IOP Conf. Series: Journal of Physics: Conf. Series **1005** (2018) 012018 doi:10.1088/1742-6596/1005/1/012018

Design of Flight Control Panel Layout using Graphical User Interface in MATLAB

A Wirawan[1, 3) *], T Indriyanto[2)]

[1)] Aeronautic Technology Centre, National Institute of Aeronautics and Space (LAPAN), Indonesia
2) Faculty of Mechanical and Aerospace Engineering, Institut Teknologi Bandung, Indonesia
3) Institut für Luft- und Raumfahrt, Technische Universität Berlin, Germany

*adi.wirawan@lapan.go.id

Abstract. This paper introduces the design of Flight Control Panel (FCP) Layout using Graphical User Interface in MATLAB. The FCP is the interface to give the command to the simulation and to monitor model variables while the simulation is running. The command accommodates by the FCP are altitude command, the angle of sideslip command, heading command, and setting command for turbulence model. The FCP was also designed to monitor the flight parameter while the simulation is running.

1. Introduction

In general, an aircraft is controlled by the following principal modes: mechanical control mode and electronic control mode. The electronic control mode can support two types of control, which are automatic control and manual control. In automatic control mode, pilot sends input commands to Electronic Flight Control System (EFCS) using Flight Control Panel (FCP). The FCP also display flight plan sequentially as a function of aircraft position.

The Flight Control Panel (FCP) also has functioned as Human Machine Interface (HMI). In this case, the FCP is placed in the avionic panel in front of the pilot. The FCP has three main functions:

1. To load and transmit initialization data to Flight Control Computer (FCC) to be read by Flight Control Law (FCL). The loaded data are aircraft and weight data, waypoints or flight path pattern and mission data, as well as runway data.
2. To allow the pilot to input data or to manually give commands to EFCS. Commands given by pilot to EFCS are engaged or disengage EFCS in normal condition, various switching command modes, and flight profile commands (airspeed, height, and heading).
3. To display information that the pilot has to observe during flight such as flight condition, EFCS status and visual warnings [9].

The FCP in these research is required to sends input commands to the flight control model and turbulence model and also to display aircraft data model.

2. Study on FCP Layout

LAPAZ Project

LAPAZ Project (abbreviation for ***L**uft-**A**rbeits-**P**lattform fur die **A**llegemeine **Z**ivilluftfahrt in German*, translated as an aerial work platform for general civil aviation). The project is intended to develop and demonstrate a reliable, high-precision automatic flight control system for the S15 aircraft to support in flight geo-exploration and surveillance tasks. The partners are German aircraft manufacturer STEMME, the Universität Stuttgart and Technische Universität Berlin (TUB).

Automatic Flight Control Panel (AFCP) of LAPAZ project is an example of human machine interface in aircraft. LAPAZ AFCP allows the pilot to engage and disengage AFCS, to set target values for various flight states (airspeed, altitude, course, etc.), to switch between flight control modes as well as to load predefined waypoints and runway data. [3]

The AFCP consists of a central processor unit, a display, push buttons and rotary encoders as shown in **Figure 1**.

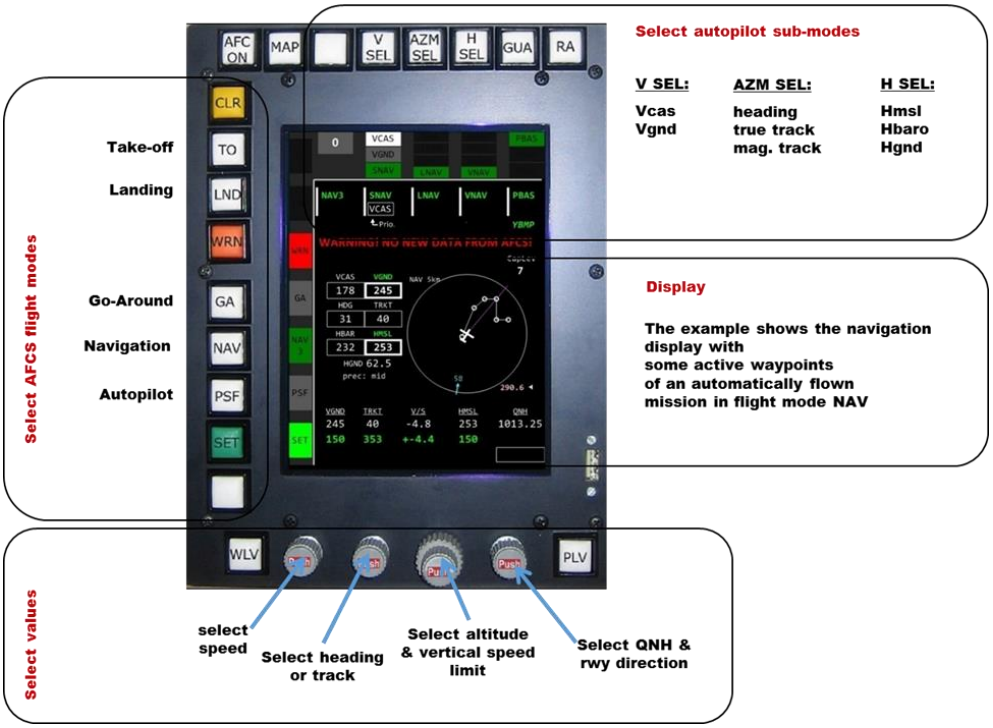


Figure 1. FCP of the LAPAZ Project

TEMO HMI

Another example of dedicated HMI platform is TEMO HMI for cockpit touch screen display as shown in **Figure 2**. The TEMO HMI consists of a Primary Flight Display (PFD), Navigation Display (ND), Vertical Situation Display (VSD) and a Time & Energy Display (TED) [8].

