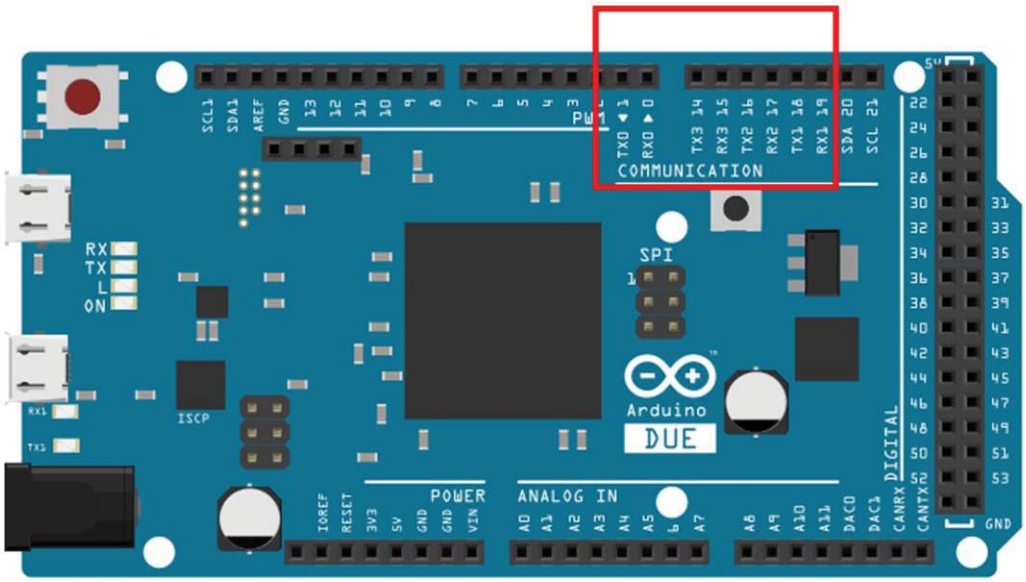
پایه 17 بعنوان Rx و پایه 16 اردوینو بعنوان Tx عمل می کند

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

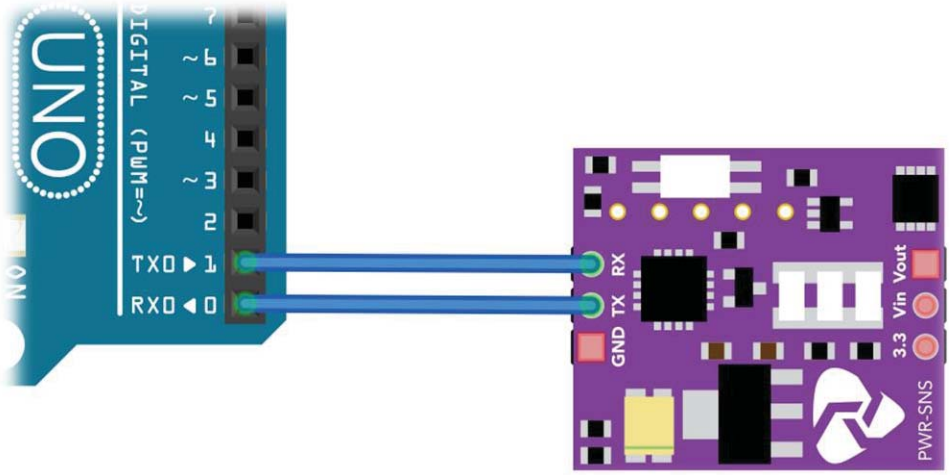
همچنین در آردوینو های سری ARM مانند ARDUINO DUE نیز مشخصات درگاه سریال دقیقا مانند اردوینو سری مگا می باشد



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

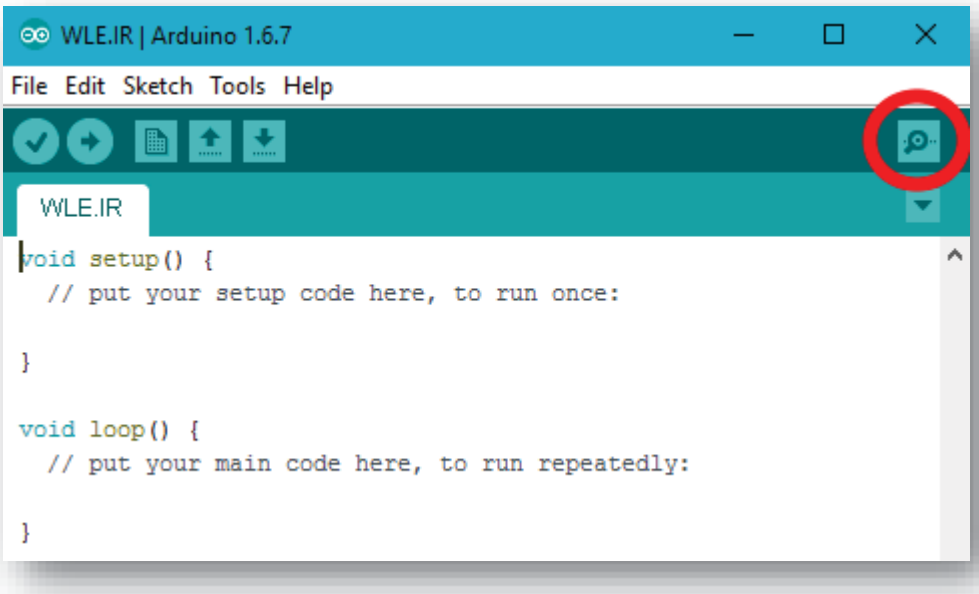
برای باز کردن سریال مانیتور می توانید به صورت های زیر عمل کنید

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

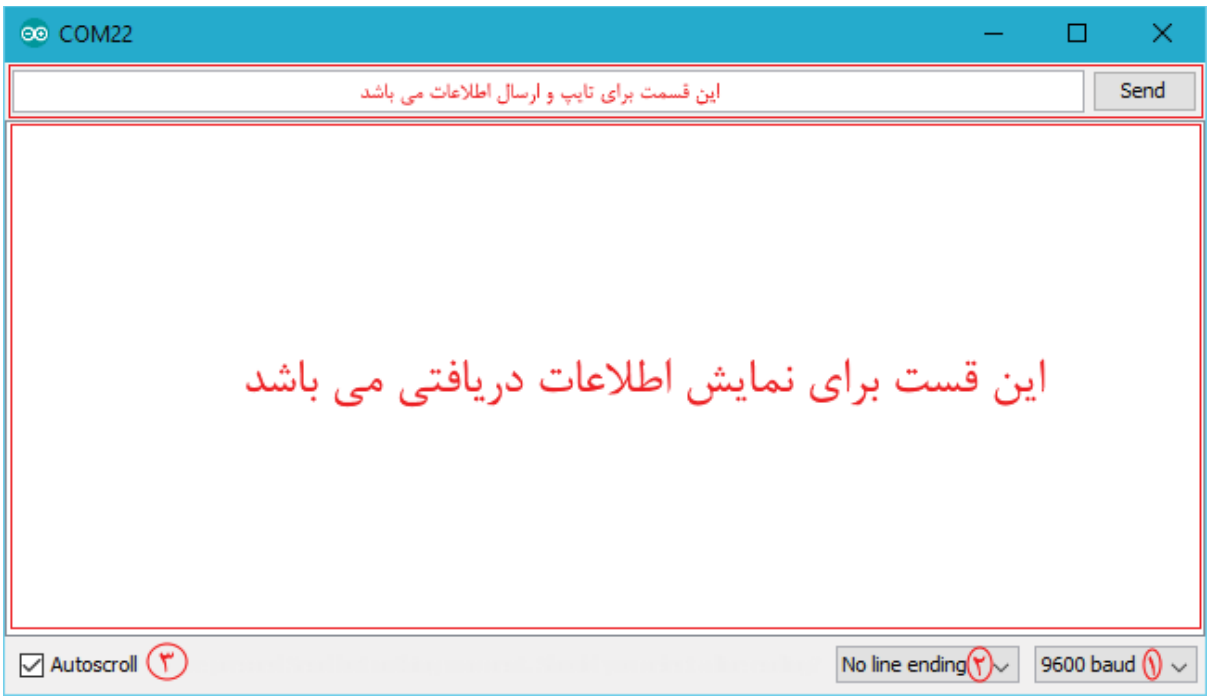
۱ – می توانید از کلید ترکیبی ctrl+shift+M استفاده کرد

