World Academy of Science, Engineering and Technology
International Journal of Aerospace and Mechanical Engineering
Vol:16, No:2, 2022

Real Time Data Communication with FlightGear Using Simulink over a UDP Protocol

Adil Loya, Ali Haider, Arslan A. Ghaffor, Abubaker Siddique

Abstract**—**Simulation and modelling of Unmanned Aerial Vehicle (UAV) has gained wide popularity in front of aerospace community. The demand of designing and modelling optimized control system for UAV has increased ten folds since last decade, as next generation warfare is dependent on unmanned technologies. Therefore, this research focuses on the simulation of nonlinear UAV dynamics on Simulink and its integration with FlightGear. There has been lots of research on implementation of optimizing control using Simulink, however, there are fewer known techniques to simulate these dynamics over FlightGear and a tedious technique of acquiring data has been tackled in this research horizon. Sending data to FlightGear is easy but receiving it from Simulink is not that straight forward, i.e. we can only receive control data on the output. However, in this research we have managed to get the data out from the FlightGear by implementation of level 2 s-function block within Simulink. Moreover, the results captured from FlightGear over a Universal Datagram Protocol (UDP) communication are then compared with the attitude signal that were sent previously. This provide useful information regarding the difference in outputs attained from Simulink to FlightGear. It was found that values received on Simulink were in high agreement with that of the FlightGear output. And complete study has been conducted in a discrete way.

Keywords**—**Aerospace, flight control, FlightGear, communication, Simulink.

I. Introduction

FLIGHT control simulations has gained wide importance in the field of avionics and aerospace engineering. The usage of flight control simulations helps aeronautical students to practically understand the phenomena of flight mechanics and its application. There is a big room for optimization and augmentation. Student has more fidelity over the aircraft control and can understand aircraft aspects in greater detail. However, designing and implementation of the flight control simulation requires deep understanding of different subject areas interconnected with the flight control topic. Moreover, with the help of affordable computational power, flight control simulations has increased rapidly.

Designing and modelling of the external Flight Dynamic Model (FDM) has gained immense importance. As now there are commercially available open source platform of testing FDM in visualization mode. FDM designing and modelling is not new and generic models are already available like JSBSim, YASim, and XPlane based FDM. However, each of them has their own pros and cons. In similar way FDM can be designed and tested on the MATLAB/Simulink platform while integrated with the open source flight gear simulation package. This gives fidelity to change and optimize settings of the aircraft in real-time [1]. Moreover, nonlinear parameters can also be validated.

Communication with FlightGear requires understanding of packet unpacking over a Universal Datagram Protocol (UDP). Protocol settings are set in internal data folder of FlightGear to fetch data and send it to Simulink in real-time.

M. Moness et al. demonstrated a successful connectivity of FlightGear to MATLAB/Simulink using UDP protocol, but, acquiring data on Simulink was not their goal rather they wanted output on image generation i.e. FlightGear. In their research, a successful test of feedback control autopilot system for C310 aircraft model was evaluated [2].

Jianbin Ye et al. also formulated a dynamic controls using MATLAB/Simulink and visualized its output on the FlightGear using UDP protocol, however, they did not work on receiving the data from the FlightGear to Simulink [3].

Ondris Daniel and Rudolf Audoga utilized the internally built in block of Simulink library to fetch and control data from the FlightGear over the UDP protocol. However, as already noted that internal block from MATLAB library only gives output of the control data from FlightGear and not orientations [4].

In addition to MATLAB Simulink based system, umanned helicopter autopilot was designed and tested by M Reid and S Manso with FlightGear as image generation tool. They designed mex based s-function for sending signals over the UDP protocol [5].

Oktay Arslan et al. designed touch screen C2 interface for realistic lab-scale flight experiments in which FlightGear was used as image generation tool and communication of flight dynamic model was performed using UDP protocol in between Simulink during Hardware in loop simulation [6].

It is being noted from a previous study of Metalli Matteo that MATLAB function can be implemented to acquire data from the Simulink using Judp MATLAB function for protocolling, however, it requires datatype conversions [7].

Therefore, this paper presents a simple algorithm and method of collecting data from the FlightGear. UDP is used for communication between the FlightGear and Simulink. Earlier before researchers who contributed to this area came up with difficult mex based s-functions on MATLAB/Simulink to fetch output parameters, however, it only gives control based parametric output from FlightGear to Simulink. Therefore, there was a need to design simple and robust method of collecting any property node from FlightGear so that one can

Adil Loya is with the Department of Mechanical Engineering, National University of Sciences and Technology, Karachi, Pakistan (Corresponding author, e-mail: loya_adil@yahoo.com).