Generally, two types of cockpit applications distinguished by symbology applications and interactive applications. Symbology applications present one-way information to pilot. Typically, symbology displays are data-driven applications. Interactive applications enable the pilot to react on the information shown by means of, for example, a toggle button, push button or combo box. This type of information exchange has an event-driven character [12].

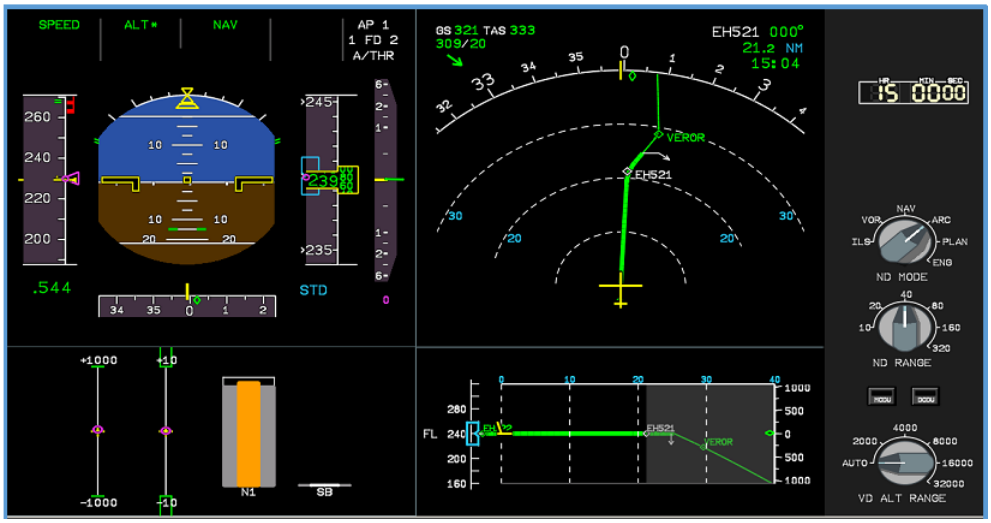


Figure 2. *TEMO cockpit HMI [8]*

HMI for A380 flight test

Another example of HMI (**Figure 3)** is a unique HMI used in flight test and On-board computing for Airbus A380 that has been designed to integrate all test monitoring applications [2].

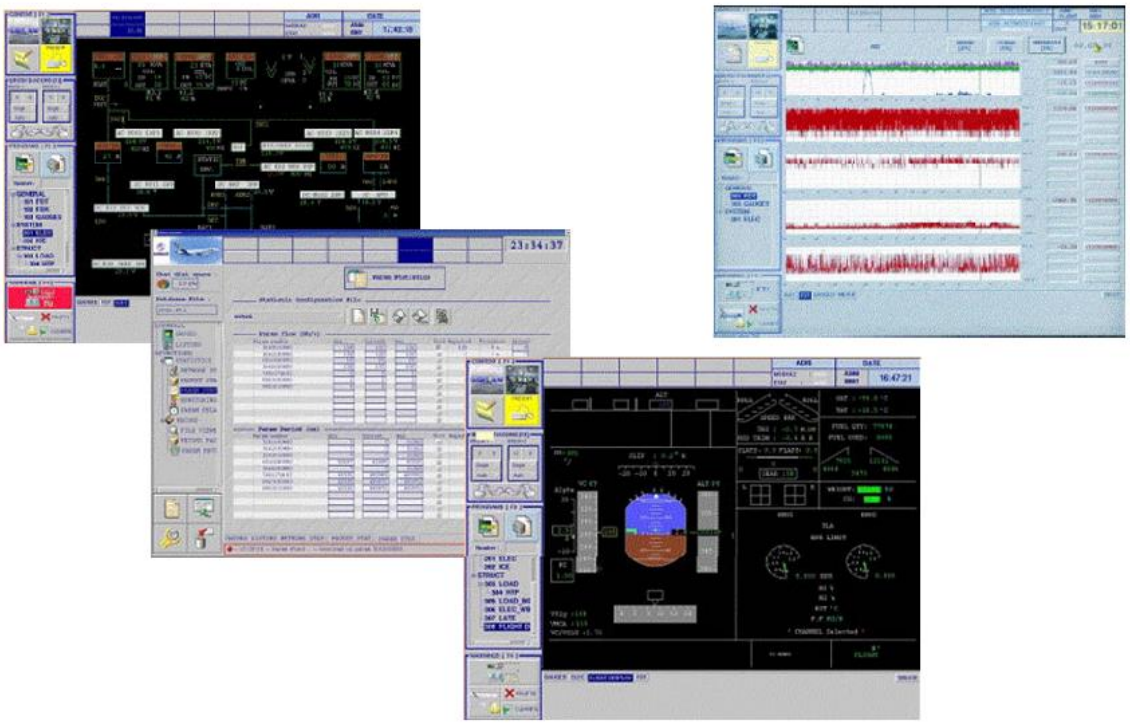


Figure 3. HMI for Flight Test of Airbus A380 [2]

3. Requirements and Design Process

The method used within the research are: define the key function of FCP, design the layout, then implement the layout by using GUIDE Matlab, iterate the design to fulfil all requirements, and finally test the communication within the FCP program and with simulation program in X-Plane and Simulink.

