Vol. 4.Special Issue ICCSII ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences
©2009-2013 CIS Journal. All rights reserved.

http://www.cisjournal.org

Design and Development of a Networked Control System Platform for Unmanned Aerial Vehicles

[1]Yücel Taş, [2]Aydın Yeşildirek, [3]Ahmet Sertbaş

[1]Istanbul University, Computer Engineering Dept., Istanbul, Turkey
[2]American University of Sharjah, Electrical Engineering Dept., Sharjah, UAE
[3]Istanbul University, Computer Engineering Dept., Istanbul, Turkey

ABSTRACT

This paper reports the design and development of a network based computer simulation and control platform for small unmanned aerial vehicles (UAV). The complete platform has been built by a complex software integration in NIST/ECMA model having user interfaces on distributed platforms, open-source and educational vertical tools accessing public domain maps, satellite images and tool libraries. Entire platform is setup to use UDP for inter-process message communication. Successful integration is illustrated for an UAV and simulation results are presented. A major element in this work is to define a complex software architecture platform unifying GNU, open source and educational tools for ground station visualization, communication, modeling, navigation and control of small scale UAV systems.

Keywords: *networked control, modeling, simulation, UAV*

1. INTRODUCTION

Most of the modern engineering systems have become multidisciplinary and complex. For this reason, need for Computer Aided Design (CAD) support is essential at every stage of engineering practices. Integration of different software bundles offers great benefits in engineering systems.

An Unmanned Aerial Vehicle (UAV) is an autonomous platform that can be remotely controlled in a ground station unit to perform different tasks. They operate without onboard human pilots. In recent years, unmanned aerial vehicles have gained much more attention worldwide due to their high impact applications in both military and civil domains. For this reason UAV development is attracting many researchers due to benefits they bring. When the UAV is flown over an area, it can send surveillance data back to a ground station where the data is analyzed. Our objective is to develop a virtual UAV simulation and control platform that requires integration of complex distributed software running multitasks connected via some network. Such platform can be used to train pilots to fly different UAVs as well as to test effectiveness of UAV navigation and control systems, since it reduces the time and the risks associated with the real flight [1][2][3][4].

In this work, we propose an integrated and comprehensive software system for the modeling, controlling and simulation of an UAV platform using open source tools glued by a C++ application. UDP/IP is selected for backbone communication protocol for its low overhead.

2. PLATFORM ARCHITECTURE

Our system mainly consists of a user, ground station and model-based virtual UAV with navigation and control capabilities as shown in Fig. 1. We include in our ground station a visual simulation of a 2D UAV cockpit view with realistic 3D terrain view, 2D navigation map and user interface module to simulate an efficient UAV environment for both monitoring and training a pilot that is depicted in Fig. 2. Finally, the UAV platform is composed of a nonlinear dynamic model, path planner and an auto pilot controller that can be switched off for manual or auto operation, subsystems that is shown in Fig. 3. Such a platform enables us to design and test effective navigation and control algorithms as well as a virtual tool to train a pilot.

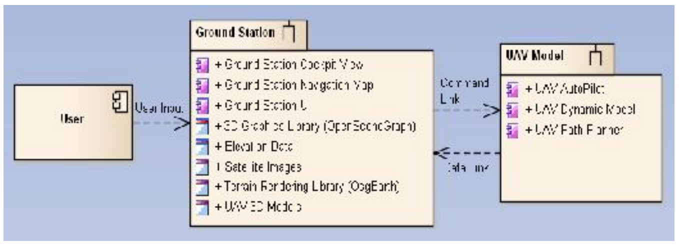


Figure 1. Main components of the UAV platform

If we want to show all the subsystems in the platform, we got the following detailed block diagrams of the platform.