Ali Haider, Arslan A. Ghaffor and Abubaker Siddique are with the Rawalpindi, Cambridge Hartford School, Rawalpindi, Pakistan.

further use it for data quantification. A structured layout of the paper has been demonstrated in graphical manner as shown in Fig. 1.

II. Methodology

The focus of this paper is designing a protocolling technique for capturing the FlightGear property nodes via MATLAB/Simulink. In this research, data was collected by running the digital signal processing tool box of Simulink library and then settings of the block properties was carried out and secondly, level 2 S-function was also used to design algorithm to send and receive property nodes of FlightGear. The connection between the FlightGear and Simulink was commenced using UDP protocol. For this purpose, already created MathWorks function i.e. Judp which is available for free on the internet was used. The data was collected from FlightGear by designing a protocol file in an xml format which was saved in the 'Protocol' folder of FlightGear. This is also demonstrated in the schematic diagram of Fig. 2. This was used to fetch data from the FlightGear in a chronological order. After the communication is established, data was displayed in Simulink output displayer.

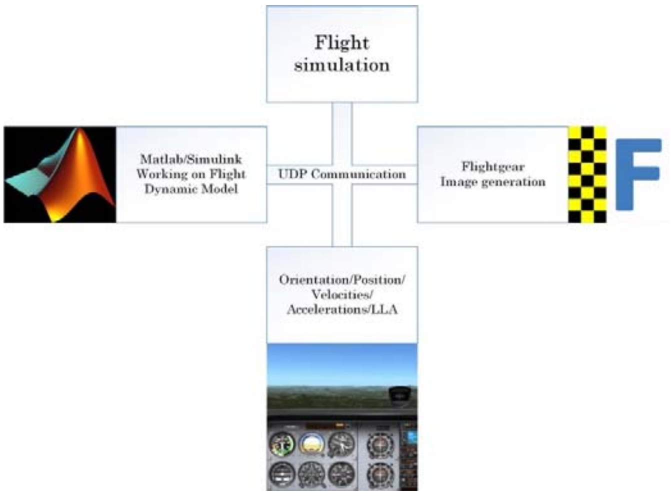
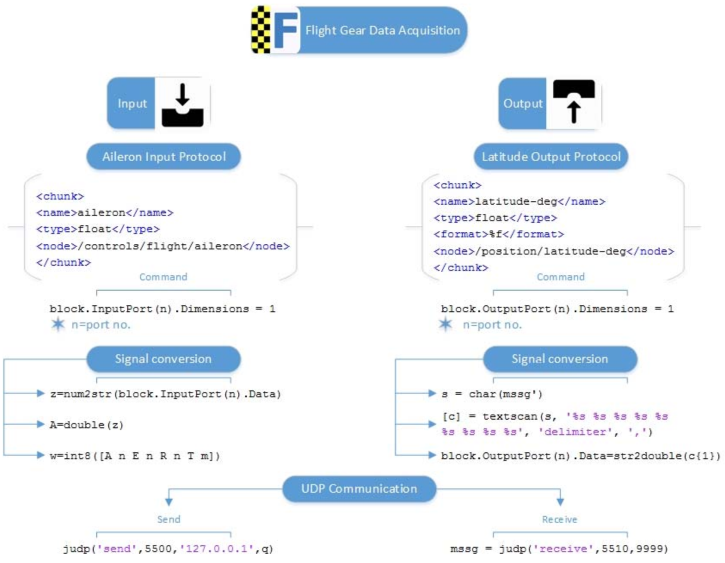

Flight simulation
Matlab/Simulink Working on Flight Dynamic Model
UDP Communication
Flightgear Image generation
Orientation/Position/ Velocities/ Accelerations/LLA

Fig. 1 Representing the UDP communication between the MATLAB/Simulink and FlightGear

The flow and working of the algorithm on MATLAB are shown in a schematic Fig. 2; it represents the steps of setting it up in the MATLAB prompt and then calling it in Simulink as a level 2 s-function. This creates a block in Simulink with receiving and sending ports. At the end of receiving ports, we plug scopes for results visualization and at input port, a constant can be plugged to send values of interest to FlightGear. Moreover, in Fig. 2, aileron input protocol is defined for an example of sending particular parameter to FlightGear, this is further defined in MATLAB using "block.InputPort(n).Dimensions=1" command. Later the dimensional parameter from Simulink is converted from numeric to string using "z= num2str(block.InputPort(n).Data)", then it is converted to Double using "A=double(z)". Finally, this parameter is defined as an 8-bit Integer parameter i.e. using "w = int8 (A)". The above generated value is then send from the Simulink to FlightGear using UDP communication by "udp('send',5500,'127.0.0.1',q)" command.