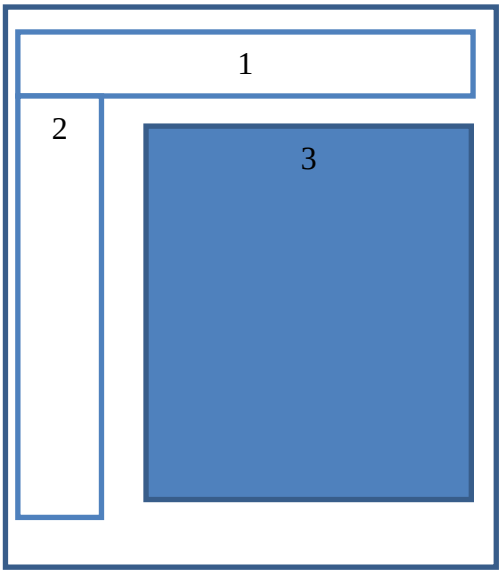
Key maps for the initial design of FCP in this research consist of three major functions as required in FCP requirements, i.e. input command, display flight parameter and load and transmit initialization. The FCP key maps function is shown in Table 1.

Table 1. FCP key maps function

Input Command	Initialization	Display
Trimming command Velocity command Heading command Beta command	Set initial condition Set loop mode Set turbulence condition	Velocity Altitude Heading Beta Visualization of aircraft heading

Layout

Because of the similarity of the functions, the layout of FCP in this research is designed similar to LAPAZ AFCP. For current task, the design will follow Matlab GUIDE in [6].



1. Set of command input
2. Sets of initialization
3. Display

Figure 4. Initial FCP layout design

Initial FCP layout design as shown in **Figure 4** consists of three main block as defined in FCP key maps function in Table 2. The layout consists of command input block, initialization block, and display block. After several evaluations and iterations, the final layout is as shown in **Figure 5**.

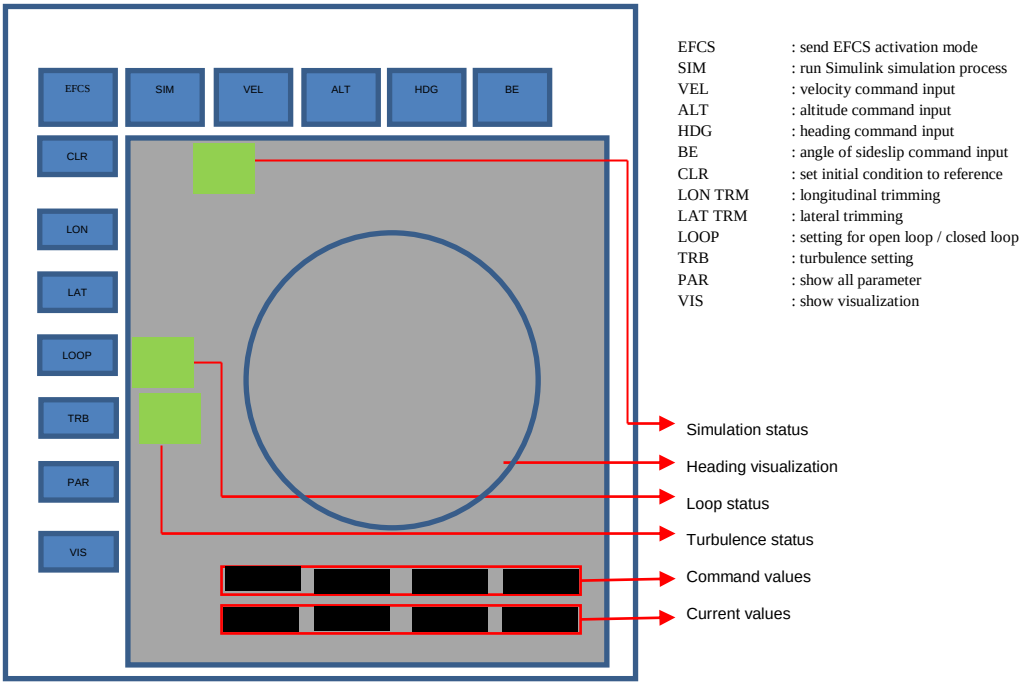


Figure 5. Final FCP Layout design

MATLAB GUIDE

MATLAB has a built-in Graphical User Interface Development Environment (GUIDE), which can be used to lay out GUI graphically and generate MATLAB code automatically [6]. The GUI is a modern tool found in MATLAB versions 7.0 or later, and not automatically compatible with version 6.5 and earlier [11]. GUIDE automatically generates two kinds of Matlab files: one file for Matlab interface figures and another M-file used to store command function of Matlab program [5]. In previous Matlab version, users had to write Matlab algorithm in an M-file and if there were any changes, they had to go back to M-file to make the changes.