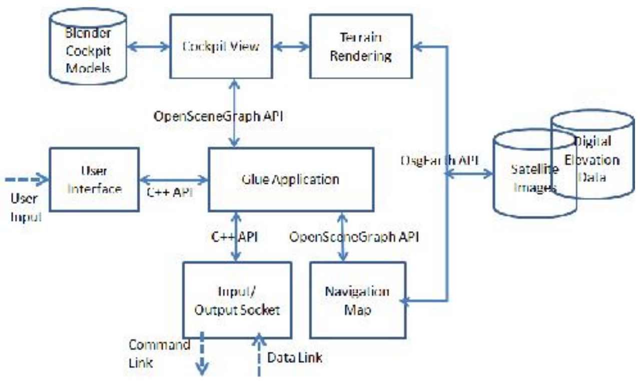


Figure 2. Ground Station and Interfaces

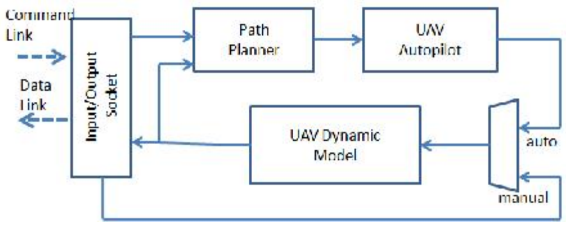


Figure 3. Virtual UAV System

The user interacts directly to the user interface of the ground control station through the command link. User directives are sent to the UAV path planner and UAV surface deflection inputs. In auto mode, UAV autopilot takes the output of the path planner and produces the necessary control commands, in manual overwrite pilot defines these inputs. The dynamic model takes the commands and produces six degree-of-freedom (DOF) UAV state information (position, orientation and velocity). The state information is sent to the ground control station through the data link that is used to update the ground station views. In Fig. 4 we can see the NIST/ECMA model for UAV environment integration. We can see the vertical tools that we used in our platform on the NIST/ECMA model.

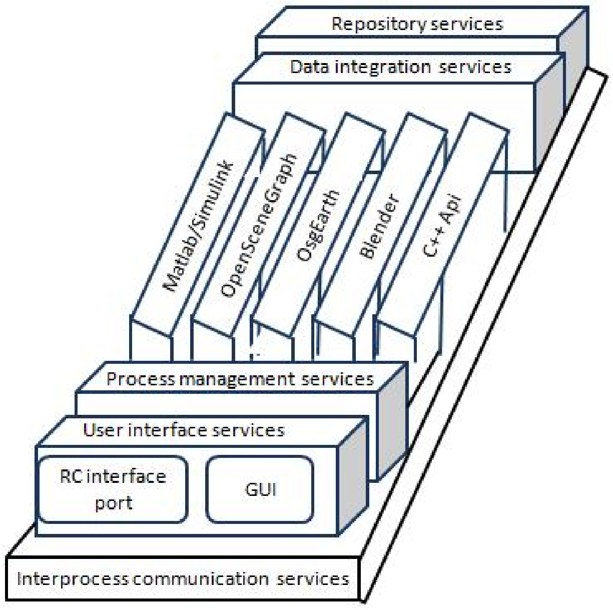


Figure 4. NIST/ECMA model for UAV environment integration

3. VIRTUAL MODEL SIMULATION

3.1. UAV Simulation System

The virtual UAV system has three main components: nonlinear dynamic model, auto pilot controller and path planner. Simulink from Mathworks provides a powerful design and development platform.

3.1.1. Nonlinear Dynamic Model Simulation

Dynamic of aircrafts have been studied extensively e.g.,[6] and models for UAV have been developed for aircrafts. Main block of UAV virtualization is to have a realistic model.

An airplane can rotate around three axes (x, y, z) from the plane's center of gravity. The position control of UAV is usually converted to the angular control: roll (φ), pitch (θ) and yaw (ψ).

The axes of motion of airplanes are shown in Fig. 5.

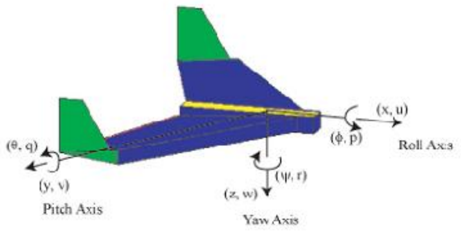


Figure 5. Euler angles which shows the angular movement of the UAV [5]

The main control surfaces or control inputs for a fixed-wing air vehicle include some or all of the following [6]:

- **Ailerons:** to control the roll angle.
- **Elevator:** to control the pitch angle (up and down).
- **Throttle:** to control the motor speed.
- **Rudder:** to control the yaw angle (left and right).

The developed unmanned air system is the 6-DOF physical air-frame that responds to servo command inputs (elevator, ailerons, rudder and throttle), wind and other disturbances.

We use following nonlinear models derived in [5].

$$\begin{pmatrix}\dot{p}_n\\ \dot{p}_e\\ \dot{p}_d\end{pmatrix}=\begin{pmatrix}c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi\\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi\\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta\end{pmatrix}\begin{pmatrix}u\\ v\\ w\end{pmatrix}$$

$$\begin{pmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{pmatrix}=\begin{pmatrix}rv-qw\\ pw-ru\\ qu-pv\end{pmatrix}+\frac{1}{m}\begin{pmatrix}f_x\\ f_y\\ f_z\end{pmatrix},$$

$$\begin{pmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}=\begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta}\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix}$$

$$\begin{pmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{pmatrix}=\begin{pmatrix}\Gamma_1 pq-\Gamma_2 qr\\ \Gamma_5 pr-\Gamma_6(p^2-r^2)\\ \Gamma_7 pq-\Gamma_1 qr\end{pmatrix}+\begin{pmatrix}\Gamma_3 l+\Gamma_4 n\\ \frac{1}{J_y}m\\ \Gamma_4 l+\Gamma_8 n\end{pmatrix}. \tag{1}$$

where the state variables of the UAV include

- p_n: the inertial (north) position.
- p_e: the inertial (east) position.
- p_d: the altitude or the height.
- u: the body frame velocity measured along body x axis.
- v: the body frame velocity measured along body y axis.
- w: the body frame velocity measured along body z axis.

