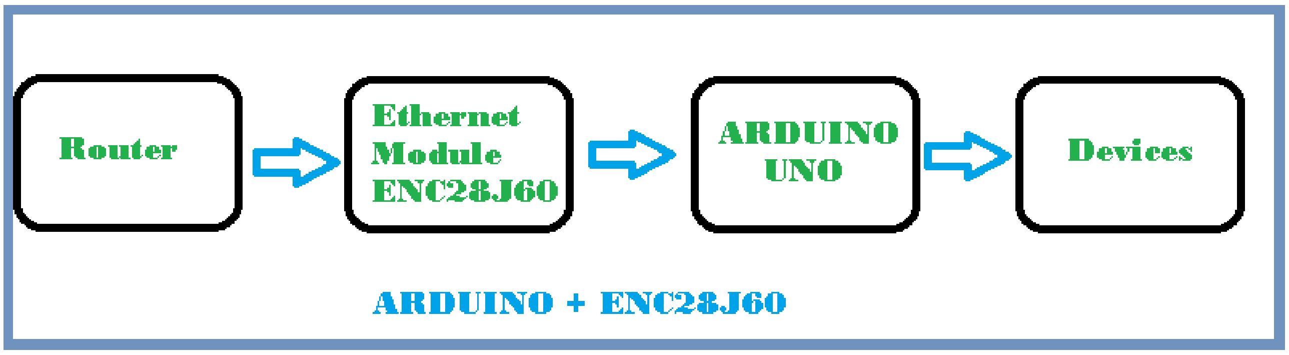
Lab-08 Ethernet(ENC28J60) Interfaced Arduino

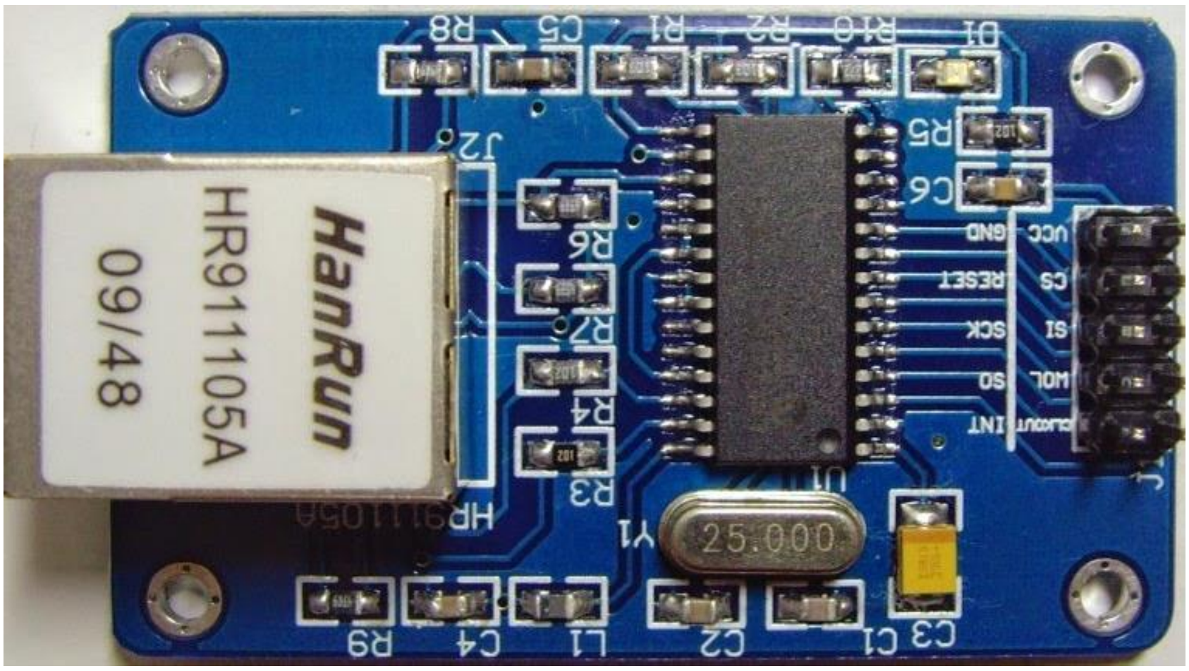
1. **Objectives: After this lab, the learner will be able to:**
   - Interface SD module with Arduino board
   - Interface
2. **Facilities**
   - Proteus software
   - Arduino IDE
3. **Prerequest**
   - Basic electronic
   - C programming
4. **Introduction**

In this Artical we will see how to establish communication between computer/mobile to Arduino in a LAN or Wireless Network. We are interfacing the ENC28J60 Ethernet controller to Arduino so that our Arduino will be the one of member of that network. Once it will done the arduino can control things in that network or the arduino can be controlled by the other devices of that network it might be computer or mobile. Interfacing Arduino with the ENC28J60 will further gives us the freedom of using the internet on Arduino but before that we need to learn how to do all that networking stuff in a locan network.

Soon I will post about how to control the things in local network, how to send data on a local network, and how to interfaced with Android devices or how to make IOT Devices. IOT Devices are quite famous.
A simple block diagram of the data flow is in the image below.



Ethernet Module(ENC28J60): Microchip's ENC28J60 is a 28-pin, Ethernet Controller with on board MAC & PHY, 8 Kbytes of Buffer RAM and an SPI serial interface. With a small foot print package size the ENC28J60 minimizes complexity, board space and cost. It used in so many application like Industrial Automation, Building Automation, Home Control, Security and Instrumentation, IOT Devices.
You can build Your own circuit but I use the Module to save my time.



VIRTUAL SIMULATION: I first simulate my design on the Proteus ISIS to make sure I am working in right direction and also to save my time. In ISIS their is already a component named "ENC28J60 ethernet controller" so I picked it from the library. Now I picked up the Arduino and connect it with the ethernet controller.

Network Setup

Before building your circuit and starting the simulation, you need to set up some necessary components on your Windows system. This includes installing a packet capture library and a virtual network driver.

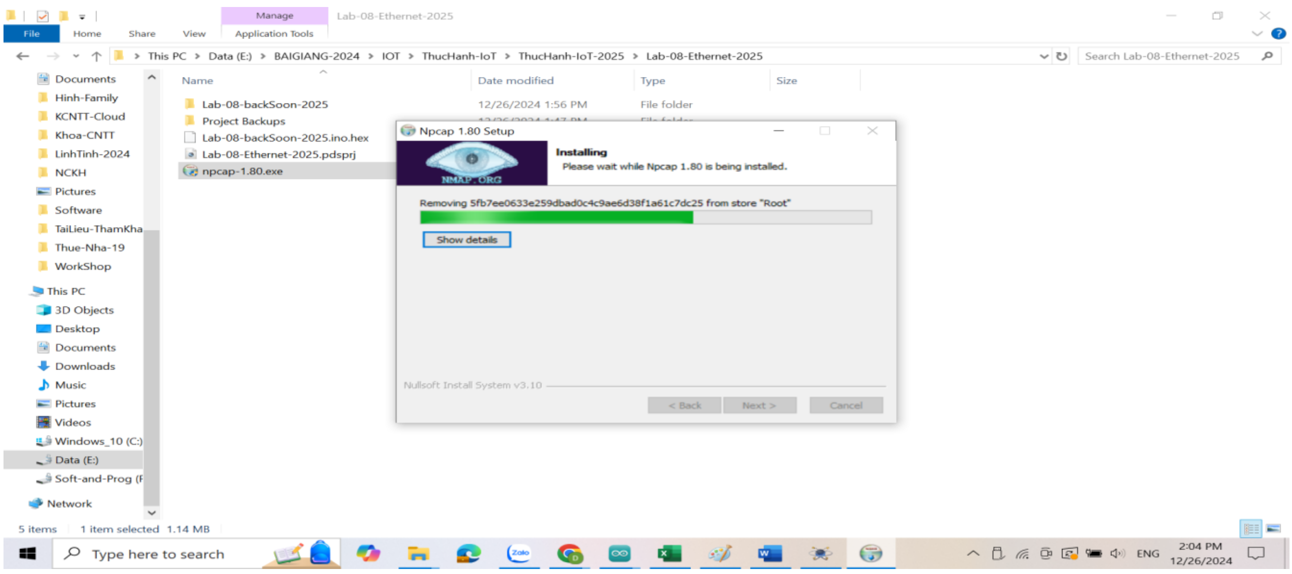
Packet capture library

Proteus recommends installing the WinPcap library, but it is no longer actively maintained. Therefore, you should use Npcap instead.

What is Npcap?

Npcap is a packet sniffing library and driver for Windows operating systems. It is an open-source project based on the well-known WinPcap library but with additional features and improvements. Npcap allows users to capture and analyze network traffic on Windows machines. It provides support for various network protocols and offers features like loopback traffic capturing, raw 802.11 packet capturing, and kernel-level packet filtering.

You can download this package from the **Npcap website**, and the installation process is straightforward. After agreeing to their license agreement, it provides a one-click install.


Npcap 1.80 Setup

Installing

Please wait while Npcap 1.80 is being installed.

Removing 5fb7ee0633e259dbad0c4c9ae6d38f1a61c7dc25 from store "Root"

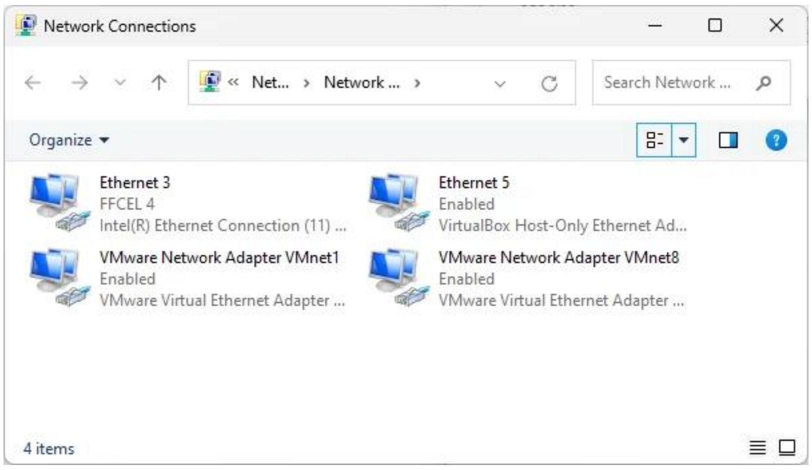
Name	Date modified	Type	Size
Lab-08-backSoon-2025	12/26/2024 1:56 PM	File folder	
Project Backups			
Lab-08-backSoon-2025.ino.hex			
Lab-08-Ethernet-2025.pdsprj			
npcap-1.80.exe			

5 items 1 item selected 1.14 MB

Virtual Network Driver

Suppose your PC doesn't have a network card, or if it does, it's already in use, or consider you are using a laptop that doesn't have an Ethernet connection. In this case, you can use a virtual network driver, which can provide you with a dummy network adapter. This allows you to easily simulate the ENC28J60 Ethernet Module in Proteus.

There are multiple ways to obtain a virtual network adapter. If you're using VMWare Workstation or VirtualBox software, you already have a virtual network adapter in your system, which you can utilize. Alternatively, if you're not using either of these, you can utilize the Microsoft KM-Test Loopback adapter, accessible through your Computer Management settings.


Network Connections

Ethernet 3
FFCEL 4
Intel(R) Ethernet Connection (11) ...

Ethernet 5
Enabled
VirtualBox Host-Only Ethernet Ad...

VMware Network Adapter VMnet1
Enabled
VMware Virtual Ethernet Adapter ...

VMware Network Adapter VMnet8
Enabled
VMware Virtual Ethernet Adapter ...

4 items

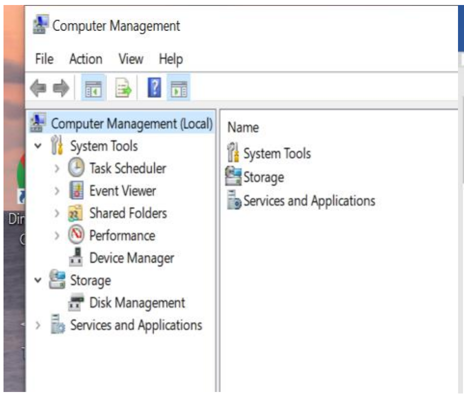
Network Adapters

As you can observe in the image above, Ethernet 3 represents my default PC network card adapter, while Ethernet 5 and others are provided by VirtualBox and VMWare Workstation software, respectively.

How to install Microsoft Loopback Adapter

Here is the installation method for the Microsoft Loopback Adapter.

