This lesson will not be held in the computer lab because it requires hands-on access to hardware that we have a limited quantity of. Parts 1, 2 and 3 of this document are independent and do not need to be done in order, although some instructions will refer to previous sections. For problems, refer to the section labeled “Note on differing versions of Visual Studio”.

Part 1: Garmin GPS unit
There are two Garmin hand-held GPS units set up on ucrobotics 6 and 7.

Begin by turning on the unit: hold the red button until it starts up, then press enter two or three times to get past the splash screens.

Then press menu, and enter, starting the simulator, which will provide data for you without finding an actual signal.
Next open Hyperterminal.
Name the connection (it doesn't matter what), set it to use **COM1, baud rate 4800**.
When you click OK, you should see data like that in the image below. Congratulations, you have connected to the Garmin.
If you want to understand the data being shown, look up the NMEA protocol. I have used the following references: http://www.kh-gps.de/nmea-faq.htm or http://home.mira.net/~gnb/gps/nmea.html. This is beyond the scope of this lesson.

Accessing the com port in C:
Next, after closing HyperTerminal (two programs cannot access the serial port simultaneously), open Visual Studio, and start a new blank win32 console application (as usual). Compile and run the code in the appendix.

The OpenComPort function below opens a com port using built in functions in windows.h. The com port is initially opened as a file by windows, but then functions specific to com ports are used to setup
the com port as desired. The return value is a “HANDLE”, which is a data type that just keeps track of something, whether a file, button, window, etc.

Then ReadFile commands are issued to the handle to the com port, requesting a single character at a time. If this character is an endline, the line is displayed and the outer loop repeats. The close statement at the end is important, because only one program is allowed to access a port at a time, so improperly closing a serial port can result in that port being unusable until the computer is restarted.

When finished, turn off the Garmin units by holding the red button.

Parsing this information and programming the robot to use it will be left to future lessons.

Part 2: Honeywell Digital Compass

The digital compass is on the cub, and I am not going to move it. Open HyperTerminal on the cub laptop, and enter a connection name. I believe the com ports on the hub are assigned such that port 1 is com3, 2 is com4, etc, but I do not trust the computer to not reset that. The digital compass should be connected to com5. It runs at 19200bps. You should see a string of numbers representing compass headings.

Follow the directions on the previous page under “Accessing the com port in C”, but modify the code in the appendix such that the line opening the com port is correct, i.e.
HANDLE m_hFile=OpenComPort(5,CBR_19200);

Build and run the program.
Part 3: Novatel GPS unit

On ucrobotics2, the computer with 2 monitors on the center left, there is a Novatel OEMV GPS unit set up. On that computer, open HyperTerminal as before, using COM1, baud rate 9600.

Go to properties, select the “settings” tab and click "ASCII Setup". Select the checkbox "echo typed characters locally". This will allow you to see what you are typing into the program.

Type "LOG COM1 GPGGA ONTIME .1" and press "enter". This command tells the Novatel to output NMEA GGA messages on its com1 port every .1s. Since the antenna is inside, it will not get a signal, and you will only see the following:

This setting is lost each time the Novatel is powered off, so must be reentered each time.

To run this device in C++, we use the same procedure (in Accessing the com port in C) and almost the same main function (see Appendix C), modified to send the command line above to the device first, using a WriteFile command, along with the same OpenComPort function as before. Cycle power (unplug and replug in the Novatel) and run this code.
Note on differing versions of Visual Studio:
Some computers have multiple versions of Visual Studio; if one doesn't work correctly, e.g. Doesn't have an option for C++ projects, try another version.
Some versions of Visual Studio do not have a “Win32 Console Application” project; instead select Win32 Application, and select Console Application in the project settings dialog afterwards.

Some versions of Visual Studio make it hard to disable unicode:
First, go to properties > C/C++ > General and change Use UNICODE response files to No.