General procedure to design GUI in MATLAB are shown in **Figure 6**. The design starts with user interface design where programmer put button, text, editor or other GUI objects and then continue with programming callback function for each GUI objects to define the behavior and responses of the objects. After programming callback functions, users then continue to build M-file as editor for user interface design. In the build M-file process if the process success then continues to the final process that is the creation of user interface for program application, but if still, any function needs to add then the process back to the callback functions.

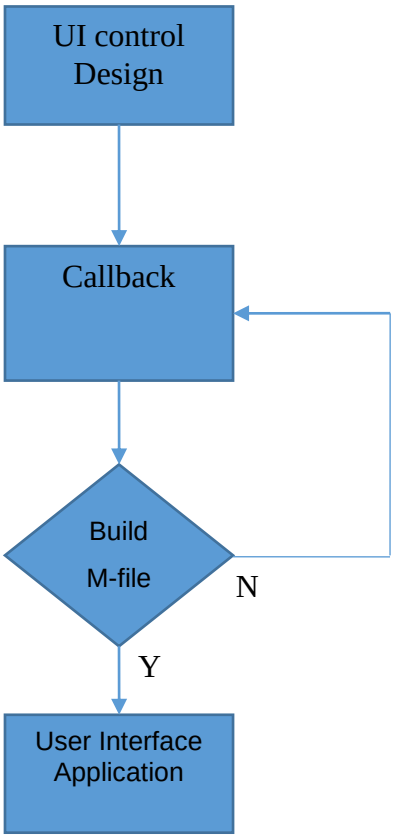


Figure 6. General flowchart of GUI design with Matlab

Figure 7 shows an example of Quick Start window to create a new GUI in MATLAB.

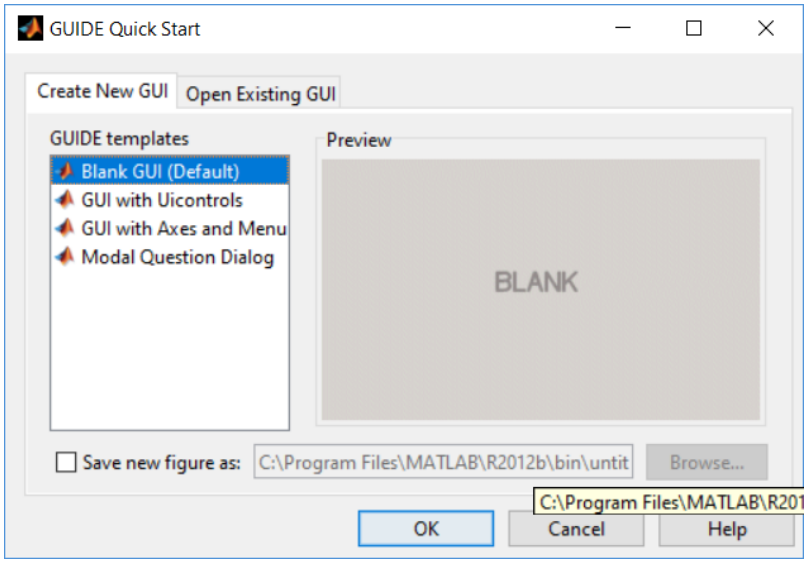


Figure 7. GUIDE Quick Start window

By using the GUI Layout Editor, the user can populate a GUI by clicking and dragging GUI components — such as axes, panels, buttons, text fields, sliders, into the layout area. From the Layout Editor, the size of GUI interface can be changed. GUIDE automatically generates an M-file that controls how the operation of the GUI. This M-file provides code to initialize the GUI and contains a

framework for the GUI click-backs, i.e. the routines that execute when a user interacts with a GUI component. Using the M-file editor, the user can add code to the click-backs to perform required functions [9]. The GUI for the development process of FCP design is shown in **Figure 8**.