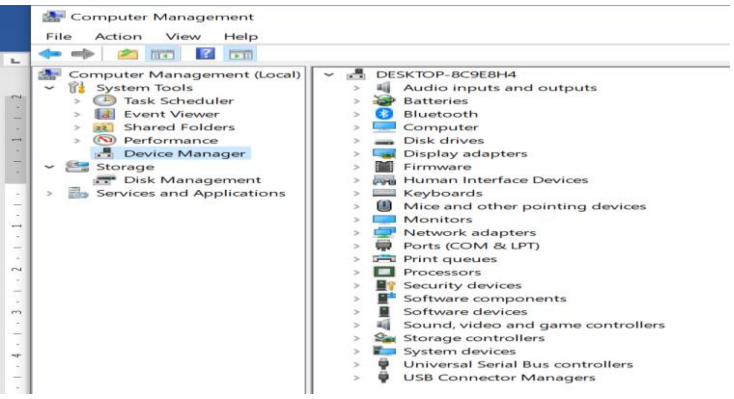
Loopback Adapter Installation


Computer Management

File Action View Help

Computer Management (Local)
- System Tools
  - Task Scheduler
  - Event Viewer
  - Shared Folders
  - Performance
  - Device Manager
- Storage
  - Disk Management
- Services and Applications

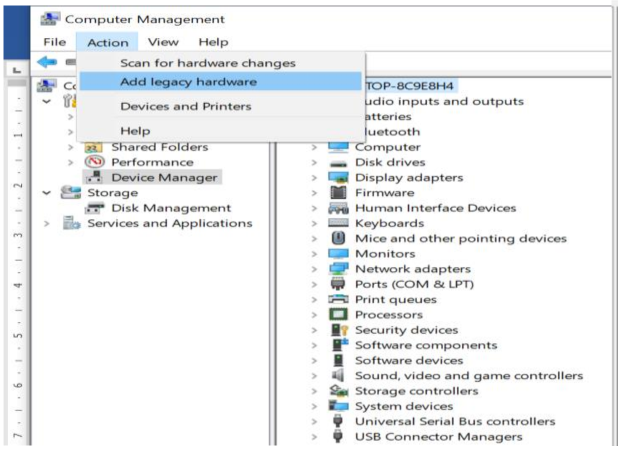
Name
- System Tools
- Storage
- Services and Applications


Computer Management

File Action View Help

Computer Management (Local)
- System Tools
  - Task Scheduler
  - Event Viewer
  - Shared Folders
  - Performance
  - Device Manager
- Storage
  - Disk Management
- Services and Applications

DESKTOP-8C9E8H4
- Audio inputs and outputs
- Batteries
- Bluetooth
- Computer
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Print queues
- Processors
- Security devices
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers
- USB Connector Managers

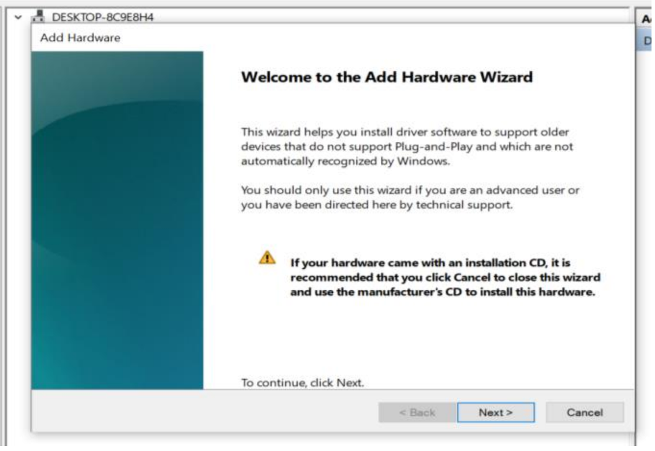

Computer Management

File Action View Help

Action menu:
- Scan for hardware changes
- Add legacy hardware
- Devices and Printers
- Help

- Shared Folders
- Performance
- Device Manager
- Storage
  - Disk Management
- Services and Applications

TOP-8C9E8H4
- udio inputs and outputs
- atteries
- luetooth
- Computer
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Print queues
- Processors
- Security devices
- Software components
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers
- USB Connector Managers


DESKTOP-8C9E8H4

Add Hardware

Welcome to the Add Hardware Wizard

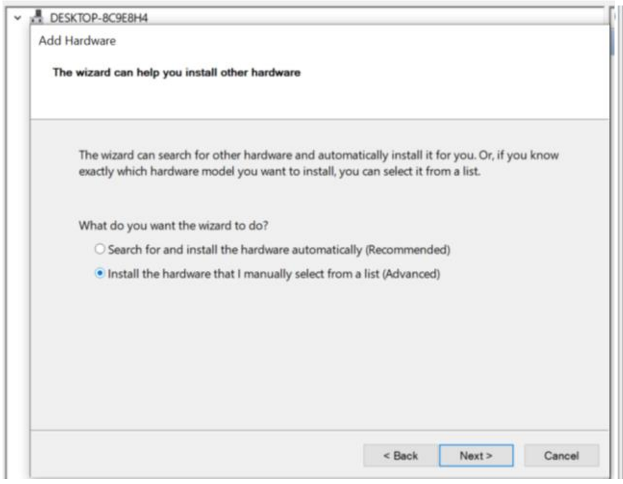
This wizard helps you install driver software to support older devices that do not support Plug-and-Play and which are not automatically recognized by Windows.

You should only use this wizard if you are an advanced user or you have been directed here by technical support.

If your hardware came with an installation CD, it is recommended that you click Cancel to close this wizard and use the manufacturer's CD to install this hardware.

To continue, click Next.

< Back | Next > | Cancel


DESKTOP-8C9E8H4

Add Hardware

The wizard can help you install other hardware

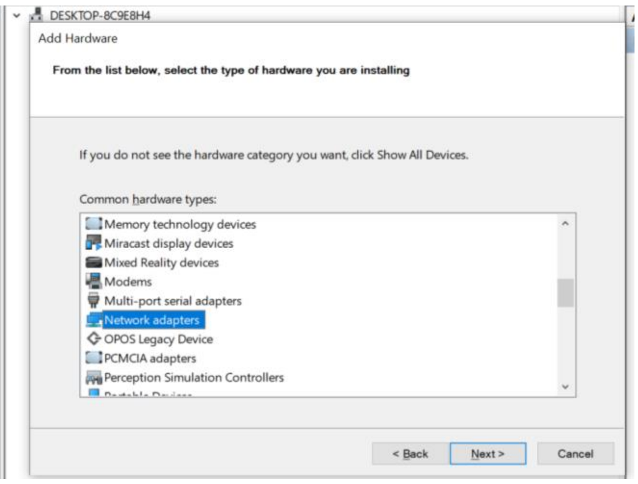
The wizard can search for other hardware and automatically install it for you. Or, if you know exactly which hardware model you want to install, you can select it from a list.

What do you want the wizard to do?

○ Search for and install the hardware automatically (Recommended)

[✓ Install the hardware that I manually select from a list (Advanced)]

< Back | Next > | Cancel


DESKTOP-8C9E8H4

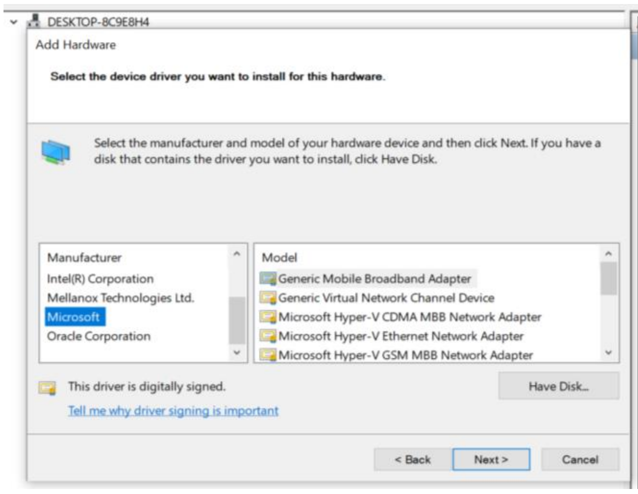
Add Hardware

From the list below, select the type of hardware you are installing

If you do not see the hardware category you want, click Show All Devices.

Common hardware types:
- Memory technology devices
- Miracast display devices
- Mixed Reality devices
- Modems
- Multi-port serial adapters
- Network adapters
- OPOS Legacy Device
- PCMCIA adapters
- Perception Simulation Controllers

< Back | Next > | Cancel


DESKTOP-8C9E8H4

Add Hardware

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

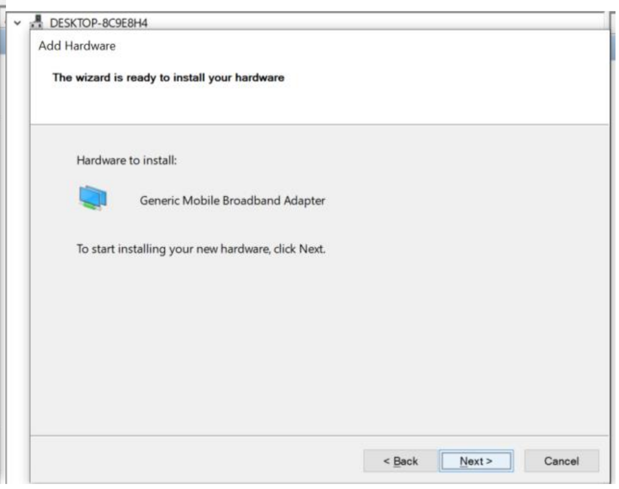
Manufacturer	Model
Intel(R) Corporation	Generic Mobile Broadband Adapter
Mellanox Technologies Ltd.	Generic Virtual Network Channel Device
Microsoft	Microsoft Hyper-V CDMA MBB Network Adapter
Oracle Corporation	Microsoft Hyper-V Ethernet Network Adapter
	Microsoft Hyper-V GSM MBB Network Adapter

This driver is digitally signed.

Tell me why driver signing is important

Have Disk...

< Back | Next > | Cancel


DESKTOP-8C9E8H4

Add Hardware

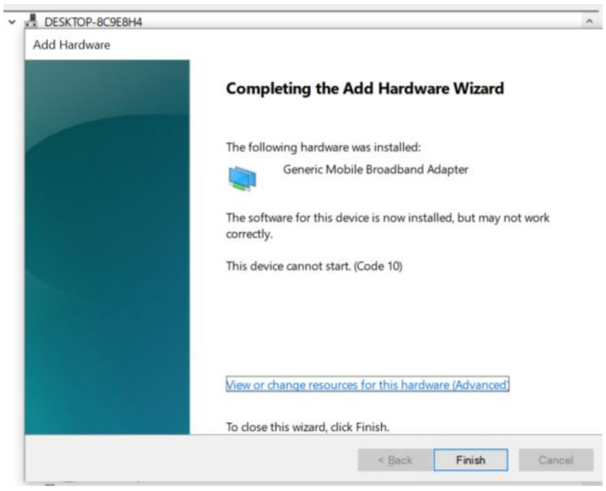
The wizard is ready to install your hardware

Hardware to install:

Generic Mobile Broadband Adapter

To start installing your new hardware, click Next.

< Back | Next > | Cancel



DESKTOP-8C9E8H4

Add Hardware

Completing the Add Hardware Wizard

The following hardware was installed:

Generic Mobile Broadband Adapter

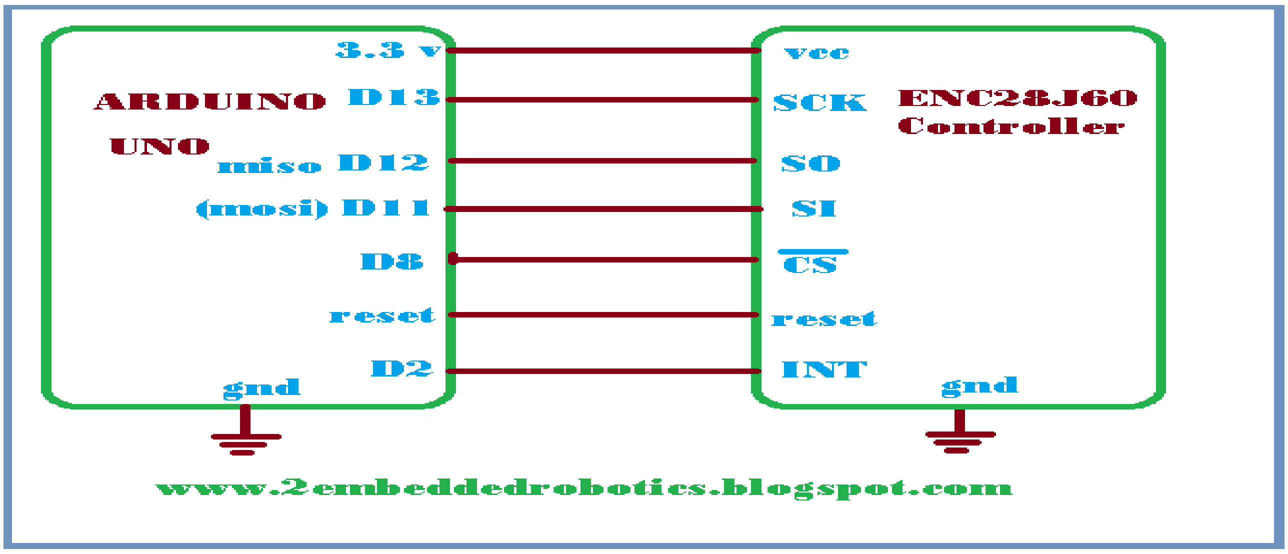
The software for this device is now installed, but may not work correctly.