Output from FlightGear is fetched in Simulink using similar way as before, however, there are some changes with commands, for receiving data "block.OutputPort(n).Dimensions = 1" is used. This introduces an output port on the level 2 S-function block, where n defines number of output ports. Furthermore, the receiving message is allocated with a variable where the message is defined in character form and then message is transposed at the same time to receive it in form of columns using "s = char(mssg')". Then from this message different text are scanned using "[c] = textscan(s, '%s %s %s %s %s', 'delimiter', ','). Here five "%s", defines five output being received from FlightGear, so user can set as many as required. Finally, the outputs are converted into double format from strings using "block.OutputPort(n).Data=str2double(c{1})" command. Receiving of the data from UDP protocol is carried out using "mssg = judp ('receive',5510,9999).



```
<chunk>
<name>aileron</name>
<type>float</type>
<node>/controls/flight/aileron</node>
</chunk>

block.InputPort(n).Dimensions = 1

z=num2str(block.InputPort(n).Data)
A=double(z)
w=int8([A n E n R n T m])

judp('send',5500,'127.0.0.1',q)
```

```
<chunk>
<name>latitude-deg</name>
<type>float</type>
<format>%f</format>
<node>/position/latitude-deg</node>
</chunk>

block.OutputPort(n).Dimensions = 1

s = char(mssg')
[c] = textscan(s, '%s %s %s %s %s
%s %s %s %s', 'delimiter', ',')
block.OutputPort(n).Data=str2double(c{1})

mssg = judp('receive',5510,9999)
```

Fig. 2 Protocol and signal conversion settings for data input and output from flight gear to simulink

III. Results

Setting up protocols on MATLAB/Simulink helped in receiving data from the FlightGear. Two methods used for communicating and receiving the data from FlightGear are presented in Figs. 3 and 5. However, Fig. 4 illustrates settings that were carried out under the UDP block of DSP Toolbox. In Fig. 3 the data was perceived in the American Standard Code for Information Interchange (ASCII) data type; in addition to this, the ASCII format string was set to "%f". IP address was set to '127.0.0.1' and port was set to '5510'.

Moreover, datasize was set to number of parameters that we needed to fetch from the protocol file. In our case it was set to 9, as 9 parameters were being fetched in real-time from FlightGear. Lastly, results received are displayed in Fig. 3, where it can be seen that 9 parameters are being fetched, out of which 3 parameters of orientation are visible in the Simulink scope.

