NORTHERN ILLINOIS UNIVERSITY

UNMANNED AERIAL VEHICAL

A Thesis Submitted to the

University Honors Program

In Partial Fulfillment of the

Requirements of the Baccalaureate

Degree

With University Honors

Department of Electrical Engineering

By Kevin Wilson

DeKalb, Illinois

May 10th, 2002
Abstract

The project's goal is to design and build an Unmanned Aerial Vehicle whose view and instruments can be viewed from a computer. This airplane will allow for surveillance and casualty inspection. Currently a full size airplane or helicopter must be used to inspect an area that has been through a major casualty. For example, during a forest fire an airplane or helicopter must be used to search the area for survivors or people that are trapped. With a low cost unmanned airplane, the search will be conducted faster and with fewer personnel. The Unmanned Aerial vehicle will also be used in situations where personnel would be in danger. During the nuclear accident at Chernobyl manned helicopters were used to inspect the damage, unfortunately the personnel on these helicopters were subjected to dangerous amounts of radiation. An unmanned aerial vehicle could have been used instead of the manned aircraft to protect the personnel from this harmful radiation.
Report

In order to figure out what parts to use the project's goals must be examined. One of the goals of this project was to use only one communication link between the airplane and the ground. This will simplify the airplane's internals. The alternative to one transmitter would be to install a UHF transmitter to transmit the video and a simple data packet transmitter for the data. Using two transmitters increases the chance of interference between the two and increases the power output required, thus lowering battery life. To get the highest bandwidth and communication compatibility between the airplane and ground computer a wireless Ethernet approach was investigated.

The project incorporates several sensors to collect the airplane's data, for example a pressure sensor for altitude and an electronic compass system using an analog output. The video is digitized and sent back to the base computer via the wireless Ethernet.
All these considerations were taken into account and it was decided to use a PC/104 based embedded computer. The PC/104 refers to the 104-pin bus on the computer card that is used to connect different PC/104 devices. This allows for great flexibility in the same way one can add to their personal computer using the ISA slots. The Micro/Sys Netsock 400 PC/104 based computer was chosen due to it's on card Ethernet port, two RS-232 communication ports and eight-channel analog to digital converter. The Netsock 400 uses an Intel 133 Megahertz 486DX processor with four megabytes of ram and eight megabyte disk on chip C: drive. MSDOS 5.0 is used as the operating system.

To digitize the video from a standard video camera a digital frame grabber is used. Since this video will have to be broadcast over a wireless Ethernet a frame grabber with on board video compression was needed, therefore the Sensoray Co. model 512 Video Frame Grabber was chosen. The model 512 uses the PC/104 bus for easy of integration and uses MSDOS drivers.
to control the data communication between the frame grabber and the computer. The 512 uses the MPEG-2 video compression, which is decoded by the ground computer using the Elecard MPEG-2 Decoder.

The video camera must be small, have a high resolution, and give a good picture of the horizon at a large distance. The choice had to be made to go with a CMOS based camera or a CCD array camera. CMOS cameras are cheaper and consume less power and a CCD camera gives a better picture and cost a little more. The Images Co. WDCC-3200S Color Board Camera with the standard lens was chosen. The camera uses a 1/3" Color CCD Image Sensor and has 420 lines of horizontal resolution for a clear picture.

The altitude sensor is the Motorola MPX4115A fully integrated pressure sensor. The MPX4115A has a linear analog voltage output for changes in pressure. Connecting the output voltage to one of the PC/104 computers analog to digital converters the pressure can be calculated, thus giving us an accurate value for calculating altitude.

Speed sensing is accomplished by the Motorola MPX2010 differential pressure sensor. The MPX2010 is a differential pressure sensor that uses the difference in
pressure on a pitot tube and the atmosphere. The output is an analog signal proportional to the difference of the two pressures.

The Pitch and Roll is determined by using the AOSI EZ-TILT-3000-045. This dual axis tilt sensor gives an output voltage proportional to sensor tilt. The sensor has a ±70° linear output range. Like the altitude sensor the output voltage is directly connected to one of the channels of the analog to digital converter. This device uses a reference voltage that is compared to the output voltage to determine tilt, thus a total of three channels of the analog to digital converter are used. The voltages are read by the PC/104 computer, subtracted from each other, and then sent to the ground computer via the Ethernet.

The compass chosen is the Dinsmore Sensors 1655 Analog Compass. The analog compass has two channels for output.

![Graph](image)

Channel one is the north-south direction and channel two is...
the east-west direction. An interesting decoding algorithm had to be implemented since the two outputs are sinusoidal based on direction then added to a DC offset. First a ratio is taken between the actual voltage output and the DC offset for the two channels. Next, if the absolute value of the north channel’s ratio is less than 0.7 the north channel is used for the direction. Now a quick comparison of the west channel to zero gives the aircrafts direction as being between 45 degrees and 135 degrees or between 225 degrees and 315 degrees. The same logic is used for the west channel if the north channel’s ratio is above 0.7.

```c
//****** Compass Algorithm **************
double NorthAngle, WestAngle;
int CompassNorthMedian = 2086; // (2640+1532) / 2
int CompassWestMedian = 2076; // (2638+1514) / 2

NorthAngle = (CompassNorth - CompassNorthMedian) / 555;
WestAngle = (CompassWest - CompassWestMedian) / 562;

if (NorthAngle > 1)
    NorthAngle = 1;
if (NorthAngle < -1)
    NorthAngle = -1;
if (WestAngle > 1)
    WestAngle = 1;
if (WestAngle < -1)
    WestAngle = -1;

if (NorthAngle > 0.7)
    Compass = 0 - int(asin(WestAngle) * 180 / PI + 0.5);
if (WestAngle < -0.7)
    Compass = 90 - int(asin(NorthAngle) * 180 / PI + 0.5);
if (NorthAngle < -0.7)
    Compass = 180 + int(asin(WestAngle) * 180 / PI + 0.5);
if (WestAngle > 0.7)
```
Compass = 270 + int(asin(NorthAngle) * 180 / PIE + 0.5);
if (Compass < 0)
    Compass = 360 + Compass;

Sending the output of the sensor to two more channels of the analog to digital converter allows us to send this information back to the computer to be converted to a heading.

The wireless Ethernet has to be fast with a long operating range. The data transfer rate on a wireless LAN (local area network) depends on the distance between the two transmitters. As the distance increases the signal to noise in the signal drops the transmitter has to slow down to ensure the data is being sent correctly. The Proxim RangeLAN2 7921 Ethernet Adapter was chosen for the plane’s Ethernet transmitter and the Proxim RangeLAN2 7421 PCMCIA with dipole antenna was chosen for the base computer. These operate on a 2.4 gigahertz transmitters that use a series of frequency hops so that they can be operated without a FCC license. These components did not perform as they were expected to. The maximum range that the components are functional at is 1800 feet outdoors, according to the data sheets. The tested maximum range outdoors was only about 300 feet. This may be due to the
type of antennas that were provided and the interference caused by the surrounding buildings at the time of testing.

The power system for the airplane must be lightweight and provide enough power for one hour of operation. To achieve this a 12V lead acid battery was chosen. For the power and weight the lead acid battery was the best choice. The power requirements for the electronic equipment for the plane are as follows.

<table>
<thead>
<tr>
<th>Part Name</th>
<th>Current Draw mA</th>
<th>dc Voltage dc</th>
<th>Power watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC-104 Board</td>
<td>850</td>
<td>5</td>
<td>4.25</td>
</tr>
<tr>
<td>MPEG2 Frame Grabber</td>
<td>280</td>
<td>5</td>
<td>1.4</td>
</tr>
<tr>
<td>Board Camera</td>
<td>100</td>
<td>12</td>
<td>1.2</td>
</tr>
<tr>
<td>Pitch and Roll sensor</td>
<td>2</td>
<td>5</td>
<td>0.01</td>
</tr>
<tr>
<td>Compass sensor</td>
<td>19</td>
<td>5</td>
<td>0.095</td>
</tr>
<tr>
<td>Altitude sensor</td>
<td>10</td>
<td>5</td>
<td>0.05</td>
</tr>
<tr>
<td>Speed sensor</td>
<td>6</td>
<td>12</td>
<td>0.072</td>
</tr>
<tr>
<td>Wireless Lan</td>
<td>300</td>
<td>12</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Because of the power loss due to using voltage regulators, a dc to dc converter from Lambda was used, the model number is PM20-12S05. With the loses of the converter calculated with the power requirements of the five and twelve volt applications the current draw from a battery would be just under one amp. So, due to the size and weight limitations, a one point two amp hour battery from Yausa was selected for use in the plane to supply the plane the required power for the estimated preparation and flight time of fifteen minutes. A recommended power decoupling and output filter
circuit was fabricated for the altitude sensor, along with the power delivery circuit and all of the wire harnesses to hook up the sensors to the power delivery system and to the A/D on the computer board.

Upon experimental implementation of the system, with everything running, the current draw from a power supply is one point three amps. Due to this the battery last only about twenty minutes, which is only good for one theoretical flight. I have checked the calculations several times, so either the advertisement of the efficiency of the regulator or advertisement of the current draws of the individual components is wrong.

When it came time to install the components into the donated plane several modifications had to be made. First the rudder and tail flap servos had to be moved rearward in the plane and a special bracket was fabricated to hold the throttle servo in place. This allowed the computer board to be mounted vertically next to the battery, which was pushed as far forward as possible along with putting the original battery, camera and power delivery system in front of them. All of this was done to try and compensate for the relocation of the servos and the installation of the LAN and the sensors behind the computer board in an attempt to keep the center of gravity in the original position. After
installation of the components it appears that the center of gravity is in the original position, but the real test will be to see if it flies.