This device cannot start. (Code 10)

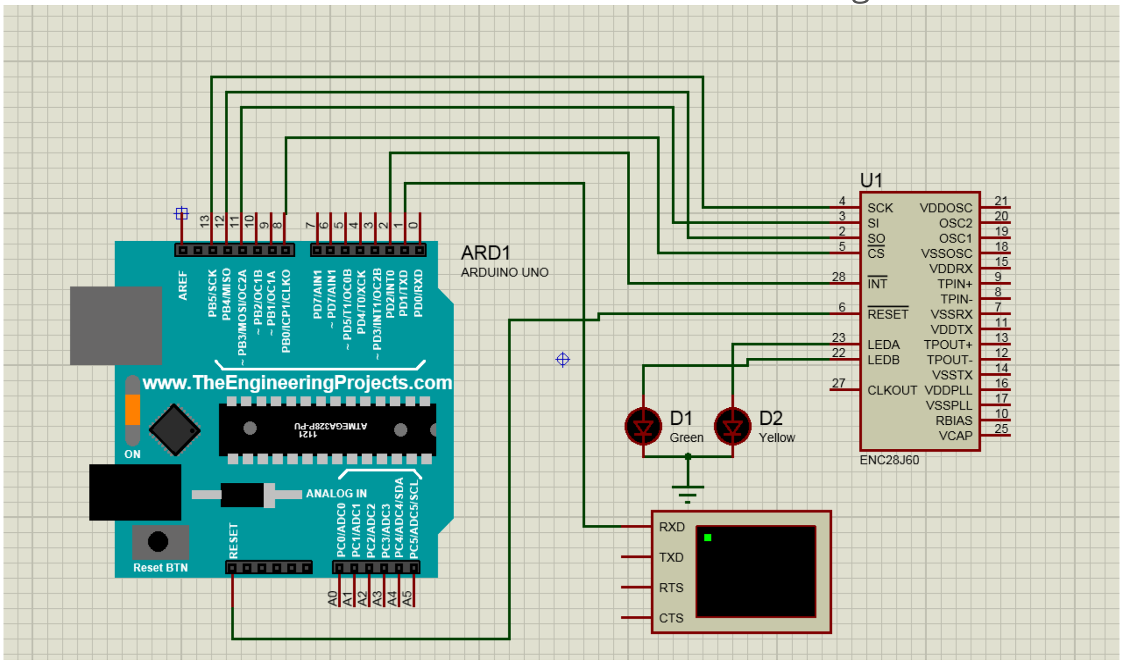
View or change resources for this hardware (Advanced)

To close this wizard, click Finish.

Now, you can choose whichever method suits you best.

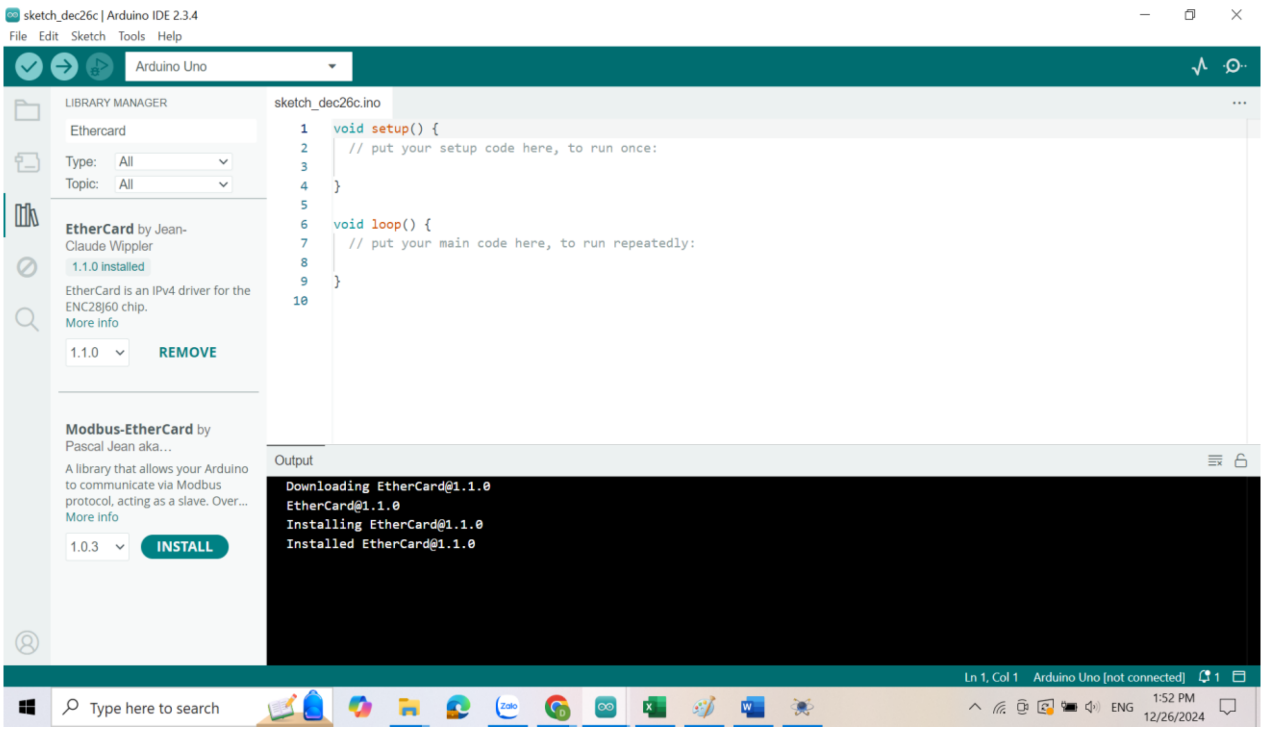


Now make these connection in the ISIS design tool.



SOFTWARE: Now we all done with the Virtual Hardware Design lets make the software. All you need is the Ethercard Library for this example. Download it from here [EtherCard. Click here]

- Extract the ZIP file, copy and paste the folder in your Arduino1.xx/library/



EtherCard by Jean-Claude Wippler
1.1.0 installed
EtherCard is an IPv4 driver for the ENC28J60 chip.
More info
1.1.0

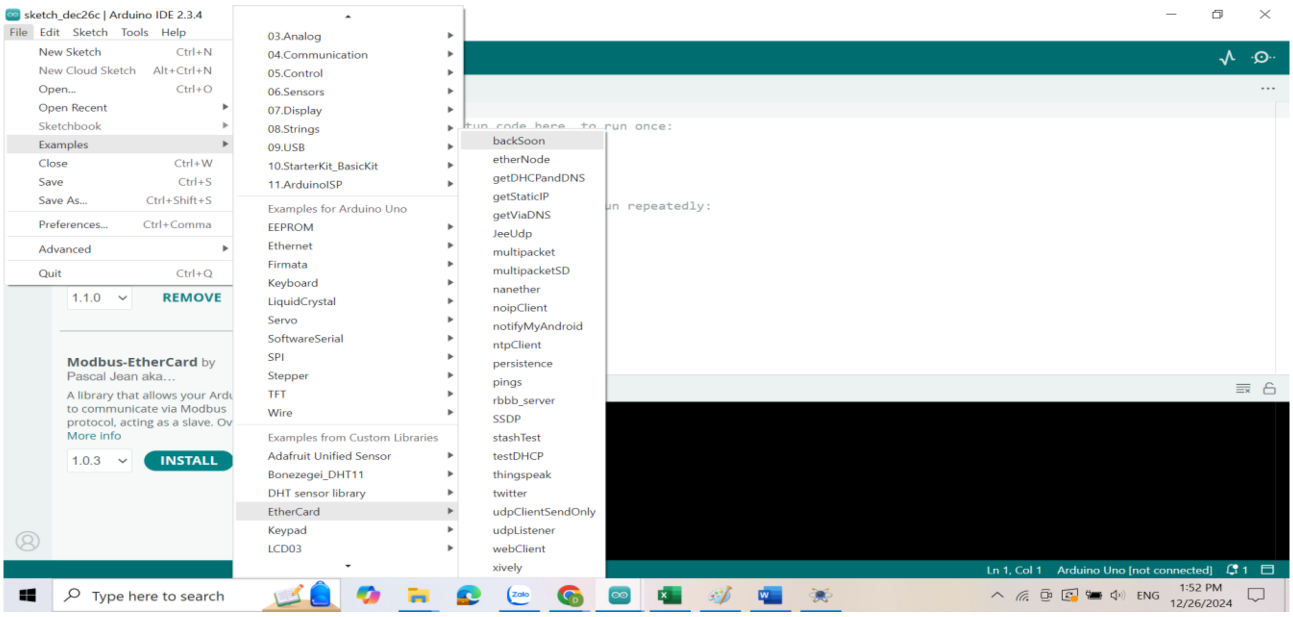
Modbus-EtherCard by Pascal Jean aka...
A library that allows your Arduino to communicate via Modbus protocol, acting as a slave. Over...
More info
1.0.3

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Downloading EtherCard@1.1.0
EtherCard@1.1.0
Installing EtherCard@1.1.0
Installed EtherCard@1.1.0
```

- once you place the folder in your arduino library folder restart your Arduino IDE

- Now go to ethercard > Examples > backsoon



Compile the code : if facing any problem then go to : [How To Simulate Arduino in ISIS]

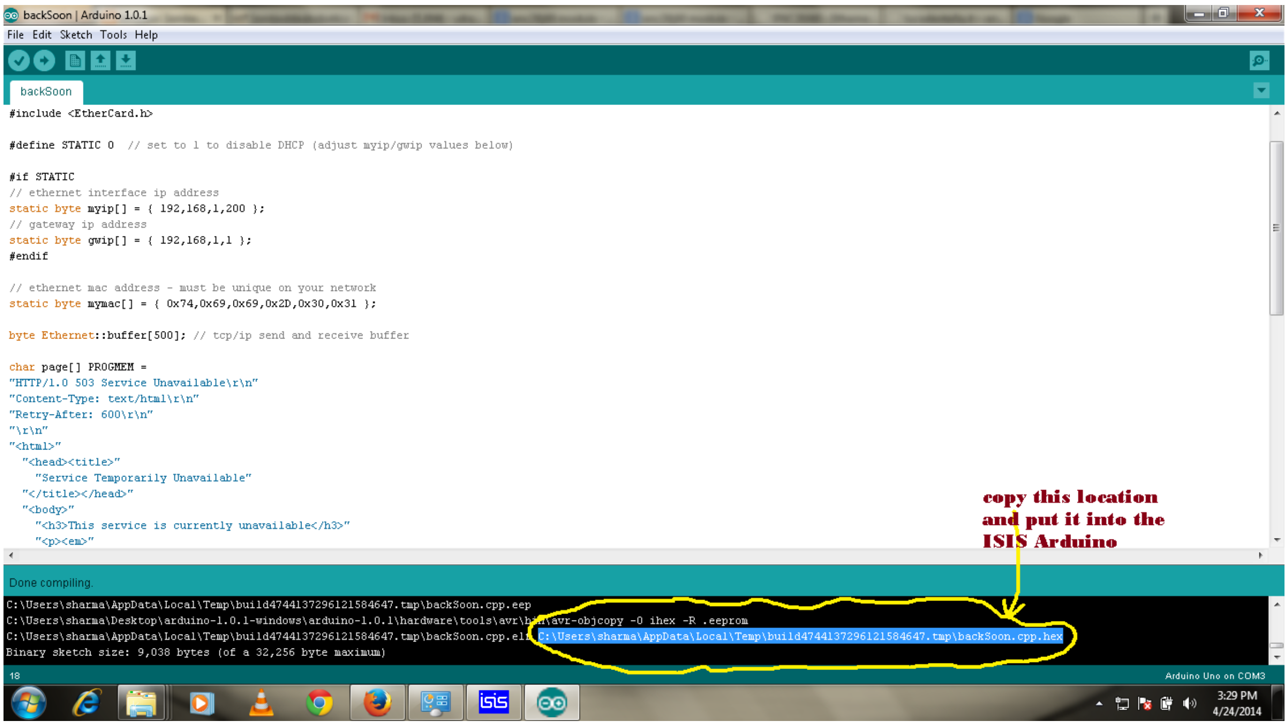
Make some changes in the program as given below

changes are in Red colour

Arduino code

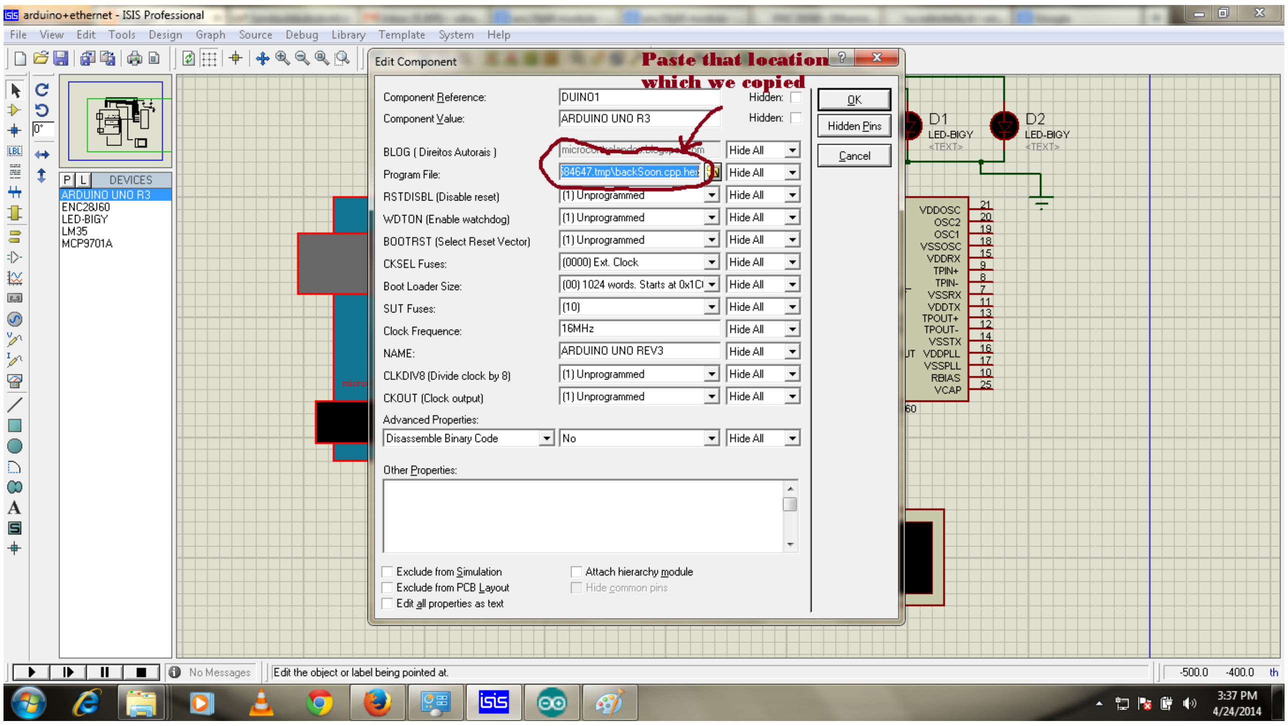
```
/*#######################################################
######################*/
// Present a "Will be back soon web page", as stand-in webserver.
// 2011-01-30 <jc@wippler.nl> http://opensource.org/licenses/mit-license.php
#include <EtherCard.h>
#define STATIC 0  // set to 1 to disable DHCP (adjust myip/gwip values below)
#if STATIC
// ethernet interface ip address
static byte myip[] = { 192,168,1,200 };
// gateway ip address
static byte gwip[] = { 192,168,1,1 };
#endif
// ethernet mac address - must be unique on your network
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
byte Ethernet::buffer[500]; // tcp/ip send and receive buffer
// char page[] is the HTML page you are uploading ..
char page[] PROGMEM =
"HTTP/1.0 503 My Seriavice \r\n"
"Content-Type: text/html\r\n"
"Retry-After: 600\r\n"
"\r\n"
"<html>"
  "<head><title>"
    "2embeddedrobotics "
  "</title></head>"
  "<body>"
    "<h3>Welcome to 2embeddedrobotics</h3>"
    "<p><em>"
      "The World of IOT Devices .<br />"
      "yipeee   Congrats...."
    "</em></p>"
   "</body>"
"</html>"
;
void setup(){
  Serial.begin(9600);
  Serial.println("\n[backSoon]");
  if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
    Serial.println( "Failed to access Ethernet controller");
#if STATIC
  ether.staticSetup(myip, gwip);
#else
  if (!ether.dhcpSetup())
    Serial.println("DHCP failed");
#endif
  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
  ether.printIp("DNS: ", ether.dnsip);
}
void loop(){
  // wait for an incoming TCP packet, but ignore its contents
  if (ether.packetLoop(ether.packetReceive())) {
    memcpy_P(ether.tcpOffset(), page, sizeof page);
    ether.httpServerReply(sizeof page - 1);
  }
}
/*############################################################
######################*/
```

now compile the code with prefrence compile. in file option.. so that you will get the .HEX file for ISIS.