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"
```

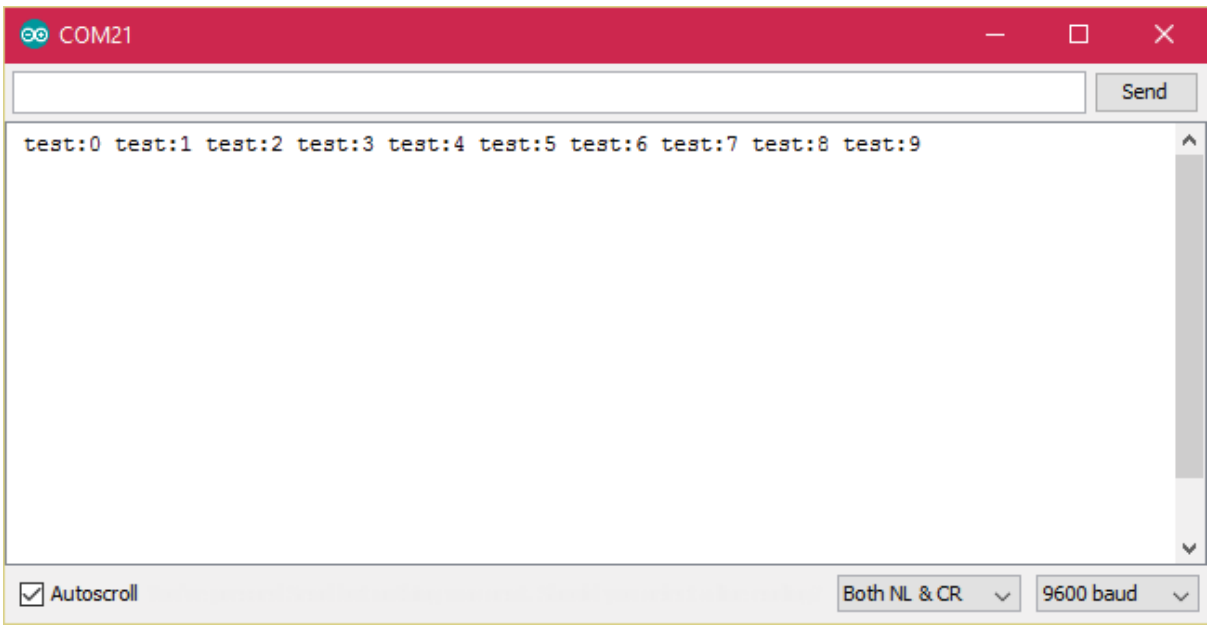
مثال ۱ : عبارت test و شمارنده افزایشی را از طریق درگاه سریال ارسال کنید بصورتی که خروجی به شکل test:1 , test:2 , test:3 , test:4 و .... چاپ شوند

