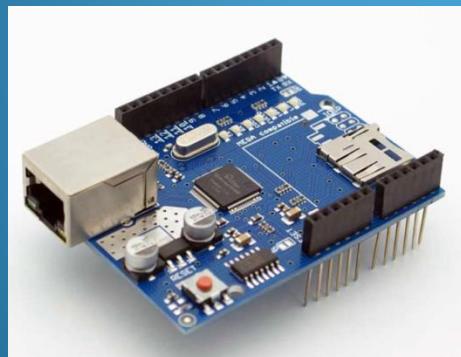


آردوینو

دانشگاه علم و فرهنگ
استاد: صالحی

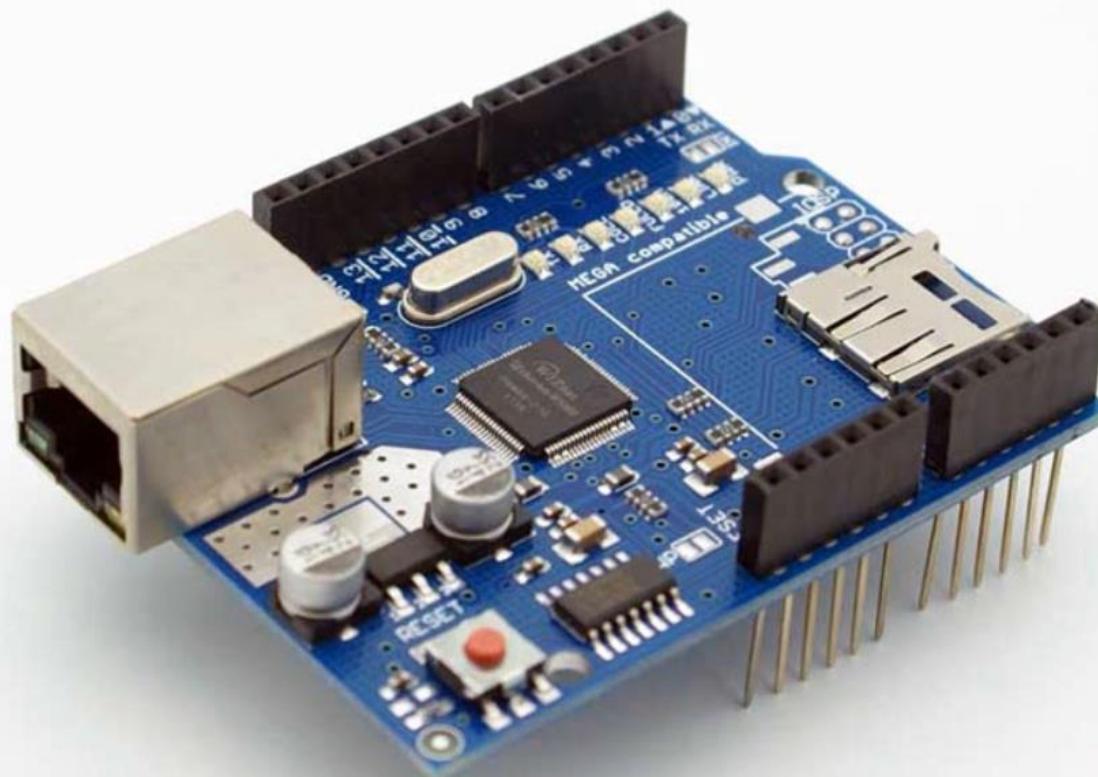


راه اندازی مازول (شیلد) شبکه (W5100)

لوازم مورد نیاز برای این پروژه

- آردوینو UNO
- شیلد اترنت W5100
- Modem
- کابل شبکه
- BreadBoard

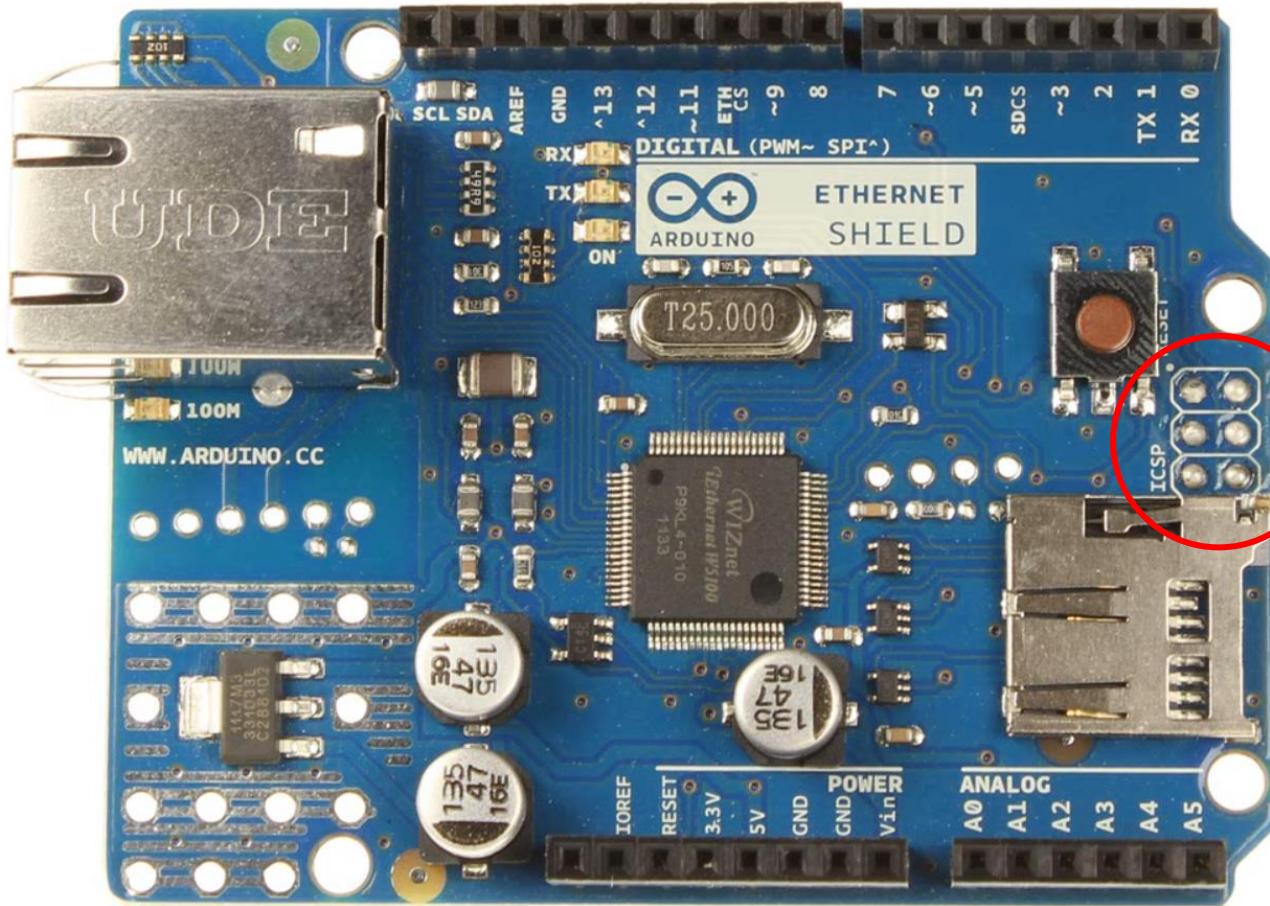
ماژول (شیلد) شبکه W5100



مشخصات مازول (شیلد) W5100

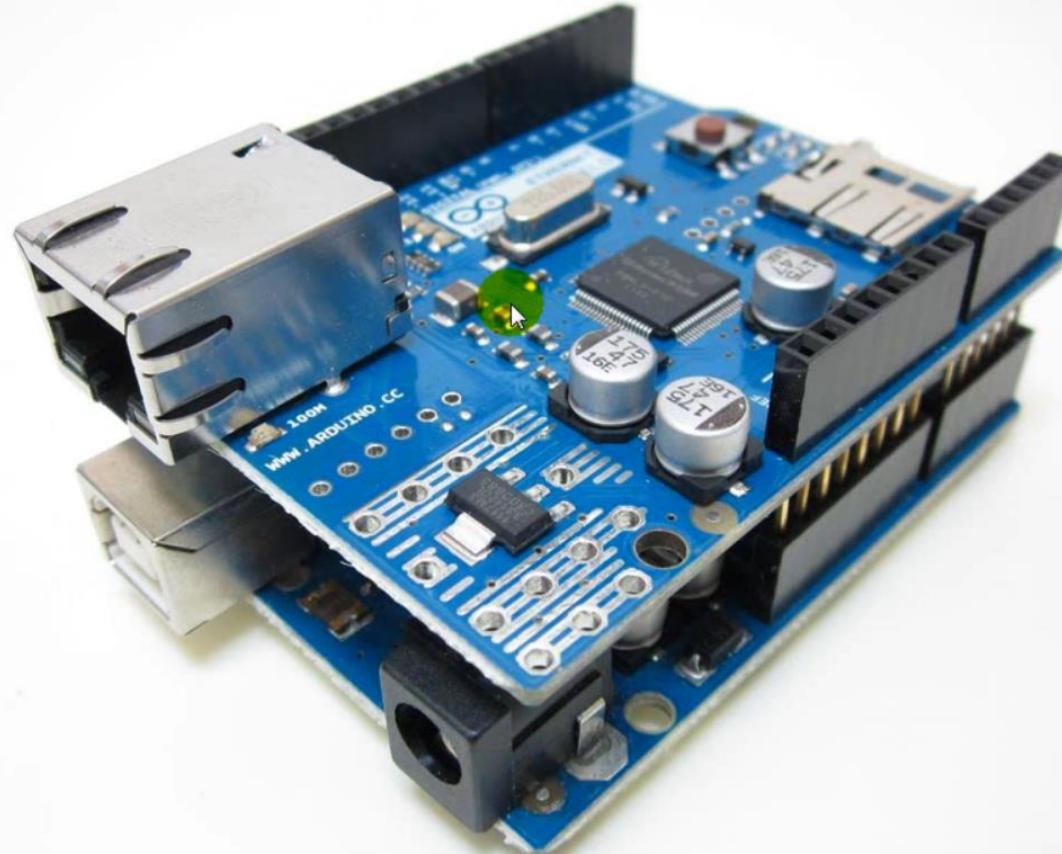
مشخصات مازول شبکه	
Wiznet W5100	چیپست
۵ ولت	ولتاژ کاری
۱۰/۱۰۰ Mbps	نرخ انتقال
SPI	نوع رابط
نشان می دهد که برد و شیلد به منبع تغذیه متصل شده اند.	PWR
وجود یک لینک شبکه را نشان می دهد و وقتی شیلد، دیتا می فرستد یا دریافت می کند چشمک می زند.	LINK
نشان می دهد که اتصال شبکه دو رشته ای کامل است.	FULLD
نشان دهنده ای حضور اتصال شبکه ۱۰۰ Mb/s (در مقابل ۱۰ Mb/s).	100M
وقتی شیلد، دیتا دریافت می کند چشمک می زند.	RX
وقتی شیلد، دیتا ارسال می کند چشمک می زند.	TX
وقتی collision در شبکه شناسایی شود چشمک می زند.	COLL