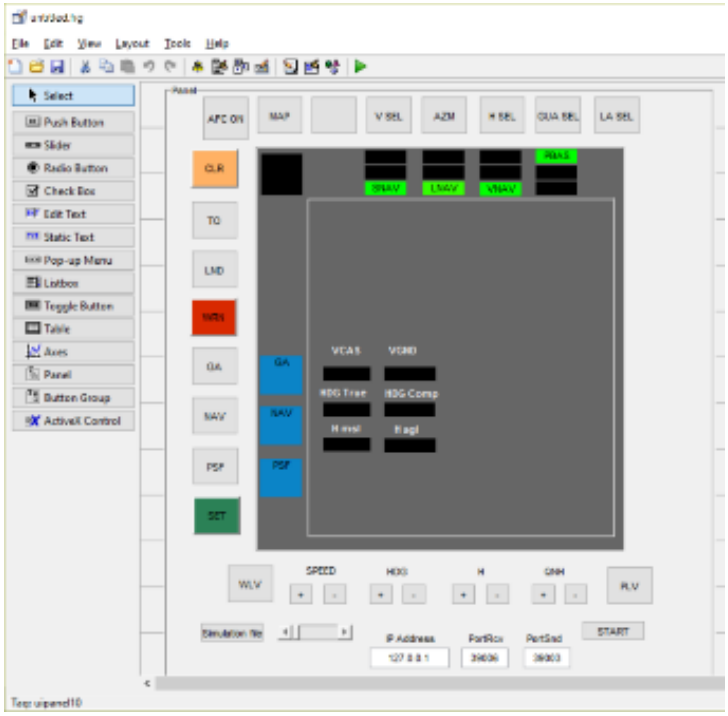


Figure 8. GUI development process for FCP

Dynamic Data Input

To get dynamic data as the input for GUI FCP, data from the flight simulation program were taken. X-plane is the flight simulation program used in this research.

The principle of operation of X-Plane is based on reading the geometric shape of an aircraft to predict its flight dynamics and performance. X-Plane is certified by the U.S. FAA (Federal Aviation Administration) to train pilots because its method ensures a reliable system since it is more detailed, flexible and advanced than flight model based on stability derivatives used by other flight simulators [1].

X-plane standard method for communicating with external processes and machines is User Datagram Protocol (UDP). UDP provides a minimal overhead communication method for passing data between nodes. Unlike Transmission Control Protocol (TCP), UDP is a non-guaranteed protocol and gives no assurances that data packets will arrive in order or at all. UDP is designed to minimize bandwidth usage but presents a possible problem resulting from corrupted data. Although this problem does exist, degradation of control and simulation has been unseen in X-plane experimentations [4].

X-Plane UDP Data

On this research, the communication established between X-Plane and Matlab was performed on a single computer using the IP address "127.0.0.1", which is the network card's internal address. In this type of configuration, it is necessary to use four access points, two for Simulink, two for X-Plane and a single IP address as shown in **Figure 9**.

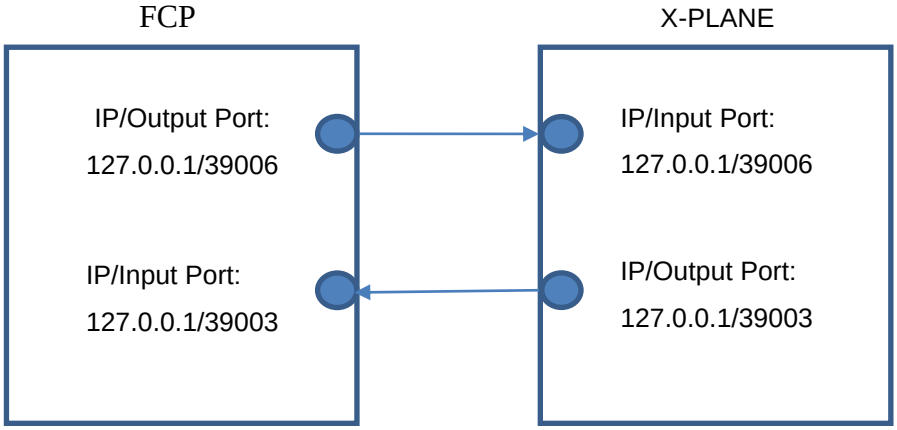


Figure 9. UDP ports connection between FCP and X-Plane

To receive data from UDP user should define IP Port and local port first. Because the program is run on the same computer, remote IP address is set to 127.0.0.1 and local IP Port to 39006. M-file code for receiving UDP data test is shown in **Figure 10**.

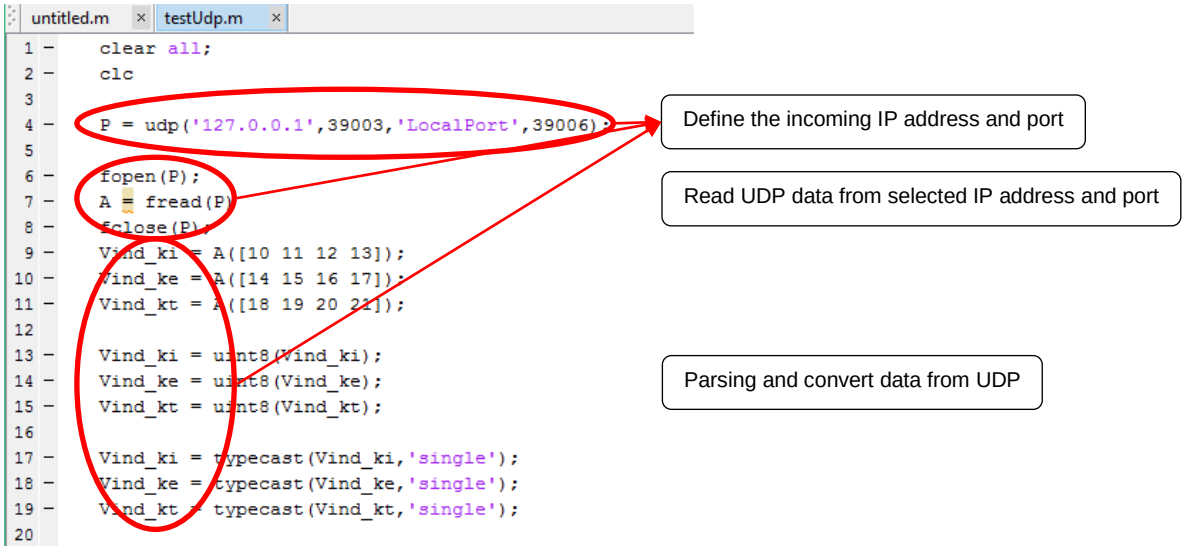


Figure 10. M-file for UDP Receive

To send data from UDP user should also define IP Port and local port. Because the program is run in the same computer remote IP address is set to 127.0.0.1 and local IP Port to 39003 as was set in X-plane. M-file code for sending UDP data test is shown in **Figure 11**.