Next, go to C/C++ > Preprocessor, and click on the “...” under preprocessor definitions.
Then deselect “Inherit from parent or project defaults”.
Appendix A: Serial port code

```c++
#include <windows.h>
#include <iostream>
using namespace std;

HANDLE OpenComPort(int nPortNo=1, int BaudRate = CBR_4800, int ByteSize = 8, int Parity = NOPARITY, int StopBits = ONESTOPBIT);

void main(void)
{
    int jj; //loop variables
    char ch;
    bool error;//variable to check for errors reading serial port (not used here).
    char line[257];//buffer used to store 1 line of output
    DWORD numberOfBytesRead = 0;  //variable receives number of bytes read reported by ReadFile
    HANDLE m_hFile=OpenComPort(1,CBR_4800);//open com port
    if (m_hFile==NULL)
    {
        cout << "Com port failed. Check hardware, and make sure no other program is accessing."<<endl;
        return;
    }

    for (jj=0;jj<100;jj++)
    {//read 100 lines from serial port
        error = ReadFile(m_hFile,LPVOID(line),255,&numberOfBytesRead,NULL);
        line[numberOfBytesRead]=0;//terminate string here, so no random data is displayed.
        cout <<line<<endl;//display data.
        Sleep(50);
    }
    CloseHandle( m_hFile ) ;//close port
}
```
HANDLE OpenComPort(int nPortNo, int BaudRate, int ByteSize, int Parity, int StopBits)
{
    char strPort[256];
    HANDLE m_hFile;
    BOOL fSuccess;
    COMMTIMEOUTS timeout;
    DCB dcb;
    sprintf(strPort, "COM%d", nPortNo);

    //Create the handle for read the COM port
    m_hFile = CreateFile(strPort, GENERIC_READ | GENERIC_WRITE, (DWORD)NULL, // exclusive access
                          NULL, // no security
                          OPEN_EXISTING,
                          FILE_ATTRIBUTE_NORMAL, NULL); // hTemplate

    if(INVALID_HANDLE_VALUE == m_hFile)
        //Check whether the handle is valid or not.
        cout<<"Error: Could not open the RS232 port!"<<endl;
        return NULL;

    SetupComm(m_hFile, (DWORD)2048, (DWORD)2048); // setup the com parameters now that we have a handle

    // Set up the DCB with our settings
    fSuccess = GetCommState(m_hFile, &dcb); // Get it first so we fill all members
    if (!fSuccess)
        // Handle the error.
        cout<<"Error: Failed to Get the System Communication Settings."<<endl;
        return NULL;

    dcb.BaudRate = BaudRate; // baud rate
    dcb.ByteSize = ByteSize; // data size
    dcb.Parity = Parity; // No Parity Bit
    dcb.StopBits = StopBits; // stop bits
    fSuccess = SetCommState(m_hFile, &dcb); // assign it

    if (fSuccess == 0)
        //Now check the configuration of the communication device is valid or not after assign the com parameters
        cout<<"Error: In Control Setting for a Serial Communications Device."<<endl;
        return NULL;

    // Set up the timeouts to use, they are quite short, since we will loop anyway.
    // Do not make them zero, else we will have a CPU load problem. Too large a value,
    // and we have to wait for comms to time out when shutting down.
    GetCommTimeouts(m_hFile, &timeout); // fill timeout structure
    timeout.ReadIntervalTimeout = 100; // 500ms between incoming chars.
    timeout.ReadTotalTimeoutConstant = 500;
    timeout.ReadTotalTimeoutMultiplier = 0;
    timeout.WriteTotalTimeoutConstant = 2000;
    timeout.WriteTotalTimeoutMultiplier = 60; // 60ms per char sent
    SetCommTimeouts(m_hFile, &timeout);
    return m_hFile;
}
Appendix B: Serial port source code for Novatel OEMV (uses OpenComPort function above)

```cpp
#include <windows.h>
#include <iostream>
using namespace std;
HANDLE OpenComPort(int nPortNo=1, int BaudRate = CBR_4800, int ByteSize = 8, int Parity = NOPARITY, int StopBits = ONESTOPBIT);

void main(void)
{
    int jj; //loop variables
    bool error; //variable to check for errors reading serial port (not used here).
    char line[257]; //buffer used to store 1 line of output
    DWORD numberOfBytesRead; //variable recieves number of bytes read reported by ReadFile
    HANDLE m_hFile=OpenComPort(1,CBR_9600); //open com port
    if (m_hFile==NULL)
    {
        cout << "Com port failed. Check hardware, and make sure no other program is accessing."<<endl;
        return;
    }
    strcpy(line,"LOG COM1 GPGGA ONTIME .1\n");
    WriteFile(m_hFile,LPVOID(line),25,&numberOfBytesRead,NULL);
    for (jj=0;jj<100;jj++)
    { //read 100 lines from serial port
        error = ReadFile(m_hFile,LPVOID(line),255,&numberOfBytesRead,NULL);
        line[numberOfBytesRead]=0; //terminate string here, so no random data is displayed.
        cout << line<<endl; //display data.
        Sleep(50);
    }
    CloseHandle( m_hFile ); //close port
}
```