ماژول W5100 از نمای بالا



ICSP
 محل اتصال با آردوینو

سوار کردن مازول شبکه روی برد آردوینو

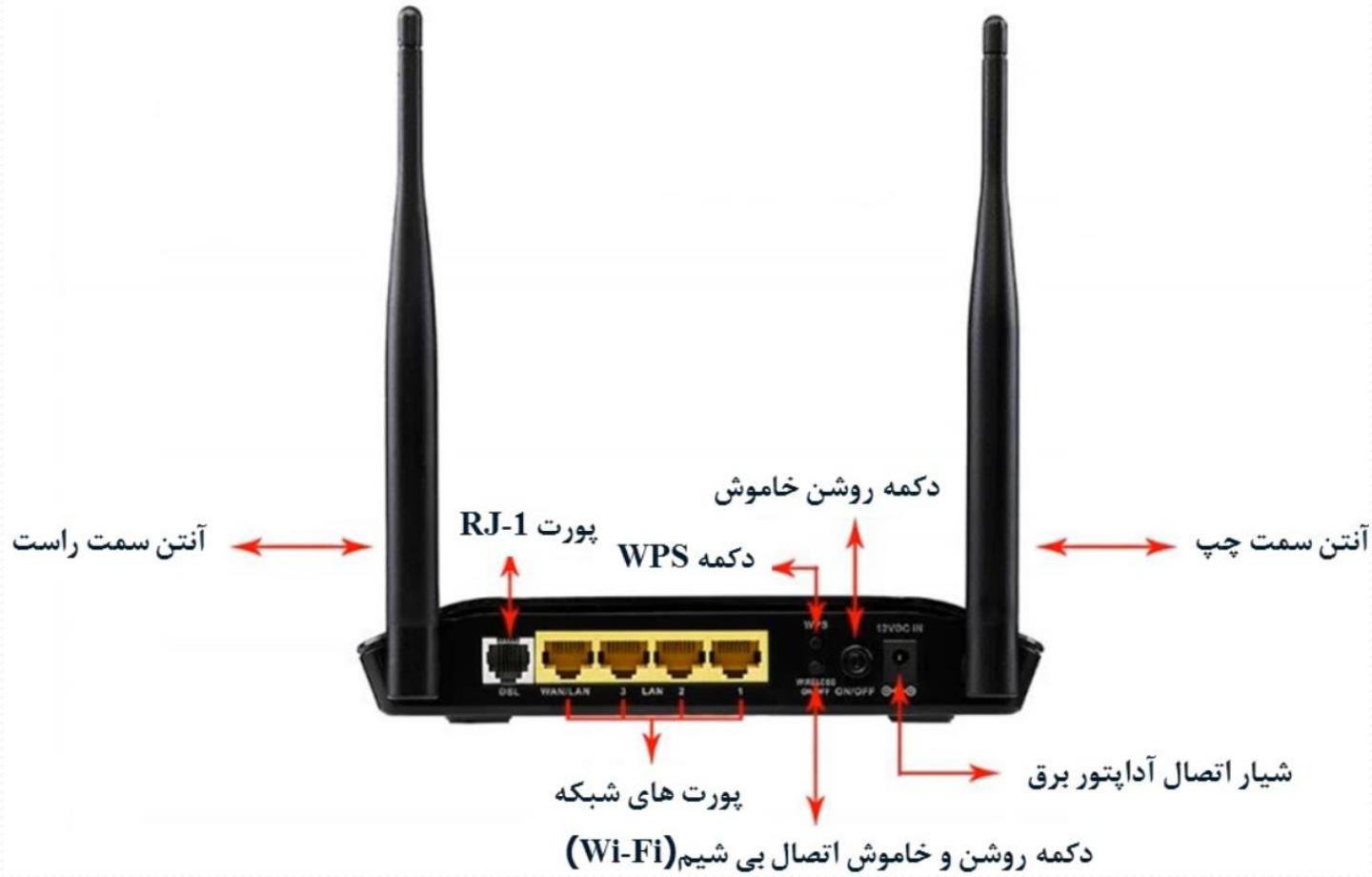


مودم



استاد: صالحی

مودم از نمای رو برو

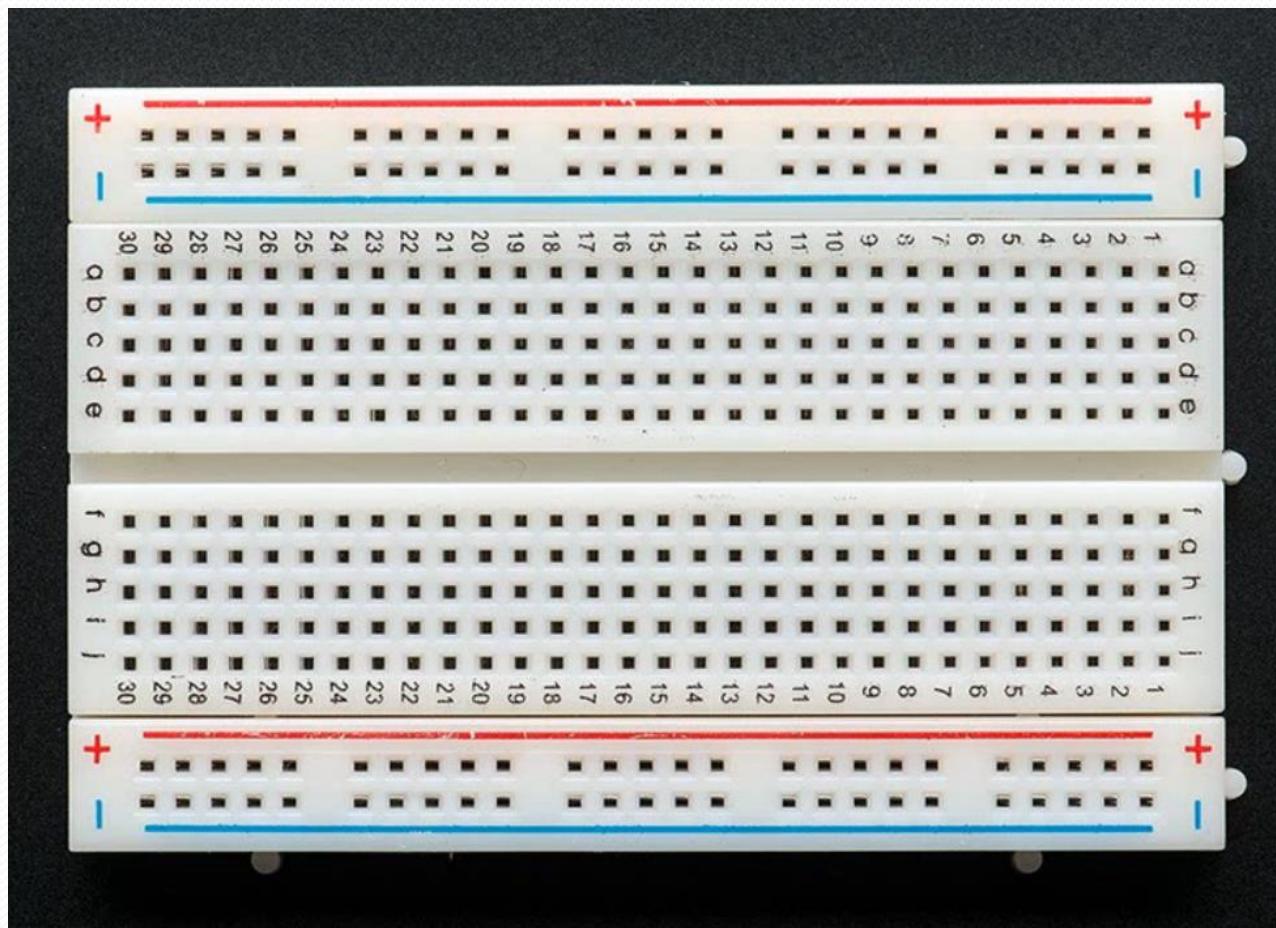


کابل شبکه

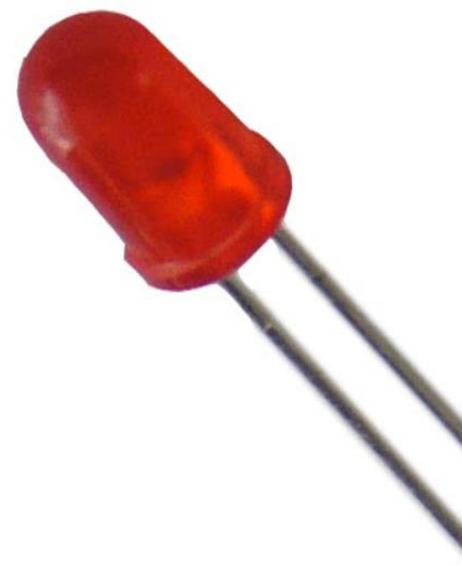


استاد: صالحی

برد بورد

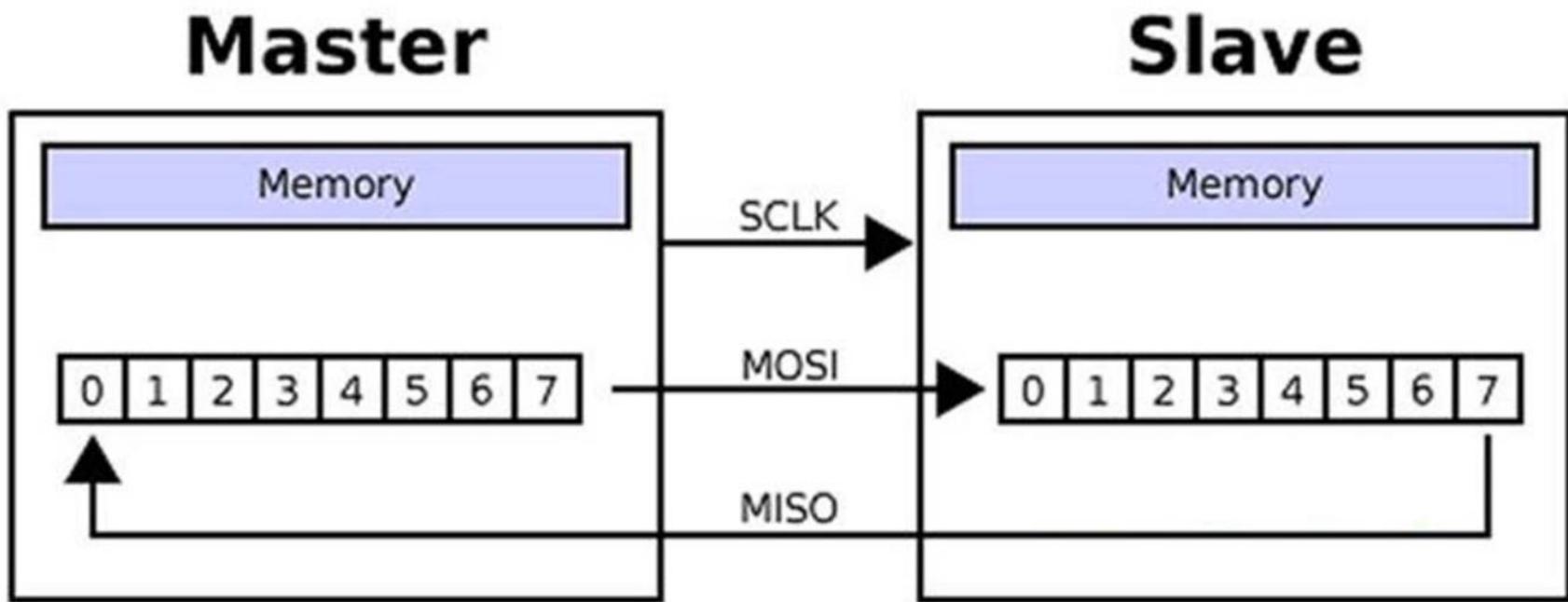


LED & LED RGB



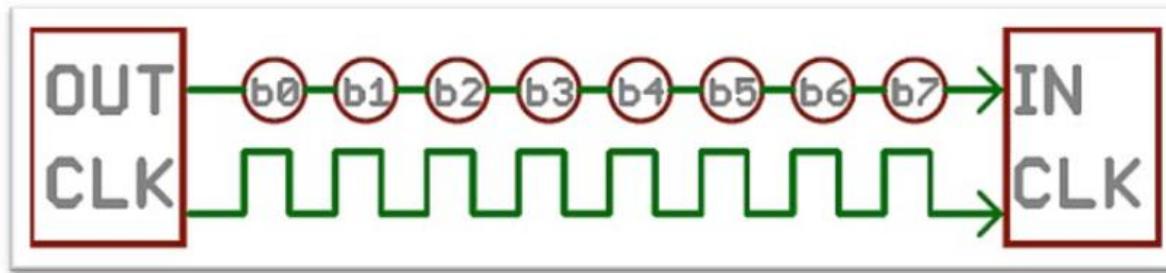
استاد : صالحی

معرفی واحد ارتباط سریال



پروتکل ارتباط سریال

- وجود تنها یک سیم داده بین فرستنده و گیرنده
- وجود یک سیم مشترک (زمین)
- قرار دادن بیتهای داده به صورت پشت سر هم توسط فرستنده و دریافت یک به یک بیت ها در گیرنده
- کاربرد وسیع در تجهیزات مخابراتی و ارتباطات راه دور



انواع ارتباط سریال

- 1- **USART** : Universal Serial Asynchronous/Synchronous Receiver Transmitter
- 2-**SPI**: Serial Peripheral Interface