The ground computer is a one-gigahertz laptop running Windows ME. The programming environment for the graphical user interface is Microsoft Visual C++. Visual C++ was chosen due to the programmer’s familiarity with the C and C++ language. Still a lot of programming skills needed to be learned.

All the data communications take place in a thread that is spun off from the main application. The thread allows the processor to share its time between retrieving the data and displaying it. These functions are run in a worker thread called CDataNetwork. Windows 98 allows this type of multitasking with a call to AFXBEGINTHREAD().

```cpp
void CUAVDoc::StartDataNetwork()
{
    RunDataNetwork = TRUE;
    CDataNetwork* m_pDataNetwork = (CDataNetwork*)
        AfxBeginThread(RUNTIME_CLASS(CDataNetwork),
        THREAD_PRIORITY_BELOW_NORMAL, 0,
        CREATE_SUSPENDED);
    m_pDataNetwork->SetOwner(this);
    m_pDataNetwork->ResumeThread();
}
```

This code starts the network thread then passes it a pointer to CUAVDoc class so variable retrieved can be passed back.
By creating a socket or a communications link between the two computers using an IP address and a socket data can be sent through the Ethernet connection. Using Winsock function such as recvfrom() and sendto() are used to retrieve and send variables over the Ethernet.

BOOL CDataNetwork::InitInstance()
{
    // TODO: perform and per-thread initialization here
    int err = 0;
    int error = 0;

    err = WSAStartup(0x0101, &DataSocketData);

    if (err != 0)
    {
        //ShowString.UpdateString("Error in
        WSAStartup!!!", 0, 0);
        //return (SOCKET_ERROR);
    }

    DataSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    CSockAddr local("192.168.1.50", 5001);
    err = connect(DataSocket, local, sizeof(local));

    if (err != 0)
    {
        error = WSAGetLastError();
        if (err == WSANOTINITIALISED)
            err = 1;
        if (err == WSAEFAULT)
            err = 2;
        if (err == WSAENOTSOCK)
            err = 3;
        //ShowString.UpdateString("Error in Connect!!!",
        0, 0);
        //ShowString.DoModal();
    }
    StartDataNetwork();
    return TRUE;
}
A local socket address is made which establishes a communication link between the two computers. In this case the IP address of 192.168.1.50 is the IP address of the PC104 computer and the port 5001 is the data port that the ground computer will listen on. Now all that has to be done is send the request for data with a simple send() function call or receive data with a recv() call.

```c
iAmtSent = send(DataSocket, strMyMessage, iLen, 0);

iAmtRecv = recv(DataSocket, RecvDatagram,
                 sizeof(RecvDatagram), 0);
```

These two functions place data on or listen to the specified datasocket and return an integer value of the amount sent or received. This provides for a nice two-way communication method between the plane and the ground.

The variables received from the airplane are then converted to the specified parameter and displayed on the screen in a gauge. The gauges are done as objects to the document class. First these objects must be given a CView pointer or device context pointer to the view class so the objects know where to draw the gauges. This is accomplished by a SetViewWindow().

```c
// Code to pass the pointer of the view window to Gauge Classes
POSITION pos = GetFirstViewPosition();
CView* pView = GetNextView(pos);
m_Altimeter.SetViewWindow(pView);
m_Horizon.SetViewWindow(pView);
```
After the CView window pointer is passed the gauges update the new variable and compute a new gauge drawing. The drawing is performed in a memory location before it is drawn to the screen to give the gauges a more fluid motion. This is accomplished by the CreateCompatibleDC() function.

```cpp
CBitmap Altimeter;
Altimeter.LoadBitmap(IDB_ALTIMETER);
CDC dcMemory;
dcMemory.CreateCompatibleDC(pDC);
```

Here the bitmap of the altimeter is loaded into the memory location pointed to by dcMemory. To draw the needles the angle of rotation must be figured out and points must be calculated. A call to Polygon() will connect the points and fill them in.

```cpp
double angle = m_Altitude * 2 * PIE / ALT_GUAGE_MAXIMUM;

POINT sides100[4];

// Compute positions for 100's gauge
sides100[0].x = int(CENTER_X + GUAGE_RADIUS * sin(angle) + 0.5);
sides100[0].y = int(CENTER_Y - GUAGE_RADIUS * cos(angle) + 0.5);
sides100[1].x = int(CENTER_X + 5 * cos(angle) + 0.5);
sides100[1].y = int(CENTER_Y + 5 * sin(angle) + 0.5);
sides100[2].x = int(CENTER_X - 5 * sin(angle) + 0.5);
sides100[2].y = int(CENTER_Y + 5 * cos(angle) + 0.5);
sides100[3].x = int(CENTER_X - 5 * cos(angle) + 0.5);
sides100[3].y = int(CENTER_Y - 5 * sin(angle) + 0.5);
```
The seven segment led is an object so that it can be in many different gauges. All that is required to place a digital display of your gauge is to first determine the ones, tens, and hundreds digits. Then make a call to the led’s DrawLedNumber().

```cpp
// % is remainder function
m_Ones.SetNumber(m_Altitude%10);
m_Tens.SetNumber((m_Altitude/10)%10);
m_Hundreds.SetNumber(int((m_Altitude/100))%10);
// Draw Digital Display
m_Ones.DrawLedNumber(10, 20, 116, 125, 0, &dcMemory);
m_Tens.DrawLedNumber(10, 20, 102, 125, 0, &dcMemory);
m_Hundreds.DrawLedNumber(10, 20, 88, 125, 0, &dcMemory);
```

When the gauge is finished in drawing in memory it should now be placed onto the screen with a call to BitBlt().

```cpp
pDC->BitBlt(755,250,200,200, &dcMemory, 0, 0, SRCCOPY);
```

pDC is pointer to the view windows device context pointer, which allows the memory window dcMemory to be drawn to the screen.

The horizon gauge needed to be drawn then masked to a black and white bitmap in order to get the affect of the horizon behind the indicator. This was achieved with a bitmask.h code from codeproject.com. The code takes two bitmaps one being the bitmap you want drawn on the screen and the other being a black and white bitmap indicating the areas you want drawn in black.
CBitMask(IDB_HORIZON, IDB_HORIZON_MASK, &dcMemory, CPoint(0, 0));

In this case the horizon is draw in the memory location, then the mask is used to insert just the colored parts of the bitmap to the memory location. The final result is realistic gauges with very smooth motion.

The PC/104 computer uses C programs and dos functions to execute its functions. The program uses software
interrupts by calling the int86() function, which runs an DOS program to retrieve information from analog to digital converters or data buffers for the video query.

```c
    cpureg.h.ah = MS_SPECIAL; /* special Micro/sys system function */
    cpureg.h.al = MS_ADCSTAT; /* check ADC conversion */
    int86(MSYS_INT, &cpureg, &cpureg);
    if (cpureg.h.bl)        /* if conversion is done, read value */
      return (cpureg.x.ax);
    if (goal < clock())
      break;
```

This code reads the analog to digital converters on the board and returns them to the calling function. The video is passed using the same int86() software interrupt and its own MSDOS commands.

```c
    Param->Buffer = vBuffer;
    regs.x.ax = X12_VI_GETDMADATA;
    //Select the "Send command' driver function
    regs.x.si = X12_SENDCOMMAND + CARD0;
    //Call the driver
    int86(X12INT,&regs,&regs);
```

This code loads the Frame Grabbers raw MPEG data from the video buffer and load it into a vBuffer contained in the Param structure. This data can then be sent over the Ethernet to the base computer.

The network connection is established using the WINsock API. The PC-104 board has the capability of having 16 sockets with each having a maximum datagram of 1024 bytes. One socket is used exclusively for data from the
sensors and another is set up for the video stream. A socket variable name is set up and then the socket() function is called to set the socket’s parameters. A bind() function is then used to associate a local port number to the newly created socket.

```c
SOCKET msgsock;
msgsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
err = bind(msgsock, &local, sizeof(local));
```

The PC-I04 board is set as a server in the plane. Two stages of programming were implemented for this project. The first being a program which allowed the base station computer to access only the A/D converters and the second extended the first program’s purpose to also transmit video data.

The data stage reads the A/D converters on the plane and sets up the data socket. After initiating the data socket, the program immediately enters an endless for() loop. This loop starts with the recvfrom() command.

```c
for(;;)
{
    fromlen = sizeof(from);
    numbytes = recvfrom(msgsock, datagram,
                        sizeof(datagram), FLAGS_ZERO, &from,
                        (int far *) &fromlen);
```

The PC-I04 will wait until it hears something from the base station computer over the data socket. The recvfrom() function returns the number of bytes that were contained in
the datagram that it received. recvfrom() also places the datagram received in an array that is called for in its function call. The first four characters in the datagram array are the command that is sent forth from the base computer. These characters are copied into another array called command. Using a series of if() statements, command is compared to the character codes for each of the A/D converters. The codes are user defined for each of the sensors. For example, “alti” was used for the channel 0, which was the channel the altitude sensor was connected to. Once the command is matched to the proper code, the A/D converter associated with the code is read using a function developed by Micro/sys called analogrd() and its value is placed in an array named parameters.

```c
if (strcmp (command, "alti") == 0)
{
    voltage = analogrd(0);
    sprintf(parameters, "=%03X", voltage)
    commanddone = 1;
}
```

The datagram array then has a “0” placed in the position after the command and the information in the parameters array is placed immediately after the “0”. The datagram is then sent to the base computer using the sendto() command and the loop is then repeated.

```c
err = sendto(msgsock, datagram, strlen(datagram),
     FLAGS_ZERO, &from, sizeof(from));
```
The second stage incorporates the data program with the necessary code to obtain and send the video stream. This program sets up the video socket and has a recvfrom() statement on the video port before the program continues into the endless loop. Once a message is received, the loop is entered and the video is retrieved using the X12_VI_GETDMADATA data command. This data is loaded into a buffer and the buffer is then transmitted to the base computer using the sendto() command. The buffer was created as a 1024 byte or character buffer so that once the data is obtained, it can be sent in a datagram unmodified. This second stage did work and transmit video data across the network. However, the data was never evaluated by the base computer due to programming difficulties. This may be because the data that was transferred may have been corrupted.