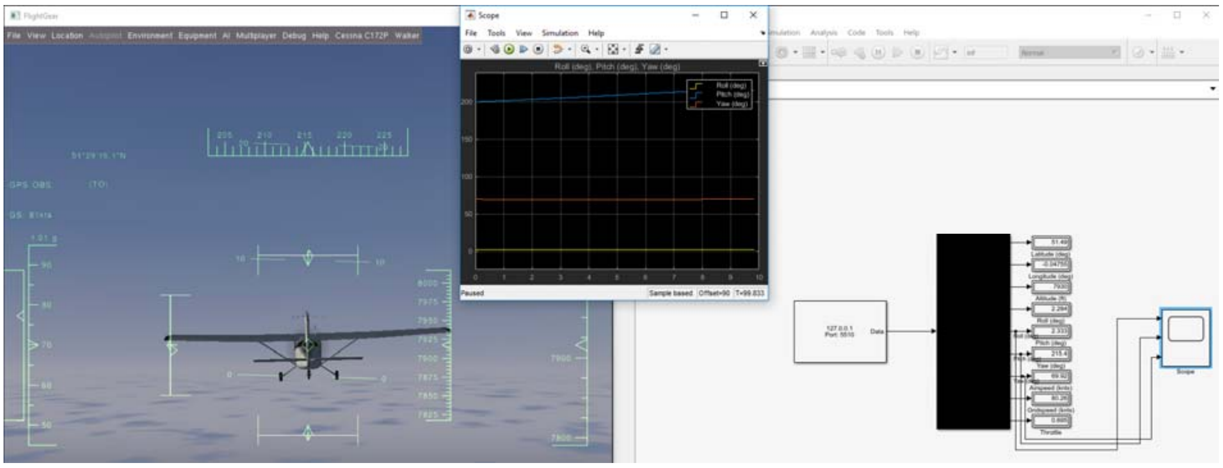
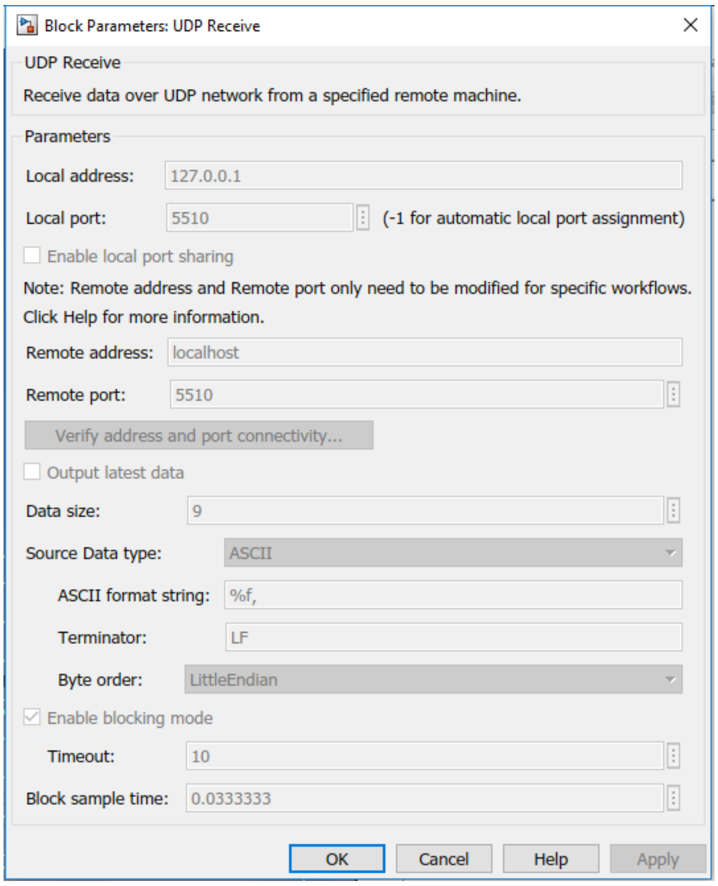


Fig. 3 Receiving data from FlightGear to Simulink using UDP protocol of DSP toolbox of MATLAB/Simulink



Block Parameters: UDP Receive

UDP Receive

Receive data over UDP network from a specified remote machine.

Parameters

Local address: 127.0.0.1

Local port: 5510 (-1 for automatic local port assignment)

Enable local port sharing

Note: Remote address and Remote port only need to be modified for specific workflows. Click Help for more information.

Remote address: localhost

Remote port: 5510

Verify address and port connectivity...

Output latest data

Data size: 9

Source Data type: ASCII

ASCII format string: %f,

Terminator: LF

Byte order: LittleEndian

[✓ Enable blocking mode]

Timeout: 10

Block sample time: 0.0333333

OK Cancel Help Apply

Fig. 4 Setting under the hood of the UDP block of DSP toolbox

Fig. 5 demonstrates receiving of the data using level-2 based S-function on Simulink. This s-function was written and tested several times. The reader has two easy methods to help them self out in receiving the parameters from the FlightGear. Moreover, this function also helps the user to receive and send data to the FlightGear simultaneously, enabling easy communication of data transfer between the FlightGear and Simulink.