```
#include <EtherCard.h>

#define STATIC 0  // set to 1 to disable DHCP (adjust myip/gwip values below)

#if STATIC
// ethernet interface ip address
static byte myip[] = { 192,168,1,200 };
// gateway ip address
static byte gwip[] = { 192,168,1,1 };
#endif

// ethernet mac address - must be unique on your network
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };

byte Ethernet::buffer[500]; // tcp/ip send and receive buffer

char page[] PROGMEM =
"HTTP/1.0 503 Service Unavailable\r\n"
"Content-Type: text/html\r\n"
"Retry-After: 600\r\n"
"\r\n"
"<html>"
  "<head><title>"
    "Service Temporarily Unavailable"
  "</title></head>"
  "<body>"
    "<h3>This service is currently unavailable</h3>"
    "<p><em>"
```

```
C:\Users\sharma\AppData\Local\Temp\build4744137296121584647.tmp\backSoon.cpp.eep
C:\Users\sharma\Desktop\arduino-1.0.1-windows\arduino-1.0.1\hardware\tools\avr\bin\avr-objcopy -O ihex -R .eeprom
C:\Users\sharma\AppData\Local\Temp\build4744137296121584647.tmp\backSoon.cpp.elf C:\Users\sharma\AppData\Local\Temp\build4744137296121584647.tmp\backSoon.cpp.hex
Binary sketch size: 9,038 bytes (of a 32,256 byte maximum)
```



Edit Component

Component Reference:	DUINO1	Hidden:
Component Value:	ARDUINO UNO R3	Hidden:
BLOG (Direitos Autorais)	microcontrolandos.blogspot.com	Hide All
Program File:	84647.tmp\backSoon.cpp.hex	Hide All
RSTDISBL (Disable reset)	(1) Unprogrammed	Hide All
WDTON (Enable watchdog)	(1) Unprogrammed	Hide All
BOOTRST (Select Reset Vector)	(1) Unprogrammed	Hide All
CKSEL Fuses:	(0000) Ext. Clock	Hide All
Boot Loader Size:	(00) 1024 words. Starts at 0x1C	Hide All
SUT Fuses:	(10)	Hide All
Clock Frequence:	16MHz	Hide All
NAME:	ARDUINO UNO REV3	Hide All
CLKDIV8 (Divide clock by 8)	(1) Unprogrammed	Hide All
CKOUT (Clock output)	(1) Unprogrammed	Hide All

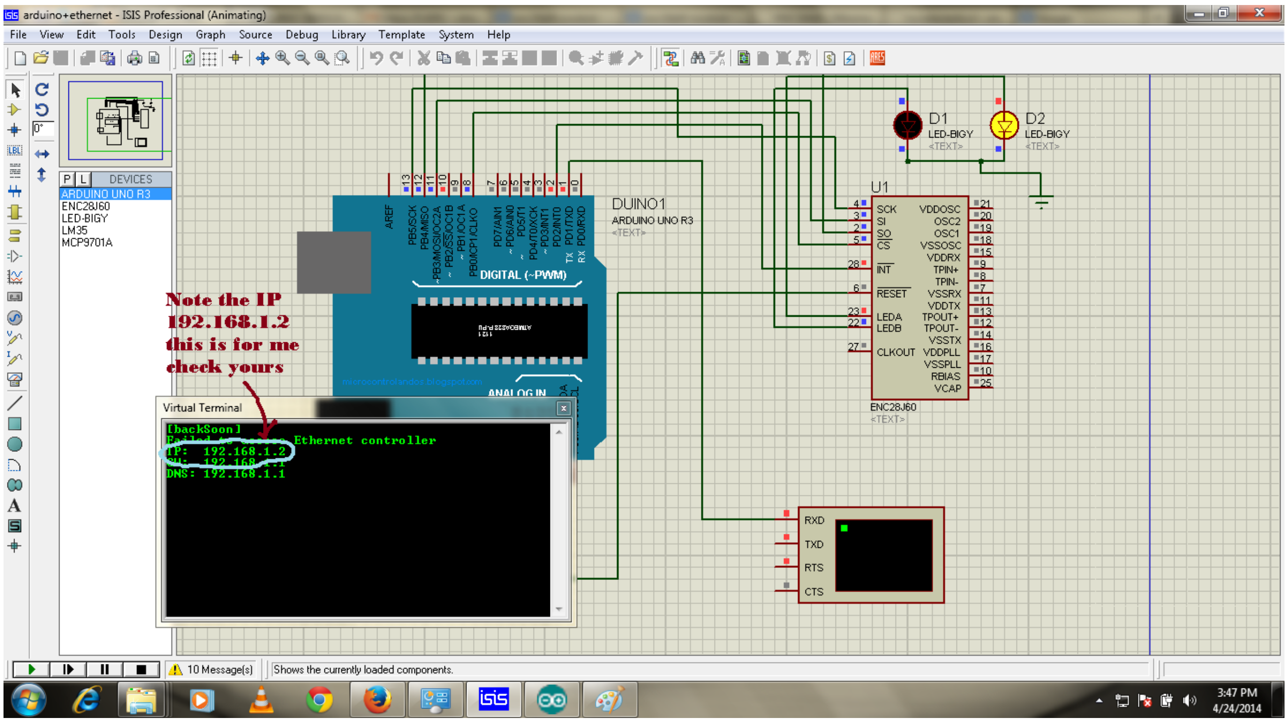
Advanced Properties:

Disassemble Binary Code	No	Hide All

Other Properties:

- [ ] Exclude from Simulation
- [ ] Exclude from PCB Layout
- [ ] Edit all properties as text
- [ ] Attach hierarchy module
- [ ] Hide common pins

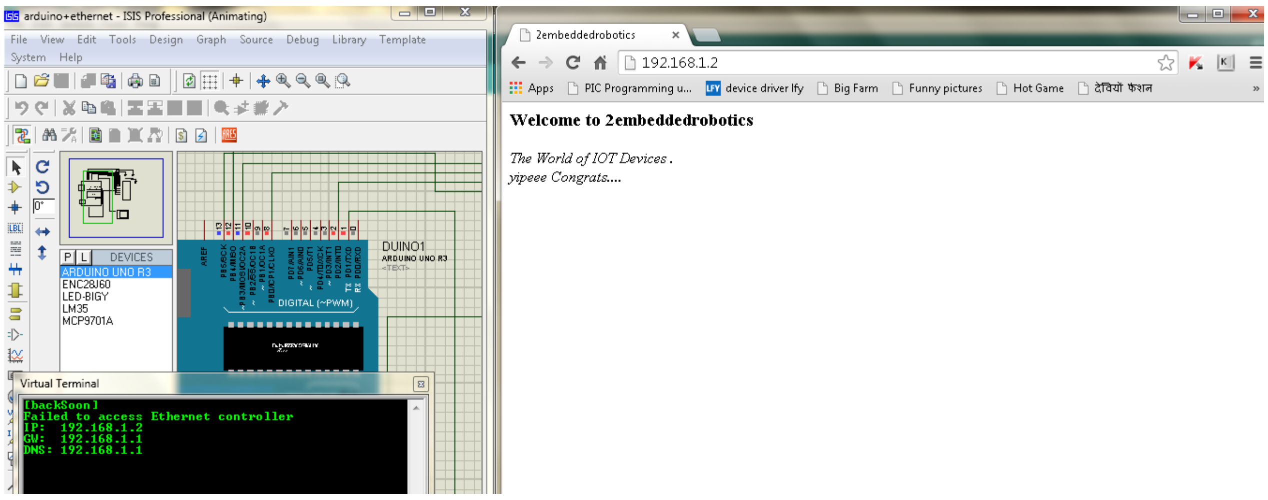
Now you have done every thing . Hit the Play Button and it will give you the assigned IP address in serial monitor now write it down on a paper.



Virtual Terminal

```
[backSoon]
Failed to access Ethernet controller
IP:  192.168.1.2
GW:  192.168.1.1
DNS: 192.168.1.1
```

open the web browser and write the IP address of your arduino in the browser.