```
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

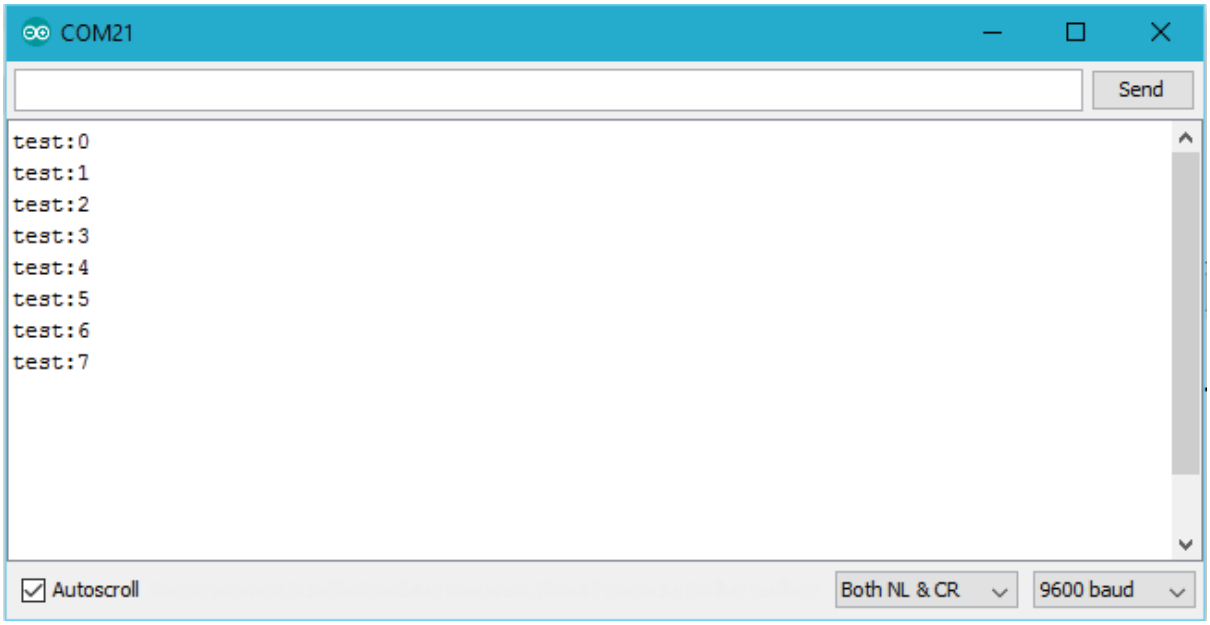
همچنین در صورتی که بخواهید عبارات پشت سر هم نوشته نشوند بلکه هر دیتا که ارسال می شود در خطی جدید نوشته شود می توانید از دستور زیر کمک بگیرید

```
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 :

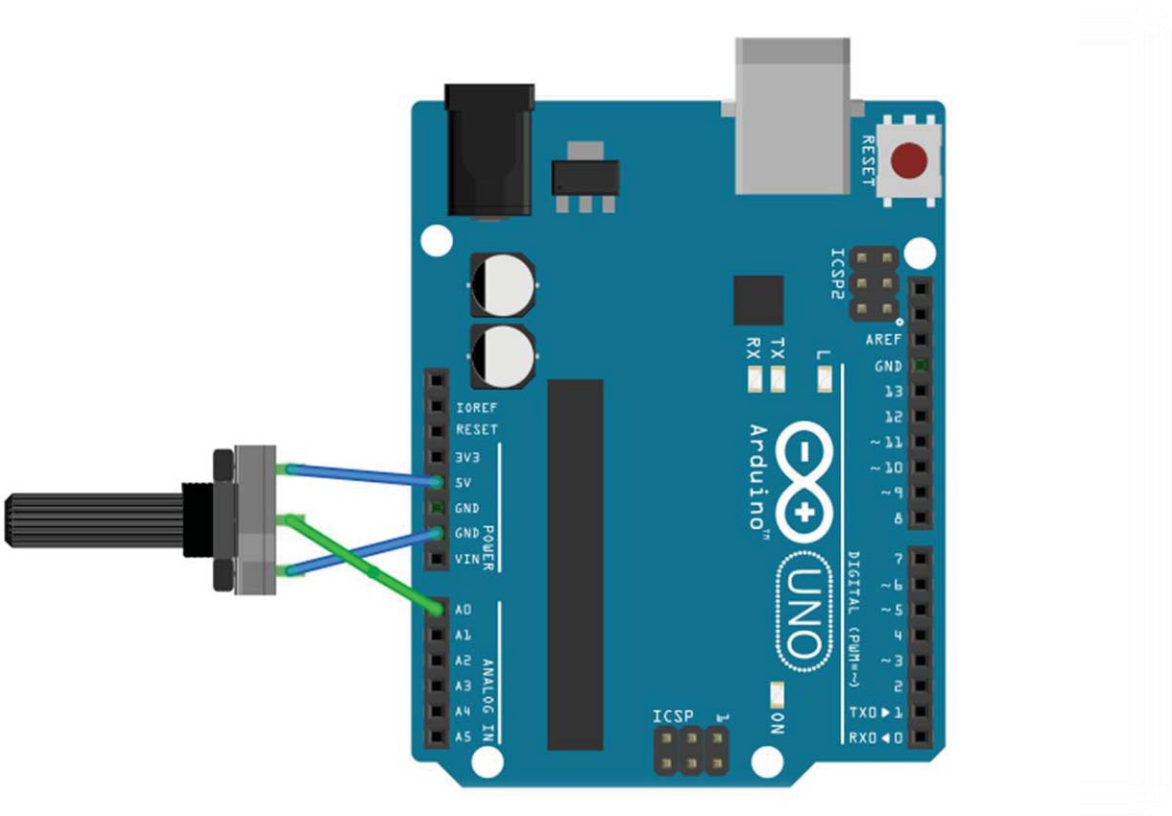
```
int x = 0 ;
void setup() {
  Serial.begin(9600);
}
void loop() {
 Serial.print("test:");
 Serial.print(x);
  Serial.print("\n");
 delay(1000);
x++;
}
```

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**

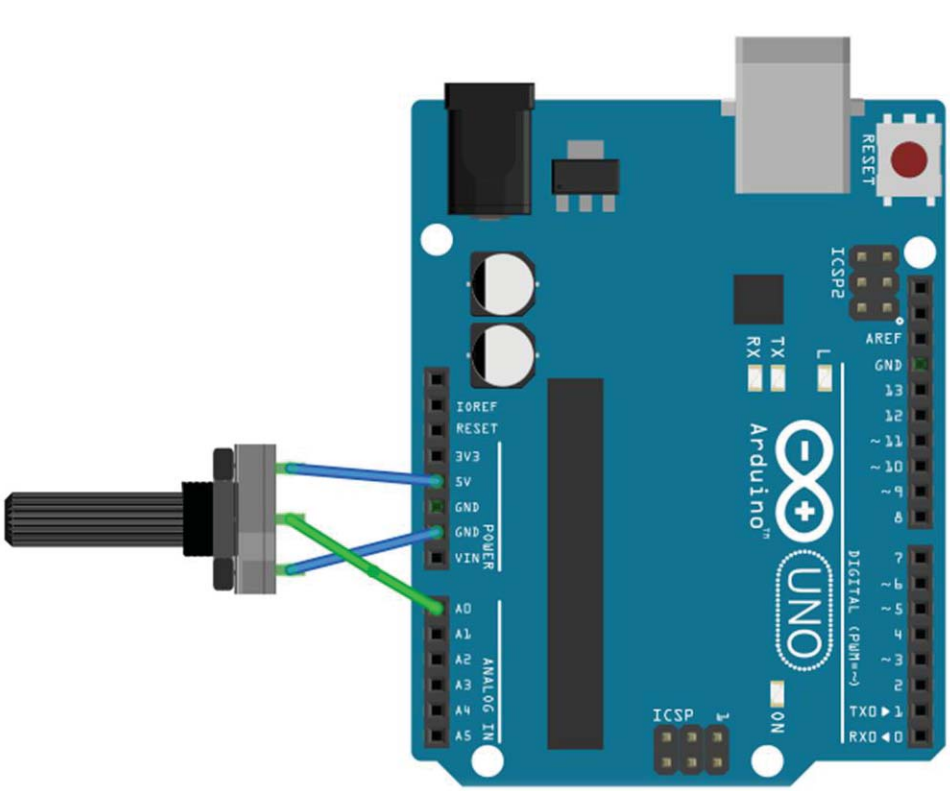
مدار : پتانسیومتر ۱۰ کیلو اهم را به شکل زیر به اردوینو وصل کنید



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
```

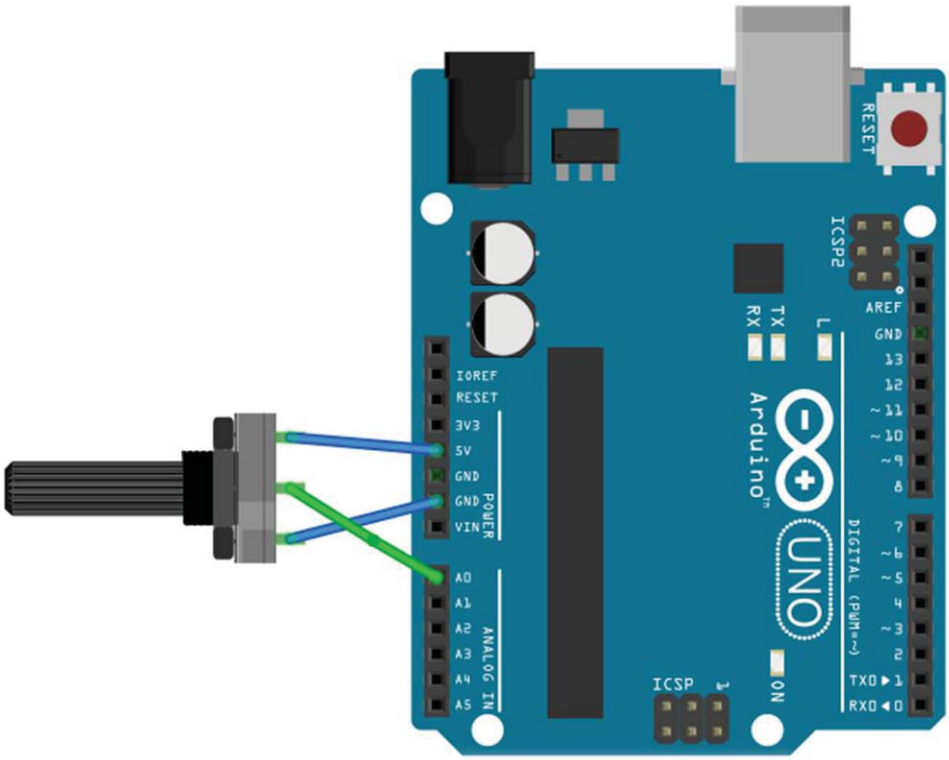
مثال ۳ : برنامه ای بنوسید که اطلاعات را هر نیم ثانیه از مبدل انالوگ صفر بگیرد و سپس به اعداد باینری (صفر و یک) تبدیل کند و از طریق درگاه سریال ارسال کند



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 محظ 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);