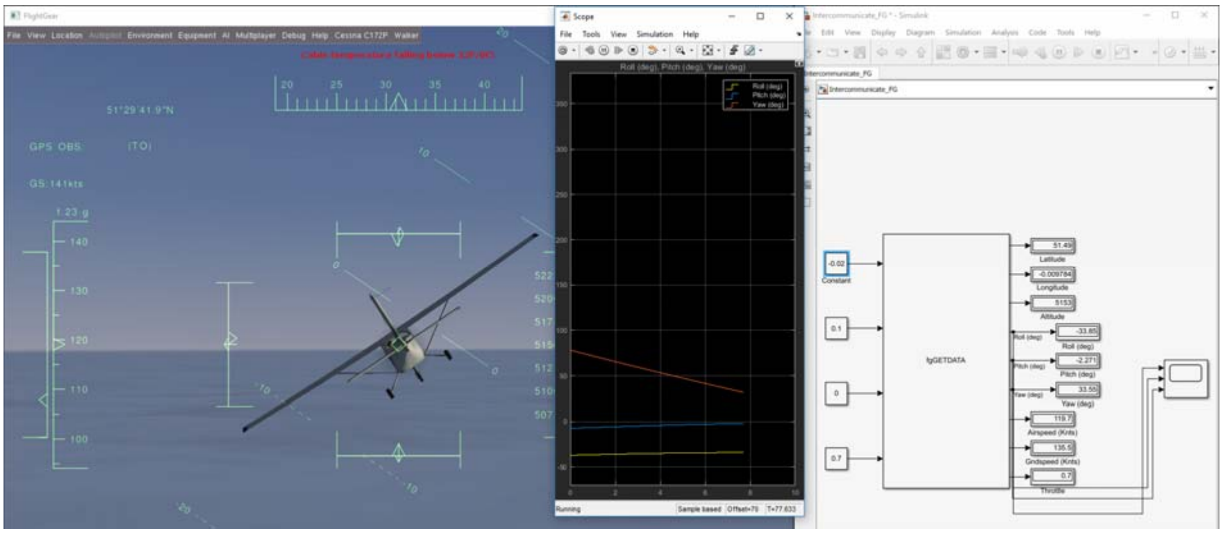


Fig. 5 Receiving data from FlightGear using level-2 S-Function in Simulink

IV. Discussion

From the above study it is depicted that achieving data from FlightGear is easy, however, requires understanding with regards to data type conversion and communication between the UDPs. Currently, this interfacing Simulink program will help in research of aerospace community to easily fetch and stream data from FlightGear. However, already mentioned before this was hard and some of the parameters i.e. related with controls were only acquirable. Researchers have tried out several ways to retrieve data from FlightGear using C, C++, java, etc.; however, with Simulink, it is scarcely found. Therefore, the novel method of this paper gives authority to user to fetch any input/output property node through FlightGear using Simulink and MATLAB. Moreover, this paper allows researchers of aerospace community to utilize this method with their embedded codes for data integration.

V. Conclusion

The research provides two promising methods of fetching any property node from FlightGear to Simulink. Several properties were fetched from FlightGear to Simulink with a data rate of 30 frames per second, which demonstrates data acquisition in real-time. Moreover, this can further be used for integration with aerospace system modellings. This research enables users to utilize two distinctive methods to acquire required property of interest from FlightGear.

References

[1] G. Aschauer, A. Schirrer, and M. Kozek, "Co-simulation of MATLAB and FlightGear for identification and control of aircraft," *IFAC-PapersOnLine,* vol. 48, no. 1, pp. 67-72, 2015.

[2] M. Moness, A. M. Mostafa, M. A. Abdel-Fadeel, A. I. Aly, and A. Al-Shamandy, "Automatic control education using FlightGear and MATLAB based virtual lab," in *8th International Conference on Electrical Engineering*, 2012, pp. 1157-1160.

[3] J. Ye, H. Guo, S. Tang, and Q. Wang, "The research on visual flight simulation for unmanned helicopter," in *AsiaSim 2012*: Springer, 2012, pp. 332-341.

[4] D. Ondriš and R. Andoga, "Aircraft modeling using MATLAB/flight gear interface," *Acta Avionica,* vol. 15, no. 27, 2013.

[5] M. Reid and S. Manso, "Development of a Rotary Wing Unmanned Aerial Vehicle (UAV) Simulation Model," DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION VICTORIA (AUSTRALIA)2014.

[6] O. Arslan, B. Armagan, and G. Inalhan, "Development of a Mission Simulator for design and testing of C2 Algorithms and HMI Concepts across Real and Virtual Manned-Unmanned Fleets," in *Optimization and Cooperative Control Strategies*: Springer, 2009, pp. 431-458.

[7] M. Metalli, "Sviluppo di un modello di drone di tipo flying wings in ambiente FlightGear interfacciato con autopiloti di Arduino implementati in Simulink."