All the files and systems are managed by the MSDOS operating system, so an autoexec.bat file needs to be placed in the C drive to tell the computer which programs to run. The video drivers and the path to the video parameters folder are among the items that need to be loaded with every start.

SET X12PATH=C:\X12DOS\ C:\X12DOS\X12TSR
Also, the C program that runs the data and video transfer to the base computer must be called by this file, so that the board will load the file with a user command each startup.

Many lessons have been learned integrating all the sensors to the CPU; and then transmitting the output of the CPU to the base station computer. By using the PC104 board for the project many upgrades will be possible, including computer-controlled flight with no operator control. This system equipped with GPS could allow the airplane to fly from one destination to another with no operator assistance.
Code

UAVDoc.cpp

// UAVDoc.cpp : implementation of the CUAVDoc class
//

#include "stdafx.h"
#include "UAV.h"
#include "UAVDoc.h"
#include "DataNetwork.h"
#include "VideoPlayer.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

///////////////////////////////////////////
// CUAVDoc
IMPLEMENT_DYNCREATE(CUAVDoc, CDocument)

BEGIN_MESSAGE_MAP(CUAVDoc, CDocument)
    //{{AFX_MSG_MAP(CUAVDoc)
    //    // NOTE - the ClassWizard will add and
    //    // remove mapping macros here.
    //    // DO NOT EDIT what you see in these
    //    // blocks of generated code!
    ////}}AFX_MSG_MAP
END_MESSAGE_MAP()

///////////////////////////////////////////
// CUAVDoc construction/destruction

CUAVDoc::CUAVDoc()
{
    // TODO: add one-time construction code here
}

CUAVDoc::~CUAVDoc()
{
}

- 22 -
BOOL CUAVDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    // TODO: add reinitialization code here
    // (SDI documents will reuse this document)
    /*
    IGraphBuilder *pGraph;

    HRESULT hr = CoCreateInstance(CLSID_FilterGraph,
                                   NULL, CLSCTX_INPROC_SERVER,
                                   IID_IGraphBuilder, (void **) &pGraph);
    */

    // Code to pass the pointer of the view window to
    // Gauge Classes
    POSITION pos = GetFirstViewPosition();
    CView* pView = GetNextView(pos);
    m_Altimeter.SetViewWindow(pView);
    m_Horizon.SetViewWindow(pView);
    m_SpeedGuage.SetViewWindow(pView);
    m_Compass.SetViewWindow(pView);

    StartVideo(pView);
    StartDataNetwork();

    return TRUE;
}

isAdmin

{ // TODO: add loading code here }
}

/#define _DEBUG

void CUAVDoc::AssertValid() const
{ 
  CDocument::AssertValid();
}

void CUAVDoc::Dump(CDumpContext& dc) const
{ 
  CDocument::Dump(dc);
}
#endif // _DEBUG

void CUAVDoc::StartDataNetwork()
{
  RunDataNetwork = TRUE;
  CDataNetwork* m_pDataNetwork = (CDataNetwork*)
  AfxBeginThread(RUNTIME_CLASS(CDataNetwork),
                 THREAD_PRIORITY_BELOW_NORMAL,
                 CREATE_SUSPENDED);
  m_pDataNetwork->SetOwner(this);
  m_pDataNetwork->ResumeThread();
}

void CUAVDoc::StartVideo(CView* View)
{
  RunVideoPlayer = TRUE;
  CVideoPlayer* m_pVideoPlayer = (CVideoPlayer*)
  AfxBeginThread(RUNTIME_CLASS(CVideoPlayer),
                 THREAD_PRIORITY_NORMAL,
                 CREATE_SUSPENDED);

-24-
m_pVideoPlayer->SetOwner(this);
m_pVideoPlayer->SetViewWindow(View);
m_pVideoPlayer->ResumeThread();
}

void CUAVDoc::UpdateAirplaneData()
{
    int iPitch;
    int iRoll;
    int iAltitude;
    int iHeading;
    int iBattery;
    int iSpeed;

    CSingleLock LockVariables(&m_mutex);

    LockVariables.Lock();

    iPitch = m_pitch;
iRoll = m_roll;
iAltitude = m_altitude;
iHeading = m_heading;
iBattery = m_battery;
iSpeed = m_Speed;
    LockVariables.Unlock();

    m_Altimeter.SetAltitude(iAltitude);
m_Horizon.SetHorizon(iPitch, iRoll);
m_SpeedGuage.SetAirspeed(iSpeed);
m_Compass.SetCompass(iHeading);
}

DataNetwork.cpp

// DataNetwork.cpp : implementation file
//
#include "stdafx.h"
#include "UAV.h"
#include "DataNetwork.h"
#include "UAVDoc.h"
#include <math.h>

#ifdef _DEBUG
#define new DEBUG NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;

- 25 -
#endif

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n
CDataNetwork::CDataNetwork()
{
    BatteryVoltage = 0;
    CompassWest = 0;
    CompassNorth = 0;
    CompassMax = 0;
    CompassMin = 5000;
    Compass = 0;
    Altitude = 0;
    AltitudePtr = 0;
    SpeedPtr = 0;
    Roll = 0;
    Pitch = 0;
    Speed = 0;
    VoltRef = 2100;
    m_bAutoDelete = TRUE;

    int i;
    for(i = 0; i < 20; i++)
    {
        if (i < 5)
            Speed_Buffer[i] = 0;
        Altitude_Buffer[i] = 3133;
    }
}

CDataNetwork::~CDataNetwork()
{
}

BOOL CDataNetwork::InitInstance()
{
    // TODO: perform and per-thread initialization here

    int err = 0;
    int error = 0;

    err = WSAStartup(Ox0101, &DataSocketData);
if (err != 0)
{
    //ShowString.UpdateString("Error in WSAStartup!!!", 0, 0);
    //return (SOCKET_ERROR);
}

DataSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
CSockAddr local("192.168.1.50", 5001);
err = connect(DataSocket, local, sizeof(local));
if (err != 0)
{
    error = WSAGetLastError();
    if (err == WSANOTINITIALISED)
        err = 1;
    if (err == WSAEFAULT)
        err = 2;
    if (err == WSAENOTSOCK)
        err = 3;
    //ShowString.UpdateString("Error in Connect!!!", 0, 0);
    //ShowString.DoModal();
}

StartDataNetwork();
return TRUE;

int CDataNetwork::ExitInstance()
{
    // TODO: perform any per-thread cleanup here
    return CWinThread::ExitInstance();
}

BEGIN_MESSAGE_MAP(CDataNetwork, CWinThread)
//{{AFX_MSG_MAP(CDataNetwork)
    // NOTE - the ClassWizard will add and remove mapping macros here.
//}}AFX_MSG_MAP
END_MESSAGE_MAP()
void CDataNetwork::StartDataNetwork()
{
    int i, iLen, value;
    int iAmtSent, iAmtRecv;
    char RecvDatagram[100];
    CString strMyMessage;
    CString datagram;
    CString parameters;
    CString command;

    ContDataNetwork = TRUE;

    strMyMessage = "alti";
    AltitudeRef = 0;
    for (i = 0; i < 20; i++)
    {
        iLen = strMyMessage.GetLength();
        iAmtSent = send(DataSocket, strMyMessage, iLen, 0);

        if (iAmtSent == SOCKET_ERROR)
        {
            //ShowString.UpdateString("Error in send!!!", 0, 0);
            //ShowString.DoModal();
        }
        iAmtRecv = recv(DataSocket, RecvDatagram, sizeof(RecvDatagram), 0);

        if (iAmtRecv == SOCKET_ERROR)
        {
            //ShowString.UpdateString("Error in recv!!!", 0, 0);
            //ShowString.DoModal();
        }
        datagram = RecvDatagram;
        datagram = datagram.Left(iAmtRecv);
        command = datagram.Left(4);
        parameters = datagram.Right(iAmtRecv - 5);
        sscanf(parameters, "%x", &value);

        //if (i == 10)
        //    AltitudeRef = 0;
        AltitudeRef = AltitudeRef + value;
    