}
```

❖ دستورات دریافت دیتای سریال در آردوینو

دستورات دریافت نسبت به دستورات ارسال هم زیاد تر هستند و هم اینکه کمی مهم تر هستند ، تقریبا اکثر کار هایی که باید رو دیتا انجام بدیم ، روی دیتای ورودی هست چون برای ارسال دیتا محدودیت کمتر داریم ، هر فرمو و شکلی که بخواهیم به راحتی دیتا را می فرستیم ولی دیتای دریافتی معمولا در فرمت های خاصی ارائه می شه که ما باید با دستورات مختلف آن را تجزیه و اطلاعات ضروری را جدا کنیم ، بعنوان مثال دیتای سریالی که مثلا یک ماژول gps میفرسته به شکل xl586#yl44R55 است ما مجبوریم بخش هایی از این رو جدا کنیم و بصورت جدا از آن استفاده کنیم نمی توانیم این فرمت ارسالی رو عوض کنیم ، پس سعی کنید این مطلب رو خوب یاد بگیرید ، سعی می کنیم دستورت را با مثال های ملموس برای شما اموزش دهیم

۱- دریافت دیتای سریال - read()

با استفاده از این دستور می توانیم اطلاعات را از پایه سریال بخوانیم ، اطلاعات فرستاده شده برای پایه سریال توسط این دستور بصورت بایت بایت (تکه تکه) دریافت می شود مثلا اگر رشته WLE را از نوع char را برای پایه سریال بفرستیم این دستور اول W را دیافت می کند بعد L و در انتها E را دریافت می کند یا اگر عدد 123 را از نوع char بفرستیم ، ابتدا 1 سپس 2 و در انتها 3 دریافت می شود همچنین اگر هیچ گونه اطلاعاتی دریافت نشود خود تابع برای متغییر char عبارت (ے) را بر میگرداند ، همچنین اگر دیتا از نوع Int را بفرستیم تابع مقدار دسیمال را بر میگرداند ، بعنوان مثال اگر عدد 0 را بفرستیم تابع 48 را بر میگرداند و یا اگر a را بفرستیم تابع عدد دسیمال آن یعنی97 را بر میگرداند و اگر هیچ دیتایی دریافت نشود خود تابع برای متغییر int عدد (-1) را بر میگرداند .

دیتا از نوع long ، short ، float و double دقیقا مثال Int عمل می کنند یعنی دیتای که از این نوع به صورت کد دسیمال تبدیل می شود و اگر دیتا دریافت نشود مقدار (-1) برگشت داده می شود ، همچنین دیتا از نوع word و byte نیز مقدار دسیمال اطلاعات دریافتی را باز می گرداند با این تفاوت که اگر دیتا دریافت نشود متغییر byte مقدار (255) را باز می گرداند و متغییر word مقدار (65535) را باز می گرداند

کاربرد دستور : دریافت اطلاعات سریال بصورت بایت به بایت

شکل کلی دستور :

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

```
Serial.read()
```

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

```
Serial1.read()
Serial2.read()
Serial3.read()
```

پارامترهای دستور :

هیچ پارامتری ندارد

مثال ۱۰ : یک برنامه بنویسید که به آن اعداد و عبارت دهیم و برنامه اعداد و عبارت را بصورت جدا جدا (اجزای سازنده را بشکند) برای ما از طریق سریال مانیتور باز گرداند

```
char DATA;
                                متغییری به اسم data را از نوع char تعریف کرده ایم
void setup()
{
  Serial.begin(9600);              سرعت ارسال داده را مشخص کرده ایم
}
void loop()
{
  DATA = Serial.read();        مقدار خوانده شده از پورت سریال را در متغییر دیتا می ریزیم
  Serial.print("data: ");        ارسال عبارت دلخواه
  Serial.print(DATA);
برای این که مقدار دیتای خوانده شده را ببینیم مجددا آن را برای پورت سریال ارسال می کنیم تا از طریق سریال مانیتور به نمایش در بیاید
 Serial.print("\n");    رفتن به خط جدید
 delay(500);        یک تاخیر نیم ثانیه ای ایجاد می کنیم تا فرصت دیدن مقادیر را داشته باشیم
}
```

وقتی کد ها را روی اردوینو اپلود کردید سریال مانیتور را باز کنید ، می بینید که علامت ے برگشت داده می شوند عبارتی را تایپ کنید و نتیجه را بررسی کنید ، می بینید که اگر شته ای بفرستید ، آن رشته به اجزای سازنده اش تبدیل می شود و هر جزء جدا جدا باز گردانده می شوند

مثال ۱۱ : یک برنامه بنویسید که به آن اعداد و عبارت دهیم و برنامه مقدار دسیمال آن را برای ما از طریق سریال مانیتور باز گرداند

```
int DATA;
                                    متغییری به اسم data را از نوع int تعریف کرده ایم
void setup()
{
  Serial.begin(9600);              سرعت ارسال داده را مشخص کرده ایم
}
void loop()
{
  DATA = Serial.read();        مقدار خوانده شده از پورت سریال را در متغییر دیتا می ریزیم
  Serial.print("data: ");        ارسال عبارت دلخواه
  Serial.print(DATA);
برای این که مقدار دیتای خوانده شده را ببینیم مجددا آن را برای پورت سریال ارسال می کنیم تا از طریق سریال مانیتور به نمایش در بیاید
 Serial.print("\n");    رفتن به خط جدید
 delay(500);        یک تاخیر نیم ثانیه ای ایجاد می کنیم تا فرصت دیدن مقادیر را داشته باشیم
}
```

وقتی کد ها را روی اردوینو اپلود کردید سریال مانیتور را باز کنید ، می بینید که -1 برگشت داده می شوند عبارتی را تایپ کنید و نتیجه را بررسی کنید ، می بینید که اگر شته ای بفرستید ، آن رشته به اجزای سازنده اش تبدیل می شود و سپس اجزا به کد های دسیمال تبدیل و هر جزء (که حالا دسیمال هستند) جدا جدا باز گردانده می شوند

توجه : در مثال های بالا می بینید که با باز کردن سریال مانیتور همیشه دیتا ارسال می شود ، یا عبارتی که ما میخواهیم باز گردانده می شود و یا مقدار تابع در صورتی که اطلاعاتی دریافت نشود ، این باز گرداندن اطلاعات همیشه صورت می گیرد حتی زمانی که ما به آن نیاز نداریم ، در صورتی که برنامه ای بنویسیم که با این دستور کار کند ، CPU اردوینو مدام درگیر دریافت اطلاعات ما و یا اطلاعات پیش فرض تابع (همان 1- و ...) می شود و پایه RX میکرو همیشه درگیر خواهد بود ، این کار باعث می شود که آردوینو قفل کند و یا بسیار کند شود ، برای حل این مشکل باید چیکار کنیم ؟؟ باید دستور زیر را همراه با دستوراتی که دیتا دریافت می کنند به کار ببریم

۲- انجام کار فقط زمانی که دیتا دریافت شود - available()

با استفاده از این دستور می توانیم تعداد بایت (کارکتر) هایی را که برای خواندن آماده هستند را بدست آوریم ، سپس می توانیم با یک دستور شرطی ساده کاری کنیم که اگر تعداد بایت ها به تعداد مشخصی رسید آن گاه اطلاعات را مورد استفاده قرار دهیم ، اطلاعاتی که به وسیله درگاه سریال دریافت می شوند در حافظه بافر سریال ذخیره می شوند ، بافر سریال تا 64 بایت اطلاعات را می تواند ذخیره کند (یعنی یک عدد ۱۶ رقمی) ، دستور available() تعداد بیت های اطلاعات موجود در بافر را می شمارد . اگر بیت ها را خواندیم (بادستور قبلی و یا دستوراتی که خواهیم آموخت) بافر سریال خالی می شود تعداد بیت ها صفر می شوند

ولی اگر درگاه سریال اطلاعات را دریافت کند و ما آن اطلاعات را نخوانیم ، اطلاعات در بافر باقی می ماند و اگر اطلاعات جدید بیاید با اطلاعات قبلی جمع می شود . و تعداد بیت کل اطلاعات موجود باز هم بیشتر می شود .

برای درک کامل این یک مثال ملموس تر می زنیم :

هر عدد و یا هر حرف انگیسی به صورت تکی (به عنوان مثال) 3 بیت فضا را اشغال می کند حالا اگر حرف a را برای درگاه سریال بفرستیم ، عدد a در بافر سریال ذخیره می شود که این حرف ، 3 بیت (از 64 بایت) بافر را اشغال می کند ، حالا مجددا اگر حرف b را بفرستیم ، این حرف هم در بافر سریال ذخیره می شود و این حرف هم 3 بیت دیگر را اشغال می کند حالا در کل دو حرف a و b در بافر سریال ذخیره شده اند و با هم 6 بیت را از بافر اشغال کرده اند (۳ بیت a و 3 بیت b)

در مثال بالا دستور available() این ۶ بیت فضای اشغال شده را برای ما حساب می کند .

کاربرد دستور : شمارش تعداد بیت های ذخیره شده در بافر سریال

شکل کلی دستور :

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