```
[backSoon]
Failed to access Ethernet controller
IP:  192.168.1.2
GW:  192.168.1.1
DNS: 192.168.1.1
```

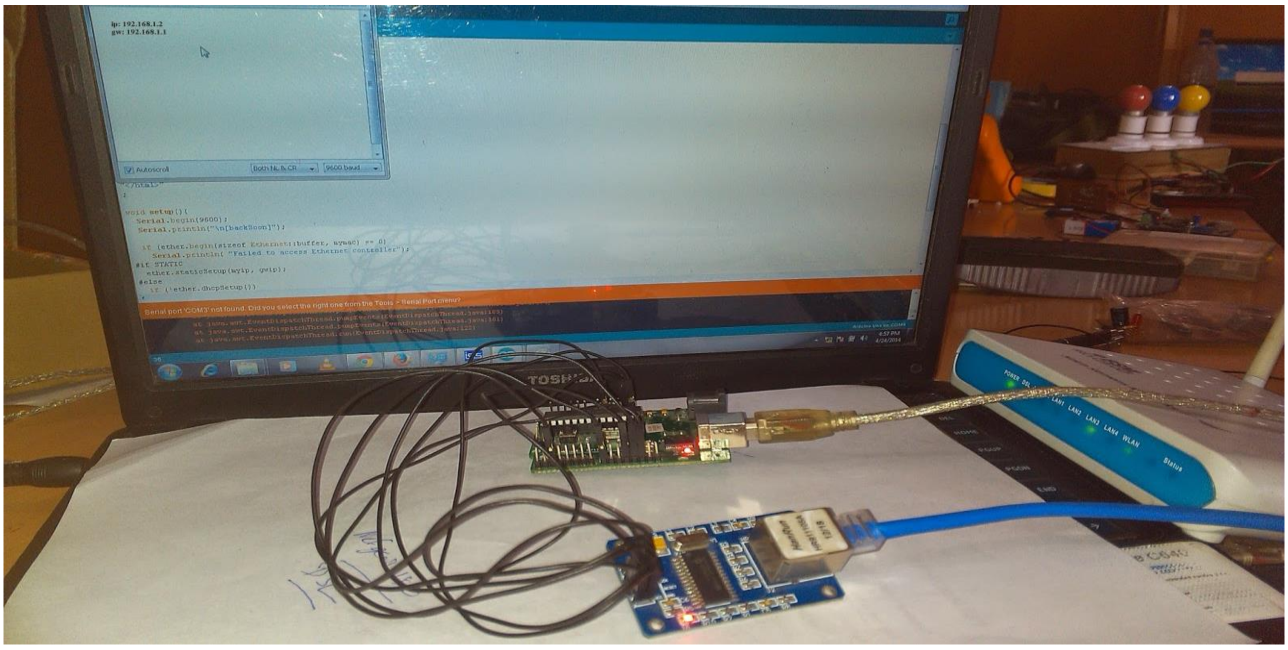
Welcome to 2embeddedrobotics

The World of IOT Devices .
yipeee Congrats....

Now you can watch this page on any device connected to that network As I show.

HARDWARE: We already did everything in virtual now we just need to connet each component as per the connection given in starting.

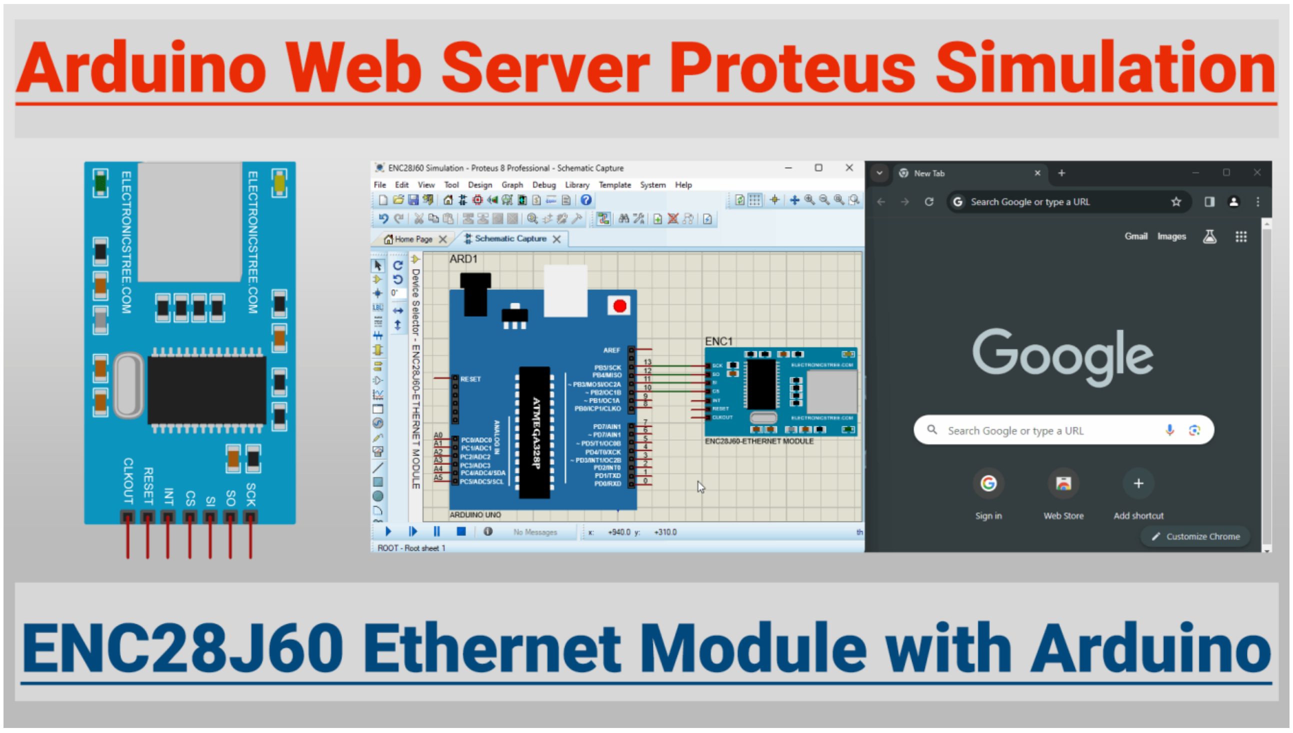
Now upload the program to your Arduino and turn on its Serial monitor so that u can get Your "IP in Serial Monitor" and just open browser and put your IP and check the webpage.



Now you have done then note your IP from the serial monitor and Enjoy..

Wish you a very good Luck.

ENC28J60 Ethernet Module: Arduino Web Server in Proteus with Simplified Setup – Part1


Arduino Web Server Proteus Simulation
ENC28J60 Ethernet Module with Arduino

Arduino Web Server - PART 1

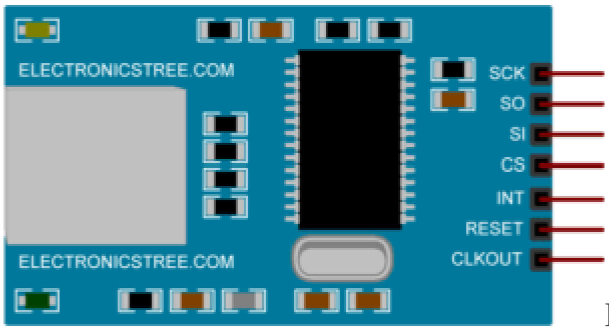
The ENC28J60 Ethernet module is a small circuit board that you can connect to your Arduino or Raspberry Pi. It acts as a bridge between your small controller and the Ethernet network. It contains a special chip called the ENC28J60, which is designed to handle the communication between your device and the network.

ENC28J60 Ethernet Module in Proteus

Currently, Proteus offers two models of Ethernet cards: the **Realtek RTL8019AS** and the **Microchip ENC28J60**. These models will not function unless your computer has a physical Ethernet card. Please verify that you have such a card installed and that any necessary drivers are both installed and enabled.

Unveiling the Upgraded ENC28J60 Ethernet Module Design

In this post, I'll be sticking to the ENC28J60 Ethernet module I personally designed. You might be wondering about the difference between the Proteus ENC28J60 and mine, but truth be told, they're pretty much the same in terms of functionality. However, the one I crafted has some extra perks. It's got a sleeker look with enhanced graphics and a cleaner interface, which matches the hardware module's aesthetics. What's more, I've even added LEDs to the module, so you can easily see if the Ethernet port is active or linked up.



ENC28J60 Ethernet Module

Network Setup

Before building your circuit and starting the simulation, you need to set up some necessary components on your Windows system. This includes installing a packet capture library and a virtual network driver.

Packet capture library

Proteus recommends installing the WinPcap library, but it is no longer actively maintained. Therefore, you should use Npcap instead.

What is Npcap?

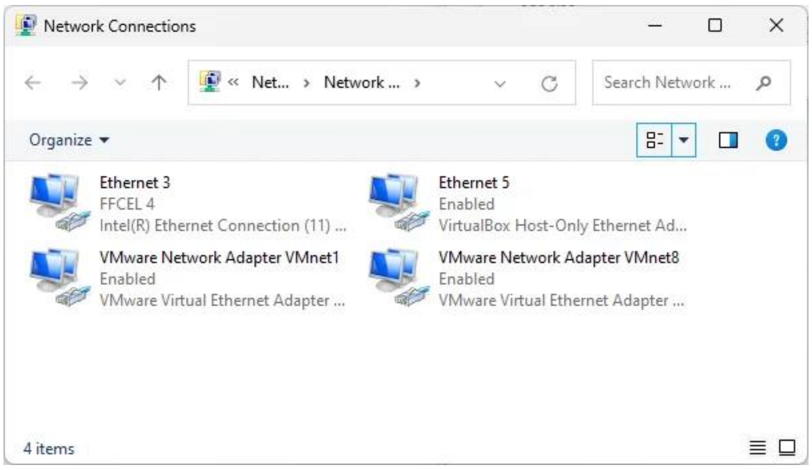
Npcap is a packet sniffing library and driver for Windows operating systems. It is an open-source project based on the well-known WinPcap library but with additional features and improvements. Npcap allows users to capture and analyze network traffic on Windows machines. It provides support for various network protocols and offers features like loopback traffic capturing, raw 802.11 packet capturing, and kernel-level packet filtering.

You can download this package from the **Npcap website**, and the installation process is straightforward. After agreeing to their license agreement, it provides a one-click install.

Virtual Network Driver

Suppose your PC doesn't have a network card, or if it does, it's already in use, or consider you are using a laptop that doesn't have an Ethernet connection. In this case, you can use a virtual network driver, which can provide you with a dummy network adapter. This allows you to easily simulate the ENC28J60 Ethernet Module in Proteus.

There are multiple ways to obtain a virtual network adapter. If you're using VMWare Workstation or VirtualBox software, you already have a virtual network adapter in your system, which you can utilize. Alternatively, if you're not using either of these, you can utilize the Microsoft KM-Test Loopback adapter, accessible through your Computer Management settings.



Ethernet 3
FFCEL 4
Intel(R) Ethernet Connection (11) ...

Ethernet 5
Enabled
VirtualBox Host-Only Ethernet Ad...

VMware Network Adapter VMnet1
Enabled
VMware Virtual Ethernet Adapter ...

VMware Network Adapter VMnet8
Enabled
VMware Virtual Ethernet Adapter ...

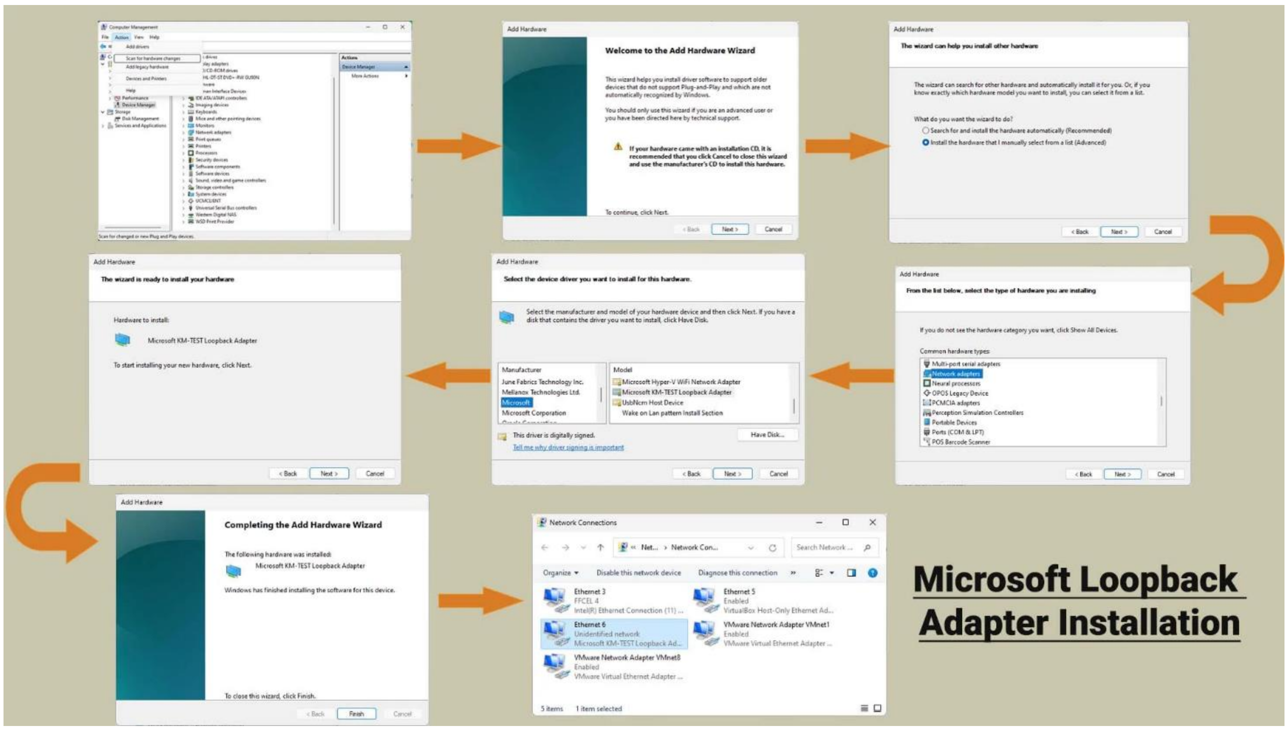
4 items

Network Adapters

As you can observe in the image above, Ethernet 3 represents my default PC network card adapter, while Ethernet 5 and others are provided by VirtualBox and VMWare Workstation software, respectively.

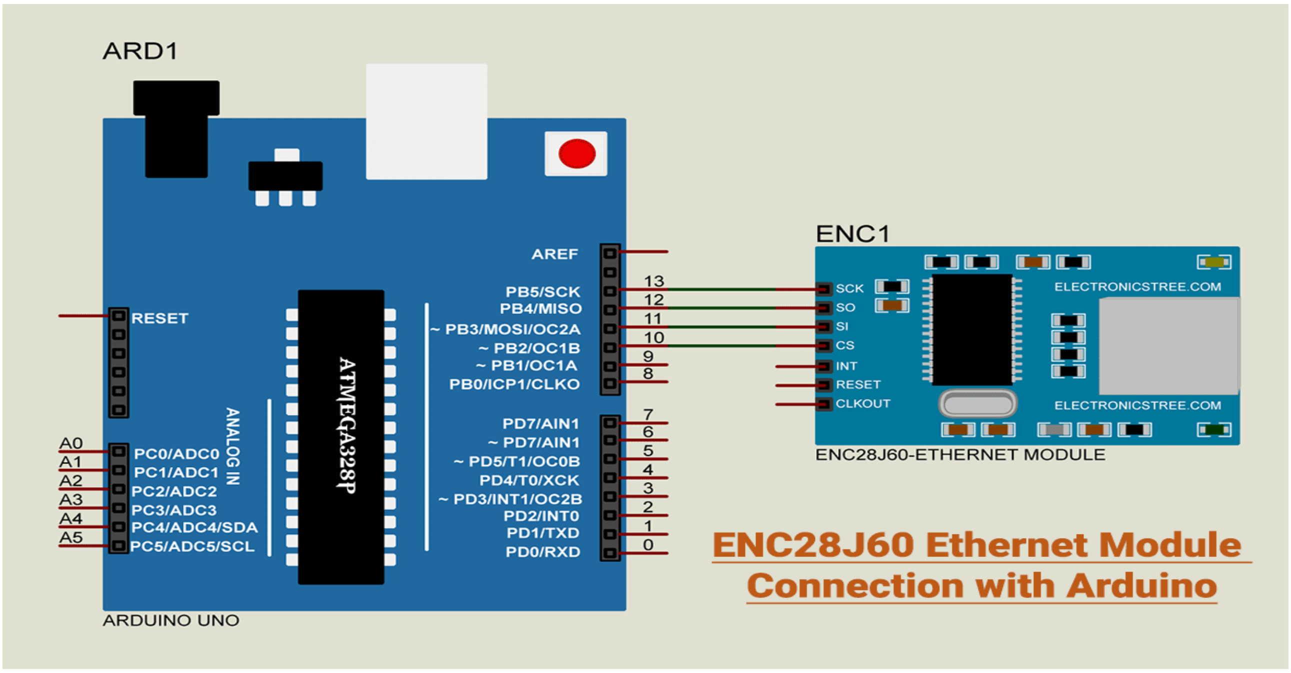
How to install Microsoft Loopback Adapter
Here is the installation method for the Microsoft Loopback Adapter.

Loopback Adapter Installation


Microsoft Loopback Adapter Installation
Welcome to the Add Hardware Wizard
The wizard can help you install other hardware
Install the hardware that I manually select from a list (Advanced)
From the list below, select the type of hardware you are installing
Network adapters
Select the device driver you want to install for this hardware.
Microsoft
Microsoft KM-TEST Loopback Adapter
The wizard is ready to install your hardware
Completing the Add Hardware Wizard
Network Connections
Ethernet 6
Unidentified network
Microsoft KM-TEST Loopback Ad...

Now, you can choose whichever method suits you best.

Proteus Setup for ENC28J60 Ethernet Module
The ENC28J60 Ethernet Module is designed to interface directly with the Serial Peripheral Interface (SPI) port available on many microcontrollers. Therefore, we will use Arduino SPI pins to connect the ENC28J60.



ENC28J60 Ethernet Module Connection with Arduino

Arduino Web Server Code

EtherCard Library

EtherCard is a library for Arduino that helps it communicate with the ENC28J60 Ethernet module, making it easier for Arduino to connect to Ethernet networks and perform tasks like sending and receiving data or serving web pages. You can find the EtherCard library for Arduino on the Arduino IDE Library Manager or on GitHub.

EtherCard *is a driver for the Microchip ENC28J60 chip, compatible with Arduino IDE. It is adapted and extended from code written by Guido Socher and Pascal Stang. High-level routines are provided to allow a variety of purposes including simple data transfer through to HTTP handling.*

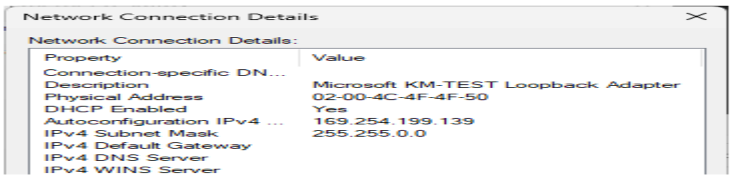
Arduino Code

This Arduino code creates a simple web server and serves a static welcome page to any HTTP requests it receives.