}
//AltitudeRef = value;
}

AltitudeRef = int(AltitudeRef / 20.0 + 0.5 - 23);

while (ContDataNetwork)
{
    for (i = 0; i < 7; i++)
    {
        commanddone = FALSE;
        if (i == 0)
        {
            // Read analog channel 0 Pitch and Roll reference voltage
            strMyMessage = "vref";
            commanddone = TRUE;
        }
        if (i == 1)
        {
            // Read analog channel 1 Pitch
            strMyMessage = "ptch";
            commanddone = TRUE;
        }
        if (i == 2)
        {
            // Read analog channel 2 Roll
            strMyMessage = "roll";
            commanddone = TRUE;
        }
        if (i == 3)
        {
            // Read analog channel 3 Altimeter
            strMyMessage = "alti";
            commanddone = TRUE;
        }
        if (i == 4)
        {
            // Read analog channel 4 compass 1
            strMyMessage = "cmpn";
            commanddone = TRUE;
        }
        if (i == 5)
        {
            // Read analog channel 5 Compass 2
            strMyMessage = "cmpw";
            commanddone = TRUE;
        }
    }
}
if (i == 6)
{
    // Read analog channel 6 Battery Voltage
    strMyMessage = "spdl";
    commanddone = TRUE;
}
if (i == 7)
{
    // Read analog channel 6 Battery Voltage
    strMyMessage = "spd2";
    commanddone = TRUE;
}
if (commanddone)
{
    iLen = strMyMessage.GetLength();
    iAmtSent = send(DataSocket,
                    strMyMessage, iLen, 0);
    if (iAmtSent == SOCKET_ERROR)
    {
        //ShowString.UpdateString("Error in send!!!", 0, 0);
        //ShowString.DoModal();
    }
}
//*************** Recieve from ***********

iAmtRecv = recv(DataSocket,
                RecvDatagram,
                sizeof(RecvDatagram), 0);

if (iAmtRecv == SOCKET_ERROR)
{
    //ShowString.UpdateString("Error in recv!!!", 0, 0);
    //ShowString.DoModal();
}

datagram = RecvDatagram;
datagram = datagram.Left(iAmtRecv);
command = datagram.Left(4);
parameters = datagram.Right(iAmtRecv - 5);
sscanf(parameters, "%x", &value);
commanddone = TRUE;

// Test incomming string
if (stricmp(command, "vref") == 0)
{
    // These functions display the incomming data in a dialog
    // to test the network and conversion from hex to int
    ShowString.UpdateString(command, value, iAmtRecv);
    //ShowString.DoModal();
    VoltRef = value;
}
if (stricmp(command, "ptch") == 0)
{
    // These functions display the incomming data in a dialog
    // to test the network and conversion from hex to int
    ShowString.UpdateString(command, value, iAmtRecv);
    //ShowString.DoModal();
    Pitch = value;
}
if (stricmp(command, "roll") == 0)
{
    // These functions display the incomming data in a dialog
    // to test the network and conversion from hex to int
    ShowString.UpdateString(command, value, iAmtRecv);
    //ShowString.DoModal();
    Roll = value;
}
if (stricmp(command, "alti") == 0)
{
    // These functions display the incomming data in a dialog
    // to test the network and conversion from hex to int
    ShowString.UpdateString(command, value, iAmtRecv);
}
//ShowString.DoModal();
Altitude = value;
}
if (stricmp(command, "cmpn") == 0)
{
    // These functions display the
    // incoming data in a dialog
    // to test the network and
    conversion from hex to int
    //ShowString.UpdateString(command,
    value, iAmtRecv);
    //ShowString.DoModal();
    CompassNorth = value;
}
if (stricmp(command, "cmpw") == 0)
{
    // These functions display the
    // incoming data in a dialog
    // to test the network and
    conversion from hex to int
    //ShowString.UpdateString(command,
    value, iAmtRecv);
    //ShowString.DoModal();
    CompassWest = value;
}
if (stricmp(command, "spdl") == 0)
{
    // These functions display the
    // incoming data in a dialog
    // to test the network and
    conversion from hex to int
    //ShowString.UpdateString(command,
    value, iAmtRecv);
    //ShowString.DoModal();
    Speed = value;
}
if (stricmp(command, "spd2") == 0)
{
    // These functions display the
    // incoming data in a dialog
    // to test the network and
    conversion from hex to int
    //ShowString.UpdateString(command,
    value, iAmtRecv);
    //ShowString.DoModal();
    Speed1 = value;
    //Speed = Speed - Speed1;

    //ShowString.DoModal();
    IISpeedl = value;
    IISpeed = Speed - Speed1;

    //ShowString.DoModal();
    CompassEast = value;
}
ContDataNetwork = pDocWnd - RunDataNetwork;

/* For Test
Altitude = Altitude + 2;
Compass = Altitude * 359 / 10000;
if (Altitude==10000) Altitude = 0;
Speed = Speed + 1;

if (Speed == 5500) Speed = 0;
if (Altitude <= 2500)
{
    Pitch = 0 - (Altitude * 50 / 2500);
    Roll = 0 - (Altitude * 90 / 2500);
}
if (Altitude > 2500 && Altitude < 7500)
{
    Pitch = -50 + ((Altitude - 2500) * 100 / 5000);
    Roll = -90 + ((Altitude - 2500) * 180 / 5000);
}
if (Altitude >= 7500)
{
    Pitch = 50 - ((Altitude - 7500) * 50 / 2500);
    Roll = 90 - ((Altitude - 7500) * 90 / 2500);
}
*/

//End of Test

UpdateUAVData();

void CDataNetwork::SetOwner(CUAVDoc * DocumentWindow)
{
    pDocWnd = DocumentWindow;
}

void CDataNetwork::UpdateUAVData()
{
    //***** Compass Algorithm ***************

- 33 -
double NorthAngle, WestAngle;

int CompassNorthMedian = 2086; // (2640 + 1532) / 2
int CompassWestMedian = 2076; // (2638 + 1514) / 2

NorthAngle = (CompassNorth - CompassNorthMedian) / 555.0;
WestAngle = (CompassWest - CompassWestMedian) / 562.0;

if (NorthAngle > 1)
    NorthAngle = 1;
if (NorthAngle < -1)
    NorthAngle = -1;
if (WestAngle > 1)
    WestAngle = 1;
if (WestAngle < -1)
    WestAngle = -1;

if (NorthAngle > 0.7)
    Compass = 0 - int(asin(WestAngle) * 180 / PIE + 0.5);
if (WestAngle < -0.7)
    Compass = 90 - int(asin(NorthAngle) * 180 / PIE + 0.5);
if (NorthAngle < -0.7)
    Compass = 180 + int(asin(WestAngle) * 180 / PIE + 0.5);
if (WestAngle > 0.7)
    Compass = 270 + int(asin(NorthAngle) * 180 / PIE + 0.5);

if (NorthAngle > 0.7 && WestAngle > 0)
    // Compass = 360 - int(asin(WestAngle) * 180 / PIE + 0.5);

if (Compass < 0)
    Compass = 360 + Compass;

****** Pitch and Roll Algorithm***********

Pitch = -(Pitch - VoltRef) / 22;
Roll = (Roll - VoltRef) / 22;

****** Altitude Algorithm **************
int i;
int count = 0;
```
Altitude_Buffer[AltitudePtr] = Altitude;
AltitudePtr++;
if(AltitudePtr > 19)
    AltitudePtr = 0;

for(i = 0; i < 20; i++)
{
    count = count + Altitude_Buffer[i];
}
//Altitude = int(count / 10.0 + 0.5);
//Altitude = int((3060.0 - count / 10.0) * 7.0 +
0.5);
Altitude = int((AltitudeRef - count / 20.0) * 7.0
+ 0.5);

if(Altitude < 0)
    Altitude = 0;

/****** Speed Algorithm

***************
if (CompassMin > Speed)
    CompassMin = Speed;
count = 0;
Speed_Buffer[SpeedPtr] = Speed;
SpeedPtr++;

if(SpeedPtr > 4)
    SpeedPtr = 0;

for (i = 0; i < 5; i++)
{
    count = count + Speed_Buffer[i];
}
//Speed = int((count / 10.0 - 58 + 0.5) * 400) ;
// max =
// min =
//Speed = int(count / 5.0 + 0.5);
//***************

CSingleLock sLock(&(pDocWnd->m_mutex));
sLock.Lock();
pDocWnd->m_pitch = Pitch;
pDocWnd->m_roll = Roll;
pDocWnd->m_altitude = Altitude;
```
pDocWnd->m_heading = Compass;
//pDocWnd->m_battery = BatteryVoltage;
pDocWnd->m_Speed = Speed;

sLock.Unlock();

//PostMessage(WM_UPDATE_AIRPLANE_DATA, 0, 0);
pDocWnd->UpdateAirplaneData();

}

Altitude.cpp
CAltimeter::CAltimeter()
{
    m_Altitude = 0;
    m_pViewWnd = NULL;
}

CAltimeter::~CAltimeter()
{
}

void CAltimeter::SetViewWindow(CWnd *pWnd)
{
    // Set the pointer to the view window
    m_pViewWnd = pWnd;
}

void CAltimeter::SetAltitude(int Alt)
{
    m_Altitude = Alt;
    //m_Altitude = 2745;
    DrawAltimeter();
}

void CAltimeter::DrawAltimeter()
{
    double angle = m_Altitude * 2 * PIE /
                   ALT_GUAGE_MAXIMUM;

    POINT sides100[4];
    POINT sides1000[4];
// Compute positions for 100's guage
sides100[0].x = int(CENTER_X + GUAGE_RADIUS * sin(angle) + 0.5);
sides100[0].y = int(CENTER_Y - GUAGE_RADIUS * cos(angle) + 0.5);
sides100[1].x = int(CENTER_X + 5 * cos(angle) + 0.5);
sides100[1].y = int(CENTER_Y + 5 * sin(angle) + 0.5);
sides100[2].x = int(CENTER_X - 5 * sin(angle) + 0.5);
sides100[2].y = int(CENTER_Y + 5 * cos(angle) + 0.5);
sides100[3].x = int(CENTER_X - 5 * sin(angle) + 0.5);
sides100[3].y = int(CENTER_Y - 5 * cos(angle) + 0.5);