- φ: the roll angle.
- θ: the pitch angle.
- ψ: the yaw angle.
- p: the roll rate measured along body x axis.
- q: the pitch rate measured along body y axis.
- r: the yaw rate measured along body z axis.

Simulink provides a powerful simulation platform and the implementation of this model is shown in Fig. 6, details are in [5].

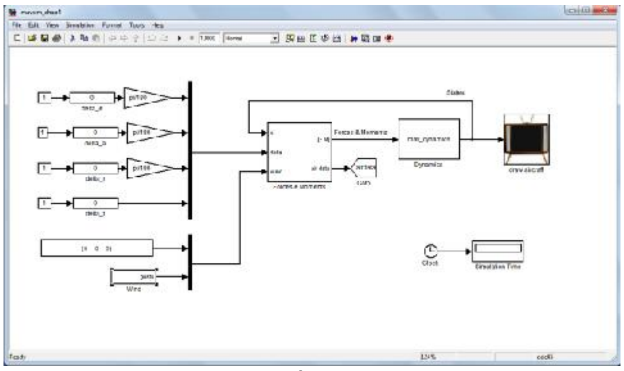


Figure 6. Simulink model of the UAV dynamic system [5]

When we look at the Simulink model in Fig. 6, there are two main blocks in the nonlinear dynamic model simulation system. First one of the main blocks is the Forces and Moments block, which takes the servo commands, wind and other disturbances as the input and outputs the forces and moments according to the inputs. Other main block is the Dynamics block, which takes the forces and moments as the input and outputs the UAV state information data according to the forces and moments applied and the UAV.

3.1.2. Autopilot System

Linearization around some predefined trim conditions has shown provide efficient controllers for such system. For the autopilot system, we commonly use proportional-integral-derivative (PID) controller loops for illustration. It is important to keep in mind that what we are after is to obtain a flexible design platform that enables us to develop different and advanced control architectures.

A general PID controller loop is given by the formula,

$$u(t) = k_p e(t) + k_\tau \int_{-\infty}^{t} e(\tau)d\tau + k_d \frac{de}{dt}(t) \qquad (2)$$

where $e(t) = y^c(t) - y(t)$ is the error between the commanded output yc(t) and the current output y(t). Controller loop tries to minimize the error between the commanded output and the current output.

Used again powerful Simulink vertical tool to design and test different controllers. In this work, we discard the climb and descend zone of the UAV model and we assume constant altitude for the UAV. The platform is flexible enough to include future extensions. Hence we need to control the heading of the UAV. For heading control, we used PID controller blocks from Simulink.

For the lateral autopilot system we used two PID controller blocks. First one of the controller blocks is chosen as the PI controller. It takes the commanded bank angle and the current bank angle, and outputs the roll rate. It is used for the UAV to meet the desired bank angle command. After tuning the parameters of the PI controller loop, we get the following response in Fig. 7 from the controller.

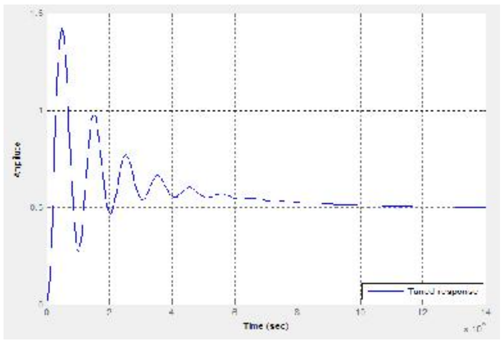


Figure 7. Roll loop response with PI controller

Second one of the controller blocks is chosen as PD controller. It takes the output roll rate of the first controller and the current roll rate of the UAV. It is used for the UAV to meet the desired roll rate. The output of the second controller is used as the aileron command for the dynamic model system. After tuning the parameters of the PD controller loop, we get the following response in Fig. 8 from the controller.

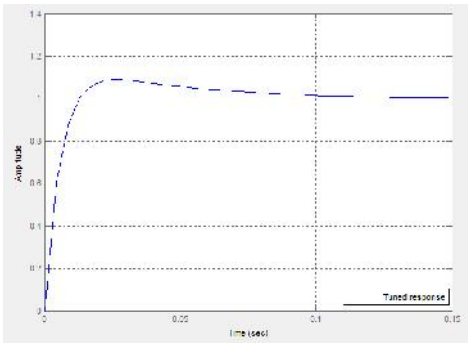


Figure 8. Roll rate loop response with PD controller

After designing the controller loops, we have a lateral autopilot system which gives the control inputs to the dynamic model automatically in order to meet the desired bank angle. It can replace the manual pilot control. It is very difficult for a normal user to control the UAV with manual control inputs. Hence this autopilot system is very useful for efficient control of the UAV model.