- 3-**TWI**: Two-Wire Interface

USART

- سرعت متوسط
- کاربرد در مسافت‌های طولانی
- ارتباط با اکثر سنسورها و ماژول‌ها
- بیشتر از حالت غیر همزمان استفاده می‌شود، چونکه مقرنون به صرفه‌تر است و نیاز به سیم اضافی ندارد
- عمدتاً از روش غیر همزمان استفاده می‌شود که در اینصورت نرخ انتقال اطلاعات و تعداد بیت‌های داده باید مشخص شود.

(واحد سریال وسایل جانبی) SPI

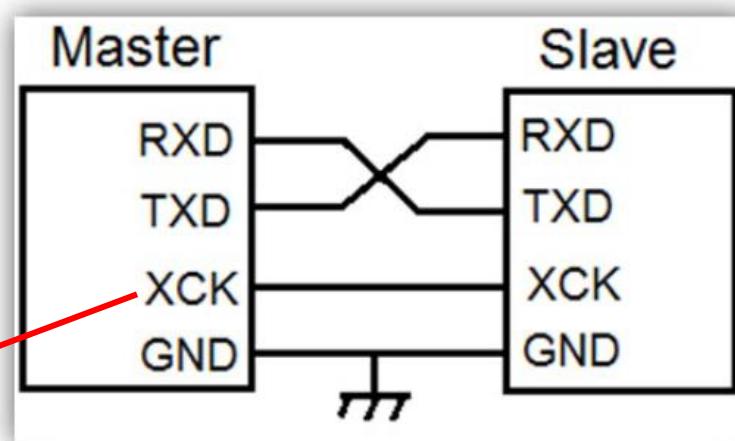
- سرعت خیلی بالا
- در فواصل خیلی کوتاه
- برای ارتباط با تجهیزاتی مثل مazzoل های شبکه ، کارت های حافظه و پروگرام بکار می رود



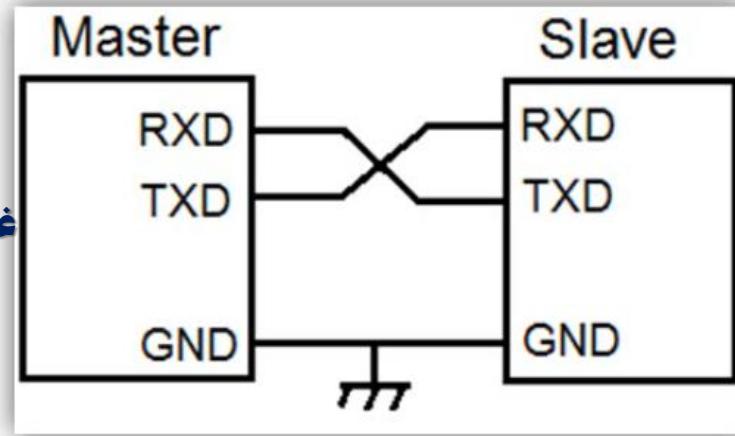
- ارتباط با المانهای جانبی نظیر سنسور ها و مازولهای سرعت پایین
- مزیت: تعدادی زیاد وسیله را از طریق این نوع ارتباط، انجام می دهیم

USART

همزمان
کلای

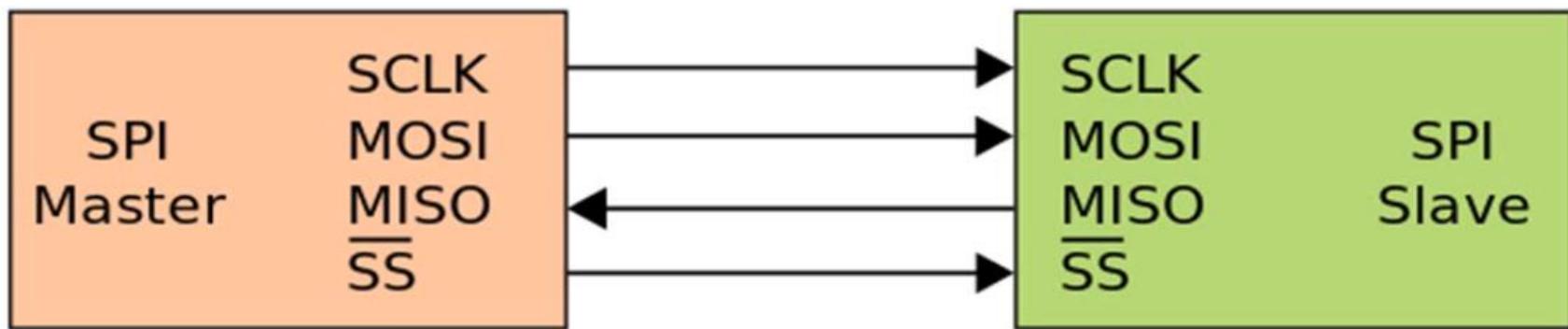


غیر همزمان(کلای ندارد)