// Compute positions for 1000's guage
angle = m_Altitude * 2 * PIE / (ALT_GUAGE_MAXIMUM * 10);
sides1000[0].x = int(CENTER_X + (GUAGE_RADIUS - 5) * sin(angle) + 0.5);
sides1000[0].y = int(CENTER_Y - (GUAGE_RADIUS - 5) * cos(angle) + 0.5);
sides1000[1].x = int(CENTER_X + 10 * cos(angle) + 0.5);
sides1000[1].y = int(CENTER_Y + 10 * sin(angle) + 0.5);
sides1000[2].x = int(CENTER_X - 10 * sin(angle) + 0.5);
sides1000[2].y = int(CENTER_Y + 10 * cos(angle) + 0.5);
sides1000[3].x = int(CENTER_X - 10 * cos(angle) + 0.5);
sides1000[3].y = int(CENTER_Y - 10 * sin(angle) + 0.5);

// % is remainder function
m_Ones.SetNumber(m_Altitude%10);
m_Tens.SetNumber((m_Altitude/10)%10);
m_Hundreds.SetNumber(int((m_Altitude/100))%10);
m_Thousands.SetNumber(int((m_Altitude / 1000)));

// Get the CDC pointer from the View Class
CDC *pDC = m_pViewWnd->GetDC();
// Create pen for use
CPen pen(PS_SOLID, 0, RGB(255, 0, 0));
// Select pen for use
CPen* pOldPen = pDC->SelectObject(&pen);

CBitmap Altimeter;
Altimeter.LoadBitmap(IDB_ALTIMETER);

CDC dcMemory;
dcMemory.CreateCompatibleDC(pDC);

// Create pen for use
CPen pen(PS_SOLID, 0, RGB(255, 255, 255));
// Select pen for use
CPen* pOldPen = dcMemory.SelectObject(&pen);
dcMemory.SelectObject(&Altimeter);
m_Ones.DrawLedNumber(10, 20, 116, 125, 0,
 &dcMemory);
m_Tens.DrawLedNumber(10, 20, 102, 125, 0,
 &dcMemory);
m_Hundreds.DrawLedNumber(10, 20, 88, 125, 0,
 &dcMemory);
m_Thousands.DrawLedNumber(10, 20, 74, 125, 0,
 &dcMemory);

CBrush brushLtGray(RGB(160, 160, 160));
CBrush* pOldBrush =
dcMemory.SelectObject(&brushLtGray);
dcMemory.Polygon(sides1000, 4);

CBrush brushDrkGray(RGB(90, 90, 90));
dcMemory.SelectObject(&brushDrkGray);
dcMemory.Polygon(sides100, 4);
dcMemory.Ellipse(CENTER_X - 15, CENTER_Y - 15,
 CENTER_X + 15, CENTER_Y + 15);

//dcMemory.MoveTo(115, 115);
//dcMemory.LineTo(115 + m_Altitude, 115);

pDC->BitBlt(755,250,200,200, &dcMemory, 0, 0,
SRCCOPY);

****** Need at end ************
dcMemory.SelectObject(pOldPen);
dcMemory.SelectObject(pOldBrush);

- 38 -
Horizon.cpp

// Horizon.cpp: implementation of the CHorizon class.

#include "stdafx.h"
#include "UAV.h"
#include "Horizon.h"
#include "Bitmask.h"
#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
define new DEBUG_NEW
#endif

CHorizon::CHorizon()
{
    m_iPitch = 0;
    m_iRoll = 0;
    m_pViewWnd = NULL;
}

CHorizon::~CHorizon()
{
}

void CHorizon::SetViewWindow(CWnd*pWnd)
{
    // Set the pointer to the view window
}
void CHorizon::SetHorizon(int ptch, int roll)
{
    m_iPitch = ptch;
    m_iRoll = roll;

    // m_iPitch = 10;
    // m_iRoll = 20;
    DrawHorizon();
}

void CHorizon::DrawHorizon()
{
    // Get the CDC pointer from the View Class
    CDC *pDC = m_pViewWnd->GetDC();

    COLORREF White(RGB(255, 255, 255));
    COLORREF Brown(RGB(170, 50, 0));
    COLORREF Blue(RGB(10, 120, 235));
    COLORREF Black(RGB(0, 0, 0));

    double angle = m_iRoll * 2 * PIE / 360;
    double tempAngle;

    POINT CenterLine[2], HorizonCenter;
    CenterLine[0].x = int(100 - 100 * cos(angle));
    CenterLine[0].y = int(100 - 100 * sin(angle));
    CenterLine[1].x = int(100 + 100 * cos(angle));
    CenterLine[1].y = int(100 + 100 * sin(angle));

    double ChordLength = sqrt(80 * 80 - m_iPitch * m_iPitch);

    double ChordAngle = atan2(ChordLength, m_iPitch);

    POINT ChordA, ChordB;
    ChordA.x = int(100 - 79 * sin(ChordAngle + angle) + 0.5);
    ChordA.y = int(100 + 79 * cos(ChordAngle + angle) + 0.5);
C borough.(x = int(100 + 79 * \sin(\text{ChordAngle} - \text{angle}) + 0.5);
C borough.(y = int(100 + 79 * \cos(\text{ChordAngle} - \text{angle}) + 0.5);

\text{HorizonCenter.} x = \text{int}((100 - \text{m_iPitch} * \sin(\text{angle}) + 0.5));
\text{HorizonCenter.} y = \text{int}((100 + \text{m_iPitch} * \cos(\text{angle}) + 0.5));

/*****************************************************************
** Insert Guage Code Here
**/*****************************************************************
C Bitmap Template;
Template.LoadBitmap(IDB TEMPLATE);

CDC dcMemory;
dcMemory.CreateCompatibleDC(pDC);
dcMemory.SelectObject(&Template);

// Create pen for use
CPen WhitePen(PS_SOLID, 2, White);
CPen BrownPen(PS_SOLID, 1, Brown);
CPen BluePen(PS_SOLID, 1, Blue);
CPen BlackPen(PS_SOLID, 2, Black);

C Brush BlueBrush(Blue);
C Brush BrownBrush(Brown);
C Brush BlackBrush(Black);

// Select pen for use
CPen* pOldPen = dcMemory.SelectObject(&WhitePen);
// Select Brush for use
C Brush* pOldBrush = dcMemory.SelectObject(&BlueBrush);

//dcMemory.SelectObject(&WhitePen);
dcMemory.Ellipse(11, 11, 189, 189);
dcMemory.MoveTo(CenterLine[0]);
dcMemory.LineTo(CenterLine[1]);
// Draw -60 roll indicator
tempAngle = 5 * \text{PIE} / 6 - \text{angle};
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 100 * \cos(tempAngle) + 0.5),
int(100 - 100 * sin(tempAngle) + 0.5));

// Draw -30 roll indicator
tempAngle = 2 * PI / 3 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 100 * cos(tempAngle) + 0.5),
                int(100 - 100 * sin(tempAngle) + 0.5));

// Draw -20 roll indicator
tempAngle = 11 * PI / 18 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 85 * cos(tempAngle) + 0.5),
                int(100 - 85 * sin(tempAngle) + 0.5));

// Draw -10 roll indicator
tempAngle = 10 * PI / 18 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 84 * cos(tempAngle) + 0.5),
                int(100 - 85 * sin(tempAngle) + 0.5));

// Draw 60 roll indicator
tempAngle = PI / 6 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 100 * cos(tempAngle) + 0.5),
                int(100 - 100 * sin(tempAngle) + 0.5));

// Draw 30 roll indicator
tempAngle = PI / 3 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 100 * cos(tempAngle) + 0.5),
                int(100 - 100 * sin(tempAngle) + 0.5));

// Draw 20 roll indicator
tempAngle = 7 * PI / 18 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 85 * cos(tempAngle) + 0.5),
                int(100 - 85 * sin(tempAngle) + 0.5));

// Draw 10 roll indicator
tempAngle = 8 * PI / 18 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 85 * cos(tempAngle) + 0.5),
                int(100 - 85 * sin(tempAngle) + 0.5));

// Draw 0 roll indicator
tempAngle = PI / 2 - angle;
dcMemory.MoveTo(100, 100);
dcMemory.LineTo(int(100 + 100 * cos(tempAngle) + 0.5),
                int(100 - 100 * sin(tempAngle) + 0.5));

dcMemory.SelectObject(&BrownBrush);
dcMemory.Ellipse(20, 20, 180, 180);
dcMemory.FloodFill(int(100 - 82 * sin(angle)), int(100 + 82 * cos(angle)), White);
dcMemory.MoveTo(ChordA);
dcMemory.LineTo(ChordB);
dcMemory.SelectObject(&BlueBrush);
dcMemory.FloodFill(int(100 - (m_iPitch - 10) * sin(angle) + 0.5),
                   int(100 + (m_iPitch - 10) * cos(angle) + 0.5),
                   White);