3.1.3. Path Planner

Path planning is another key component to achieve UAV virtualization. For the path planner system, the problem is to follow a prescribed path or ordered path from the ground station in real-time. We get the waypoints defining the path from the ground station as an ordered sequence of waypoints.

$$W = (w_1, w_2, ..., w_N), \text{ where } w_i = (w_{xi}, w_{yi})^T \in R^2.$$

For the path planning system, we used an algorithm of Ducard’s adaptive path planning technique [7]. The guidance system presented by Ducard provides an adaptive path planning algorithm that allows an UAV to navigate to waypoints by connecting the waypoints with straight lines called segments which are tracked by the UAV. Tracking the line segments is done through roll commands which are calculated continuously instead of yaw commands. This is a common practice in aircraft navigation. Fig. 9 shows the guidance law geometry of the path planning system.

In this figure, V is the aircraft velocity, L_1 is the distance from the center of the aircraft to a reference point, η is the angle between velocity vector and the line L_1, a_1 is the acceleration command, R is the radius of the circular segment and P is the reference point.

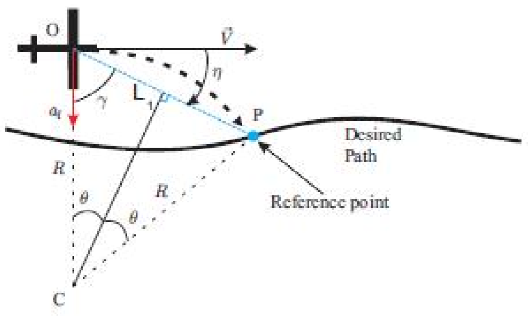


Figure 9. Guidance law geometry

The design parameter in the lateral guidance logic is the distance L_1 between the vehicle and the reference point. With a nominal flight velocity around 25 m/s, the distance L_1 has been chosen to be L_1: 150 m. We have approximately 20 m/s velocity in our UAV model, for this reason we used L1 = 150 m in our path planning system. In every update of the UAV model, we calculate the reference point according to the guidance algorithm and the route entered from the ground station. After calculating the reference point, lateral acceleration can be calculated as:

$$a_l = \frac{2V^2}{L_l} \sin \eta \tag{3}$$

In turn the lateral acceleration a_1 is converted to a bank angle command as $\emptyset_{com} \approx \frac{a_l}{g}$ and the commanded bank angle is applied to the autopilot of the UAV model.

After the path planner system, complete Simulink model of the UAV can be shown in the Fig. 10.

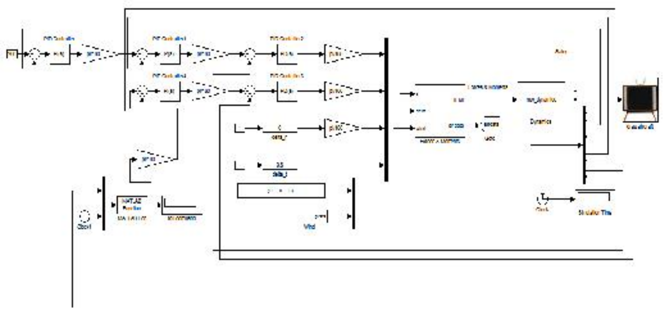


Figure 10. UAV Simulink model

4. GROUND CONTROL STATION

The ground control station plays a role as a terminal for end users to monitor and command the aircraft. The task of the ground station is to provide a friendly interface for users to monitor the flight from the ground station. Because the environment of the aircraft is not known from the ground station, visualization of the environment is very important. We particularly highlight the development of a 3D view interface which consists of a realistic flight simulation with terrain rendering.

Ground station software runs on a PC and consists of three views: User Interface view, Virtual Cockpit Display and a Navigation Map. User Interface screen is used for entering route for the UAV and for viewing the state information data gathered from the UAV. Virtual cockpit is used to get the affect to be on the aircraft, while controlling the aircraft. With simulated cockpit gauges we can easily visualize, manage, and control the UAV. Navigation map displays the map of the terrain and the aircraft can be tracked on the map.

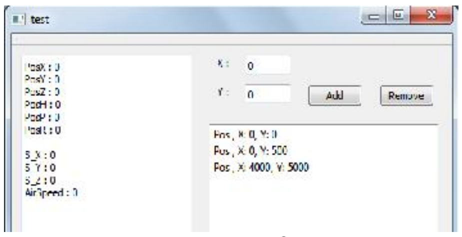
4.1. User Interface

User Interface screen is the only interface that the user interacts with the simulation. We implemented the UI screen in C++ language. The user can enter the route of the UAV on the screen from the ground station as an ordered sequence of waypoints.

$$W = (w_1, w_2, \ldots, w_N), \text{ where } w_i = (w_{xi}, w_{yi})^T \in R^2.$$

After entering the route, the route points are sent to the UAV model with the network connection.

