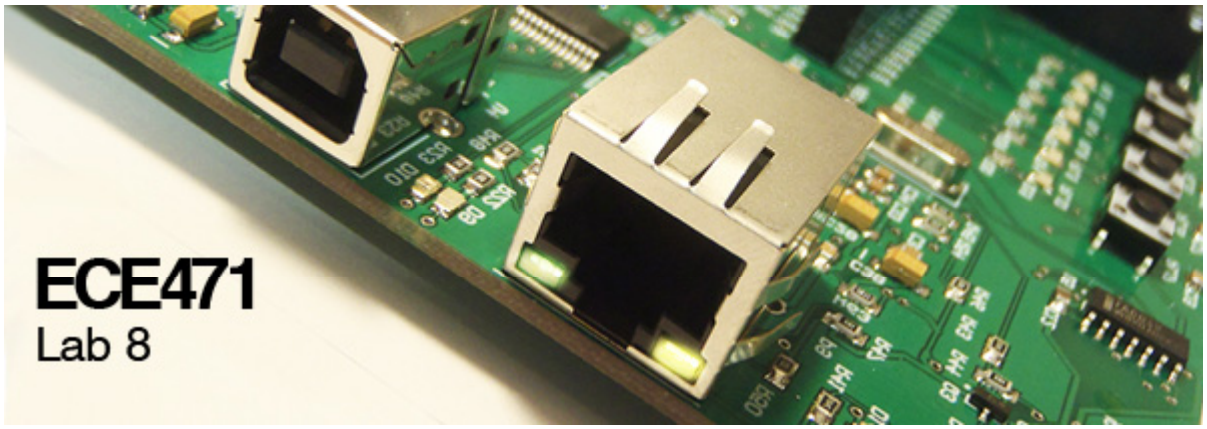


NDSU Embedded Systems II

TCP Ethernet

Embedded Web Servers

Objective: To obtain familiarity with the following:

- Ethernet
- TCP/IP Protocols
  - TCP: Transfer Control Protocol
  - HTTP: Hyper Text Transfer Protocol

By
Nathan Zimmerman

Requirements for lab 8:

Introduction: As technology has progressed, Ethernet is experiencing frequent utilization in modern embedded systems. Ethernet is practical for embedded systems that consist of multi-device networks, networks over large distance, high speed networks, and networks that require data high levels of data integrity. Consequently, Ethernet is becoming a large future player in embedded as a method of external communication. In this lab, we will be studying implementing Ethernet on an embedded system as well as using TCP, a TCP/IP protocol, over an Ethernet connection. We will be using HTTP on top of TCP in order to host an embedded web server.

Requirements for Lab 8:

1. Create a simple HTML page for an embedded web server.
2. Obtain ADC data from POT on embedded board
3. ADC data should be present on the web browser and be dynamically updating
4. ADC data format should be in volts

Result should look similar to Lab8.axf .

Useful Resources

WireShark:

http://www.wireshark.org/

Wikipedia:

http://en.wikipedia.org/wiki/Ethernet
http://en.wikipedia.org/wiki/Transmission_Control_Protocol
http://en.wikipedia.org/wiki/HTTP

Networking Basics:

Fundamentals TCP/IP: (worth the watch unless you know them very well)
http://www.youtube.com/watch?v=EkNq4TrHP_U

TCP/HTTP:
http://www.youtube.com/watch?v=iySa4zBYScE
http://www.youtube.com/watch?v=JA3aTmJ_77Y

Web Server Example Code:

See our project repo.

