

arduino 上使用 ENC28J60 以太网控制器的教程—ping 通你的 Arduino

Arduino 的官方以太网盾是基于 Wiznet W5100 芯片, Arduino IDE 的 [Ethernet 库](#)也是针对这个芯片。

一种广泛使用的以太网控制器是 Microchip 的 ENC28J60, 在互联网上你可以找到基于该芯片设计的盾或模块, 通常比官方的以太网盾便宜。

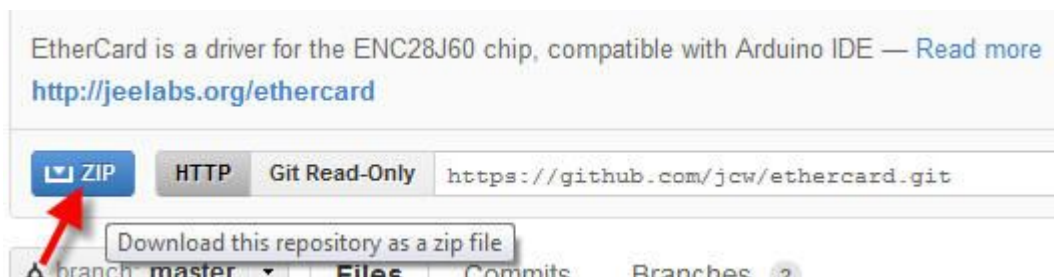


Testato 建议我说明这两个控制器之间的差异: 主要的不同, 之所以 Arduino 的团队选择了 W5100 芯片, 是该芯片集成了成熟且经过验证的 TCP/IP 协议栈, 节省了 MCU 资源。

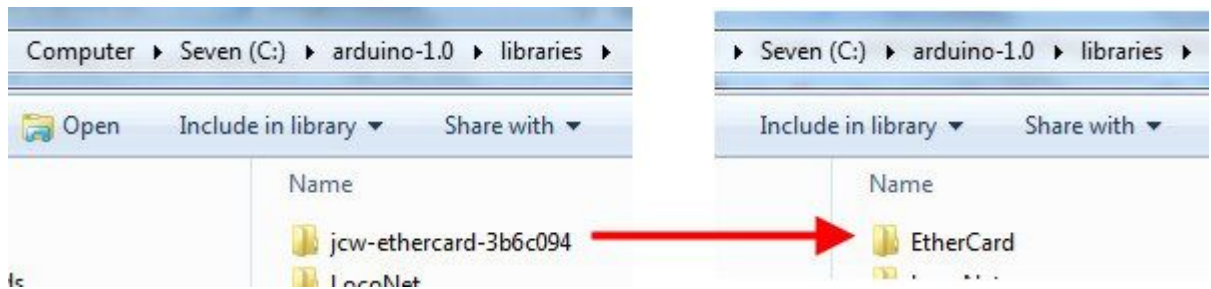
文库

很多公司和玩家开发了 Arduino 的 Microchip enc28j60 的库, 我这些教程使用的库是 [Jeelabs Café](#)编写的 **EtherCard** 库。

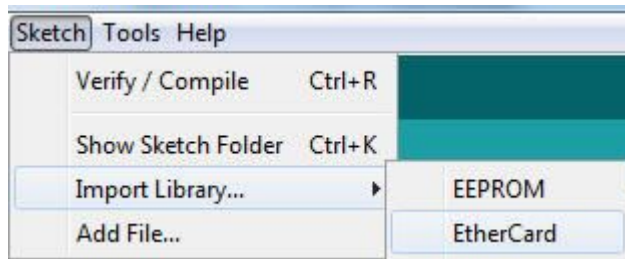
Jeelabs Café把它开源在了 [GitHub](#)上, 先下载一个 zip 文件的最新版本:



打开 Arduino IDE 安装主文件夹 (比如 D:\arduino-1.0.1\), 解压缩下载的文件到 **libraries** 子文件夹。在 ZIP 压缩包中, 所有的库文件都在一个名为 "jcw-ethercard- xxx" 的文件夹中, 重命名该文件夹为 "EtherCard":



运行 IDE，检查库是不是已经加载：



网络参数

在把网络控制器连接到网络之前，你需要了解一些参数：

MAC 地址又叫物理地址，是由 48 比特长，12 位的 16 进制数字组成，0 到 23 位是厂商向 IETF 等机构申请用来标识厂商的代码，也称为“编制上唯一的标识符”（Organizationally Unique Identifier）。是识别 LAN（局域网）结点的标志。地址的 24 到 47 位由厂商自行分派，是各个厂商制造的所有网卡的一个唯一编号；

IP 地址就是给每个连接在 Internet 上的主机分配的一个 32bit 地址。按照 TCP/IP 协议规定，IP 地址用二进制来表示，每个 IP 地址长 32bit，比特换算成字节，就是 4 个字节。例如一个采用二进制形式的 IP 地址是“00001010000000000000000000000001”，这么长的地址，人们处理起来也太费劲了。为了方便人们的使用，IP 地址经常被写成十进制的形式，中间使用符号“.”分开不同的字节。于是，上面的 IP 地址可以表示为“10.0.0.1”。IP 地址的这种表示法叫做“点分十进制表示法”，这显然比 1 和 0 容易记忆得多，它用于在本地网络中识别设备；

子网掩码是一个 32 位地址，是与 IP 地址结合使用的一种技术。它的主要作用有两个，一是用于屏蔽 IP 地址的一部分以区别网络标识和主机标识，并说明该 IP 地址是在局域网，还是在远程网上。二是用于将一个大的 IP 网络划分为若干小的子网络。子网掩码通畅也用“点分十进制表示法”。

网关实质上是一个网络通向其他网络的 IP 地址，它通常是分配给一个可以到达不同网络的设备（路由器等）。

你需要编一个与你所在的网络其他设备不同的 MAC 地址，在这个例子中，我们通常会选择一个尚未分配的组织机构代码（如：DD-DD-DD）开头的 MAC 地址。

网络控制器的网络参数（地址，子网掩码，网关）与您的本地网络中的其他设备的配置相比：IP 地址应该是唯一的，子网掩码和网关应该是相同的。

如果你的网络中有 DHCP 服务器，它会自动对新设备的网络参数进行配置。

ping !