In addition to the route entering, state information of the aircraft can be seen on the user interface. All the state information (position, orientation, velocity, …) are received from the UAV model and displayed on the UI screen as shown in Fig. 11.



```
PosX : 0
PosY : 0
PosZ : 0
PosH : 0
PosP : 0
PosR : 0

S_X : 0
S_Y : 0
S_Z : 0
AirSpeed : 0
```

X : 0
Y : 0

```
Pos , X: 0, Y: 0
Pos , X: 0, Y: 500
Pos , X: 4000, Y: 5000
```

Figure 11. User Interface screen

4.2. Virtual Cockpit

Virtual Cockpit displays the essential state and environment variables where the UAV is to give feedback to user. The virtual view of the aircraft is very useful when the UAV flies beyond the visible range of the users in the ground station. It provides users the realistic knowledge of the UAV status and it enables creation of a flight simulator platform to train pilots. Such a requirement is common for unmanned vehicles to perform practical tasks.

There are six main flight instruments shown in Fig. 12, which give the overall picture of the aircrafts flight conditions. These are called “standard six” and nearly all off the aircrafts have these instruments. These six primary instruments are airspeed indicator, attitude indicator, altimeter, turn and bank indicator, direction indicator and vertical speed indicator [8].

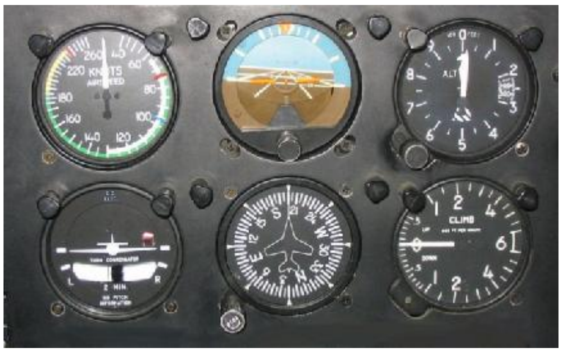


Figure 12. Six primary flight instruments

Grouping these indicators in the right way is very important in order to let the pilot clearly interpret the situation without confusion. We inform about the flight instruments shortly for better understanding [8].

- **Airspeed Indicator**

The airspeed indicator is a differential pressure gauge, that measures the difference between the air pressure in the pitot tube and the static, relatively undisturbed air surrounding the airplane. A needle displays this difference as airspeed.

- **Attitude Indicator**

Sometimes called the "artificial horizon," the attitude indicator is the only instrument that simultaneously displays both pitch and bank information. Pitch, bank and heading attitude are represented by one moving element. This is a surface that symbolizes the natural horizon. This moves in three axes to indicate the change in all three parameters simultaneously. A fixed horizontal line on the indicator represents the aircraft.

- **Altimeter**

This is a pressure-sensing device that requires direct input from the pressure sensing system. It's calibrated to display that air pressure as height, usually in feet above mean sea level.

- **Turn and Bank Indicator**

The turn coordinator shows the yaw and roll of the aircraft around the vertical and longitudinal axes. The airplane symbol shows the rate of turn, not bank angle.

- **Direction Indicator**

The direction indicator, sometimes called the “directional gyro” or “DG” is one of the three gyroscopic instruments. When aligned with the compass, it shows the aircrafts direction as a magnetic compass bearing.

- **Vertical Speed Indicator**

Also known as a rate-of-climb indicator, this is the third of the primary group of pitot-static flight instruments. The vertical speed indicator shows how fast an aircraft is climbing or descending.

After doing a trade-off analysis, we have chosen Blender horizontal tool to model and create 3D flight instruments. Blender is an open-source 3D computer graphics software product used widely for creating animated films, visual effects, interactive 3D applications or video games [9].

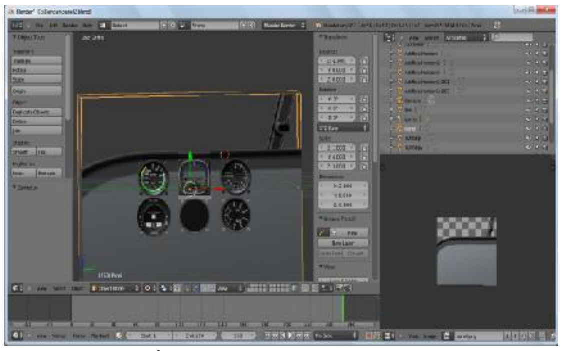


Figure 13. Modeled flight instruments and cockpit panel in Blender

After modeling the cockpit and flight instruments in Blender program as shown in Fig. 13, we used OpenSceneGraph graphics library for rendering the given models in C++. The OpenSceneGraph is an open source high performance 3D graphics toolkit, used by application developers in fields such as visual simulation, games, virtual reality, scientific visualization and modeling [10].