- معرفی ارتباط USART
- تنظیمات مورد نیاز
- ویژگی های ارتباط USART

ارتباط SPI

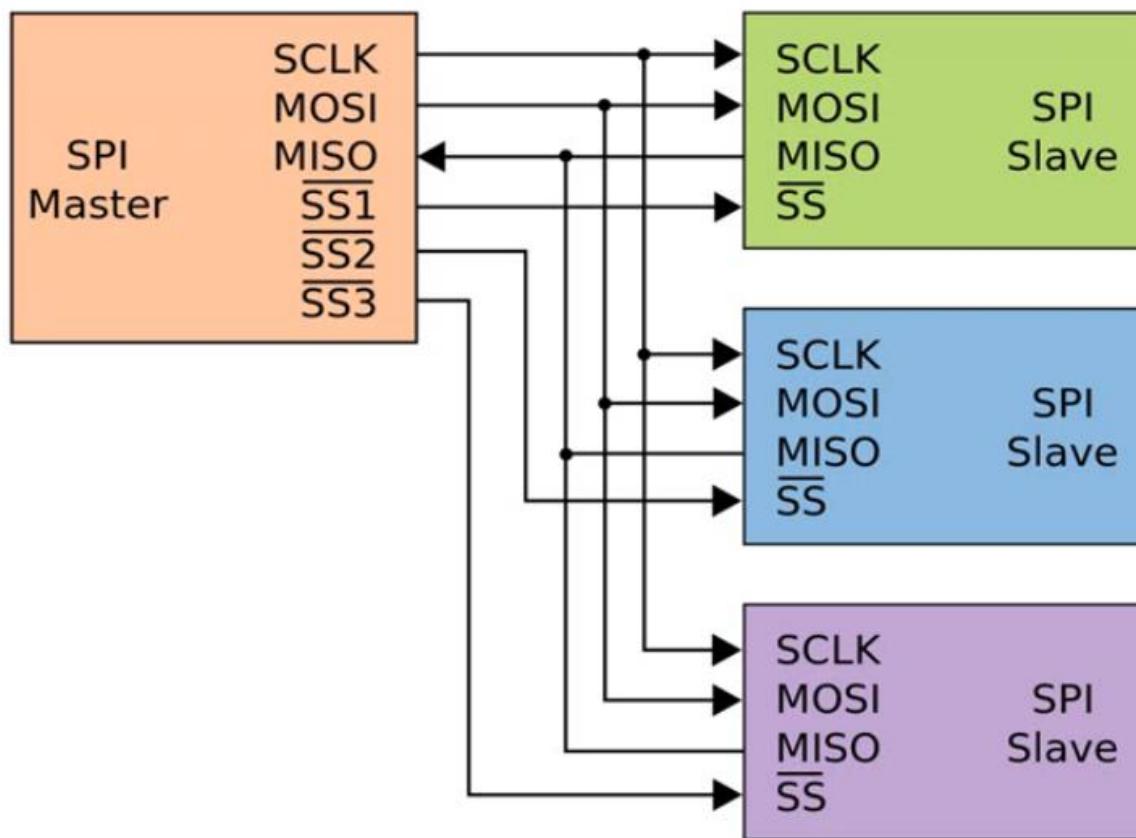


- به شکل سریال و فقط همزمان و full-duplex می باشد
- هر دو سمت Master و Slave می توانند به شکل فرستنده یا گیرنده باشند
- آغاز کننده ارتباط Master است
- از ۳ سیم استفاده می شود و ارتباط سنکرون می باشد.
- قابلیت تنظیم سرعت انتقال اطلاعات دارد

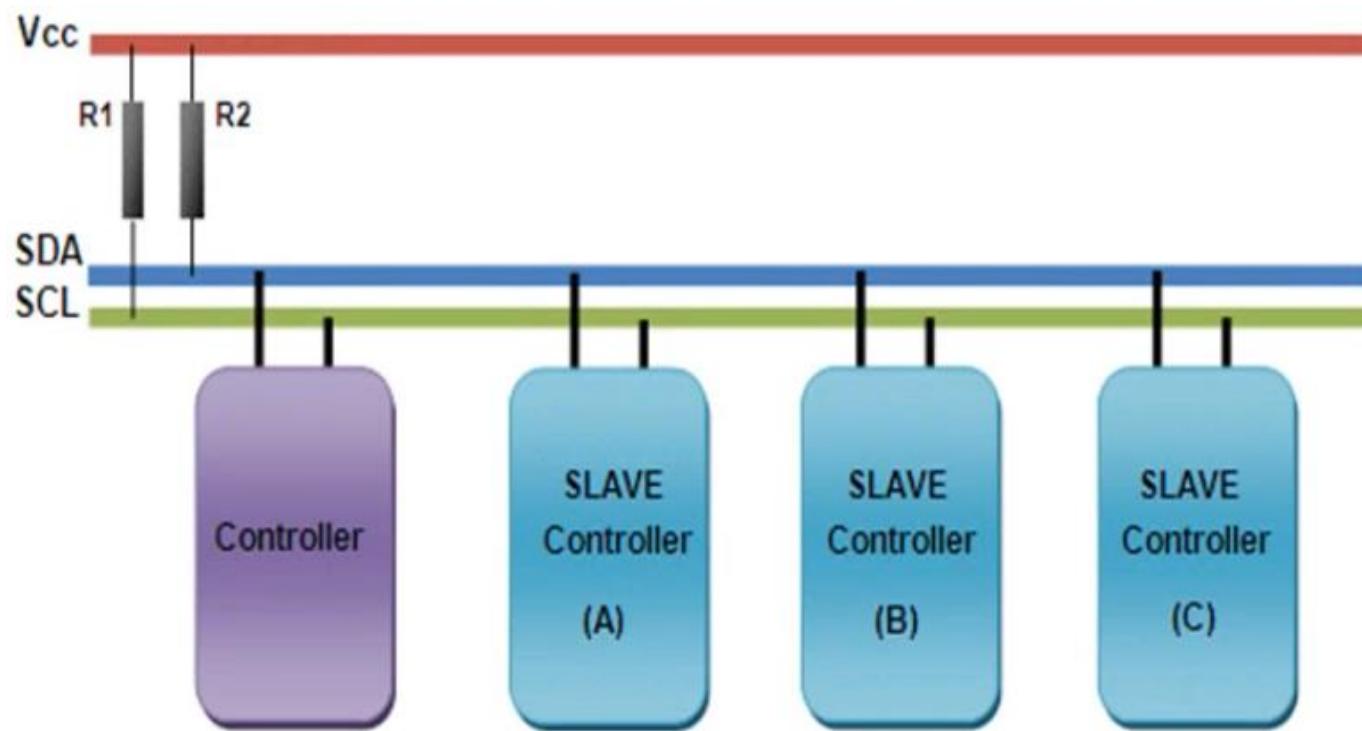
تشریح پایه ها در ارتباط SPI

نام پایه	شرح عملکرد
SS	مخف Slave Select بوده و از طریق آن دستگاه Slave تعیین می شود. اگر آن را Low در نظر بگیریم، دستگاه در مدل Slave قرار میگیرد.
SCK	مخف Serial Clock بوده و زمانی که SPI از طریق مقداردهی ثبات داده خود، مقدار دهی می شود، تولید کننده کلک در سمت Master، پالس های ساعت را بین Master و Slave فراهم کرده و هماهنگ می کند.
MOSI	مخف Master Out Slave In بوده و از طریق آن Master بیت ها را پشت سر هم برای ارسال می کند. در واقع از طریق این پایه Master خروجی و Slave ورودی در نظر گرفته می شود
MISO	مخف Master In Slave Out بوده و از طریق آن Slave بیت ها را پشت سر هم برای ارسال می کند. در واقع از طریق این پایه Master ورودی و Slave خروجی در نظر گرفته می شود

شبکه بندی چند slave با یک Master



TWI

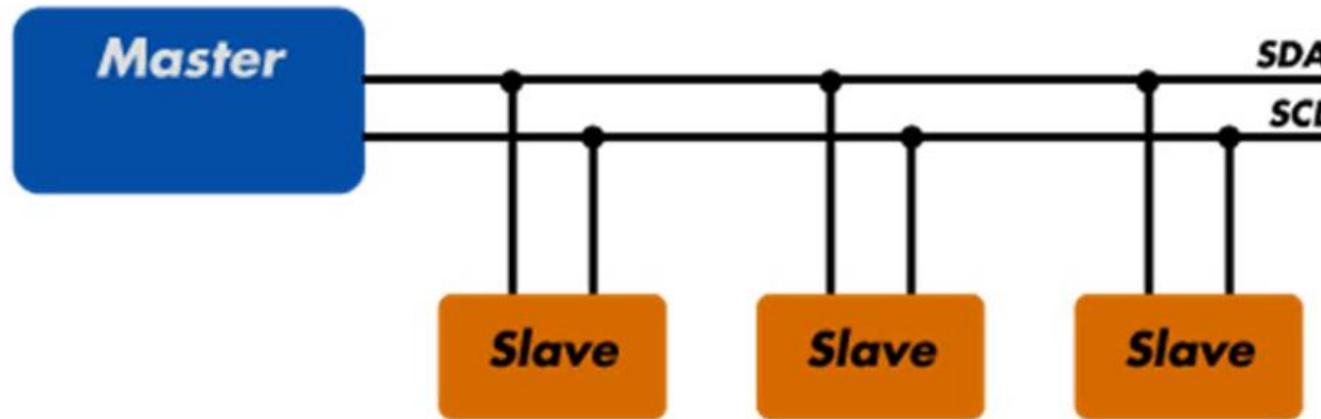


در این نوع ارتباط، حداقل می‌توان ۱۲۸ دستگاه را بهم متصل کرد که هر دستگاه دارای یک آدرس منحصر به فرد خواهد بود و همچنین هر دستگاه می‌تواند در فرکانس ۴۰۰ کیلو هرتز داده‌ها را ارسال یا دریافت کند.

I2C رابط

در اینجا با دو خط ارتباط انجام می شود

- (Serial Data Line) SDA
- (Serial Clock Line) SCL

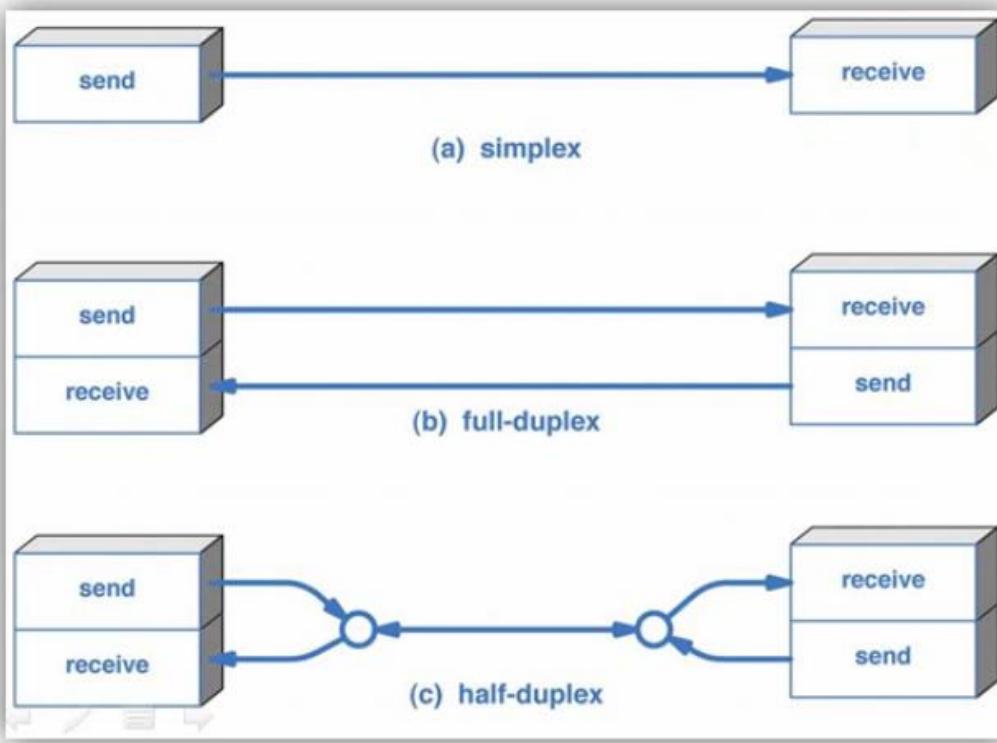


I2C

در سه مود مختلف کار میکند :