    //Draw horizon angle lines
    //dcMemory.MoveTo(HorizonCenter);
    //dcMemory.LineTo(int(HorizonCenter.x - 79 * sin(PIE / 8 + angle) + 0.5),
                    int(HorizonCenter.y + 79 * cos(PIE / 8 + angle) + 0.5));

dcMemory.SelectObject(&BlackPen);
    if (m_iPitch > -68)
        {  
dcMemory.MoveTo(int(HorizonCenter.x + 10 * sin(angle) - 5 * cos(angle) + 0.5),
                   int(HorizonCenter.y - 10 * cos(angle) - 5 * sin(angle) + 0.5));
    dcMemory.LineTo(int(HorizonCenter.x + 10 * sin(angle) + 5 * cos(angle) + 0.5),
                   int(HorizonCenter.y - 10 * cos(angle) + 5 * sin(angle) + 0.5));
        }
    if (m_iPitch > -58)
        {  
dcMemory.MoveTo(int(HorizonCenter.x + 20 * sin(angle) - 10 * cos(angle) + 0.5),
                   int(HorizonCenter.y - 20 * cos(angle) - 10 * sin(angle) + 0.5));
    dcMemory.LineTo(int(HorizonCenter.x + 20 * sin(angle) + 10 * cos(angle) + 0.5),
                   int(HorizonCenter.y - 20 * cos(angle) + 15 * sin(angle) + 0.5));
        }
    if (m_iPitch > -48)
        {  
dcMemory.MoveTo(int(HorizonCenter.x + 30 * sin(angle) - 5 * cos(angle) + 0.5),
                   int(HorizonCenter.y - 30 * cos(angle) - 5 * sin(angle) + 0.5));
    dcMemory.LineTo(int(HorizonCenter.x + 30 * sin(angle) + 5 * cos(angle) + 0.5),
...
int(HorizonCenter.y - 30 * cos(angle) + 5 *
  sin(angle) + 0.5));
}
if (m_iPitch > -38)
{
  dcMemory.MoveTo(int(HorizonCenter.x + 40 *
    sin(angle) - 10 * cos(angle) + 0.5),
    int(HorizonCenter.y - 40 * cos(angle) - 10 *
    sin(angle) + 0.5));
  dcMemory.LineTo(int(HorizonCenter.x + 40 *
    sin(angle) + 10 * cos(angle) + 0.5),
    int(HorizonCenter.y - 40 * cos(angle) + 10 *
    sin(angle) + 0.5));
}
}
dcMemory.SelectObject(&WhitePen);
if (m_iPitch < 68)
{
  dcMemory.MoveTo(int(HorizonCenter.x - 10 *
    sin(angle) - 5 * cos(angle) + 0.5),
    int(HorizonCenter.y + 10 * cos(angle) - 5 *
    sin(angle) + 0.5));
  dcMemory.LineTo(int(HorizonCenter.x - 10 *
    sin(angle) + 5 * cos(angle) + 0.5),
    int(HorizonCenter.y + 10 * cos(angle) + 5 *
    sin(angle) + 0.5));
}
if (m_iPitch < 58)
{
  dcMemory.MoveTo(int(HorizonCenter.x - 20 *
    sin(angle) - 10 * cos(angle) + 0.5),
    int(HorizonCenter.y + 20 * cos(angle) - 10 *
    sin(angle) + 0.5));
  dcMemory.LineTo(int(HorizonCenter.x - 20 *
    sin(angle) + 10 * cos(angle) + 0.5),
    int(HorizonCenter.y + 20 * cos(angle) + 10 *
    sin(angle) + 0.5));
}
if (m_iPitch < 48)
{
  dcMemory.MoveTo(int(HorizonCenter.x - 30 *
    sin(angle) - 5 * cos(angle) + 0.5),
    int(HorizonCenter.y + 30 * cos(angle) - 5 *
    sin(angle) + 0.5));
  dcMemory.LineTo(int(HorizonCenter.x - 30 *
    sin(angle) + 5 * cos(angle) + 0.5),

int(HorizonCenter.y + 30 * cos(angle) + 5 *
sin(angle) + 0.5));
}
if (m_iPitch < 38)
{
    dcMemory.MoveTo(int(HorizonCenter.x - 40 *
sin(angle) - 10 * cos(angle) + 0.5),
    int(HorizonCenter.y + 40 * cos(angle) - 10 *
sin(angle) + 0.5));
    dcMemory.LineTo(int(HorizonCenter.x - 40 *
sin(angle) + 10 * cos(angle) + 0.5),
    int(HorizonCenter.y + 40 * cos(angle) + 10 *
    sin(angle) + 0.5));
}

CBitMask(IDB_HORIZON, IDB_HORIZON_MASK, &dcMemory,
CPoint(0, 0));

pDC->BitBlt(25, 250, 199, 199, &dcMemory, 0, 0, SRCCOPY);

//****** Need at end **************
dcMemory.SelectObject(pOldPen);
dcMemory.SelectObject(pOldBrush);
m_pViewWnd->ReleaseDC(pDC);
}

Compass.cpp

// Compass.cpp: implementation of the CCompass class.
//
///////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "UAV.h"
#include "Compass.h"
#include <math.h>

#ifdef__DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

-45-
CCompass::CCompass()
{
    m_iHeading = 0;
    m_pViewWnd = NULL;
}

CCompass::~CCompass()
{

}

void CCompass::SetCompass(int Heading)
{
    m_iHeading = Heading;
    //m_iHeading = 85;
    DrawCompass();
}

void CCompass::SetViewWindow(CWnd *pWnd)
{
    // Set the pointer to the view window
    m_pViewWnd = pWnd;
}

void CCompass::DrawCompass()
{
    double angle;

    COLORREF White(RGB(255, 255, 255));
    COLORREF Brown(RGB(170, 50, 0));
    COLORREF Blue(RGB(10, 120, 235)); // (10, 120, 235)
    COLORREF Black(RGB(0, 0, 0));

    angle = -(m_iHeading * PI / 180);
    // Get the CDC pointer from the View Class
CDC *pDC = m_pViewWnd->GetDC();

// % is remainder function
m_Heading1.SetNumber(m_iHeading%10);
m_Heading10.SetNumber((m_iHeading/10)%10);
m_Heading100.SetNumber(int((m_iHeading/100))%10);

CBitmap Compass;
Compass.LoadBitmap(IDB_COMPASS);

CDC dcMemory;
dcMemory.CreateCompatibleDC(pDC);
dcMemory.SelectObject(&Compass);

m_Heading1.DrawLedNumber(10, 20, 110, 60, 0, &dcMemory);
m_Heading10.DrawLedNumber(10, 20, 95, 60, 0, &dcMemory);
m_Heading100.DrawLedNumber(10, 20, 80, 60, 0, &dcMemory);

// Create pen for use
CPen WhitePen(PS_SOLID, 2, White);
CPen BluePen(PS_SOLID, 2, Blue);
// Select pen for use
CPen* pOldPen = dcMemory.SelectObject(&WhitePen);
// Select Brush for use
//CBrush* pOldBrush =
dcMemory.SelectObject(&BlueBrush);

double sumAngle;
for (sumAngle = 0; sumAngle < 2 * PIE; sumAngle = sumAngle + PIE / 6)
{
    dcMemory.MoveTo(int(100 + 88 * sin(angle + sumAngle) + 0.5),
                    int(100 - 88 * cos(angle + sumAngle) + 0.5));
    dcMemory.LineTo(int(100 + 77 * sin(angle + sumAngle) + 0.5),
                     int(100 - 77 * cos(angle + sumAngle) + 0.5));
}

for (sumAngle = PIE / 18; sumAngle < 2 * PIE; sumAngle = sumAngle + PIE / 18)
{ 
    dcMemory.MoveTo(int(100 + 88 * sin(angle + sumAngle) + 0.5),
    int(100 - 88 * cos(angle + sumAngle) + 0.5));
    dcMemory.LineTo(int(100 + 83 * sin(angle + sumAngle) + 0.5),
    int(100 - 83 * cos(angle + sumAngle) + 0.5));
}

for (sumAngle = PIE / 36; sumAngle < 2 * PIE; sumAngle = sumAngle + PIE / 36)
{
    dcMemory.MoveTo(int(100 + 88 * sin(angle + sumAngle) + 0.5),
    int(100 - 88 * cos(angle + sumAngle) + 0.5));
    dcMemory.LineTo(int(100 + 85 * sin(angle + sumAngle) + 0.5),
    int(100 - 85 * cos(angle + sumAngle) + 0.5));
}

dcMemory.SelectObject(&BluePen);

dcMemory.MoveTo(100, 12);
dcMemory.LineTo(100, 25);