Ethernet:

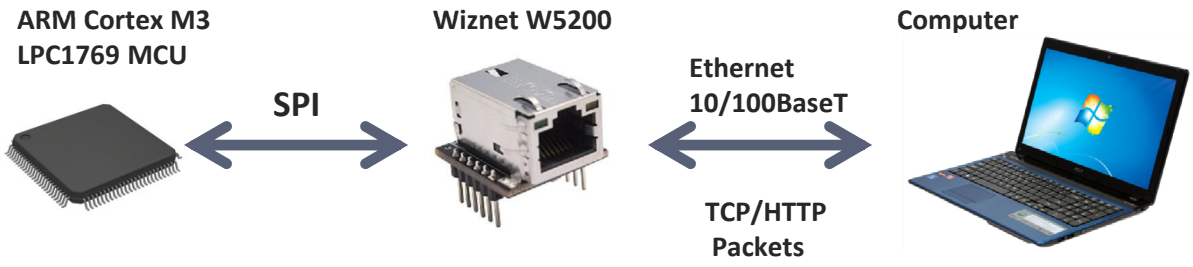
Ethernet is a series of specifications by IEEE 802.3 to implement a physical layer that is used for sending data over Local Area Networks. The physical layer of a network references the literal physical medium such as wires that raw data bits will travel over. The physical layer is also a with respect to a network model which is the Open System Interconnect model which is used the represent complex networking systems such as a network that implements TCP/IP protocols such as the Transfer Control Protocol (TCP) which is a base communication protocol used in order to send web data over the internet. TCP is used to send Hypertext Transfer Protocol (HTTP) data between user's web browsers and a server. For this lab, our embedded web server will be acting as a http server.

Layer #	Layer Name	Data Type	TCP/IP Protocol
7	Application	Data	HTTP
6	Presentation	Data	----
5	Session	Data	----
4	Transport	Segments	TCP
3	Network	Packets	IP
2	Data Link	Frame	PPP
1	Physical	Bit	100Base-T

As far as implementing Ethernet on our Embedded II board, we will be using the W5200 wiznet module as an Ethernet Transceiver. This transceiver can implements 10/100Base-T Ethernet which uses a common RJ-45 jack and uses a CAT5+ Ethernet cable.

Ethernet Communication Diagram:

Below is a diagram of how our communication scheme works with the AR Drone.



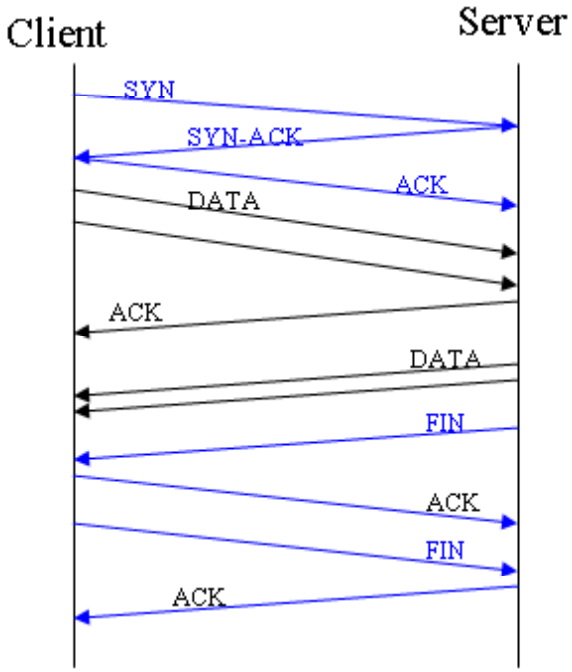
Note: Our server currently does not have DHCP support. I haven't had the time to write that. Consequently, its IP Address is hardcoded into the device. Meaning, in order to talk to this, you have to be on a network that uses/permits static IP address as opposed to DHCP assigned addresses. A direct link between your PC and the development board would work. As a side note, one could use a router and port forwarding in order to be able to access this Ethernet connection from anywhere on the WWW. Regrettably I'm on the NDSU network or else I'd show a live demo of our server.

We interface with the W5200 SPI Ethernet controller since this particular Ethernet ASIC already has an internalized TCP/IP stack making the implementing of sending UDP packets over Ethernet significantly easier. We use this as opposed to the RMII MAC Ethernet interface already on our processor in order to avoid having to implement a TCP/IP stack in software. With the W5200, sending out TCP/IP protocols is relatively easy since it only consists of writing to a few SPI registers.