```
#include <EtherCard.h>
// Static IP configuration
static byte myip[] = { 169, 254, 238, 216 }; // Arduino's static IP address
static byte gwip[] = { 169, 254, 238, 215 }; // Gateway IP address
// MAC address of the Arduino
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
// Ethernet buffer size
byte Ethernet::buffer[700];
// Welcome page HTML content (stored in PROGMEM)
const char welcomePage[] PROGMEM =
  "<!DOCTYPE html>"
  "<html lang='en'>"
  "<head>"
  "<meta charset='UTF-8'>"
  "<meta name='viewport' content='width=device-width, initial-scale=1.0'>"
  "<title>Electronics Tree</title>"
  "<style>"
  "body {"
  "font-family: Arial, sans-serif;"
  "background-color: #f0f0f0;"
  "text-align: center;"
  "padding: 20px;"
  "}"
  "h1 {"
  "color: #333333;"
  "}"
  "p {"
  "color: #666666;"
  "}"
  "</style>"
  "</head>"
  "<body>"
  "<h1>Electronics Tree's Hands-On Journey</h1>"
  "<p>Welcome to Electronics Tree, where our name isn’t just a title—it’s a "
  "commitment to practical growth in your knowledge and skills."
  "Think of it like planting a seed.</p>"
  "</body>"
  "</html>";
void setup() {
  // Begin Ethernet communication with buffer size and MAC address
  ether.begin(sizeof Ethernet::buffer, mymac, SS);
  // Configure static IP and gateway IP
  ether.staticSetup(myip, gwip);
}
void loop() {
  // Handle incoming Ethernet packets and get the position of the data
  word pos = ether.packetLoop(ether.packetReceive());
  // If data is received
  if (pos) {
    // Extract data from Ethernet buffer
    char *data = (char *)Ethernet::buffer + pos;
    // Copy the welcome page content to the Ethernet buffer
    memcpy_P(ether.tcpOffset(), welcomePage, sizeof welcomePage);
    // Send HTTP response with the welcome page to the client
    ether.httpServerReply(sizeof welcomePage - 1);
  }
}
```

IP Address

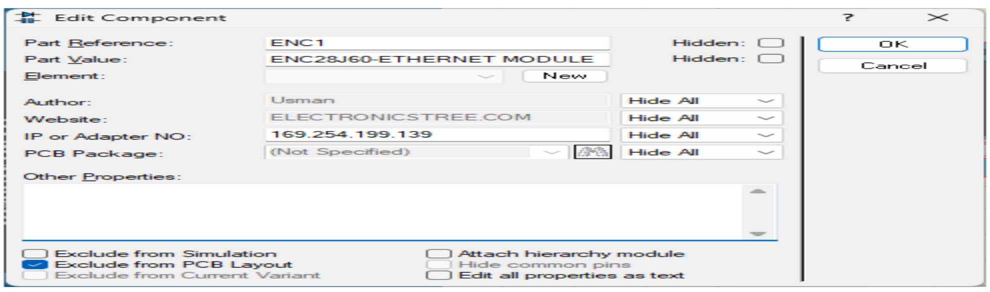
The above code uses the static IP address for the ENC28J60 Ethernet Module, which is 169.254.199.140. The gateway IP, 169.254.199.139, is the IP address of the adapter I’m using. Therefore, for the gateway IP address, use the IP address of the network adapter you want to use. In this case, I’m using the Microsoft Loopback adapter, so you can check the IP address of this adapter from the connection details window. For the Ethernet Interface IP address, you can change the host ID to your own preference.



Network Connection Details

Network Connection Details:

Property	Value
Connection-specific DN...	
Description	Microsoft KM-TEST Loopback Adapter
Physical Address	02-00-4C-4F-4F-50
DHCP Enabled	Yes
Autoconfiguration IPv4 ...	169.254.199.139
IPv4 Subnet Mask	255.255.0.0
IPv4 Default Gateway	
IPv4 DNS Server	
IPv4 WINS Server	



Edit Component

Part Reference:	ENC1	Hidden: ☐	
Part Value:	ENC28J60-ETHERNET MODULE	Hidden: ☐	
Element:		New	
Author:	Usman	Hide All	
Website:	ELECTRONICSTREE.COM	Hide All	
IP or Adapter NO:	169.254.199.139	Hide All	
PCB Package:	(Not Specified)	Hide All	

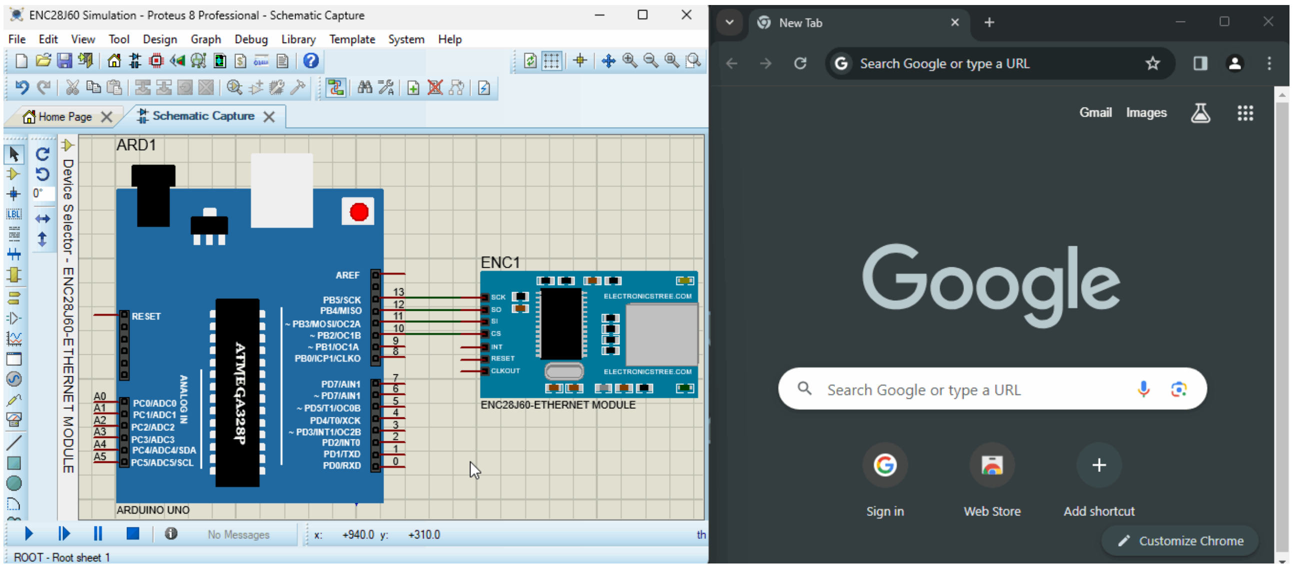
Other Properties:

☐ Exclude from Simulation
☑ Exclude from PCB Layout
☐ Exclude from Current Variant
☐ Attach hierarchy module
☐ Hide common pins
☐ Edit all properties as text

IP Address – ENC28J60 Ethernet Module

Now, in Proteus, open the properties window of the ENC28J60 Ethernet Module. You will find the field for the IP address. In this field, write the same IP address as your adapter’s.

Proteus Simulation
Write the code in the Arduino IDE and compile it. Copy the path of the resulting hex file and paste it into the Arduino Model in Proteus. Then, run the simulation. When you type the IP address into the browser and press Enter, Arduino shows the HTML web page.



Arduino Webserver with ENC28J60 Ethernet Module in Proteus

Download Library
Simply click on the button to download the library. You can refer to this post for instructions on how to install the library in Proteus 8. **How to Download and install Library in Proteus (electronicstree.com)**

Create Simple and Easy Arduino Web Server in Proteus with ENC28J60 – Part2

In this post, we will create an Arduino web server using the ENC28J60 within Proteus. Additionally, we will transmit data from Arduino to the webpage and control the LED via the webpage using Arduino.

Arduino Web Server using ENC28J60
In the previous post, we already covered the ENC28J60 Ethernet Module, discussing its purpose and how to integrate it with Arduino in Proteus. If needed, you can refer back to that post.
ENC28J60 Ethernet Module: Arduino Web Server in Proteus with Simplified Setup – Part1
In that discussion, we established an Arduino Web server that serves a static page. Now, in this post, our focus shifts to creating a dynamically generated HTML response page to update Arduino data.

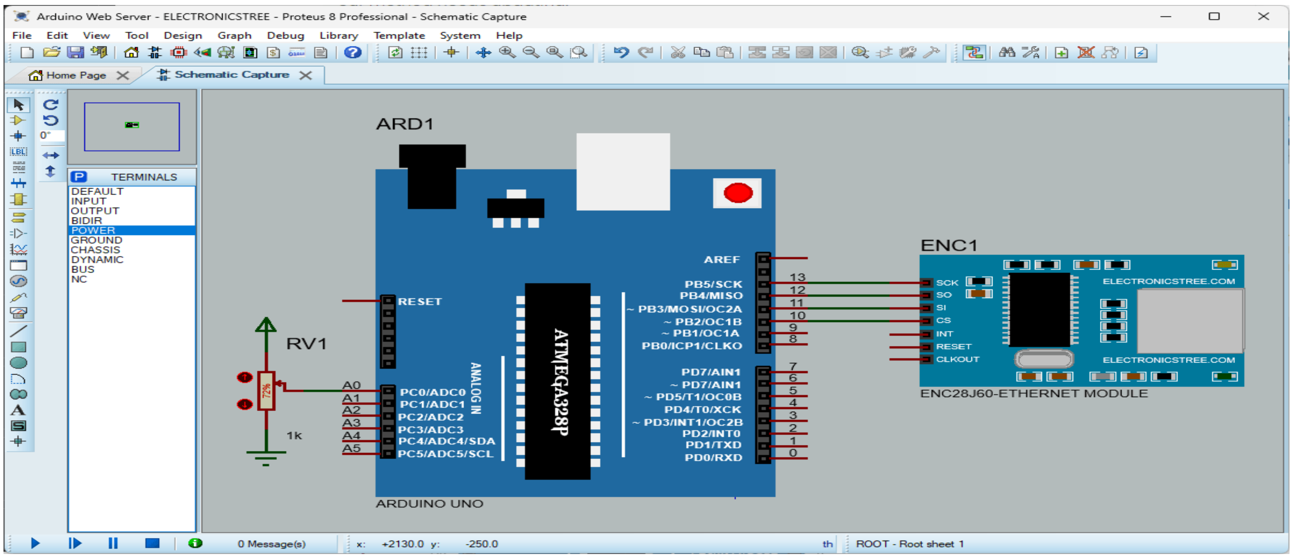
Arduino Web Server : Static vs Dynamic Method
In our previous approach, we used a pre-made HTML page stored in the program memory, which was served for every request. But as we focus on specific tasks, like displaying the potentiometer value, it's clear our method needs updating.

Instead of using a static HTML page, we'll create the HTML response dynamically. This means we'll construct the response on the spot, including the current potentiometer value directly.

This change not only saves memory space in our Arduino but also lets us customize the response as needed. With the potentiometer value embedded in the response, the webpage reflects real-time changes, making our project more responsive and efficient.

Proteus Setup for ENC28J60 Ethernet Module

The ENC28J60 Ethernet Module is designed to interface directly with the Serial Peripheral Interface (SPI) port available on many microcontrollers. Therefore, we will use Arduino SPI pins to connect the ENC28J60.



Arduino Web Server – Proteus Setup

Arduino Web Server Code : Sending Data from Arduino to Webpage

This code serves a simple webpage that displays the current value of a potentiometer connected to the Arduino. The webpage automatically refreshes every 3 seconds to display real-time updates of the potentiometer value.