Created cockpit and flight instruments models are added into the scene by the help of OpenSceneGraph library. After adding the models into the scene, the models are rotated and translated to the proper positions. A camera is placed to a certain distance behind the cockpit model to visualize the cockpit display from the eye of a pilot. By this way, we obtained the cockpit view.

State information of the aircraft is received from the UAV model and the cockpit view is arranged according to the state information. At the same time the flight gauges display the UAV data.

For increasing the reality of the virtual cockpit display, it is needed to get the effect of flying on the cockpit of the aircraft over a terrain. Hence we need 3D terrain modeling and rendering. For this purpose we used a free, open source terrain rendering library osgEarth that is compatible with OpenSceneGraph.

OsgEarth is a terrain generation system for OpenSceneGraph applications. OsgEarth enables the development of geospatial applications in OSG, makes it as easy as possible to visualize terrain models and interoperate with open mapping standards, technologies, and data [11].

Combining satellite images and elevation data is a common technique for terrain generation and rendering. By the help of osgEarth library, we can combine satellite images and elevation data for the terrain. As the data, we used NASA satellite images and elevation data. After terrain modeling and rendering, we can see the virtual cockpit display more realistic as shown in Fig. 14.

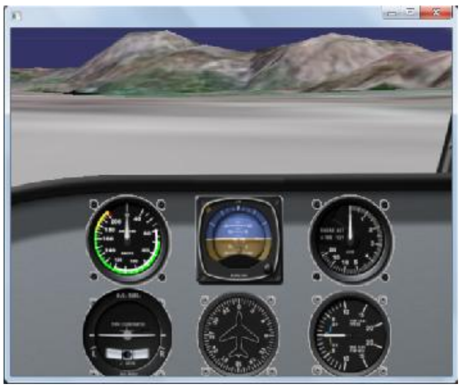


Figure 14. Virtual cockpit display

4.3. Navigation Map

Following the aircraft on a navigation map is another essential part of the ground control station. Virtual Cockpit view is not enough for efficient controlling and following the aircraft from ground station. It is very useful for the ground control user to see the position and the route of the aircraft on a map.

For navigation purpose, we used again osgEarth to visualize a 2D map of the terrain that the aircraft flies. The entered route that consists of waypoints and current position of the aircraft is shown on the map as in Fig. 15. In every update of the aircraft state info, position of the aircraft is updated.

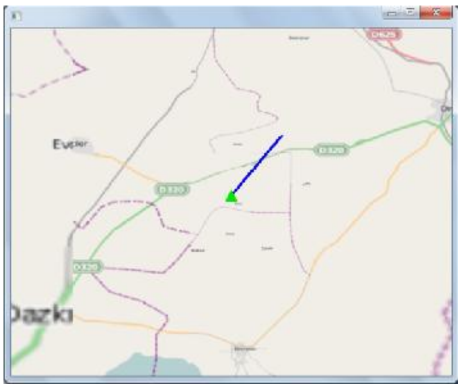


Figure 15. Navigation map display

5. NETWORK COMMUNICATIONS

One of the essential parts of the system is the network communications. A fundamental capability of the ground station is to broadcast and receive network distributed simulation messages. For this reason we define network architecture to enable inter-tool communication.

In real UAV systems, the communication between the ground station and the UAV is very important. All of the state information and useful data should be received from the UAV and the commands should be sent to the aircraft. This is done by a wireless communication channel. In our simulation system, we have the same situation. We have a distributed simulation system. Our ground station software is written in C++ language and works on any computer. On the other hand our UAV model is developed in another environment, Matlab/Simulink. So, we need a communication framework between them. For this purpose, we used UDP transmission protocol because of its advantages.

UDP, widely known as User Datagram Protocol, an alternative to the Transmission Control Protocol (TCP) and, together with IP, is also referred to as UDP/IP. Like the Transmission Control Protocol, UDP uses the Internet Protocol to get a data unit (datagram) from one computer to another. UDP is a simple and lighter protocol specifically because it doesn't have all the complexities of a TCP. That is UDP does not support reliability or acknowledgements and hence uses a best effort delivery model. In short the only real objective of the protocol is to serve as an interface between networking application processes running at the higher layers, and the internetworking capabilities of IP [12][13].

Strictly speaking, UDP is a “connectionless” protocol, a program can use a single UDP socket to communicate with more than one host and port number, but it is convenient for most UDP client programs to maintain the fiction that there is a connection, by keeping a local record of each server host and port number. To be a UDP client, a program must open a UDP socket with the host name and port.

fd = open (host_and_port, “udp-client-socket”);

Then, the program can start creating and sending datagrams from this UDP socket.

The *send (fd, datagram)* API is used for sending messages where datagram is simply a string. For listening incoming messages we should check periodically the UDP socket for incoming datagrams using *recv(fd, *buffer, length)* API or we can *bind* to the incoming socket and the API notifies us about the incoming datagrams. For this simple transmission mechanism, there is no need to call other API calls such as connect(), waitForConnection(), accept(), etc., we have chosen UDP protocol for our platform and in our simulation system.