```
Serial.available()
```

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

```
Serial1.available()
Serial2.available()
Serial3.available()
```

پارامترهای دستور :

هیچ پارامتری ندارد

مثال ۱۲: یک برنامه بنویسید که تعداد بیت های های که در بافر سریال ذخیره می شود را بشمارد و به ما نمایش دهد

```
int Number_BIT;                                  تعریف یک متغییر عددی برای ذخیره تعداد بیت
void setup() {
  Serial.begin(9600);                            تعیین سرعت ارسال اطلاعات سریال
}
void loop(){
 Number_BIT = Serial.available();                تعداد بیت اطلاعات موجود در بافر شمارش و در متغییر ریخته می شود
  Serial.print("number of bytes: ");             یک رشته دلخواه ارسال می شود
  Serial.print(Number_BIT);                      تعداد بیت های شمارش شده برای سریال مانیتور ارسال می شود
  Serial.print("\n");                            رفتن به خط جدید
  delay(500);                                    توقف نیم ثانیه ای
}
```

برای دیدن نتیجه سریال مانیتور را باز کنید ، می بینید در ابتدا عدد 0 باز گردانده می شود چون بافر خالی است ، با ارسال یک کارکتر مثل a عدد 3 برگشت داده می شود و با ارسال مجدد یک کارکتر مثل a عدد به 6 افزاریش می یابد

مثال ۱۳: برنامه ای بنویسید که تعداد بیت های موجود در بافر و هم اطلاعات موجود در بافر را برای ما بخواند و به ما نمایش دهد

```
 int bits;                                       تعریف متغییر عددی برای نمایش تعداد بیت های موجود در بافر
char data;                                       تعریف متغییر برای نمایش
void setup(){
  Serial.begin(9600);
}
void loop(){
   bits = Serial.available();                    تعداد بیت اطلاعات موجود را بشمار و در متغییر بریز
  Serial.print("bits: ");                        عبارت را بفرست
  Serial.print(bits);                            تعداد بیت ها را بفرست
  Serial.print("\n");                            برو خط جدید
  data = Serial.read();                          اطلاعات را از بافر بخوان و در متغییر بریز
  Serial.print("data: ");                        عبارت را بفرست
  Serial.print(data);                            اطلاعات خوانده شده از را بفرست
  Serial.print("\n");                            برو خط جدید
  Serial.print("---------");                     این عبارت را بفرست
  Serial.print("\n");  برو خط جدید
  delay(1000);     یک ثانیه صبر کن
}
```

در مثال بالا وقتی اطلاعاتی را برای پورت سریال بفرستیم ابتدا در بافر ذخیره میشود و تعداد بیت های ان ارسال می شود سپس اطلاعات از بافر به وسیله دستور Serial.read خوانده می شود ، هر اطلاعات که خانده شد کمی ار حجم اطلاعات بافر کم می شود و تعداد بیت های آن کاهش می یابد تا اینکه بافر کاملا خالی می شود

حالا نوبت آن رسیده بریم سر مثال اصلی

دو مثال قبل پیش نیاز بود تا شما بصورت کامل کارکرد دستور و عملکرد بافر را درک کنید ، حالا این دستور را با یک دستور شرطی ترکیب می کنیم تا بصورت کاربردی در بیاید

مثال ۱۴: برنامه ای بنویسید ، تنها زمانی که اطلاعات سریال دریافت می شود پورت سریال را درگیر شود و اطلاعات را از طریق پایه سریال برای ما بفرستد

```
char data;                                  تعریف متغییر
void setup() {
Serial.begin(9600);
}
void loop() {
if (Serial.available() > 0) {
```

در خط بالا یک دستور شرطی قرار دادیم که اگر تعداد بیت های موجود در داخل بافر بیشتر از 0 شد آنگاه وارد این حلقه شود که در این حلقه دستورات را میخوانیم و دوباره بافر خالی میشود ، وقتی بافر خالی شد و تعداد بیت ها برابر 0 شد از حلقه خارج م شویم

```
      data = Serial.read();          خواندن اطلاعات موجود در بافر سریال
      Serial.print("data: ");          نمایش عبارت مد نظر
      Serial.print(data);          نمایش اطلاعات خوانده شده از بافر
      Serial.print("\n");          رفتن به خط جدید
        }
}
```

در برنامه بالا تا زمانی که اطلاعات را نفرستیم و حجم اطلاعات بافر از 0 بیشتر نشود وارد حلقه نمیشود و اطلاعات را نمیخواند ، سریال مانیتوررا باز کنید و عبارت wle را تایپ کنید ، می بینید که مقادیر به شکل زیر بازگردانده می شوند

```
data: w
data: l
data: e
data:
data:
```

می بینید ۲ کارکتر خالی هم بازگردانیده شده اند ، حالا در مثال ذکر شده عدد شرط 0 را به 2 تغییر دهید و مجددا تست کنید ، ببینید چه نتیجه ای می گیرید

۳- خواندن یک رشته از درگاه سریال - readString()

در دستور قبلی دیدید که وقتی یک رشته مانند WLE را ارسال می کردیم ، کل رشته یک جا دریافت نمیشد بلکه کارکتر به کارکتر دریافت می شد ، یعنی اول W دریافت می شد ، بعد L دریافت میشد و در انتها E دریافت میشد ، با دستور readString() کل یک رشته (متن) را یکجا دریافت می کنیم ، در درگاه سریال بصورت پیش فرض برای هر بار دریافت اطلاعات 1 ثانیه صبر می کند ، یعنی برای خواندن اطلاعات از بافر سریال نهایتا ۱ ثانیه وقت میذاره و بعدش میره سراغ دستورات بعدی ، حالا این دستور readString() در این ۱ ثانیه هر چقدر در توانش باشد اطلاعات را از بافر سریال بیرون می کشد و بعداز این که ۱ ثانیه سپری شد هر چقدر را که توانسته بود از بافر دریافت کند بصورت یک رشته برمی گرداند (در ادامه آموزش ، نحوه تغییر این زمان را هم بیان می کنیم)

چون ۱ ثانیه پیش فرض زمان زیادی است پس دستور می تونه هر چی در بافر ذخیره شده باشد را دریافت کند ، این دستور زمانی ملموس تر می شود که زمان ۱ ثانیه را کم کنیم مثلا ۱۰ میلی ثانیه کنیم

کاربرد دستور : دریافت کل اطلاعات موجود در بافر سریال بصورت رشته یکپارچه

شکل کلی دستور :

```
Serial.readString()
```

پارامترهای دستور :

هیچ پارامتری ندارد

مثال ۱۵ : برنامه ای بنویسید که کل اطلاعات سریال خوانده شده از بافر را یک جا بصورت رشته نمایش دهد