CFont m_fN;
CFont m_fE;
CFont m_fS;
CFont m_fW;
CFont m_f30, m_f60, m_f120, m_f150;
CFont m_f210, m_f240, m_f300, m_f330;
    m_fN.CreateFont(22, 10, int(-(angle * 1800 / PIE) + 0.5), 0,
    FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
    OUT_CHARACTER_PRECIS,
    CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
    m_fE.CreateFont(22, 10, int(-(angle + PIE / 2) * 1800 / PIE) + 0.5), 0,
    FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
    OUT_CHARACTER_PRECIS,
    CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
    m_fS.CreateFont(22, 10, int(-(angle + PIE) * 1800 / PIE) + 0.5), 0,
    FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
    OUT_CHARACTER_PRECIS,
CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_fW.CreateFont(22, 10, int(-((angle + PIE * 1.5) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f30.CreateFont(18, 7, int(-((angle + PIE / 6) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f60.CreateFont(18, 7, int(-((angle + PIE / 3) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f120.CreateFont(18, 7, int(-((angle + 2 * PIE / 3) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f150.CreateFont(18, 7, int(-((angle + 5 * PIE / 6) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f210.CreateFont(18, 7, int(-((angle + 7 * PIE / 6) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f240.CreateFont(18, 7, int(-((angle + 4 * PIE / 3) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
  CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
  m_f300.CreateFont(18, 7, int(-((angle + 5 * PIE / 3) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman"));
    m_f330.CreateFont(18, 7, int(-((angle + 11 * PIE / 6) * 1800 / PIE) + 0.5), 0,
FW_BOLD, 0, 0, 0, DEFAULT_CHARSET,
OUT_CHARACTER_PRECIS,
CLIP_CHARACTER_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH | FF_DONTCARE, "Times New Roman");
    CFont* pOldFont = dcMemory.SelectObject(&m_fN);

    dcMemory.SetTextColor(White);
dcMemory.SetBkColor(Black);

    sumAngle = PIE / 25;
dcMemory.TextOut(int(100 + 74 * sin(angle - sumAngle) + 0.5),
    int(100 - 74 * cos(angle - sumAngle) + 0.5),
    "N");

    dcMemory.SelectObject(&m_fE);
dcMemory.TextOut(int(100 + 74 * sin(angle + PIE / 2 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + PIE / 2 - sumAngle) + 0.5), "E");

    dcMemory.SelectObject(&m_fS);
dcMemory.TextOut(int(100 + 74 * sin(angle + PIE - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + PIE - sumAngle) + 0.5), "S");

    dcMemory.SelectObject(&m_fW);
dcMemory.TextOut(int(100 + 74 * sin(angle + PIE * 3 / 2 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + PIE * 3 / 2 - sumAngle * 1.2) + 0.5), "W");

    sumAngle = PIE / 28;
dcMemory.SelectObject(&m_f30);
dcMemory.TextOut(int(100 + 74 * sin(angle + PIE / 6 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + PIE / 6 - sumAngle * 1.2) + 0.5), "30");

    dcMemory.SelectObject(&m_f60);
dcMemory.TextOut(int(100 + 74 * sin(angle + PIE / 3 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + PIE / 3 - sumAngle * 1.2) + 0.5), "60");

    sumAngle = PIE / 20;
    dcMemory.SelectObject(&rn_f120);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 2 * PIE / 3 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 2 * PIE / 3 - sumAngle * 1.2) + 0.5), "120");

    dcMemory.SelectObject(&m_f150);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 5 * PIE / 6 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 5 * PIE / 6 - sumAngle * 1.2) + 0.5), "150");

    dcMemory.SelectObject(&m_f210);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 7 * PIE / 6 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 7 * PIE / 6 - sumAngle * 1.2) + 0.5), "210");

    dcMemory.SelectObject(&m_f240);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 4 * PIE / 3 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 4 * PIE / 3 - sumAngle * 1.2) + 0.5), "240");

    dcMemory.SelectObject(&m_f300);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 5 * PIE / 3 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 5 * PIE / 3 - sumAngle * 1.2) + 0.5), "300");

    dcMemory.SelectObject(&m_f330);
    dcMemory.TextOut(int(100 + 74 * sin(angle + 11 * PIE / 6 - sumAngle) + 0.5),
    int(100 - 74 * cos(angle + 11 * PIE / 6 - sumAngle * 1.2) + 0.5), "330");

    pDC->BitBlt(25,15,199,234, &dcMemory, 0, 0, SRCCOPY);

    //****** Need at end **************
dcMemory.SelectObject(pOldPen);
dcMemory.SelectObject(pOldFont);
//dcMemory.SelectObject(pOldBrush);
m_pViewWnd->ReleaseDC(pDC);

}  

Led.cpp

// Led.cpp: implementation of the CLed class.
//
///////////////////////////////////////////////////////////
///////////
#include "stdafx.h"
#include "UAV.h"
#include "Led.h"

#ifdef __DEBUG
#undef THIS_FILE
#define THIS_FILE __FILE__
#endif new DEBUG_NEW

///////////////////////////////////////////////////////////
///////////
// Construction/Destruction
///////////////////////////////////////////////////////////
///////////
CLed::CLed()
{   
    m_Number = 0;
}

CLed::~CLed()
{
}

void CLed::SetNumber(int number)
{   
    m_Number = number;
}
void CLeD::DrawLedNumber(int width, int height, int x, int y, int color, CDC *pDC)
{
    int Red, Green, Blue;
    POINT BkGnd[4];
    POINT a[4];
    POINT b[4];
    POINT c[4];
    POINT d[4];
    POINT e[4];
    POINT f[4];
    POINT g[6];

    if (color == 0)
    {
        Red = 255;
        Green = 255;
        Blue = 255;
    }

    if (color == 1)
    {
        Red = 255;
        Green = 0;
        Blue = 0;
    }

    BkGnd[0].x = x;
    BkGnd[0].y = y;
    BkGnd[1].x = x + width;
    BkGnd[1].y = y;
    BkGnd[2].x = x + width;
    BkGnd[2].y = y + height;
    BkGnd[3].x = x;
    BkGnd[3].y = y + height;

    a[0].x = int(x + 0.1 * width + 0.5);
    a[0].y = int(y + 0.05 * height + 0.5);
    a[1].x = int(x + 0.9 * width + 0.5);
    a[1].y = int(y + 0.05 * height + 0.5);
    a[2].x = int(x + 0.7 * width + 0.5);
    a[2].y = int(y + 0.15 * height + 0.5);
    a[3].x = int(x + 0.3 * width + 0.5);
    a[3].y = int(y + 0.15 * height + 0.5);

    b[0].x = int(x + 0.9 * width + 0.5);
}
b[0].y = int(y + 0.1 * height + 0.5);
b[1].x = int(x + 0.9 * width + 0.5);
b[1].y = int(y + 0.48 * height + 0.5);
b[2].x = int(x + 0.7 * width + 0.5);
b[2].y = int(y + 0.38 * height + 0.5);
b[3].x = int(x + 0.7 * width + 0.5);
b[3].y = int(y + 0.2 * height + 0.5);

c[0].x = int(x + 0.9 * width + 0.5);
c[0].y = int(y + 0.52 * height + 0.5);
c[1].x = int(x + 0.9 * width + 0.5);
c[1].y = int(y + 0.9 * height + 0.5);
c[2].x = int(x + 0.7 * width + 0.5);
c[2].y = int(y + 0.8 * height + 0.5);
c[3].x = int(x + 0.7 * width + 0.5);
c[3].y = int(y + 0.62 * height + 0.5);

d[0].x = int(x + 0.1 * width + 0.5);
d[0].y = int(y + 0.95 * height + 0.5);
d[1].x = int(x + 0.9 * width + 0.5);
d[1].y = int(y + 0.95 * height + 0.5);
d[2].x = int(x + 0.7 * width + 0.5);
d[2].y = int(y + 0.85 * height + 0.5);
d[3].x = int(x + 0.3 * width + 0.5);
d[3].y = int(y + 0.85 * height + 0.5);

e[0].x = int(x + 0.1 * width + 0.5);
e[0].y = int(y + 0.9 * height + 0.5);
e[1].x = int(x + 0.1 * width + 0.5);
e[1].y = int(y + 0.52 * height + 0.5);
e[2].x = int(x + 0.3 * width + 0.5);
e[2].y = int(y + 0.62 * height + 0.5);
e[3].x = int(x + 0.3 * width + 0.5);
e[3].y = int(y + 0.8 * height + 0.5);

f[0].x = int(x + 0.1 * width + 0.5);
f[0].y = int(y + 0.1 * height + 0.5);
f[1].x = int(x + 0.1 * width + 0.5);
f[1].y = int(y + 0.48 * height + 0.5);
f[2].x = int(x + 0.3 * width + 0.5);
f[2].y = int(y + 0.38 * height + 0.5);
f[3].x = int(x + 0.3 * width + 0.5);
f[3].y = int(y + 0.2 * height + 0.5);

g[0].x = int(x + 0.17 * width + 0.5);
g[0].y = int(y + 0.5 * height + 0.5);
g[1].x = int(x + 0.3 * width + 0.5);
g[1].y = int(y + 0.43 * height + 0.5);
g[2].x = int(x + 0.7 * width + 0.5);
g[2].y = int(y + 0.43 * height + 0.5);
g[3].x = int(x + 0.83 * width + 0.5);
g[3].y = int(y + 0.5 * height + 0.5);
g[4].x = int(x + 0.7 * width + 0.5);
g[4].y = int(y + 0.57 * height + 0.5);
g[5].x = int(x + 0.3 * width + 0.5);
g[5].y = int(y + 0.57 * height + 0.5);

// Create pen for use
CPen BkGndPen(PS_SOLID, 0, RGB(0, 0, 0));
// Select pen for use
CPen* pOldPen = pDC->SelectObject(&BkGndPen);
CBrush BkGndBrush(RGB(0, 0, 0));
CBrush* pOldBrush = pDC->SelectObject(&BkGndBrush);
pDC->Polygon(BkGnd, 4);
CPen LedPen(PS_SOLID, 0, RGB(Red, Blue, Green));
CBrush LedBrush(RGB(Red, Green, Blue));
pDC->SelectObject(&LedPen);
pDC->SelectObject(&LedBrush);

if (m_Number == 0 || m_Number == 2 || m_Number == 3 || m_Number == 5 || m_Number == 6 || m_Number == 7 || m_Number == 8 || m_Number == 9)
{
    pDC->Polygon(a, 4);
}

if (m_Number == 0 || m_Number == 1 || m_Number == 2 || m_Number == 3 || m_Number == 4 || m_Number == 7 || m_Number == 8 || m_Number == 9)
{
    pDC->Polygon(b, 4);
}

if (m_Number == 0 || m_Number == 1 || m_Number == 3 || m_Number == 4 || m_Number == 5 || m_Number == 6 || m_Number == 7 || m_Number == 8 || m_Number == 9)
Analog1.c (Program installed on the PC104 Computer)