Our UAV model side, Matlab/Simulink, supports UDP transmission with its useful API too, as:

Open a socket: *fd = udp(host, port); fopen(fd);*

Send message: *fwrite(fd, datagram);*

Receive message: *[datagram, size] = fread(fd);*

A network manager is developed in C++ for socket based programming using UDP stack. By the help of the network manager, we can broadcast and receive network distributed simulation messages. We can see the network architecture in Fig. 16.

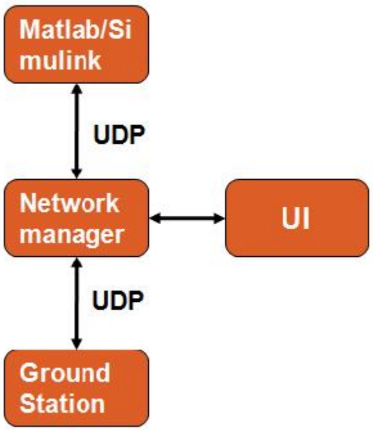


Figure 16. Network architecture

Commanded route points are sent to the UAV model from the ground station and state information is received from the UAV model by the help of UDP connection. Because of this distributed mechanism, we can run the ground station and UAV model on different computers easily.

6. SIMULATION

After developing the platform, we run a simulation of a flight for demonstration. For the simulation environment we used a reference point on the world as our reference (0,0) location. We used a point in Turkey and used Turkey maps from *http://readymap.org* for this environment. We entered five waypoints from the Ground Station UI as the route of the UAV.

w1: x: 0, y: 0
w2: x: 0, y: 1000
w3: x: 500, y: 2000
w4: x: 2000, y: 2000
w5: x: 3000, y: 3000

For the dynamic model system it is essential to use the points in meters for calculations. Hence, the waypoints are all in meters. But to show the UAV on the correct position on the map, coordinate transformation is needed. After calculating the state information of the UAV in every update, position of the UAV which are in local coordinates (in meters) should be converted into geodetic coordinate system (longitude and latitude positions). The local coordinates are transformed into the geographic coordinates according to the reference point.

When we run both of the ground station and UAV model simultaneously, we got the following figures, Fig. 17-19, from the ground station software.

While the simulation is running, we can see the 3D terrain and the flight instruments on the cockpit view. They are updated simultaneously with the UAV state information received from the UAV model.

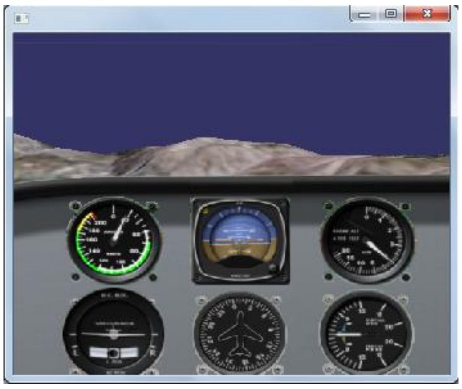


Figure 17. Virtual cockpit display

From the navigation map, we can see the entered route as a blue line and the travelled route as a red line. Current position of the UAV is shown as a green triangle. From this map, the UAV can be tracked on the terrain easily.

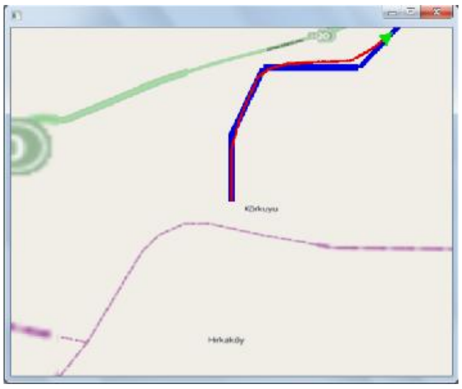
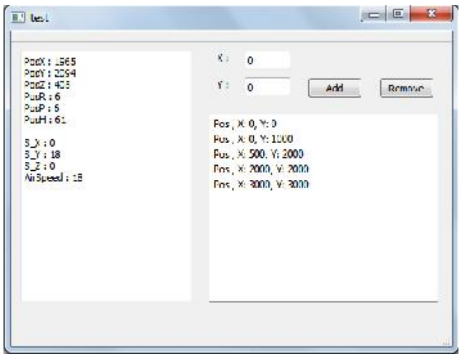


Figure 18. Navigation map



```
PosX : 1965
PosY : 2094
PosZ : 405
PsuR : 6
PsuP : 5
PsuH : 61

S_X : 0
S_Y : 18
S_Z : 0
AirSpeed : 18
```

```
Pos , X: 0, Y: 0
Pos , X: 0, Y: 1000
Pos , X: 500, Y: 2000
Pos , X: 2000, Y: 2000
Pos , X: 3000, Y: 3000
```

Figure 19. UI display

We did not put all the ground station views on one screen, because user can want to see one of the screens in full screen mode. For example, user can full screen the virtual cockpit view only to get the effect of using the UAV on it.