```
String  data ;                                  تعریف یک متغییر از نوع رشته (متن)
void setup() {
Serial.begin(9600);
}
void loop() {
if (Serial.available() > 0) {                   تعریف شرط برای شروع دریافت
      data = Serial.readString()  ;             خواندن کل رشته موجود در بافر و ریختن در متغییر مد نظر
      Serial.print("data: ");                   ارسال عبارت مد نظر
      Serial.print(data);                       ارسال رشته خوانده شده از بافر
      Serial.print("\n");                       رفتن به خط تازه
        }
}
```

سریال مانیتور را باز کنید و یک عبارت مثل wle را بنویسید و enter را بزنید می بنید که عین عبارت بازگردانده می شود

❖ دستورات کار با دیتای سریال در آردوینو

در دستورات قبلی که آموزش دادیم دید که اطلاعات یا ارسال می شد و یا دریافت میشد و هیچ کاری روی دیتا انجام نمی شد ، در این قسمت دستوراتی را بیان می کنیم که به واسطه آنها می توانیم روی دیتا کار هایی انجام دهیم ، قبلا هم اشاره کردیم که دستورات دریافت اطلاعات مهم تر از دستورات ارسال دیتا هستند و تقریبا همه دستوراتی که روی دیتا تغییراتی ایجاد می کنند مربوط به دریافت دیتا هستند . دلیل این که فقط روی دیتای ورودی مانور داده شده و به دستوراتی برای ایجاد تغییرات در ارسال اطلاعات پرداخته نشده این است که ما برای ارسال دیتا محدودیت نداریم و هر طور که خواستیم می توانیم دیتا را بفرستیم و فرمت و شیوه های مختلفی را انتخاب کنیم ولی برای دیتای ورودی ما ازاد نیستیم و هر فرستنده و ماژول و ... که اطلاعات را برای ما می فرستد یک فرمت دیتای خاص دارد که ما باید توانایی کار با دیتای دریافتی را داشته باشیم .

در واقع دستورات این بخش همان دستورات دریافت اطلاعات هستند فقط می توانند کار هایی روی این اطلاعات دریافتی انجام دهند

۱ - خواندن دیتا تا کارکتر مدنظر - readStringUntil()

در این دستور ما یک کارکتر یا عبارت یا عدد را مشخص می کنیم تا هر وقت در خواندن اطلاعات به آن رسیدیم کار خواندن اطلاعات به پایان برسد ، وقتی یک رشته بزرگ را میفرستیم این رشته در بافر سریال ذخیره می شود و این دستور از دیتای ذخیره شده در بافر شروع به خواندن اطلاعات می کند وقتی به کارکتر مشخص شده رسید خواندن اطلاعات از بافر را قطع می کند و اطلاعات خوانده شده از بافر حذف می شود و مابقی اطلاعات باقی می ماند

کاربرد دستور : خواندن اطلاعات تا رسیدن به کارکتر مشخص شده

شکل کلی دستور :

```
Serial.readStringUntil(terminator)
```

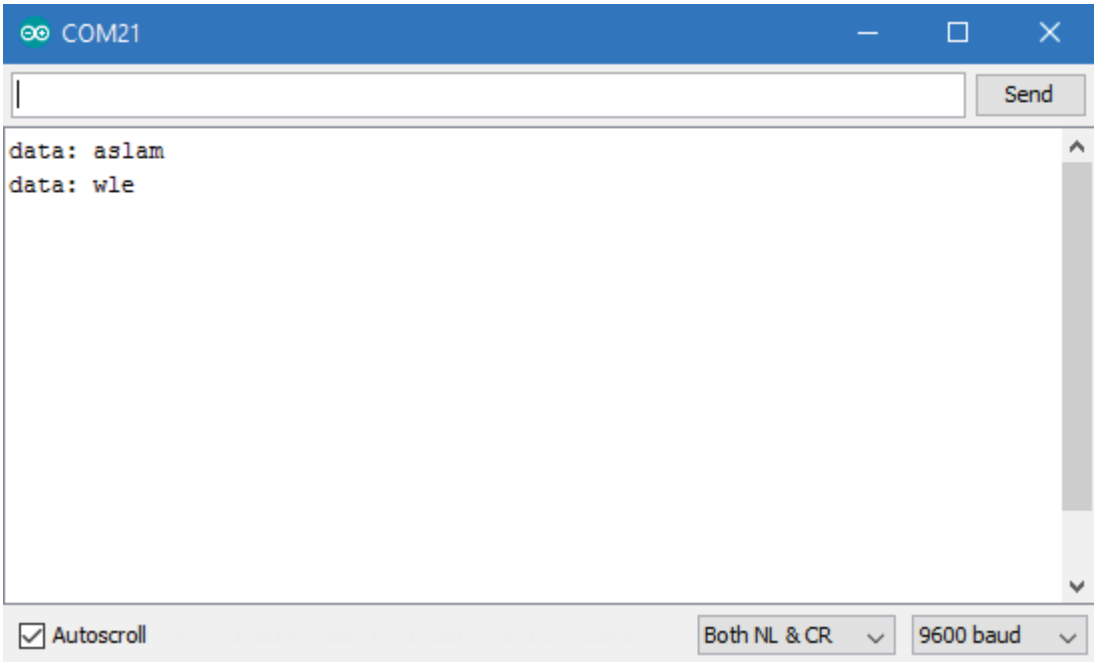
پارامترهای دستور :

Terminator : کارکتر یا عبارت مد نظر برای توقف خواندن اطلاعات از بافر سریال (char)

مثال ۱۶: برنامه ای بنویسید که در ان کارکتر # بعنوان کارکتر قطع عمل کند و فقط اطلاعات تا رسیدن به # خوانده شود

```
String  data ;                                  تعریف متغییر از نوع رشته برای ریختن اطلاعات دریافتی
void setup() {
Serial.begin(9600);
}
void loop() {
if (Serial.available() > 0) {                   تعریف شرط برای شروع کردن خواندن اطلاعا
 data = Serial.readStringUntil('#') ;
                                                تعیین کارکتر # برای توقف خواندن اطلاعات از بافر سریال
 Serial.print("data: ");                        ارسال عبارت مدنظر
 Serial.print(data);                            ارسال دیتای خوانده شده
 Serial.print("\n");                           رفتن به خط بعدی
        }
}
```

حالا برای تست برنامه ، سریال مانیتور را باز کنید و عبارت aslam#wle را تایپ و ارسال کنید ببینید چه اتفاقی می افتد ، بعداز انجام این کار می بینید خروجی به شکل زیر خواهد بود



حالا یه سوال مگه قرار نبود اطلاعات بعداز کارکتر # حذف شود ؟ چرا در تصویر بالا عبارت بعداز کارکتر # یعنی wle هم آماده است ؟

بدیهی است اگر عبارتی مانند aslam#wle را برای پورت سریال فرستاده باشیم این عبارت ابتدا در بافر سریال ذخیره می شود و اگر ما کارکتر # را برای توقف در نظر گرفته باشیم ، در ابتدا رشته aslam خوانده می شود وقتی به کارکتر # می رسیم خواندن از بافر سریال متوقف می شود ولی هنور عبارت wle در بافر باقی مانده است و چون

هنوز شرط خواندن اطلاعات `if (Serial.available() > 0)` درست است و حجم دیتا مانده در بافر از 0 بیشتر است (چون هنوز wle در بافر مانده است) ، بار دیگر وارد حلقه شده و مابقی دیتا را نیز میخواند

این دستور به شکل زیر نیز می تواند مورد استفاده قرار گیرد

```
String  data ;
char search = '#' ;
data = Serial.readStringUntil(search) ;
```

۲ - پیدا کردن رشته مد نظر - Serial.find()

با استفاده از این دستور می توان رشته مد نظر را بین اطلاعات موجود در بافر سریال جستجو کرد و بود و یا نبود آن رشته را تعیین کرد ، بعنوان مثال می توانیم عبات `wle` را به تابع معرفی کنیم در این صورت هر بار که پورت سریال اطلاعاتی را دریافت و ذخیره می کند تابع ما به دنبال رشته `wle` می گردد ، اگر رشته وجود داشته باشد عدد 1 را بر میگرداند و اگر رشته وجود نداشته باشد عدد 0 را باز می گرداند ، در واقع این تابع فقط با بله یا خیر به ما اعلام می کند چیزی که به دنبال آن هستیم پیدا شد یا نشد و اطلاعات دیگری درباره موقعیت و ... به ما نمی دهد

کاربرد دستور : جستجوی اطلاعات ورودی برای پیدا کردن رشته مد نظر

شکل کلی دستور :