检查一个设备是否正确联网，最简单的方法是 ping 它。在一个联网的计算机上输入 ping 你要查看设备的 IP 地址

让我们来编写一个简单例子回答 ping 请求：

ARDUINO 代码

```
#include <EtherCard.h>
static byte mymac[] = {0x74,0x69,0x69,0x2D,0x30,0x31};
static byte myip[] = {192,168,1,10};
byte Ethernet::buffer[700];

void setup () {
    Serial.begin(57600);
    Serial.println("PING Demo");

    if (ether.begin(sizeof Ethernet::buffer, mymac, 10) == 0)
        Serial.println( "Failed to access Ethernet controller");

    if (!ether.staticSetup(myip))
        Serial.println("Failed to set IP address");
}

void loop() {
    ether.packetLoop(ether.packetReceive());
}
```

说明：

```
#include <EtherCard.h>
static byte mymac[] = {0x74,0x69,0x69,0x2D,0x30,0x31};
static byte myip[] = {192,168,1,10};
byte Ethernet::buffer[700];
```

首先，你需要包括以 EtherCard 库，并定义一些变量：MAC 地址(mymac[])，IP 地址(myip[])和用来存储传入和传出的数据缓冲（Ethernet::buffer[700]）。

ether.begin(sizeof Ethernet::buffer, mymac, 10)

用 begin()方法开始网络连接，需要 3 个参数，分别为缓冲大小、MAC 地址和 Arduino 的片选（CS）引脚（通常为 10，这个参数可以不写，如果不写的话默认为 8（2560 默认也是 8），所以要根据你的电路进行设置）。

顺便说下接线，特别是模块的，很多同学容易弄错，必要接的 7 根，Vcc 接 3.3V（电路接好通电后，一定要用万用表量一下，至少要 3V 以上）；CS 根据程序接；SI 接 D11；SO 接 D12；SCK 接 D13；RESET 接 RESET；GND 接 GND。

arduino 2560 接线：Vcc 接 3.3V（电路接好通电后，一定要用万用表量一下，至少要 3V

以上)；cs 根据程序接；SI 接 D51；SO 接 D50；SCK 接 D52；RESET 接 RESET；GND 接 GND。有些 2560 可能 3.3V 电压不够，如果调试 Vcc 可接 5V 电压（建议不要长时间运行，芯片会比较热）。

```
ether.staticSetup(myip)
```

用 `staticSetup()` 方法配置静态的 IP 地址，参数有 3 个，分别为 ip 地址、网关和 DNS，IP 地址是必须的，网关和 DNS 是可选的。大家看下这个函数的参数定义。

```
static bool staticSetup (const uint8_t* my_ip =0,  
                        const uint8_t* gw_ip =0,  
                        const uint8_t* dns_ip =0);
```

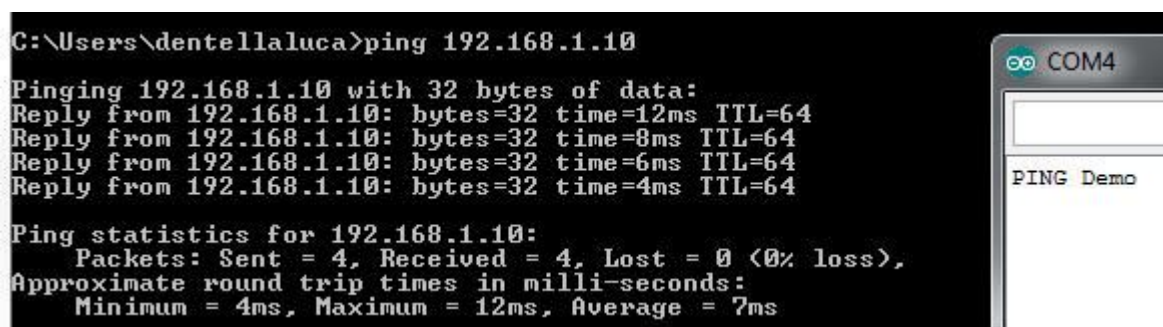
```
ether.packetLoop(ether.packetReceive());
```

in the loop, 你只需要 2 条命令：

`packetReceive()` 方法： 从网络接收一个新传入的数据包；

`packetLoop()` 方法： 对具体收到的信息作出回应，包含“ping”请求(ICMP echo 请求)。

现在你的 arduino 联网了：



```
C:\Users\dentellaluca>ping 192.168.1.10  
  
Pinging 192.168.1.10 with 32 bytes of data:  
Reply from 192.168.1.10: bytes=32 time=12ms TTL=64  
Reply from 192.168.1.10: bytes=32 time=8ms TTL=64  
Reply from 192.168.1.10: bytes=32 time=6ms TTL=64  
Reply from 192.168.1.10: bytes=32 time=4ms TTL=64  
  
Ping statistics for 192.168.1.10:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
Approximate round trip times in milli-seconds:  
    Minimum = 4ms, Maximum = 12ms, Average = 7ms
```