```
#include <EtherCard.h>
static byte myip[] = { 169, 254, 238, 216 }; // Arduino's static IP address
static byte gwip[] = { 169, 254, 238, 215 }; // Gateway's IP address
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 }; // Arduino's MAC address
byte Ethernet::buffer[500]; // Ethernet buffer size
void setup() {
  ether.begin(sizeof Ethernet::buffer, mymac, SS); // Initialize Ethernet with buffer size and MAC
address
  ether.staticSetup(myip, gwip); // Set static IP and gateway
}
void loop() {
  int potValue = analogRead(A0); // Read analog value from potentiometer connected to A0
  char potString[6]; // Buffer for converting potentiometer value to string
  itoa(potValue, potString, 10); // Convert potentiometer value to string
  char htmlResponse[250]; // Buffer for HTML response
  // Construct HTML response with potentiometer value
  sprintf(htmlResponse, "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r
                      "<html>\r\n"
                      "<head>\r\n"
```

```
                        "<meta http-equiv='refresh' content='3'>\r\n"
                        "</head>\r\n"
                        "<body>\r\n"
                        "Potentiometer value: %s\r\n"
                        "</body>\r\n"
                        "</html>\r\n", potString);
    // Check for incoming packets
    word pos = ether.packetLoop(ether.packetReceive());
    if (pos) {
      char *data = (char *)Ethernet::buffer + pos; // Pointer to packet data
      // Copy HTML response to Ethernet buffer
      memcpy(ether.tcpOffset(), htmlResponse, strlen(htmlResponse));
      // Send HTTP server reply with HTML response length
      ether.httpServerReply(strlen(htmlResponse));
    }
  }
```
Code Explanation

The code above creates a basic web server on the Arduino. It shows a webpage with the potentiometer's current value. To see this webpage, just type the Arduino's IP address into a web browser. Now, let's understand how this Arduino Web Server operates.

How the Arduino Web Server Works

TCP Packet Reception:

- When a TCP packet is received by the Arduino, it contains a payload. In the context of web servers, this payload typically consists of an HTTP request sent by a client (such as a web browser) to the Arduino.

Extracting TCP Payload:

- The `ether.packetLoop()` function processes the received TCP packet. It identifies the position of the TCP payload within the Ethernet buffer and returns this position (`pos`). This position indicates where the payload starts within the buffer.

Generating HTML Response:

- After identifying that a TCP packet has been received and processed, the Arduino constructs an HTML response. This response includes the current value of the potentiometer, formatted as HTML content.

Replacing TCP Payload with HTML Response:

- Once the HTML response is constructed, it is copied into the same memory space previously occupied by the TCP payload. This effectively replaces the content of the TCP payload with the HTML response containing the potentiometer value.

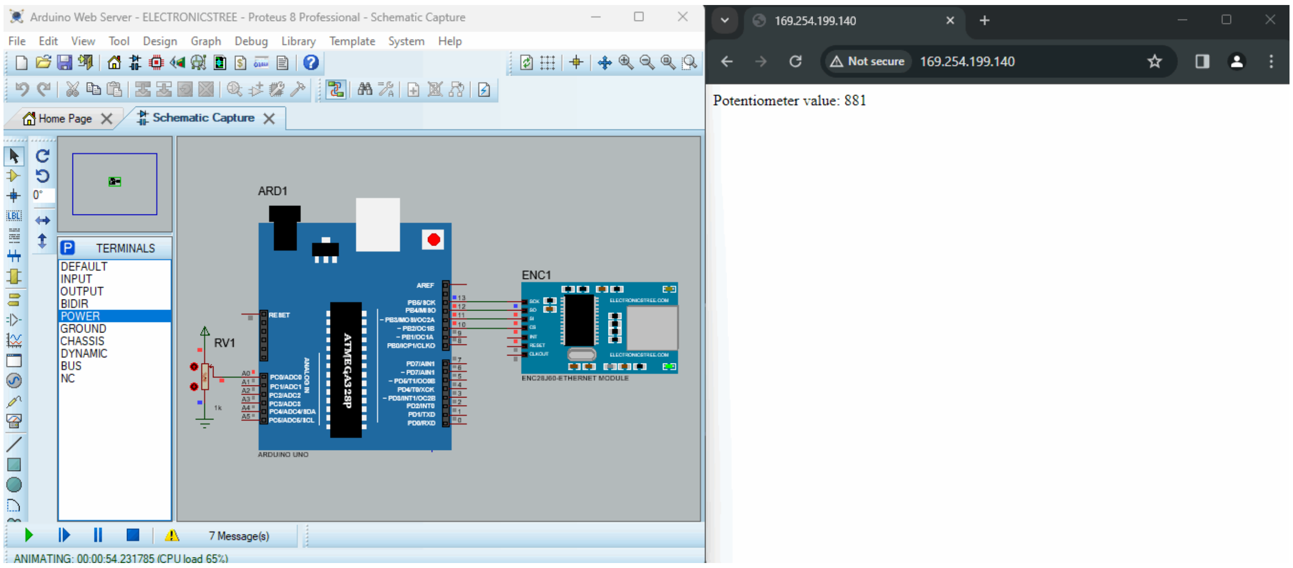
Sending HTTP Response:

- Finally, the Arduino sends an HTTP response back to the client (the device that made the HTTP request). This response now contains the HTML content generated by the Arduino, including the potentiometer value. The response is sent using the `ether.httpServerReply()` function, which takes the length of the HTML response as an argument to specify how much data to send.

this process allows the Arduino to act as a simple web server. When a client sends an HTTP request (like accessing a webpage), the Arduino dynamically generates an HTML response containing the current value of the potentiometer and sends it back to the client, effectively displaying the potentiometer value in a web browser or any other HTTP client.

Proteus Simulation

Here are the results of the Proteus simulation. What's interesting to observe is that any changes made to the potentiometer value within Proteus are instantly reflected in the browser interface as well.

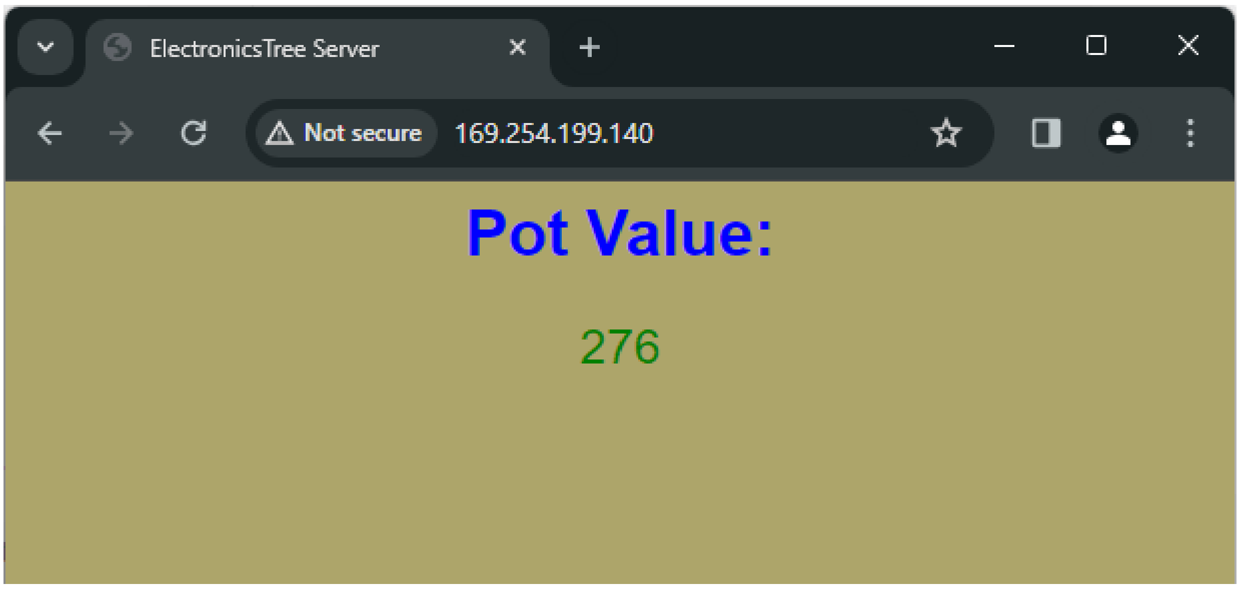


Potentiometer value: 881

Arduino Web Server – Simulation Result

Add CSS styling in the Webpage

You have the option to enhance the HTML response with styling elements like titles, headings, and colors directly within the code. By adjusting the HTML content string, you can incorporate HTML tags to introduce styling features.



Pot Value:

276

Arduino Web Server – CSS Styling

```
#include <EtherCard.h>
// Define static IP addresses for Arduino and gateway
static byte myip[] = { 169, 254, 238, 216 };
static byte gwip[] = { 169, 254, 238, 215 };
// Define Arduino's MAC address and Ethernet buffer size
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
byte Ethernet::buffer[500];
```

```
void setup() {
  // Initialize Ethernet with buffer size and MAC address
  ether.begin(sizeof Ethernet::buffer, mymac, SS);
  // Set static IP and gateway
  ether.staticSetup(myip, gwip);
}
void loop() {
  // Read analog value from potentiometer connected to A0
  int potValue = analogRead(A0);
  // Convert potentiometer value to string
  char potString[6];
  itoa(potValue, potString, 10);
  // Ensure the string is null-terminated
  potString[sizeof(potString) - 1] = '\0';
  // Construct HTML response with potentiometer value
  char htmlResponse[500];
  sprintf(htmlResponse, "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\
                      "<html>\r\n"
                      "<head>\r\n"
                      "<meta http-equiv='refresh' content='3'>\r\n"
                      "<title>ElectronicsTree Server</title>\r\n"
                      "<style>\r\n"
                      "body {\r\n"
                      "  background-color: #ADA56A;\r\n"
                      "  font-family: Arial, sans-serif;\r\n"
                      "}\r\n"
                      "  color: blue;\r\n"
                      "  text-align: center;\r\n"
                      "}\r\n"
                      "p {\r\n"
                      "  font-size: 24px;\r\n"
                      "  color: green;\r\n"
                      "  text-align: center;\r\n"
                      "}\r\n"
                      "</style>\r\n"
                      "</head>\r\n"
                      "<body>\r\n"
                      "<h1>Pot Value:</h1>\r\n"
                      "<p>%s</p>\r\n"
                      "</body>\r\n"
                      "</html>\r\n", potString);
  // Check for incoming packets
  word pos = ether.packetLoop(ether.packetReceive());
  if (pos) {
    char *data = (char *)Ethernet::buffer + pos;
    // Copy HTML response to Ethernet buffer and send HTTP server reply
    memcpy(ether.tcpOffset(), htmlResponse, strlen(htmlResponse));
```

```
      ether.httpServerReply(strlen(htmlResponse));
    }
  }
```

Limitations of Basic Arduino Boards Compared to ESP Boards

Basic Arduino boards don’t come with Wi-Fi built-in. To get them connected to the internet, you’ll need to add extra hardware like an Ethernet shield or a Wi-Fi module. That’s why it’s not really fair to compare this Arduino Web Server with the ones on ESP boards. ESP boards are more powerful because they have more memory and speed. So, you can’t do all the cool stuff with Arduino that you can with ESP boards.

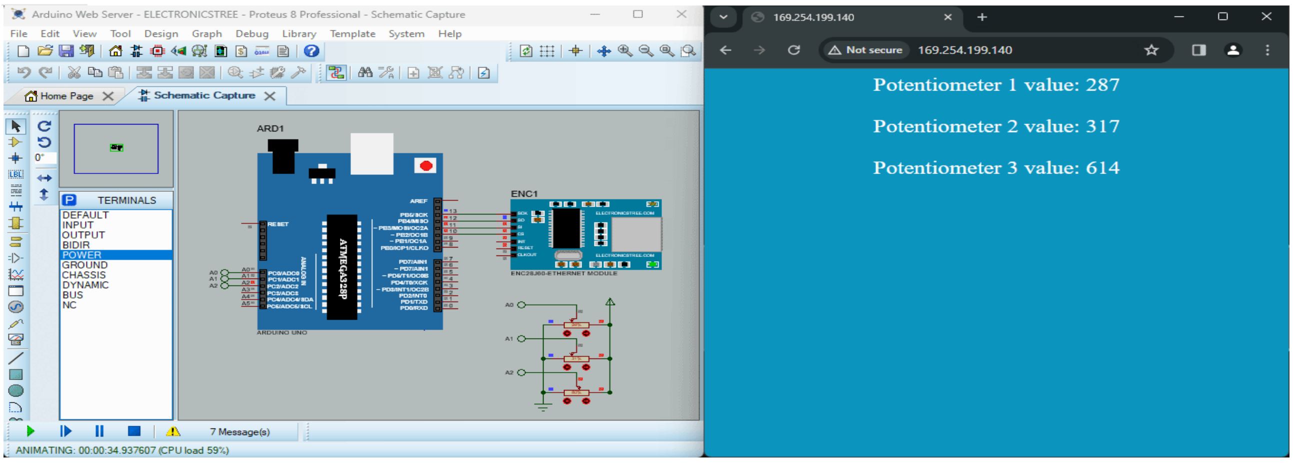
To make sure your Arduino runs smoothly, it’s a good idea to avoid using fancy CSS styles on your webpage. That way, you can save memory and make everything run faster.

Adding More Data to The Arduino Web Server

If you want to add more potentiometers to the code, you’ll have to read their values in the loop() function and incorporate them into the HTML response. Here’s how you can adjust the code to include readings from additional potentiometers connected to analog pins A1 and A2.