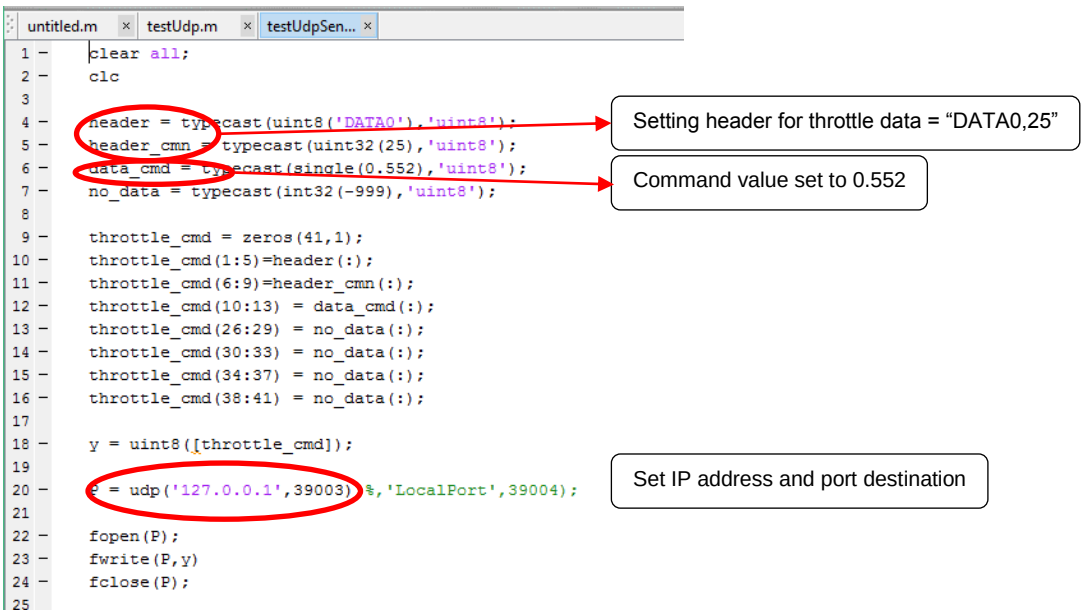


Figure 11. M-file for UDP Send

4. Results and Discussion

To demonstrate the integration according to the scenario, some parameter values need to be set up. Those parameter values are velocity command, altitude command, the angle of sideslip command and heading command. The setting of parameter values is done by using FCP function as shown in **Figure 12**.

