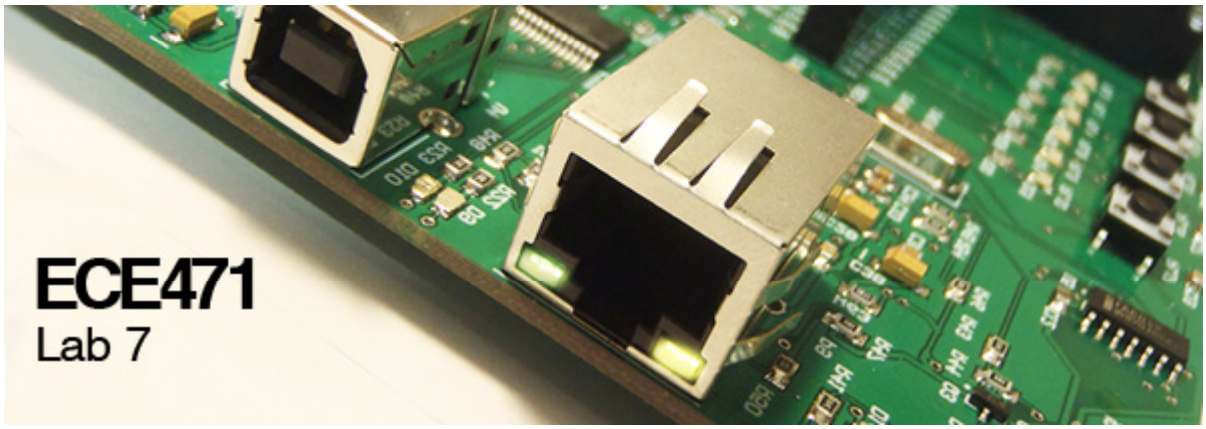


NDSU Embedded Systems II

UDP Ethernet

Flying the AR Drone

Objective: To obtain familiarity with the following:

- Ethernet
- TCP/IP Protocols

  UDP: User Datagram Protocol

By
Nathan Zimmerman

Requirements for lab 7:

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 UDP, a TCP/IP protocol, over an Ethernet connection. The application the UDP packets sent over the network will be used in order to control the AR Drone.

Requirements for Lab 7:

Our goal is to create a control system in FreeRTOS for the AR Drone:

1. Formulate UDP packets for AR Drone commands such as land, takeoff, move up, move down, move left, move right, move forward, move backwards rotate right, rotate left.
2. Implement some physical hardware interface (such as buttons / joystics) to interface the user to these commands. Create a separate FreeRTOS Task to manage these inputs.
3. Create a task for sending data to the drone. This task will need to manage info from the physical interface task as well to ensure Watch Dog Commands are sent to the drone every 50ms to stop it from crashing.

Result should look similar to Lab7.axf .

Useful Resources:

AR Drone 2.0 SDK User Guide:

www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf

Wireshark:

Download & Info:
http://www.wireshark.org/

Example UDP Packet code for W5200:

(See repo for lpcxpresso project)

PC WINDEV16 AR Drone Control AP:

http://repository.windev.com/resource.awp?file_id=76&lang=UK
(Use this AP to control the AR Drone from your computer)

Google!

If you are un familiar with any term used in this lab writeup, google it! The best learning is done based upon independently motivated study.

Ethernet:

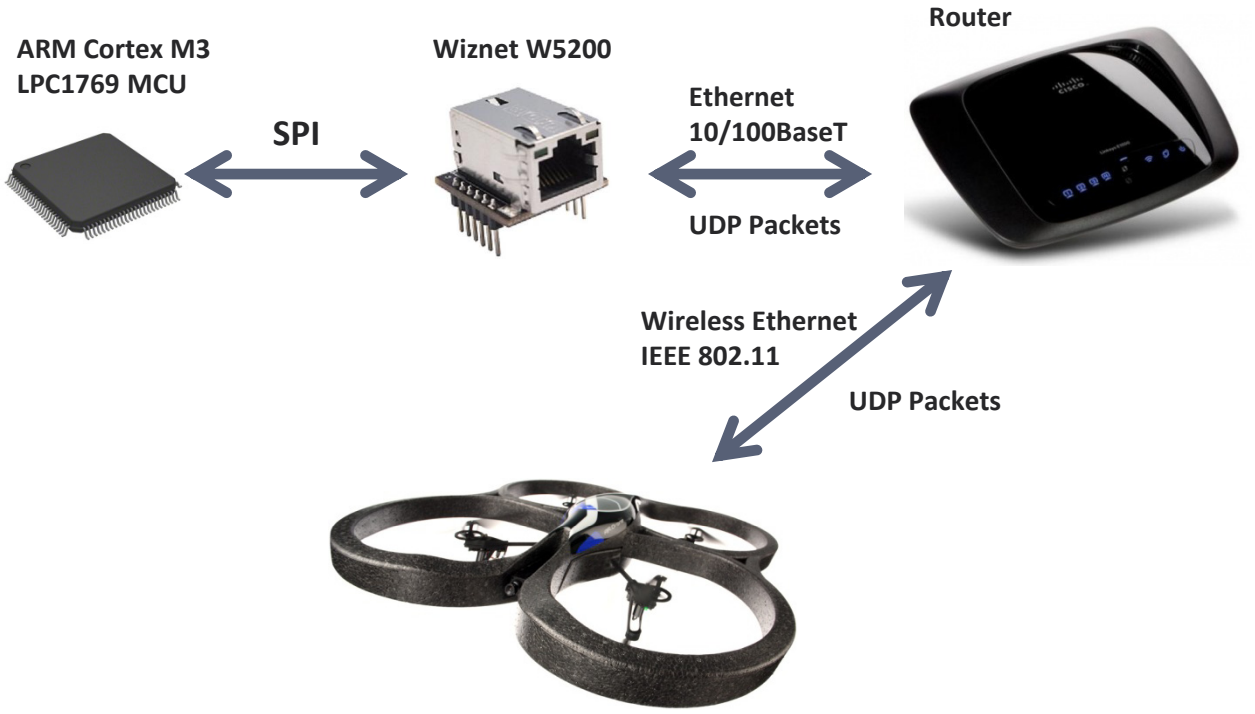
Ethernet is a series of specifications by IEEE 802.3 to implement a physical layer that is used for sending data over Local Area Networks. The physical layer of a network references the literal physical medium such as wires that raw data bits will travel over. The physical layer is also a with respect 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 User Datagram Protocol(UDP) which is the communication that is used in order to send commands to the AR Drone 2.0. Below is a example of the OSI Model.

Layer #	Layer Name	Data Type	TCP/IP Protocol
7	Application	Data	Our C Code
6	Presentation	Data	----
5	Session	Data	----
4	Transport	Segments	UDP
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.



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.

AR DRONE SDK Manual:

The Parrot released an SDK (software development kit) manual to go with their product to assist open source developers for app creation. This manual is crucial in order to understand how to control the AR Drone.

AR Drone 2.0 SDK User Guide:

www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf

Ensure you read section 2.10 and all of chapter 4 and 6, and 7.1.2 in this manual. This contains critical information for flying the drone.

In summary, the drone is controlled via UDP Packets. These packets must be sent over UDP on port 5556 for basic control commands. The drones IP address is 192.168.1.1 and will by default broadcast its own network. The drone then uses ASCII commands in UDP packets for various actions. For example:

One element in controlling the drone is continuously resetting the watch dog timer on the drone via UDP packets. A udp packet sent to 192.168.1.1 on the same network with a source & destination port of 5556 with the following ascii data: “AT*COMWDG=23\r” would reset the watchdog. Note that the ascii “AT*COMWDG=” is the WDG reset command. “23” is the packet #. This 16bit number must continuously increment every time one sends a UDP packet. Lastly, the “\r” is the ascii character for carriage return. This carriage return terminates every command.

Before writing your own code, it would be wise to fly the AR Drone 2.0 with either the PC AP, or my example code and run an instance of wireshark in order to obtain the UDP packet data that is transmitted while communicating with the drone. Simply cloning this and re-broadcasting the packet data will allow you to control the drone.

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.

Setting up Drone Communication

By default, the drone will broadcast its own SSID such as ARDRONE2.0_14873482 or something of that nature. This works fine for connecting to the drone and controlling it with an IPAD or phone device. However, this does not work when attempting to connect it to the W5200 Ethernet controller since that uses a physical Ethernet connection. Consequently, you will need to use a wireless router in order to control the drone. Furthermore, you will need to establish a telnet terminal with the drone and tell the drone to connect to the SSID of your router. Below are the instructions to do this w/ putty.

1. Set your PCs IP address to 192.168.1.2
2. Connect to the wireless network being broadcasted from the drone.
3. Ping the drone in command prompt via its IP address (192.168.1.1) and verify that you are connected to it.
4. Open up putty and open up a telnet session with the ip 192.168.1.1
5. Send the following command over telnet to the AR Drone

iwconfig atho mode managed essid drone_control ; ifconfig atho 192.168.1.103 netmask 255.255.255.0 up

Paste this command into telnet without any extra spaces / line returns and then hit enter key once entered into telnet. This will set the drones SSID to “Drone_Control” You will have to match it with the SSID of your router. This command also sets the drone IP address to 192.168.1.103. This will have to be repeated every time power up the drone. Final steps below:

1. Verify that the drones wireless network is no longer broadcasting.
2. Optional: Connect your PC to the wireless router and ping the new IP address of the drone to verify that the drone is indeed connected to your wireless router.

You should now be ready to send UDP commands to the drone provided your W5200 is plugged into your wireless router and your wireless router is configured properly.

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.**