```
#include <EtherCard.h>
// Define static IP addresses for Arduino and gateway
static byte myip[] = { 169, 254, 238, 216 };
static byte gwip[] = { 169, 254, 238, 215 };
// Define Arduino's MAC address and Ethernet buffer size
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
byte Ethernet::buffer[500];
void setup() {
  // Initialize Ethernet with buffer size and MAC address
  ether.begin(sizeof Ethernet::buffer, mymac, SS);
  // Set static IP and gateway
  ether.staticSetup(myip, gwip);
}
void loop() {
  // Read analog values from potentiometers connected to A0, A1, and A2
  int potValue1 = analogRead(A0);
  int potValue2 = analogRead(A1);
  int potValue3 = analogRead(A2);
  // Convert potentiometer values to strings
  char potString1[6], potString2[6], potString3[6];
  itoa(potValue1, potString1, 10);
  itoa(potValue2, potString2, 10);
  itoa(potValue3, potString3, 10);
  // Construct HTML response with potentiometer values
  char htmlResponse[600];
  sprintf(htmlResponse, "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r
                        "<html>\r\n"
                        "<head>\r\n"
                        "<meta http-equiv='refresh' content='3'>\r\n"
```

```
                  "<style>"
                  "body { text-align: center; font-size: 24px; background-color: #0b94be; color: whi
                  "</style>"
                  "</head>\r\n"
                  "<body>\r\n"
                  "<div>Potentiometer 1 value: %s</div><br>\r\n"
                  "<div>Potentiometer 2 value: %s</div><br>\r\n"
                  "<div>Potentiometer 3 value: %s</div>\r\n"
                  "</body>\r\n"
                  "</html>\r\n", potString1, potString2, potString3);
    // Check for incoming packets
    word pos = ether.packetLoop(ether.packetReceive());
    if (pos) {
      char *data = (char *)Ethernet::buffer + pos;
      // Copy HTML response to Ethernet buffer and send HTTP server reply
      memcpy(ether.tcpOffset(), htmlResponse, strlen(htmlResponse));
      ether.httpServerReply(strlen(htmlResponse));
    }
  }
```



```
Potentiometer 1 value: 287

Potentiometer 2 value: 317

Potentiometer 3 value: 614
```

Arduino Web Server - Multi Pot Values

In the next post, we will demonstrate how to control the Arduino digital pins directly from the webpage. If you have any questions or encounter any issues related to this Arduino Web Server, feel free to ask in the comments section or contact me via the Contact page.

Creating an Easy Arduino Web Server in Proteus without ESP8266 Boards – Part3

Introduction to Arduino Web Server in Proteus

This marks the final installment of our series on creating an Arduino web server in Proteus. In the initial segment, we configured **ENC28J60** with Arduino in Proteus and established a web server within Proteus to serve a static web page. Following that, we adopted a dynamic approach to serve the web page and transmit data from the web server in Proteus to the client (web browser). Now, in this concluding segment, we will accept commands from the client to control the web server in Proteus. As this is a step-by-step tutorial, I recommend newcomers to refer back to the previous posts for a better grasp of the concepts.

ENC28J60 Ethernet Module: Arduino Web Server in Proteus with Simplified Setup – Part1

Create Simple And Easy Arduino Web Server In Proteus With ENC28J60 – Part2

Sending Data from Webpage to Arduino Web Server in Proteus

This code creates a simple web server on the Arduino board that can receive HTTP requests to control an LED. It listens for specific requests, processes them to toggle the LED, and sends an HTML response back to the client indicating the current LED state

```
#include <EtherCard.h>
// Ethernet interface IP address
static byte myip[] = { 169, 254, 199, 140 };
// Gateway IP address
static byte gwip[] = { 169, 254, 199, 139 };
// Ethernet MAC address - must be unique on your network
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
// LED pin
const int ledPin = 3;
bool ledState = LOW;
byte Ethernet::buffer[700];  // TCP/IP send and receive buffer
void setup() {
  // Initialize Ethernet connection
  ether.begin(sizeof Ethernet::buffer, mymac, SS);
  // Set static IP if defined
  ether.staticSetup(myip, gwip);
  // Set LED pin as output
  pinMode(ledPin, OUTPUT);
}
void loop() {
  // Listen for incoming packets and respond
  word pos = ether.packetLoop(ether.packetReceive());
  if (pos) {
    // Data received, process HTTP request
    char *data = (char *)Ethernet::buffer + pos;
    // Check if the request contains "GET /led/on"
    if (strstr(data, "GET /led/on")) {
```

```
    // Turn LED on
    digitalWrite(ledPin, HIGH);
    ledState = HIGH;
   } else if (strstr(data, "GET /led/off")) {
    // Turn LED off
    digitalWrite(ledPin, LOW);
    ledState = LOW;
   }
   // Send HTML response
   sendHttpResponse();
  }
}
void sendHttpResponse() {
  // Create HTML response with LED state
  char htmlResponse[400];
  sprintf(htmlResponse,
        "HTTP/1.1 200 OK\r\n"
        "Content-Type: text/html\r\n"
        "Connection: close\r\n"
        "\r\n"
        "<!DOCTYPE html>\r\n"
        "<html>\r\n"
        "<head>\r\n"
        "<title>LED Control</title>\r\n"
        "</head>\r\n"
        "<body>\r\n"
        "<h1>LED Control</h1>\r\n"
        "<p>LED is %s</p>\r\n"
        "<p><a href=\"/led/on\">Turn On</a> | <a href=\"/led/off\">Turn Off</a></p>\r\n"
        "</body>\r\n"
        "</html>\r\n",
        (ledState == HIGH) ? "On" : "Off");
  // Send response
  memcpy(ether.tcpOffset(), htmlResponse, strlen(htmlResponse));
  ether.httpServerReply(strlen(htmlResponse));
}
```

Code Explanation

Library Inclusion:

```
#include <EtherCard.h>
```

This line includes the EtherCard library, which provides functions for Ethernet communication on Arduino

IP Address and MAC Address Setup:

```
static byte myip[] = { 169, 254, 199, 140 };
static byte gwip[] = { 169, 254, 199, 139 };
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
```

Here, we set up the IP address (`myip`), gateway IP address (`gwip`), and MAC address (`mymac`) for the Ethernet interface. These values are used to configure the Ethernet connection.

LED Pin and State:

```
const int ledPin = 3;
bool ledState = LOW;
```

We define the pin connected to the LED (`ledPin`) and initialize the LED state (`ledState`) as LOW (off).

Ethernet Buffer and Initialization:

```
byte Ethernet::buffer[700];
```

This line declares a buffer for TCP/IP communication.

Here, we initialize the Ethernet connection using `ether.begin`. The `sizeof Ethernet::buffer` specifies the size of the buffer, `mymac` is the MAC address, and `SS` is the slave select pin used for SPI communication with the Ethernet module.

```
ether.begin(sizeof Ethernet::buffer, mymac, SS);
```

Static IP Configuration:

```
ether.staticSetup(myip, gwip);
```

This line configures the Ethernet connection with a static IP address (`myip`) and gateway IP address (`gwip`).

Loop Function:

```
void loop() {
word pos = ether.packetLoop(ether.packetReceive());
if (pos) {
// Process HTTP request
}
}
```

In the `loop` function, we continuously listen for incoming packets using `ether.packetLoop` and process them if a packet is received.

HTTP Request Processing:

```
char *data = (char *)Ethernet::buffer + pos;
if (strstr(data, "GET /led/on")) {
// Turn LED on
} else if (strstr(data, "GET /led/off")) {
// Turn LED off
}
```

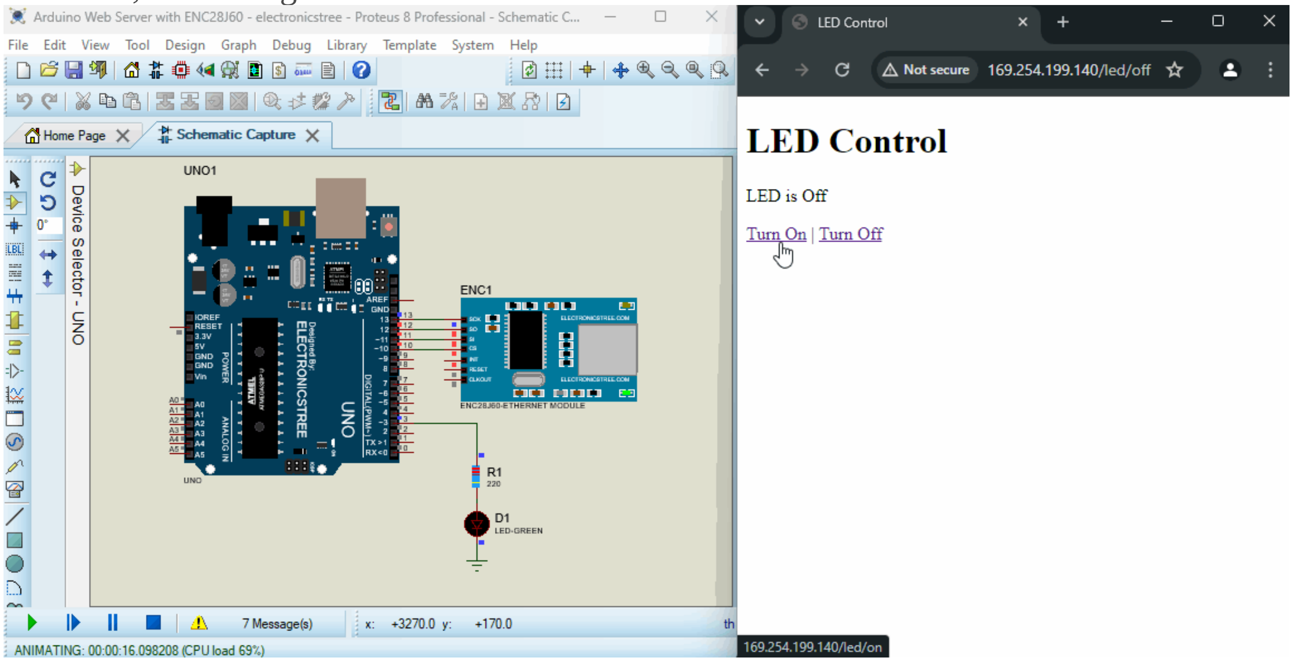
Inside the packet processing block, you extract the HTTP request data and check if it contains either “GET /led/on” or “GET /led/off”. Depending on the request, you either turn the LED on or off.

HTML Response Generation:

This function constructs an HTML response based on the current LED state (`ledState`) and sends it back to the client. The response includes the LED state and links to turn the LED on or off.

Web Server in Proteus Simulation

Here are the results of the Proteus simulation. As you can see, the Arduino controls the LED from the webpage. This not only works in Proteus simulation; if you have the actual hardware, this configuration and code will also function the same.



Arduino Web Server with ENC28J60 - electronicstree - Proteus 8 Professional - Schematic C...
File Edit View Tool Design Graph Debug Library Template System Help
Home Page
Schematic Capture
Device Selector - UNO
UNO1
ENC1
ENC28J60-ETHERNET MODULE
R1
220
D1
LED-GREEN
7 Message(s)
x: +3270.0 y: +170.0
ANIMATING: 00:00:16.098208 (CPU load 69%)
LED Control
Not secure 169.254.199.140/led/off
169.254.199.140/led/on

LED Control

LED is Off

Turn On | Turn Off

Web Server in Proteus – Simulation

Limitations of Basic Arduino Boards Compared to ESP Boards

Basic Arduino boards don’t come with Wi-Fi built-in. To get them connected to the internet, you’ll need to add extra hardware like an Ethernet shield or a Wi-Fi module. That’s why it’s not really fair to compare this Arduino Web Server with the ones on ESP boards. ESP boards are more powerful because they have more memory and speed. So, you can’t do all the cool stuff with Arduino that you can with ESP boards.

If you have any questions or encounter any issues related to this Arduino Web Server, feel free to ask in the comments section or contact me via the Contact page.