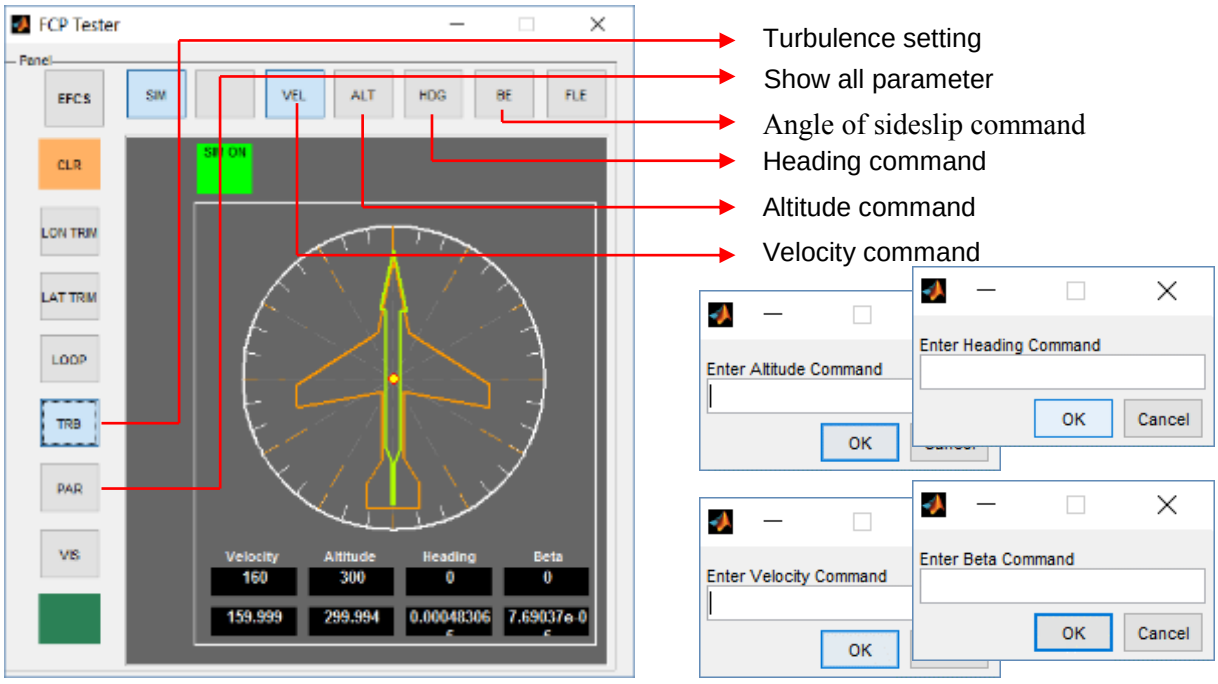


Figure 12. Parameter values setting in FCP

Turbulence setting consists of turbulence intensity selection, current turbulence intensity mode run on Simulink models as well as a button to activated and deactivated turbulence models. The turbulence setting on FCP is shown in **Figure 13**.

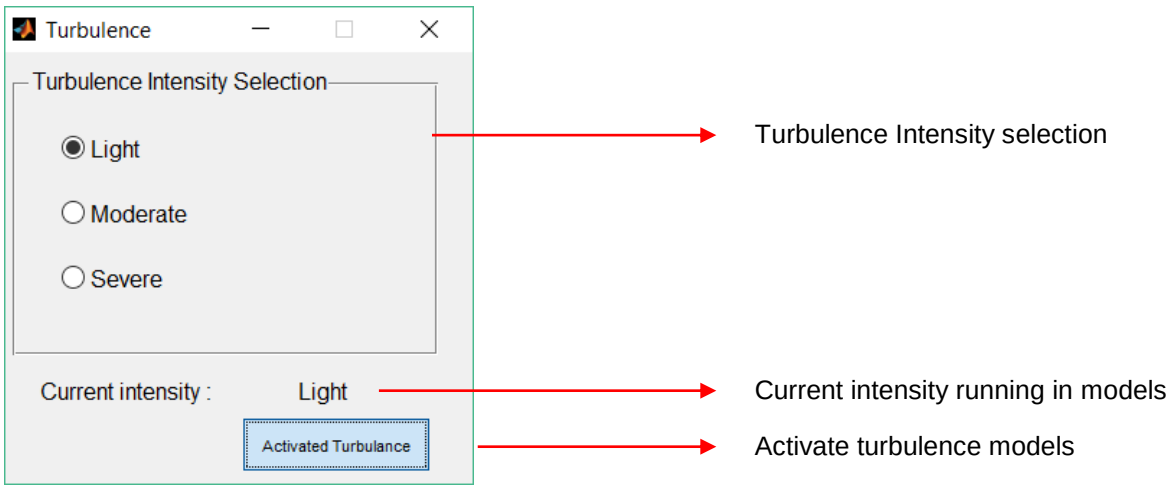


Figure 13. Turbulence models setting in FCP

While the simulation in Simulink is run, flight parameters observe by using PAR function on the FCP as shown in **Figure 14**.

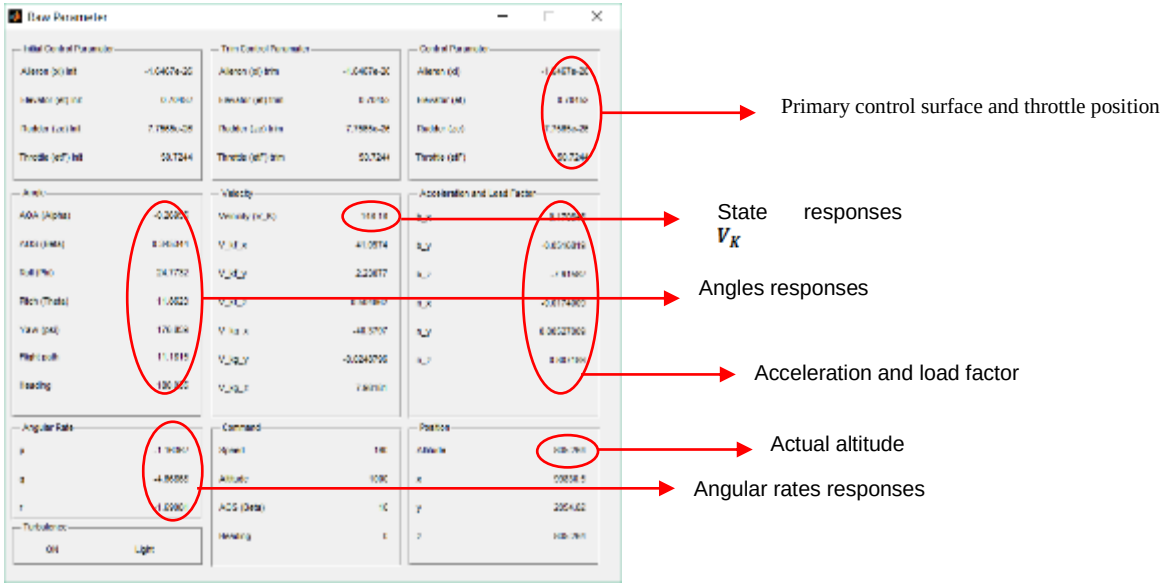


Figure 14. Current Parameters values

Final layout of FCP is shown in **Figure 15** below with description and function of each button and display.