```
Serial.find(target)
```

پارامترهای دستور :

target : رشته مد نظر ما که به دنبال بود و یا نبود آن هستیم

اطلاعات خروجی تابع : خروجی تابع از جنس بولین است (بولین یعنی صحیح یا غلط ، بود یا نبود و ...)

مثال ۱۷ : برنامه ای بنویسید که به دنبال رشته wle بگردد ، اگر رشته پیدا شد عبارت found را برگرداند و اگر رشته پیدا نشد عبارت NOT found را برای ما باز گرداند

```
boolean finds;        یک متغییر از جنس بولین تعریف می کنیم (متغییر بولین میتواند تنها دو مقدار ۰ یا ۱ را در خود جای دهد)
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {          تعریف شرط برای شروع خواندن اطلاعات

    finds = Serial.find("wle");
```

خط بالا به دنبال رشته wle می گردد ، اگر رشته پیدا شد ، خروجی آن یعنی finds برابر با 1 می شود و اگر نبود برابر با 0 می شود

```
    if (finds == 1)  {
                                  دستور شرطی که اگر متغییر finds برابر با 1 شود انگاه دستور پایین را اجرا کند
      Serial.print("found");
      Serial.print("\n");
    }
    else if (finds == 0)      همچینین اگر متغییر برابر با ۰ بود  انگاه دستورات پایین را اجرا کن
    {
      Serial.print("NOT found");
      Serial.print("\n");
    }

  }
}
```

برای اجرای این برنامه بعداز باز کردن سریال مانیتور یک عبات دلخواه را وارد کنید ببینید چه چیزی مشاهده می کنید ، همچنین عباراتی بنویسید که رشته wle را داشته باشند و مجددا نتیجه را بررسی کنید

۳ – پیدا کردن رشته مد نظر همراه با رشته قطع – Serial.findUntil()

این دستور دقیقا مثل دستور قبلی عمل می کند با این تفاوت که این دستور برای بررسی دو رشته است ، رشته هدف و رشته قطع که در صورت رسیدن به یکی از این دو مورد تابع عدد 1 را باز میگرداند و در صورت نیافتن هیچ کدام عدد 0 را بر میگرداند در صورت رسیدن به هدف عدد صفر و در صورت رسیدن به رشته قطع عدد یک را بر میگرداند و هم اینکه دریافت اطلاعات سریال را پایان می دهد

کاربرد دستور : جستجوی اطلاعات ورودی برای پیدا کردن رشته مد نظر و یک رشته قطع

شکل کلی دستور :

```
Serial.findUntil(target, terminal)
```

پارامترهای دستور :

target : رشته مد نظر ما که به دنبال بود و یا نبود آن هستیم

terminal : رشته قطع که با رسیدن به آن دریافت اطلاعات متوقف می شود

اطلاعات خروجی تابع : خروجی تابع از جنس بولین است (بولین یعنی صحیح یا غلط ، بود یا نبود و ...)

مثال : در اخرین نسخه کامپلر در حال حاضر که از آن استفاده می کنم (۱,۶,۷) اپلود این ستور بصورت کامل انجام نمیشه در نتیجه بخش دوم که مربوط به رشته قطع است به درستی عمل نمی کند

۴ - انتخاب بایت ها از پورت سریال - Serial.readBytes()

با استفاده از این دستور ما اطلاعات را از پورت سریال میخوانیم و و به تعداد معیین (length) در یک آرایه (buffer) می ریزیم ، حالا می توانیم با توجه به نیازمان اطلاعات را از خانه های ارایه ها بیرون بیاوریم

کاربرد دستور : ریختن تعداد بایت مد نظر در متغییر ارایه

شکل کلی دستور :

```
Serial.readBytes(buffer, length)
```

پارامترهای دستور :

buffer: بافر برای قرار ذخیره بایت ها که می تواند char یا byte باشد

length : تعداد بایت هایی که باید دریافت و در ارایه قرار بگیرد ، باید از جنس متغییر int انتخاب شود (فقط عدد)

اطلاعات خروجی تابع : خروجی تابع تعداد بایت هست

مثال ۱۸ : مثالی بنویسید که رشته دریافتی را به بخش های ۶ قسمتی تقسیم کند

```
char data[6];  یک ارایه هفت عضوی تعریف می کنیم (صفر هم یک خانه حساب می شود)
int n = 0 ;
void setup()
{ Serial.begin(9600); }
void loop(){
  if (Serial.available() > 0){
    n = Serial.readBytes(data, 6);
      data[5] data[4] data[3] data[2] data[1] data[0]  در خط بالا ما اطلاعات ورودی را در خانه ۰تا۵ ارایه قرار می دهیم یعنی
    for (int i = 0; i <= n - 1; i++)     یک حلقه فور ایجاد می کنیم که به تعداد بایت خروجی تابع تکرار شود
      Serial.print(data[i]);    در انجا هم اطلاعات از ۰ تا ۵ پشت سر هم چاپ می شوند
    Serial.print("\n");
  }
}
```

سریال مانیتور را باز کنید و عبارت wle.irwle.irwle.irwle.irwle.ir را پرینت کنید و نتیجه را مشاهده کنید

خروجی تابع Serial.readBytes(buffer, length) **تعداد بایت است یعنی اگر**

n = Serial.readBytes(buffer, length)

در این صورت n **تعداد بایت های ذخیره شده در بافر را نشان می دهد حالا اگر یادتان باشد تابع** Serial.available() **نیز دقیقا همین کار را انجام میداد و تعداد بایت های بافر را می شمارد**

تابع Serial.readBytes(buffer, length) **شبیه به تابع** Serial.read() **است ، با این تفاوت که در تابع جدید می توانیم اطلاعات ورودی را مدیریت کنیم و یعنی مشخص کنیم اولا چند بایت لازم داریم و دوما این بایت های دریافت شده در ارایه ذخیره بشن به این ترتیب می توانیم هر بایت را که لازم داشتیم بر داریم ،**

در مثال بالا وقتی به دستور زیر می رسیم

```
n = Serial.readBytes(data, 6);
```

اطلاعات به ترتیب از خانه 0 **وارد ارایه** data **می شوند ، بعنوان مثال ما** wle.ir **را برای درگاه سریال فرستادیم انگاه اطلاعات به صورت زیر وارد ارایه می شوند**

data[0] = w

data[1] = l

data[2] = e

data[3] = .

data[4] = i

data[5] = r

حالا ما به اطلاعات بصورت جدا دسترسی داریم یعنی اگر فقط با حرف e کار داشته باشیم از data[2] استفاده می کنیم به مسال زیر دقت کنید

```
char data[6];
int n;
void setup()
{Serial.begin(9600);}

void loop()
{
  if (Serial.available() > 0)
  {
    n = Serial.readBytes(data, 6);
      Serial.print(data[2]);
      Serial.print("\n");}
}
```

این مثال را اجرا کنید و سپس عبارت WLE.IR را وارد کنید ، عبارت E برگشت داده می شود

۵ - انتخاب بایت ها از پورت سریال و تعیین پایان دریافت - Serial.readBytesUntil()

این تابع دقیقا شبیه تابع قبلی است با این تفاوت که یک کارکتر هم تعیین می کنیم تا وقتی به کارکتر رسیدیم کار خواندن اطلاعات پایان یابد ، با استفاده از این دستور ما اطلاعات را از پورت سریال میخوانیم و به تعداد معیین (length) در یک آرایه (buffer) می ریزیم ، حالا می توانیم با توجه به نیازمان اطلاعات را از خانه های ارایه ها بیرون بیاوریم

کاربرد دستور : ریختن تعداد بایت مد نظر در متغییر ارایه

شکل کلی دستور :