#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <stdio.h>
#include <memory.h>
#include <string.h>
#include <ctype.h>

{  
pDC->Polygon(c, 4);
}

if (m_Number == 0 || m_Number == 2 || m_Number == 3 ||
    m_Number == 5 || m_Number == 6 || m_Number == 8 ||
    m_Number == 9)
{
  pDC->Polygon(d, 4);
}

if (m_Number == 0 || m_Number == 2 || m_Number == 6 ||
    m_Number == 8)
{
  pDC->Polygon(e, 4);
}

if (m_Number == 0 || m_Number == 4 || m_Number == 5 ||
    m_Number == 6 || m_Number == 8 || m_Number == 9)
{
  pDC->Polygon(f, 4);
}

if (m_Number == 2 || m_Number == 3 || m_Number == 4 ||
    m_Number == 5 || m_Number == 6 || m_Number == 8 ||
    m_Number == 9)
{
  pDC->Polygon(g, 6);
}

//***** Need at end **************
pDC->SelectObject(pOldPen);
pDC->SelectObject(pOldBrush);
}
#include <time.h>

#define NETSOCK_MASTER
#include "netsock.h"

#define DACQ_PORT 5001  // the UDP port this server will listen on

// Micro/sys Hardware Feature Control Subfunctions
#define MS_SPECIAL 0xDD  // Micro/sys INT1A
#define MS_LEDCTRL 0x10  // Micro/sys Control LED
#define MS_FINDDIO 0x11  // Micro/sys locate 82C55
#define MS_STARTADC 0x20  // Micro/sys start A-to-D conversion
#define MS_ADCSTAT 0x21  // Micro/sys check A-to-D done status
#define MS_SETDAC 0x30  // Micro/sys set D-to-A converter
#define MS_DACRANGE 0x31  // Micro/sys find output range of DAC

int DIOBASE;
int DIOSIZE;

#define DCMDR DIOBASE+3*(DIOSIZE)  // (+3 for 8-bit)
or (+6 for 16-bit)

// Local prototypes
int analogrd(int chan);
void analogwr(int chan, int dta);

// CPU register structures for int86(), int86x() calls
union REGS cpureg;
struct SREGS cpusreg;
WSAData SocketData;

int main(void)
{
    int fromlen,numbytes,debug,commanddone,err;
    char datagram[80];
char command[10];
char parameters[11];
int voltage;
char channel, pin, state, mask;
unsigned int port, value, allocseg;

SOCKET msgsock;
struct sockaddr_in local, from;

debug = 0;

// Find and Initialize 82C55

cpureg.h.ah = MS_SPEC; /* special Micro/sys system function */
cpureg.h.al = MS_FINDDIO; /* locate 82C55 */

int86(MSYS_INT, &cpureg, &cpureg);
DIOBASE = cpureg.x.ax;
DIOSIZE = cpureg.h.bl + 1;
outp(DCMDR, Ox82); // A out, B in, C out

// Initialize D/A converters
analogwr(0,0);
analogwr(1,0);
analogwr(2,0);
analogwr(3,0);

printf("\n\r\rMicro/sys Embedded Netsock Network-based Control\r\n\r\n");

err = WSAStartup(0x101, &SocketData);
if (err)
{
printf("WSAStartup failed with error %d\n",err);
switch (EmbeddedNetsockLoadError)
{
case 0:
break;
case ENE_LDERR_BIOS:
printf("System BIOS does not support
Embedded Netsock!\n");
break;
case ENE_LDERR_ADAPTER:
printf("No network adapter found!\n");
break;
case ENE_LDERR_MEM:

-58-
printf("Error allocating memory!\n\r")
break;

case ENETDERR_NETSOCK:
    printf("Netsock not available!\n\r")
    break;
}
WSACleanup();
exit(l);
}

printf("WINsock version: %x\n", SocketData.wVersion);
printf("Highest version supported: %x\n", SocketData.wHighVersion);
printf("%s\n", SocketData.szDescription);
printf("Netsock status: %s\n", SocketData.szSystemStatus);
printf("Maximum number of sockets: %d\n", SocketData.iMaxSockets);
printf("Maximum datagram length: %d bytes\n", SocketData.iMaxUdpDg);
printf("Embedded Netsock Version: %x\n", EmbeddedNetsockVersion);

msgsock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if (msgsock == SOCKET_ERROR)
    printf("Error opening socket: Error %d\n\r", WSAGetLastError());

local.sin_family = AF_INET;
local.sin_port = htons(DACQ_PORT);

err = bind(msgsock, &local, sizeof(local));
if (err == SOCKET_ERROR)
    {
        printf("Error binding socket: Error %d\n\r", WSAGetLastError());
        WSACleanup();
        return(-1);
    }

printf("Initiating server mode on UDP port %d....\r\n\r\n", DACQ_PORT);

for (;;) {
    /* Get a datagram and process it */
fromlen = sizeof(from);
numbytes = recvfrom(msgsock, datagram, 
sizeof(datagram), FLAGS_ZERO, &from,
    (int far *) &fromlen);
if (numbytes == SOCKET_ERROR)
    printf("Failed to receive message: Error
%d\n",WSAGetLastError()) ;
memcpy(command, datagram, 4);
command[4] = 0;
//memcpy(parameters, datagram+4, numbytes-4);
//parameters[numbytes-4] = 0;
commanddone = 0;
if (debug)
    {
        printf("\nReceived %d byte
message.\n",numbytes);
        printf("Command = %s\n", command);
        //printf("Parameters = %s\n", parameters);
    }

/**********************************************************/

if (stricmp(command, "alti") == 0)
    {
        voltage = analogrd(0);
        sprintf(parameters, "%03X", voltage);
        commanddone = 1;
    }

/**********************************************************/

if (stricmp(command, "cmpn") == 0)
    {
        voltage = analogrd(1);
        sprintf(parameters, "%03X", voltage);
        commanddone = 1;
    }

/**********************************************************/

if (stricmp(command, "cmpw") == 0)
    {
        voltage = analogrd(2);

- 60 -
sprintf(parameters, "=%03X", voltage);
commanddone = 1;
*/

if (stricmp(command, "spd1") == 0)
{
    voltage = analogrd(3);
    sprintf(parameters, "=%03X", voltage);
    commanddone = 1;
}

/*---------------------------------------------*/

if (stricmp(command, "spd2") == 0)
{
    voltage = analogrd(4);
    sprintf(parameters, "=%03X", voltage);
    commanddone = 1;
}

/*---------------------------------------------*/

if (stricmp(command, "vref") == 0)
{
    voltage = analogrd(5);
    sprintf(parameters, "=%03X", voltage);
    commanddone = 1;
}

/*---------------------------------------------*/

if (stricmp(command, "roll") == 0)
{
    voltage = analogrd(6);
    sprintf(parameters, "=%03X", voltage);
    commanddone = 1;
}

/*---------------------------------------------*/

if (stricmp(command, "ptch") == 0)
{
    voltage = analogrd(7);
    sprintf(parameters, "=%03X", voltage);
    commanddone = 1;
}
// return response to host that rcvfrom() indicated in 'from' structure
strcpy(datagram, command); // start with command
if (commanddone)
    strcat(datagram, parameters); // append valid command results
else
    strcat(datagram, "??"); // append invalid command response
if(debug)
    printf(" Returning message: %s", datagram);
err = sendto(msgsock, datagram, strlen(datagram),
FLAGS_ZERO,
    &from, sizeof(from));
if (err == SOCKET_ERROR)
    printf("Failed to send datagram: Error %d\n", WSAGetLastError());
/*--------------------------------------------------------*/
if (stricmp(command, "break") == 0)
{
    exit(O);
}
/*--------------------------------------------------------*/

/* Analog input and output routines */
/* ------------------------------------ */
#define RNGE5VUNIP 0 /* 0 to +5v range */
#define RNGE5VBIP 1 /* -5 to +5v range */
#define RNGE10VUNIP 2 /* 0 to +10v range */
#define RNGE10VBIP 3 /* -10 to +10v range */
int analogrd(int chan)
{
    clock_t goal;
    cpureg.h.ah = MS_SPECIAL; /* special
    Micro/sys system function */
    cpureg.h.al = MS_STARTADC; /* start ADC
    conversion */
    if(chan == 3)
        cpureg.h.bh = RNGE10VUNIP;
    else
        cpureg.h.bh = RNGE5VUNIP; /* 5v
    unipolar range */
    cpureg.h.bl = chan;
    int86(MSYS_INT, &cpureg, &cpureg);

    goal = 200 + clock();
    for(;;)
    {
        cpureg.h.ah = MS_SPECIAL; /* special
        Micro/sys system function */
        cpureg.h.al = MS_ADCSTAT; /* check ADC
        conversion */
        int86(MSYS_INT, &cpureg, &cpureg);
        if (cpureg.h.bl) /* if conversion is done,
        read value */
            return (cpureg.x.ax); /* if conversion is done,
            break;
            return (0);
    }
References


Steffen, Mike., John Eley, and William M. Cavanaugh, "Remote Reality Vehicle", Diss. Northern Illinois University, DeKalb, IL
Capstone Title: (print or type):

UNMANNE AERIAL VEHICAL

Student Name (print or type): KEVIN WILSON

Faculty Supervisor (print or type): DR. MILLER

Faculty Approval Signature: [Signature]

Department of (print or type): ELECTRICAL ENGINEERING

Date of Approval (print or type): 5/7/2002