- استاندارد 100 kbps
- سریع 400 kbps
- فوق العاده سریع 3.4 Mbps

حالتهای ارتباط سریال



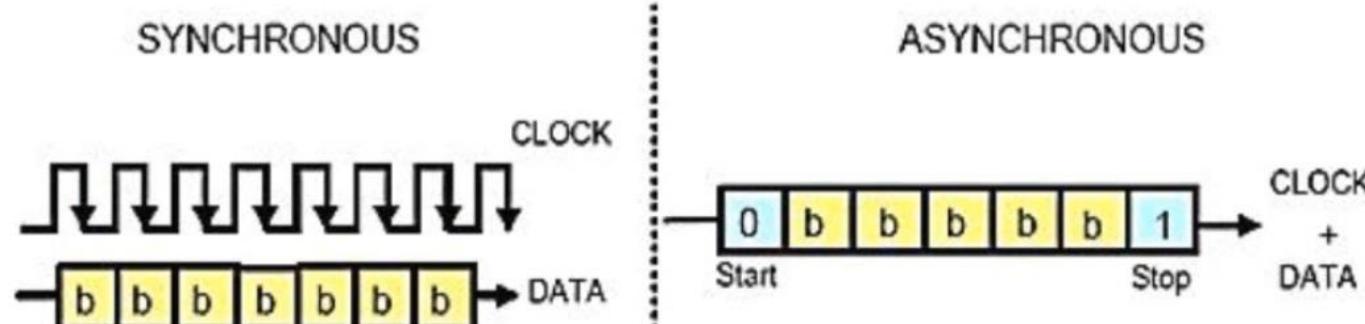
- ارتباط یک طرفه (Simplex)

- ارتباط نیمه دوطرفه (Half Duplex)

- ارتباط دوطرفه (Full Duplex)

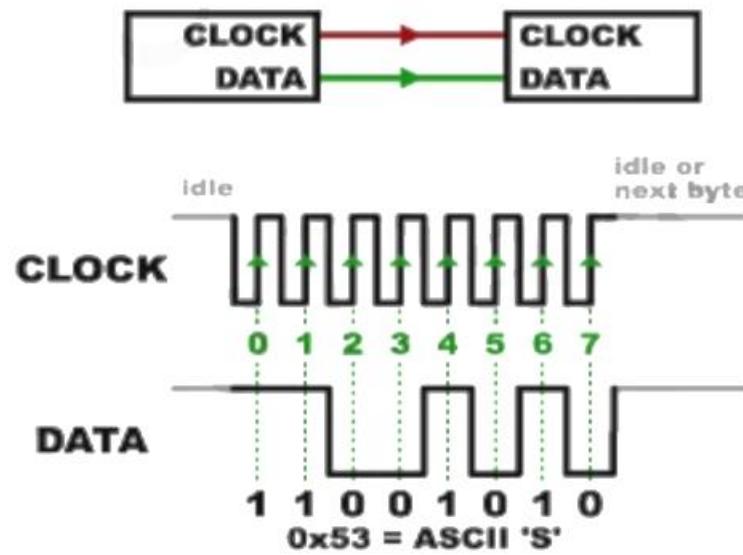
روش های ارسال اطلاعات سریال

- روش همگام (Synchronous)
- روش ناهمگام (Asynchronous)



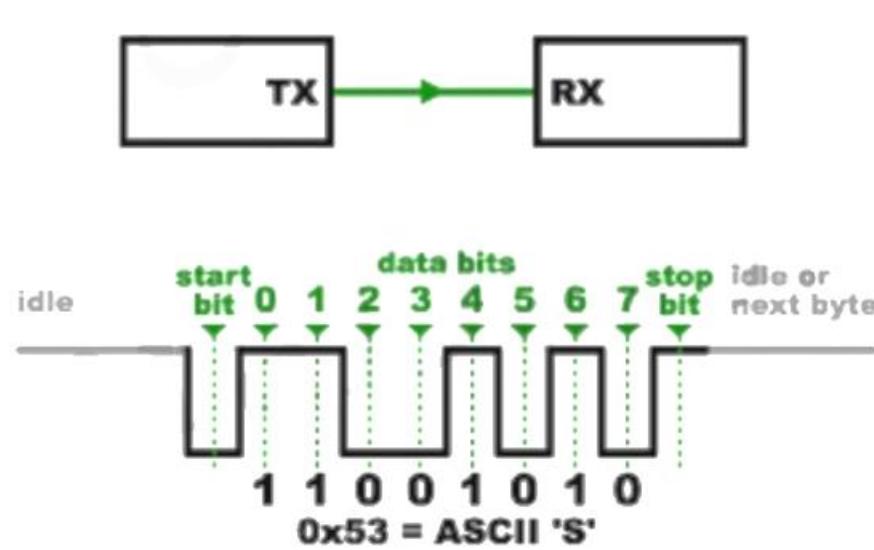
روش همکام

- فرستنده به همراه داده ها، یک سیگنال ساعت ارسال می کند.



روش غیر همکام

قبل از آغاز ارتباط، نرخ داده بین فرستنده و گیرنده مورد توافق قرار میگیرد.



تفاوت ارتباط همگام با ناهمگام

- عدم محدودیت طول بسته در ارتباط همگام
- تغییر سرعت ارتباط در ارتباط همگام (با تغییر ساعت انتقال)
- وجود سربار اطلاعات در ارتباط غیرهمگام و کاهش کارایی آن
- عدم نیاز به سیگنال ساعت (CLK) در ارتباط غیرهمگام

قالب انتقال ارتباط سریال ناهمگام

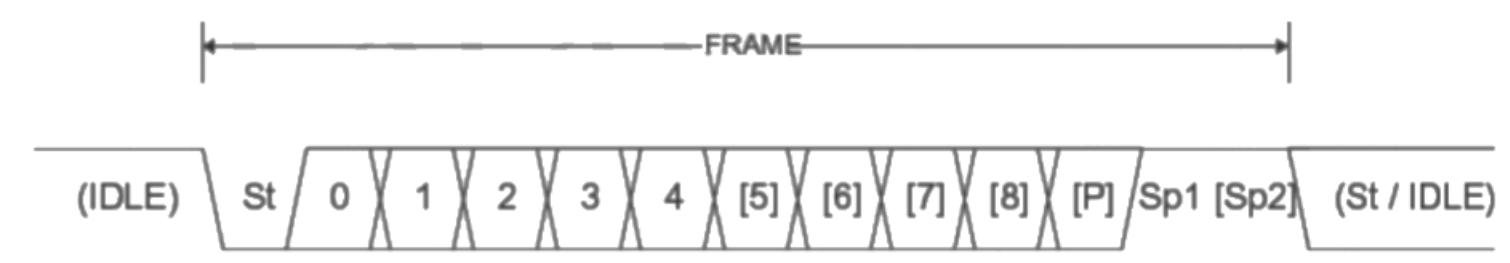
ارسال و دریافت داده ها در قالب مشخص (Frame)

ارسال سیگنال Start به گیرنده در زمان ارسال دیتا

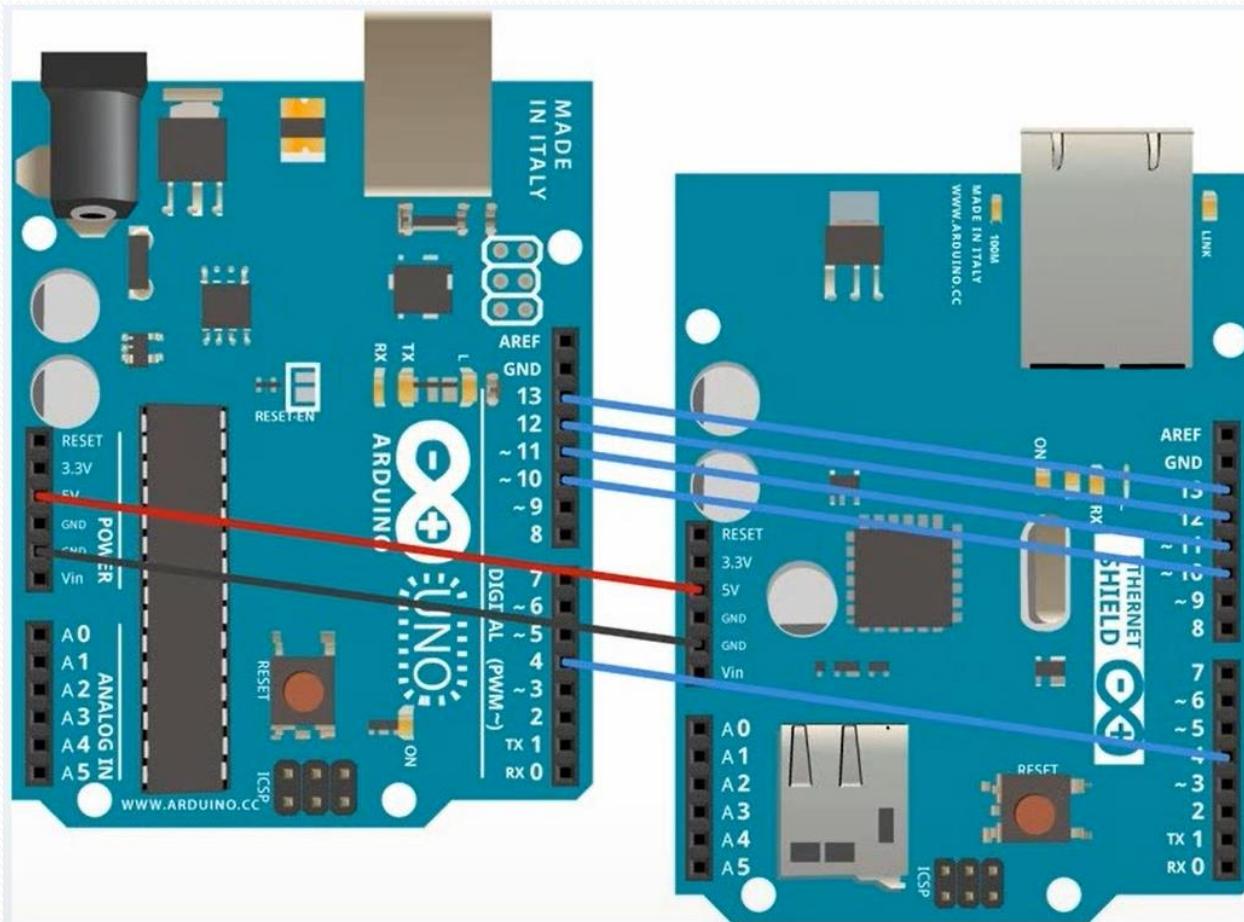
ارسال دیتای مورد نظر (حداکثر ۸ بیت)