TCP: Transmission Control Protocol

TCP, the Transmission Control Protocol is a reliable transport protocol (reference OSI model) that is used to transfer data most commonly to World Wide Web. Unlike UDP, this protocol ensures the order and the integrity of the data transmitted. This software flow control is achieved through a series of ACK & SYN packets among other checks. Any data that fails to meet proper software checks is re-requested by the receiver of the corrupt data and resent. Consequently, there is considerable overhead with this protocol since it is not a connectionless protocol.

In regards to the actual transmission of data. TCP appear as a host/client relationship.



To the left is a diagram symbolizing how most TCP transactions of data occur. Generally a device (server) is listening on a bus and waiting for clients to request data from it. Opening up a live session in wireshark and connecting to a website via a browser will display this flow control. Consequently, debugging TCP as opposed to UDP data is considerably more difficult since much of the data that will appear on the bus is simply error-checking.

Luckily for our application, our W5200 Ethernet ASIC has a built in TCP/IP stack which will handle the flow control required for TCP communication. As a result of this and the scope of the class, I will not go into further detail on the protocol. However, more information can be easily obtained on the internet should you wish to learn more.

Numerous protocols communicate over TCP such as HTTP which brings us to our next protocol of brief study.

HTTP: Hypertext Transfer Protocol

HTTP, the Hypertext Transfer Protocol is a communication protocol on top of TCP which generally communicates with the WWW (World Wide Web). HTTP is a request/response protocol which is considered to be on the application layer of the OSI model. Examples of these requests include but are not limited to GET, HEAD, POST, PUT, DELETE, and CONNECT.

This protocol is generally conducted in sessions where a HTTP client will request information of a HTTP host. For example, a web browser is the HTTP Client and when one types in a URL of a website and hits enter, they are making a HTTP GET request from the HTTP HOST . A GET request contains relevant information to the transfer of data such as the HTTP server's URI (Uniform Resource Identifiers) or URL (Uniform Resource Locators) , the HTTP clients acceptable data formats…ect. Below is an example of a GET request I made when I tried to connect to a website:

Example GET request from HTTP Client

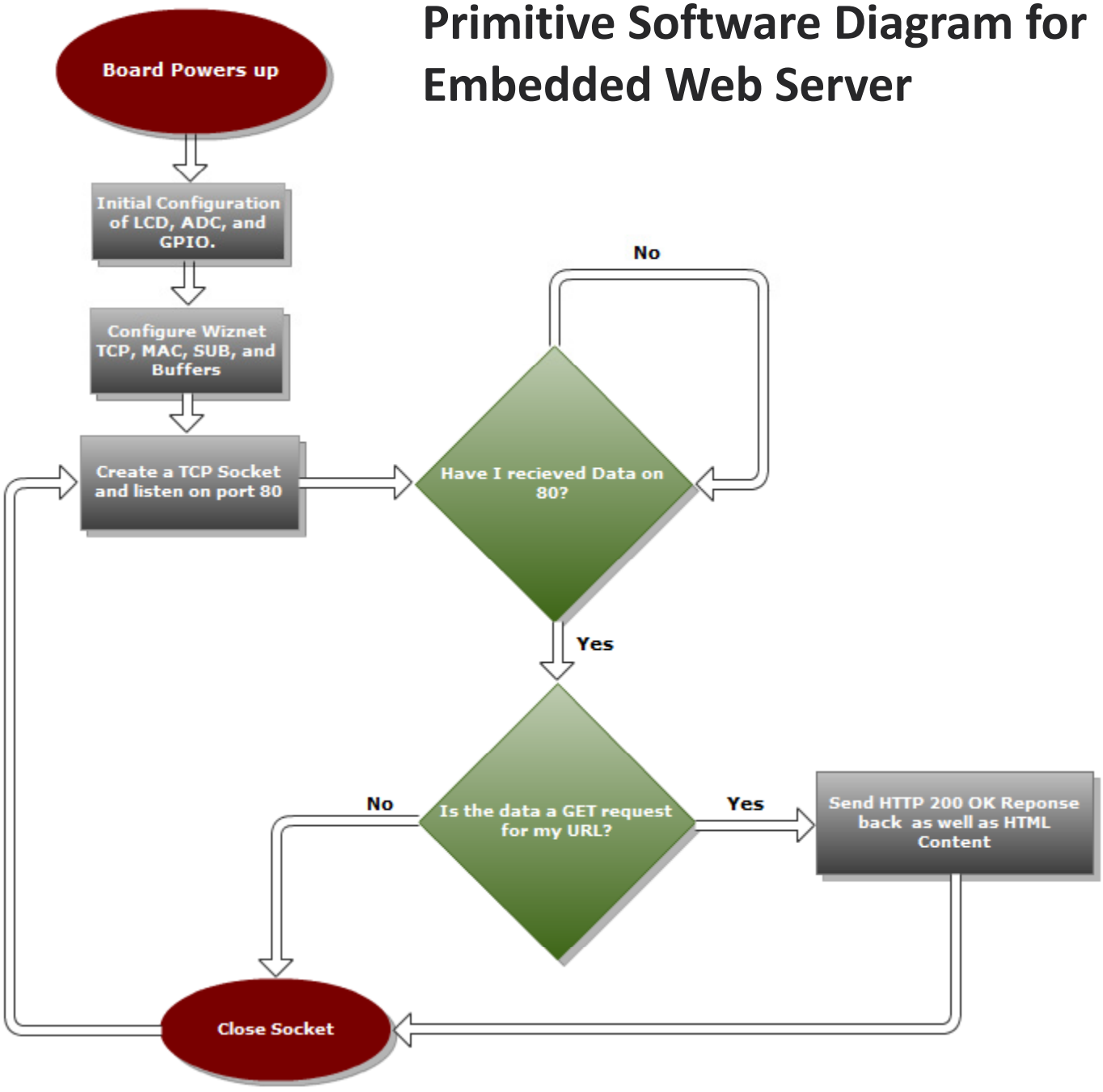
GET / HTTP/1.1
HOST: www.mysite.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW 64; rv:1:.9.0) Gecko20100101 Firefox/19.0
Accept: text/html,application/xhtml + xml,application/xml,;q=0.9,/*/*;q=0.8
Accept-Language: en-US,en;q=0.5
….

Example GET response from HTTP Server

HTTP/1.1 200 OK
Server: qws/0.0
Content-Type: text/html
Content-Length: 440
….Data such as HTML

The GET request informs the server about the client. Once the server decides how to interpret the GET request, the server will send back a response. This response can include data such as the entire data length, the file types it will be sending, and cache information. In addition, these responses generally have raw data with them such as HTML data.

Primitive Software Diagram for Embedded Web Server



Trouble on where to start?
Study the following files in the example code:

main.c (obviously)
tcp.c/h
web_page.h

Lab Report Guidelines:

1. Upload your code files to your assigned GIT repo. You must first create an account on github and email me your account name in order to gain access to your repo.

2. If you are having trouble getting GIT to work initially, email me your code at nathan.zimmerman@my.ndsu.edu . The code should be attached in the form of .c and .h files or in a complete zip archive . Your email title should be : Lab_Number : Myname. For example: Lab_1 : Nathan Zimmerman. This method of turning in assignments is cumbersome and will not be available long term so attempt to get GIT working ASAP.

   You are allowed to work in groups but you are all expected to turn in your own individual code. You should write the code yourself and it should not be a duplication of someone elses code.

Grading:

100% **: Code runs and by majority meets requirements**

Xx% **: An attempt was made to implement as many of the requirements as possible for partial credit.**

0% **: Code was not turned in or significantly falls short of meeting requirements.**