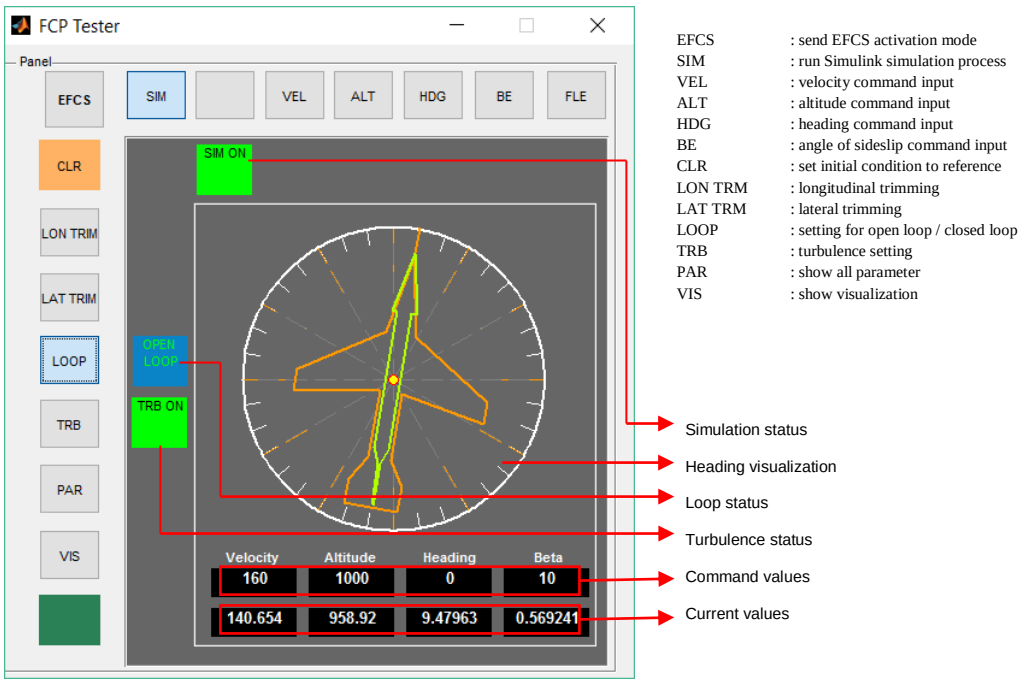


Figure 15. Final FCP layout

5. Conclusions and Future Works

Flight Control Panel Layout using Graphical User Interface MATLAB has been successfully designed. The FCP is used as interface and monitor for command input and aircraft data model including an input for turbulence model setting.

The setting of command can also be done using FCP function with velocity, altitude, the angle of sideslip and heading command values. While simulation in Simulink is run, flight parameter can be observed by using PAR function on the FCP.

Later on, the FCP will be combined with the flight mechanical model and disturbance model. The FCP will be the is used as an interface to command the input to the flight control model and turbulence model and to be an interface for aircraft data model.

Acknowledgements

This work is the final report of minor research of the first author, held in ITB Bandung. The authors would like to thank Drs. Gunawan Setyo Prabowo, MT the director of Aeronautic Technology Center (Pustekbang) LAPAN for providing administrative support, and also Dr. Ony Arifianto, Dr. Rianto Adhi Sasangko and Dr. Yazdi Ibrahim Jenie for supporting this research.

References

[1] A. Bittar, H. Figuereido, P. Guimaraes and A. Mendes, "Guidance Software-in-The-Loop Simulation Using X-Plane and Simulink for UAVs," *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 993-1002, 2014.

[2] D. Lafourcade, "Conduct of Flight Tests and On-Board Computing for the A380," in *Meeting Proceedings RTO-MP-SCI-162*, France, 2005.

[3] M. Lamp, "Konzept und hardware fur das Automatic Flight Control Panel," FMRA, TU Berlin, Berlin, 2008.

[4] R. Garcia and L. Barnes, "Multi-UAV Simulator Utilizing X-Plane," *Intelligent and Robot system*, pp. 393-406, 2010.

[5] R. C.Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing Using Matlab, Gatesmark Publishing, 2009

[6] H. Silver, "Creating Graphical User Interfaces with MATLAB," in *Conference of the American Society of Engineering Education*, 2013.

[7] T. MathWorks, "Creating Graphical User Interfaces," 2014.

[8] R. Verhoeven, F. Bussink, P. X. and D. R., "FLIGHT TESTING TIME AND ENERGY MANAGED OPERATIONS (TEMO)," in *Society of Flight Test Engineers - European Chapter Symposium*, 2016.

[9] T. Hakim, "BEFCS Definition, Requirements and Concept," Berlin, 2016.

[10] B. Jähne, Digital Image Processing, Springer, 2002

[11] I. Pitas, Digital Image Processing Algorithms and Applications, Willey, 2000

[12] R.P.M. Verhoeven and A.J.C. de Reus, Prototyping interactive cockpit applications, NLR, 2005