ارسال دیتا از کم ارزش ترین بیت (LSB) آغاز می گردد

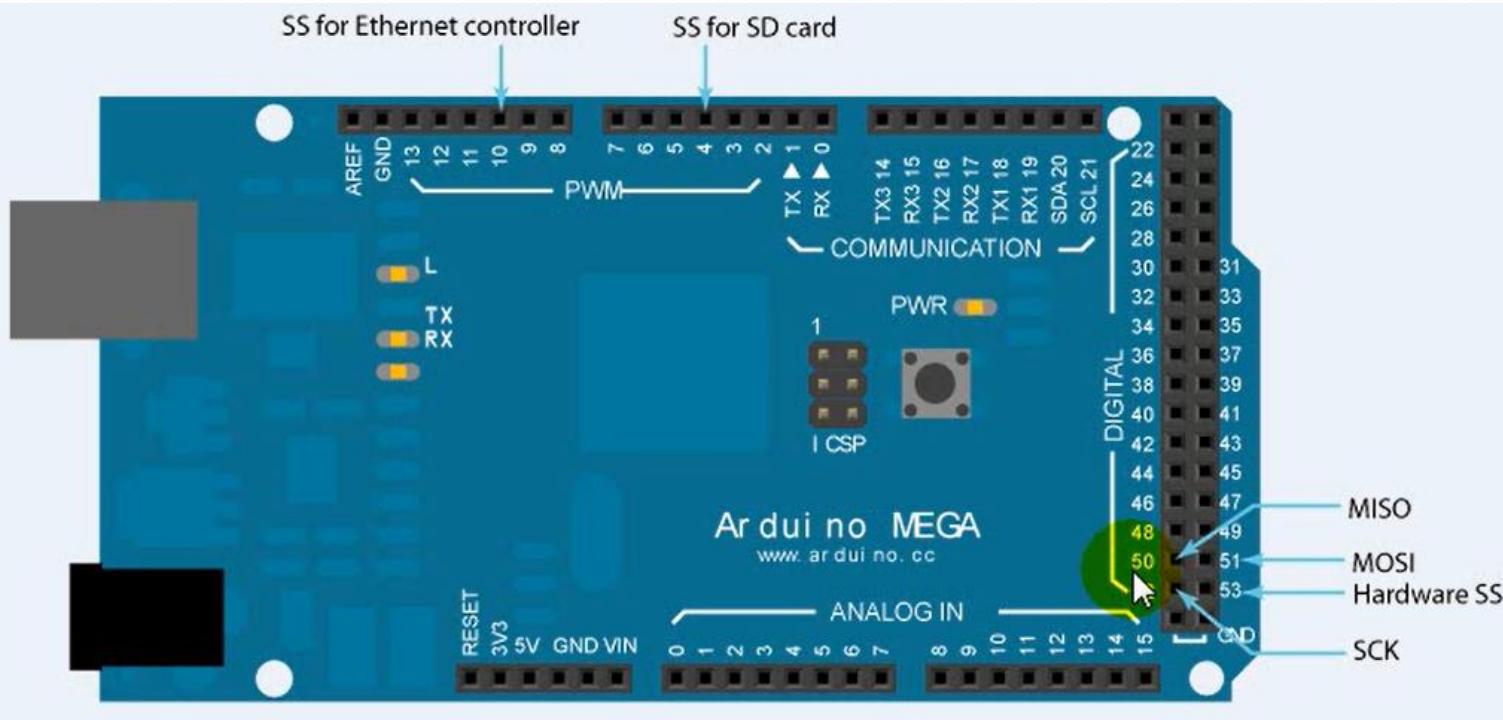
ارسال بیت Parity Check در صورت وجود و در نهایت بیت Stop



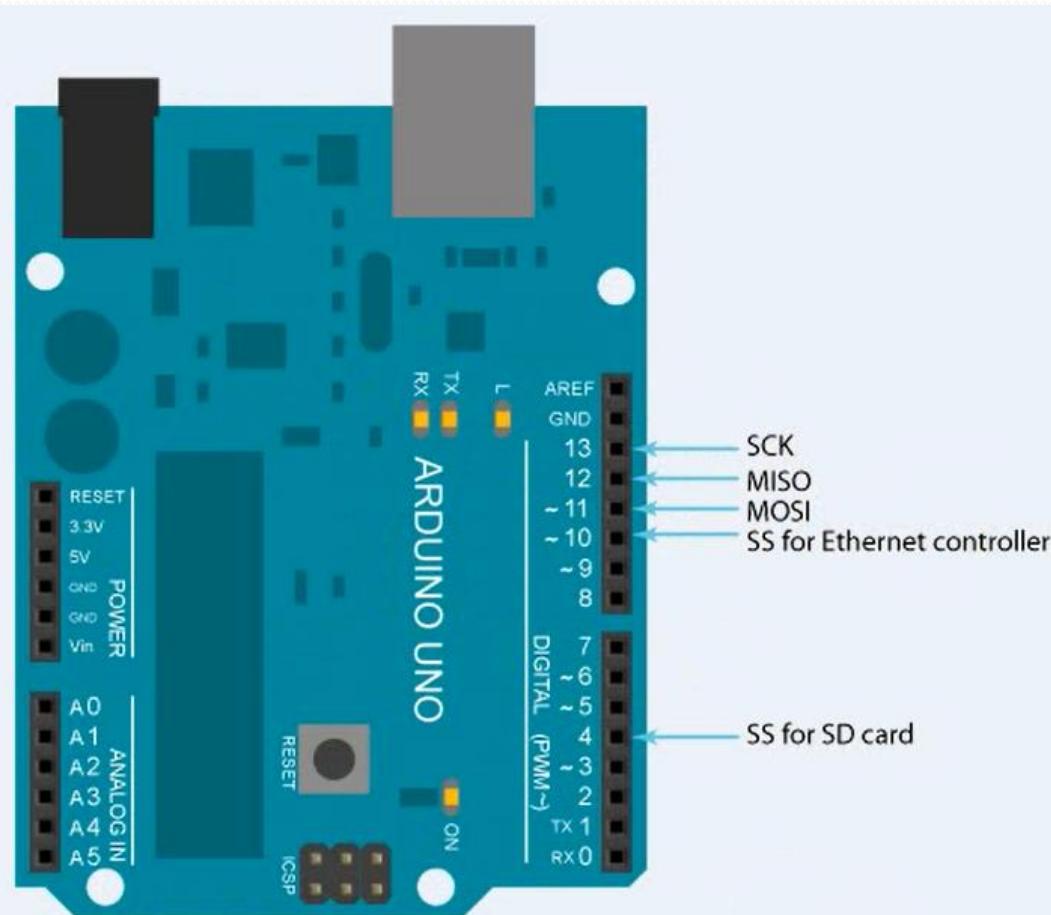
اتصال شیلد شبکه به آردوینو



پایه های مورد نظر برای SPI



پایه های مورد نظر برای SPI



کتابخانه Ethernet در آردوینو

برای مقدار دهی اولیه کتابخانه



Ethernet Class

-

برای تنظیمات IP



IPAddress Class

-

ایجاد یک سرور و ارسال و دریافت داده ها به کلاینتها



Server Class

-

ایجاد یک کلاینت و ارسال و دریافت داده ها به سرور



Client Class

-

Ethernet کلاس

عنوان خروجی دو مقدار صفر و یک را بر می گرداندو زمانی که مقدار یک است یعنی اینکه باموفقیت به سرور DHCP وصل شده ایم و IP اختصاص داده شده است و نیاز به تنظیم دستی IP نیست.

begin()

➤ مقدار دهی اولیه کتابخانه اترنت و تنظیمات شبکه

زمانیکه IP از طریق DHCP اختصاص داده می شود بکار می رود.

localIP()

➤ آدرس IP اختصاصی را به دست می آورد

دارای مقادیر خروجی • (هیچ اتفاقی نیفتاده) و ۱ (IP اختصاص داده نشده) و ۲ (IP جدید اعمال شده) و ۳ (تا زمانیکه IP اختصاص داده نشده سعی میکند IP اختصاص بدهد ولی نمیشود) و ۴ (در دفعات بعدی توانسته IP را اختصاص بدهد)

maintain()

➤ اختصاص مجدد IP از طریق سرور DHCP

Syntax

Ethernet.begin(mac);

Ethernet.begin(mac, ip);  اختصاص IP به شکل دستی

Ethernet.begin(mac, ip, dns);

Ethernet.begin(mac, ip, dns, gateway);

Ethernet.begin(mac, ip, dns, gateway, subnet);

متدهای کلاس IPAddress

IPAddress()

➤ برای اختصاص آدرس به صورت دستی

متدهای کلاس Server

- **Server**
- یک کلاس پایه برای کلیه متدهای مربوط به **Ethernet server**
- **EthernetServer()**
- یک سرور ایجاد میکند که به اتصال های دریافتی بر روی یک پورت مشخص گوش میکند
- **begin()**
- شروع به گوش کردن برای اتصالات ورودی بر روی یک پورت

متدهای کلاس Server

available()

کلاینتی که به سرور متصل شده و داده های برای خواندن دارد را به دست می آورد.