```
Serial.readBytesUntil(character, buffer, length)
```

پارامترهای دستور :

character : کارکتر مد نظر تا هر وقت به ان رسیدیم کار دریافت اطلاعات متوقف شود

buffer : بافر برای قرار ذخیره بایت ها که می تواند char یا byte باشد

length : تعداد بایت هایی که باید دریافت و در ارایه قرار بگیرد ، باید از جنس متغییر int انتخاب شود (فقط عدد)

اطلاعات خروجی تابع : خروجی تابع تعداد بایت هست

مثال ۱۹ : مثالی که در زیر می بینید همان مثال قبلی است که با تابع جدید باز سازی شده است ، در این مثال ما کارکتر دات (.) را بعنوان کارکتر قطع در نظر گرفته ایم تا در صورتی که به آن رسید دریافت اطلاعات متوقف شود ، برای تست این مثال سریال مانیتور را باز کنید و عبارت WLE.IR را وارد کنید

```
char data[6];
int n;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    n = Serial.readBytesUntil('.',data, 6);
    for (int i = 0; i <= n-1 ; i++)
      Serial.print(data[i]);
    Serial.print("\n");
  }
}
```

۵ - خواندن اعداد اعشاری - parseFloat()

این تابع اعداد اعشاری داخل بافر را تا دو رقم اعشار میخواند و برای اعداد صحیح دو رقم اعشار 0 در نظر می گیرد

یعنی عددی مثل 5658 . 5 را به صورت 5 . 56 باز می گرداند و عدد مثل 6 را بصورت 6 . 00 باز میگرداند

کاربرد دستور : خواندن اعداد اعشاری

شکل کلی دستور :

```
Serial.parseFloat()
```

پارامترهای دستور :

هیچ پارامتری ندارد

اطلاعات خروجی تابع:

اعداد اعشاری

مثال ۲۰ : مثالی بنویسید که اعداد اعشاری را باز گرداند

```
float n;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    n = Serial.parseFloat() ;
    Serial.print(n);
    Serial.print("\n");
  }
}
```

سریال مانیتور را باز کنید و اعداد را وارد کنید ببینید اردوینو چه چیزی بر میگرداند

۶ - خواندن اعداد صحیح - parseInt()

این تابع بافر سریال را میخواند و تنها اعداد صحیح مثل 2- , 1- , 3 , 2 , 1 , 0 را بر میگرداند ، این تابع هر نوع کارکتری مثل ممیز ، حرف ، علامت و ... را حذف می کند و کارش تنها خواندن اعداد صحیح است ، اگر کارکتر یا ممیز یا علامتی را به تنهایی به ان بدهیم عدد 0 را بر میگرداند

بعنوان مثال اگر عدد 56 . 87 را به آن بدهیم به صورت زیر عمل می کند

تابع ابتدا عدد 56 را میخواند وقتی به . رسید کار خواندن اطلاعات از بافر را متوقف می کند ، چون بافر هنوز خالی نشده است ، باری دیگر به سراغ خواندن دیتا می رود و این بار عدد87 را نیز می خواند

کاربرد دستور : خواندن اعداد صحیح از بافر سریال

شکل کلی دستور :

```
all Arduino:
Serial.parseInt()
Arduino Mega only:
Serial1.parseInt()
Serial2.parseInt()
Serial3.parseInt()
```

پارامترهای دستور :

هیچ پارامتری ندارد

اطلاعات خروجی تابع:

اعداد صحیح

مثال ۲۱ : مثالی بنویسید که فقط اعداد صحیح را باز گرداند

```
int n;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
```

مثال ۲۲ : مثالی بنویسید که اطلاعات بافر سریال را بخواند ولی بافر را پاک نکند

```
char n;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
 n = Serial.peek()    ;  خواندن بافر بدون پاک کردن
    Serial.print(n);     نمایش مقادیر خانده شده
    Serial.print("\n");
    delay(1000);
}
```

۸ - تنظیم زمان بین دریافت های سریال - Serial.setTimeout()

اگر خاطرتان باشد در بخش های اولیه این آموزش ذکر کردیم که زمان بین هر بار دریافت از بافر سریال برابر با یک ثانیه است ، حالا اگر بخواهیم این زمان را تغییر دهیم از این دستور استفاده می کنیم ، این دستور زمان را بر حسب میلی ثانیه تنظیم می کند در حالت پیش فرض روی ۱۰۰۰ میلی ثانیه (۱ ثانیه) قرار دارد که ما می توانیم ان را تغییر دهیم این دستور بیشتر همراه تابع های `Serial.readBytesUntil()` , `Serial.readBytes()` `Serial.parseInt()` , `Serial.parseFloat()` کاربرد دارد

کاربرد دستور : تغییر زمان بین هر بار دریافت اطلاعات از بافر

شکل کلی دستور :

```
Serial.setTimeout(time)
```

پارامترهای دستور :

time : زمان بر حسب ثانیه ، نوع متغییر هم اگر برایش تعریف کنیم باید **long** باشد

```
void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(1000);
}
```

این دستور باید به شکل بالا و داخل حلقه setup مورد استفاده قرار گیرد ، در مثال بالا می توان بجای ۱۰۰۰ عدد مد نظر خود را قرار داد

۹ - حلقه مخصوص دریافت اطلاعات سریال - serialEvent()

این دستور بصورت اختصاصی یک لوپ یا همان حلقه را به ما می دهد که تنها زمانی وارد حلقه می شویم که اطلاعاتب دریافت شود ، قبلا هم برای اینکه میکرو همیشه درگیر نباشد با استفاده از دستور شرطی و `Serial.available()` یک حلقه به شکل زیر درست کرده بودیم

```
 if (Serial.available() > 0)
  {
// فعالیت مد نظر
  }
```

کاربرد دستور : ایجاد حلقه برای زمانی که اطلاعات سریال دریافت شود

شکل کلی دستور :

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

```
void serialEvent(){
//statements
}
```

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

```
void serialEvent1(){
//statements
}

void serialEvent2(){
//statements
}

void serialEvent3(){
//statements
}
```

پارامترهای دستور :

statements : دستوراتی که باید بنویسیم

ما می توانیم این حلقه را قبل و یا بعداز حلقه اصلی قرار دهیم به مثال زیر توجه کنید

مثال 23 :

```
String  data;
void setup()
{
  Serial.begin(9600);

}

void loop()
{
//
}
void serialEvent(){
      data = Serial.readString()  ;
      Serial.print("data: ");
      Serial.print(data);
      Serial.print("\n");
}
```

در مثال بالا در صورتی که اطلاعات سریال دریافت شود وارد حلقه serialEvent **می شود و ان اطلاعات را برای ما به نمایش در می آورد ، مثال بالا را تست کنید می بینید که با باز کردن سریال مانیتور و وارد کردن عبارت یا عدد یا هر چیزی دیگر یک ثانیه طول می شود تا آن اطلاعات برای ما برسد برای تست تابع قبلی که زمان را تغییر می داد اکنون خط زیر را به مثال بالا اضاف کنید تا ببینید این فعه اطلاعات را که وارد می کنید با سرعت بیشتری اطلاعات برای ما بازگردانیده می شوند**

```
Serial.setTimeout(15);
```

در حالت پیش فرض زمان روی ۱۰۰۰ میلی ثانیه است ما آن را روی ۱۵ میلی ثانیه قرار می دهید

```
String  data;
void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(15);
}

void loop()
{
//
}
void serialEvent(){
      data = Serial.readString()  ;
      Serial.print("data: ");
      Serial.print(data);
      Serial.print("\n");
}
```

این مثال را نیز حتما تست کنید