At the same time, we are displaying the UAV model info at the UAV model simulation side to see the produced values, as in Fig. 20-21.

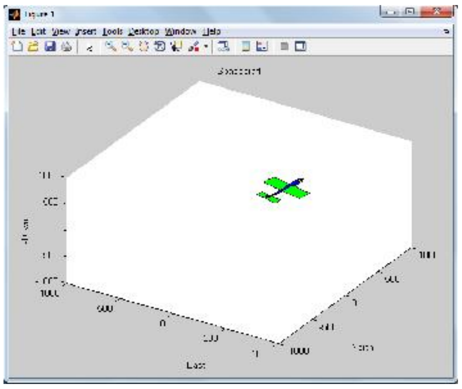


Figure 20. UAV model [5]

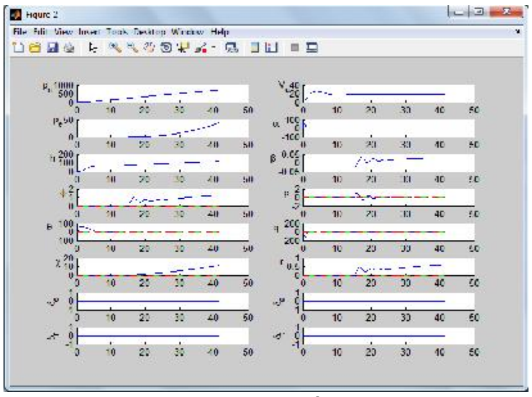


Figure 21. UAV State info graphics [5]

These figures make it possible to compare the produced values on the UAV model with the received values on the ground control station.

7. CONCLUSION

Recent advancements in software and hardware technologies made possible to develop advanced design, analysis and development platforms. Sometimes they are built in an integrated environment but in some other cases they need to be built in distributed environments. In this work, we put together such a distributed platform targeting UAV systems. Our architecture is composed of user interfaces, open source tools, public domain data and libraries, together with educational Matlab/Simulink tool. Our platform offers user interface for monitoring, analysis, design and pilot training. The platform supports:

- Analysis and simulation of UAV systems
- Advanced control algorithm development
- Guidance and navigation support
- Instrumentation with 2D Map support
- 3D Earth terrain visualization
- Ground station simulation

All key components have successfully integrated and needed glue code is developed in C++ to stick the tools around an UDP communication channel by the help of a developed network manager. A simulation example is illustrated for a flight over Turkey using algorithms from literature

ACKNOWLEDGMENT

I would like to thank TUBİTAK BILGEM for their support in this work.

REFERENCES

[1] Stephen A. Cambone, Keinheth J. Krieg, Peter Pace, and Linton Wells, "*USA's Unmanned Aircraft Roadmap, 2005-2030*" National Defense, August 2005.

[2] D. Jung, E. J. Levy, D. Zhou, R. Fink, J. Moshe, A. Earl, and P. Tsiotras, "*Design and development of a low-cost test-bed for undergraduateeducation in UAVs*", in 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on Decision and Control, December 2005,pp. 2739–2744.

[3] H. Wu, D. Sun, and Z. Zhou, "*Micro air vehicle: Configuration, analysis, fabrication and test*", IEEE/ASME Transactions on Mechatronics , vol.9, no. 1, pp. 108–117, March 2004.

[4] Sorton, E. F. and Hammaker, S., "*Simulated Flight Testing of an Autonomous Unmanned Aerial Vehicle Using FlightGear*" Infotech@Aerospace, Arlington, VA, Sept. 2005, AIAA 2005-7083.

[5] Randal W. Beard and Timothy W. McLain "*Small Unmanned Aircraft: Theory and Practice*", Princeton Press, 2012.

[6] R. C. Nelson, "*Flight Stability and Automatic Control*", Boston, Massachusetts: McGraw-Hill, 2nd ed., 1998.

[7] G. Ducard, K. C. Kulling, H. P. Geering, "*A Simple and Adaptive On-Line Path Planning System for a UAV*", Proceedings of the IEEE 15th Mediterranean Conference on Control and Automation,, T34-009, pp. 1-6, Athens, Greece, June 2007.

[8] Pallett, E. H. "*Aircraft Instruments*", Longman Scientific & Technical, Essex, 2nd ed. 1991.

[9] http://www.blender.org/, Date accessed 28/09/2012.

[10] http://www.openscenegraph.org/projects/osg, Date accessed 28/09/2012.

[11] http://osgearth.org/, Date accessed 28/09/2012.

[12] W. Richard Stevens. "*UNIX Network Programming*", volume 1. Prentice-Hall, Upper Saddle River, NJ, second edition, 1998.

[13] content@ipv6.com. "UDP Protocol Overview". Ipv6.com. Date accessed 28/09/2012.