می تواند دارای پaramترهای buffer(length) و value باشد ← write()

داده ها را بر روی تمامی کلاینت های متصل شده به سرور، رایت میکند.

print() - println()

داده ها را بر روی تمامی کلاینت های متصل شده به سرور، مینویسد

متدهای Client

- **Client**
- یک کلاس پایه برای کلیه متدهای مربوط به **Ethernet Client** می باشد.
- **EthernetClient()**
- یک کلاینت ایجاد میکند که میتواند به یک آدرس IP و پورت مشخص متصل شود.
- **if (EthernetClient)**
- اگر کلاینت مشخصی آماده باشد. آن را نشان میدهد.

متدهای Client

- **connected()** ➤ وضعیت یک کلاینت را مشخص میکند که آیا متصل است یا خیر؟
- **connect()** ➤ کلاینت به یک آدرس و پورت مشخص متصل می شود.
- **write()** ➤ داده های کلاینت متصل شده به سرور، در سرور نوشته می شوند.

Syntax

Client.connect()

Client.connect(ip, port)

Client.connect(URL, port)

متدهای کلاس client

print() •

- داده های کلاینت متصل شده به سرور، در سرور نوشته می شوند.

println() •

- دقیقا مثل متدهای بالا است؛ با این تفاوت که بعد از ارسال به خط بعدی میروند

available() •

- تعداد بایت های موجود برای خواندن را برمی گرداند.

متدهای کلاس client

- **read()**
 - این متده، بایت بعدی دریافت شده از سرور را می خواند.
- **flush()**
 - منتظر می ماند تا همه کاراکترها (بایت های) خروجی موجود در بافر فرستاده شوند.
- **stop()**
 - منجر به قطع اتصال از سرور می شود.

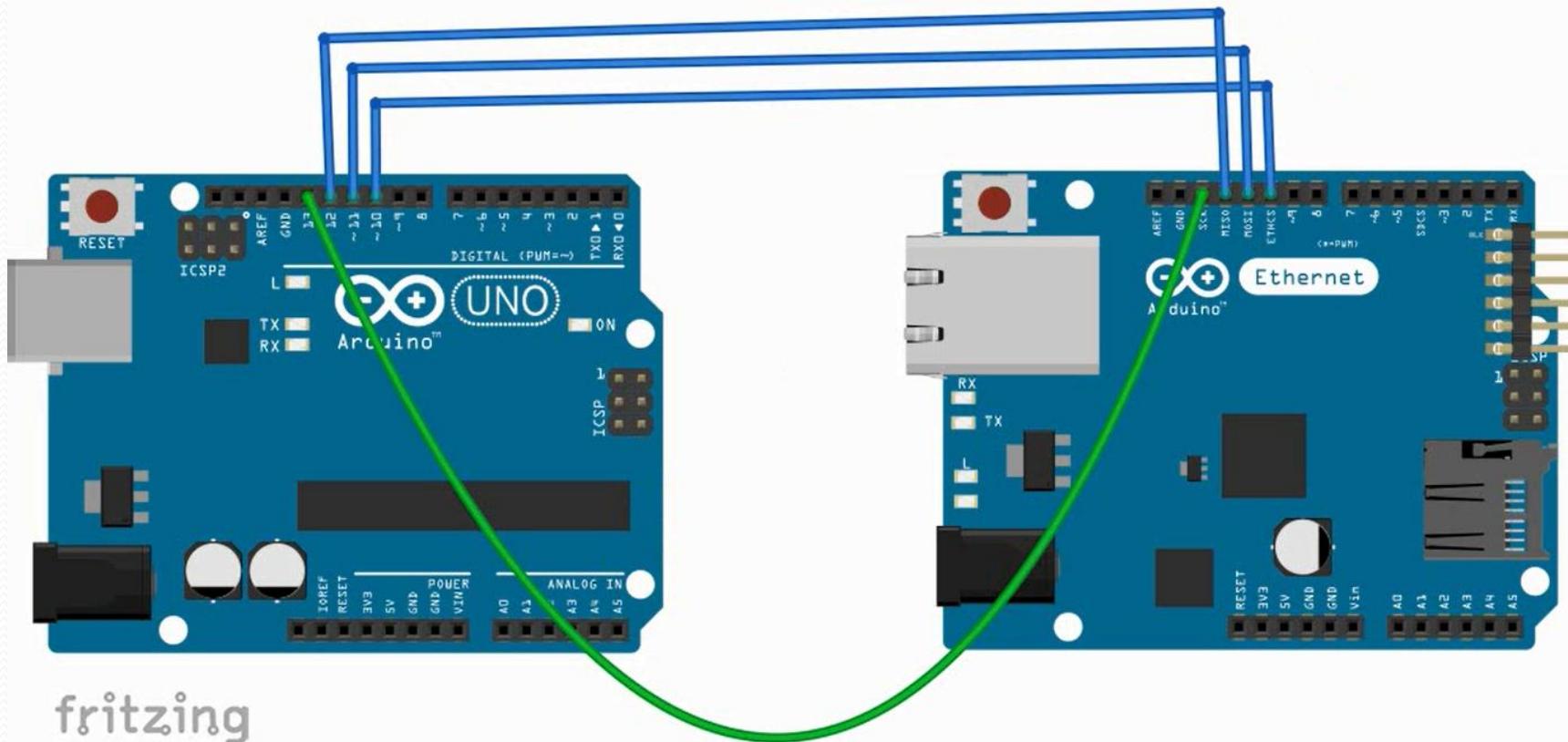
کتابخانه Ethernet در برد آردنو

کلاس	متدها	توضیحات
Ethernet class	<code>begin()</code>	برای مقدار دهی اولیه کتابخانه اترنت و تنظیمات شبکه مورد استفاده قرار میگیرد.
	<code>localIP()</code>	آدرس IP اختصاصی را به دست می آورد، زمانی مفید می باشد که IP از طریق DHCP اختصاص می یابد
	<code>maintain()</code>	برای اختصاص مجدد IP از طریق سرور DHCP مورد استفاده قرار میگیرد.
Server class	<code>IPAddress()</code>	برای اختصاص IP آدرس به صورت دستی (استاتیک) مورد استفاده قرار میگیرد
	<code>Server</code>	سرور (Server)، یک کلاس پایه برای کلیه متدهای مربوط به Ethernet server می باشد. این کلاس به صورت مستقیم مورد استفاده قرار نمیگیرد، ولی برای متدهای وابسته به آن استفاده می شود.
	<code>EthernetServer()</code>	این متدها، یک سرور ایجاد میکند که به اتصال های دریافتی بر روی یک پورت مشخص گوش میکند.
	<code>begin()</code>	این متدها، به سرور میگوید شروع به گوش کردن برای اتصالات ورودی بر روی یک پورت مشخص بکند.
	<code>available()</code>	این متدها، کلاینتی که به سرور متصل شده و داده های (اطلاعاتی) برای خواندن دارد را به دست می آورد، این ارتباط همچنان ادامه می یابد تا زمانیکه کلاینت از محدوده خارج می شود (ارتباط را ترک میکند)
	<code>write()</code>	این متدها را بر روی تمامی کلاینت های متصل شده به سرور، رایت میکند؛ این داده ها به صورت یک بایت و یا مجموعه ای از بایت ها ارسال می شوند.
	<code>print()</code>	این متدها را بر روی تمامی کلاینت های متصل شده به سرور، رایت میکند؛ این داده ها به صورت دنباله هایی از اعداد فرستاده می شوند. (مثلاً به صورت کاراکترهای اسکی)
	<code>println()</code>	این متدها دقیقاً مثل متدهای بالاست؛ با این تفاوت که بعد از ارسال به خط بعدی میروند (New Line)

Client class

کلاینت (Client) ، یک کلاس پایه برای کلیه متدهای مربوط به Ethernet Client می باشد. این کلاس به صورت مستقیم مورد استفاده قرار نمیگیرد، ولی برای متدهای وابسته به آن استفاده می شود.	Client
این متدها، یک کلاینت ایجاد میکند که میتواند به یک آدرس IP و پورت مشخص متصل شود. (اتصال توسط متدهای Connect() و Disconnect())	EthernetClient()
اگر کلاینت مشخصی آماده باشد. آن را نشان میدهد (اگر کلاینت در دسترس باشد True برگردانده خواهد شد).	if (EthernetClient)
این متدها، وضعیت یک کلاینت را مشخص میکند که آیا متصل است یا خیر؟ (نکته مهم اینکه، اگر اتصال یک کلاینت بسته شود ولی داده آن هنوز خوانده نشده باشد، در آن صورت کلاینت متصل است)	connected()
توسط این متدها، کلاینت به یک آدرس و پورت مشخص متصل می شود. و با توجه به خروجی متدهای مذکور، وضعیت موفقیت آمیز بودن اتصال، مشخص میشود.	connect()
توسط این متدها، داده های کلاینت متصل شده به سرور، در سرور نوشته می شوند. این داده ها به صورت یک بایت یا مجموعه ای از بایت ها نوشته می شوند.	write()
توسط این متدها، داده های کلاینت متصل شده به سرور، در سرور نوشته می شوند. این داده ها به صورت دنباله هایی از اعداد فرستاده می شوند.	print()
این متدها، دقیقاً مثل متدهای بالاست؛ با این تفاوت که بعد از ارسال به خط بعدی میروند (New Line)	println()
این متدها، تعداد بایت های موجود برای خواندن را بر می گردانند. (در واقع مقدار داده هایی که توسط سرور در کلاینت نوشته می شوند را بر میگردانند.)	available()
این متدها، بایت بعدی دریافت شده از سرور را (توسط کلاینتی که به آن متصل است)، می خواند.	read()
این متدها، منتظر می مانند تا همه کاراکترها (بایت های) خروجی موجود در بافر فرستاده شوند. (در واقع توسط این متدها بافر تخلیه می شود).	flush()
این متدها، منجر به قطع اتصال از سرور می شود.	stop()

اتصال شیلد اترنت به آردوینو



fritzing



برنامه نویسی مازول شبکه

```
#include<SPI.h>
#include<Ethernet.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192,168,1,10);
IPAddress myDns(192,168,1,1);
IPAddress gateway(192,168,1,1);
 IPAddress subnet(255,255,255,0);
```

یک **server** را مشخص میکنیم تا از پورت ۲۳ بخواند

EthernetServer server(23);  تعین میکند که آیا کلاینت به سرور متصل شده است یا خیر؟

boolean check = false;  چون دفعه اول هنوز متصل نشده، مقدارش را **false** می دهیم.

```
void setup() {
  Ethernet.begin(mac, ip, myDns, gateway, subnet);
  server.begin();
  Serial.begin(9600);
```

تا زمانیکه هیچ ارتباط سریالی را شروع نکردیم داخل حلقه می مانیم 

```
while(!Serial)
{
  ;
}
```

برنامه نویسی مازول شبکه ...

```
}
```

```
Serial.print("Address :");
```

نمايش IP اختصاص داده شده به کاربر →

```
}
```

```
void loop() {
```

یک کلاینت به یک آدرس و پورت مشخص متصل می شود ;
حال می خواهیم بینیم که آیا اطلاعاتی برای خواندن دارد یا خیر ؟

اگر کلاینتی وجود داشت →

```
{
```

اگر دفعه اول بود که کلاینت متصل شده بود →

```
{
```

تخلیه بافر →

```
Serial.println("New Client...");
```

```
client.println("Hello....");
```

مقدار را True میکنیم تا دفعه بعد وارد این قسمت نشود →

```
}
```

```
}
```

برنامه نویسی مازول شبکه ...

```
if(client.available()>0)
{
    char data = client.read();
    server.write(data);
    Serial.write(data);
}

}
```

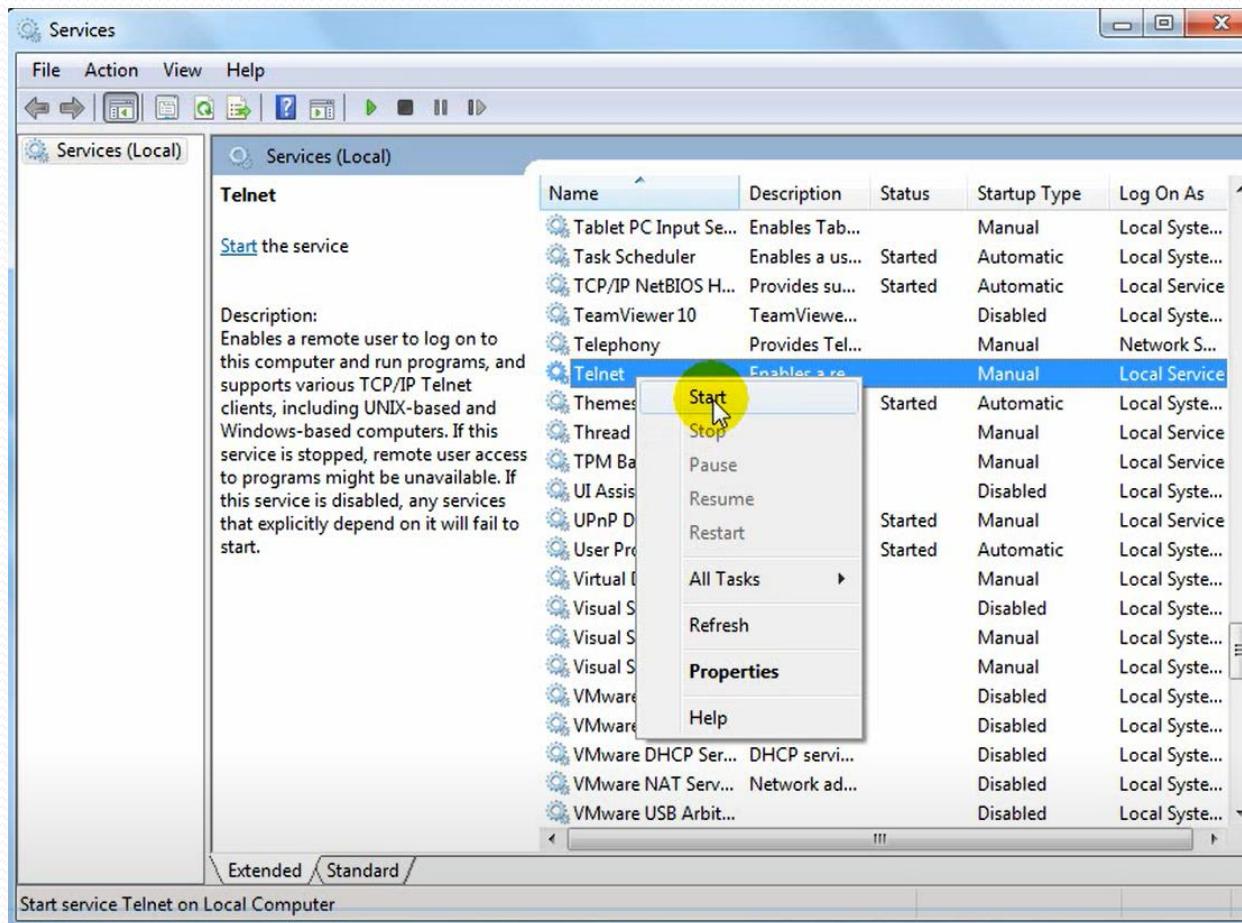
Ping کردن w5100 توسط رایانه

- Ping 192.168.1.10

فعال کردن سرویس telnet client در ویندوز از طریق control panel



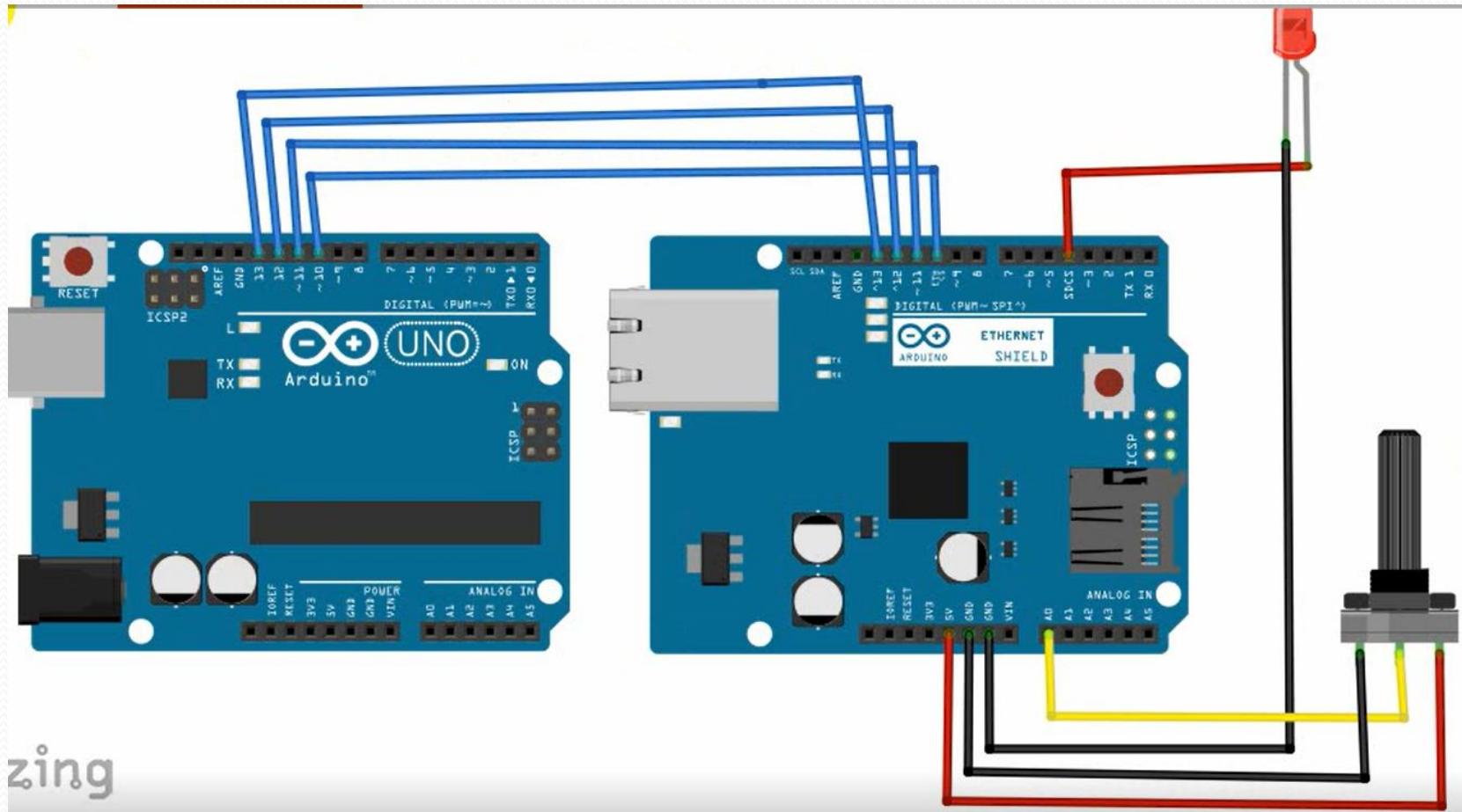
فعال کردن سرویس telnet با دستور services.msc



- حال میتوان از طریق کامپیوتر به آردوینو telnet زد.

مفهوم web service

- برنامه ها یا قطعه کدهایی هستند که از طریق وب قابل دسترسی هستند و از طریق پروتکلهایی مثل http به انتقال و مبادله داده ها با سایر برنامه های کاربردی می پردازند.
- اساس وب سرویسها بر پایه دریافت و تولید پیام است.
- به سیستم عاملها و زبانهای برنامه نویسی خاصی وابسته نیستند.
- سهولت در برقراری ارتباط با سایر برنامه ها



```
1 #include <SPI.h>
2 #include <Ethernet.h>
3
4 byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
5 byte ip[] = {192,168,1,10};
6 byte gateway[] = {192,168,1,1};
7 byte subnet[] = {255,255,255,0};
8
9 int LED = 4;
10 int POT = A0;
11 int reading;
12
13 //-----
14 EthernetServer server(80);
15 String readString;
16 //-----
```

I

```
18 void setup() {  
19     Serial.begin(9600);  
20     pinMode(LED, OUTPUT);  
21  
22     while(!Serial) {  
23         ;  
24     }  
25  
26     Ethernet.begin(mac, ip, gateway, subnet);  
27     server.begin();  
28     Serial.println("IP Address Is :");  
29     Serial.println(Ethernet.localIP());  
30 }  
31
```

```
32 void loop() {  
33     EthernetClient client = server.available();  
34     if (client)  
35     {  
36         while(client.connected())  
37     {  
38         if (client.available())  
39         {  
40             char c = client.read();  
41             //-----  
42             if ()  
43             //-----  
44             readString += c;  
45             Serial.print(c);
```