A MATLAB-BASED LOW-COST AUTOPILOT FOR AUTONOMOUS HELICOPTER DEPLOYMENT

Ismaila B. Tijani, Rini Akmeliawati, Ari Legowo,
Intelligent Mechatronics System Research Units,
Faculty of Engineering, IIUM,Malaysia
Kuala Lumpur, Malaysia

Tun Muzamil L.A
MISA sdn Bhd Shah Alam,
Selangor Darul Ehsan, Malaysia

Abstract— **The challenges associated with the software and hardware integration activities in development of flight autopilot system for autonomous helicopter have called for a change of tactics. The resulting effect is for example, a long time delay in autopilot system design, testing and deployment coupled with the fact that several other autonomous helicopter development tasks depend largely on availability of the autopilot system. Though, the use of off-the-shelf autopilot for a flight control system may ease these challenges, they are generally characterized with limited functionalities, and restrict the user's design authority. As alternative approach, this paper presents the development of a MATLAB-based autopilot system for autonomous helicopter development. This approach provides an integrated design environment for rapid-prototyping of a low-cost autopilot system. The results of real-time application of the autopilot for flight data logging are presented. The performance shows the effectiveness of the developed autopilot system in small scale autonomous helicopter design and implementation. This is hope to reduce the design cycle time involves in the deployment of small scale autonomous helicopter in various civil low-cost, small payload applications.**

Keywords-MATLAB-SIMULINK, dsPIC blcoksets, autopilot, autonomous helicopter, embedded programming

I. INTRODUCTION

The need for simple, cost effective, and reliable autopilot system represents a core requirement in low-cost flight control system for autonomous helicopter deployment for several potential civil applications [1]. Two major approaches are available in the FCS development: (i) development of custom system and (ii) procurement of off-the-shelf autopilot. The design and development of customized autopilot for FCS has been reported alongside research activities in autonomous helicopter or sometimes as a standalone research study. Among the prominent achievements in this regards are the development of custom FCS using PIC104 processor board with QNX Neutrino real-time operating system (RTOS) for the MIT Xcell-60 [2], and for the NUS autonomous helicopter [3]. Carnegie Mellon (CMU) autonomous helicopter (based on Yamaha R-50) was developed with Motorola 68060 processor board and programmed with VxWorks RTOS. The use of MP555 PowerPC board together with a standard PC board (VIA^{TM}) for autonomous helicopter platform development is reported in [4]. Here the MPC555 was used for sensors and actuator interface while the PC board handled the computations. The challenges in this design approach have been anchored on the difficulties involve in software and hardware integrations for the real-time embedded platform. The resulting effect is for example, a long time delay in autopilot system design, testing and deployment coupled with the fact that several other autonomous helicopter development tasks depend largely on availability of the autopilot system.

A possible way out of this is to go for off-the-shelf (commercial) autopilots. These commercial autopilots are available either with proprietary software such as Piccolo [5], Micropilot [6], Kersel [7], or with open-source software among which are ArduPilot [8] and Paparazzi [9]. The use of any of these autopilots for flight control system design (FCS) provides a simplified solution to the customized based FCS design. . A comparative study of these autopilots is reported by [10]. Apart from the software options, all of them share similar characteristics and functionalities. Though they simplified the FCS design, there are many drawbacks in the adoption of this approach. First, they are difficult to modify [11], and hence, user are constraint within available functionalities provided even with open-source based autopilot option. Second, they have limited computational capacities, and presently most of them are equipped with classical PID controller [10]. These have responsible for the limitation of these autopilots to either preliminary autonomous aerial vehicle design cycle or for a micro aerial vehicle applications especially fixed wing based autonomous aerial vehicles.

Alternatively, the use of user-friendly programming environment like National Instrument (NI) Labview [12] and MATLAB-based real-time embedded programming ([13] has been proposed. Konku university reported recently successful completion of rotary wing autopilot using NI single board computer [12] with NI virtual instrument real-time programming tools. The simplicity of the graphical programming tool is major benefit of this design approach. This is obtained at a relative high cost in order of four digits US dollars, also it requires suitable amount of payload to carry the computing target which could weigh up to 4Kg with power supply module.

Exploration of MATLAB Simulink's Real Time Workshop (RTW) for embedded code generation is reported in [13].

Considering the fact that, most of the autonomous helicopter design activities are commonly carried out using MATLAB tools, this embedded design approach promises an integrated design environment for rapid-prototyping of FCS design and deployment. In addition, the general believe that the automatically code generated through this process is neither reliable nor as efficient as handwritten codes has been noted to be an outdated views in embedded programming [11]. It has been demonstrated that the code generated is only 5% larger and 15% slower than the handwritten code, and this approach has been successfully employed in several applications such as industrial PLC programming, navigation and flight control system and hypersonic scrampjets [11]. Moreso, the use of this automatic code generation has benn shown to reduce the design cycle by around 75% [14]. Meanwhile, the generic code generated by MATLAB RTW is currently limited to specific features of a given processor via Simulink Embedded target (ET). There may be need to do few code verification prior to compilation into the target hardware. Recently, alternative ET has been proposed targeting dedicated microcontroller such as Microchip PIC24 upwards [15]. The integration of this third party ET tools provides automatic generation of ready-to-deploy code for microchip PIC24, dsPIC33 and PIC32 series. The application of this integrated design approach is reported in [11].

Hence, as part of an ongoing research on autonomous helicopter, this study presents the development of a MATLAB-based autopilot for flight control system deployment. Though, the overall goal of the FCS is to fully provide essential functionalities for autonomous helicopter deployment as shown in Fig. 1, the present study focuses on the flight data acquisition stage. The rapid-prototyping of low-cost 16-bits digital signal processor (dsPIC33) is facilitated using the integrated design environment involving MATLAB-SIMULINK-RTW, a third party dsPIC-MATLAB blocksets [2] and microchip design tools. The performance of the developed autopilot is evaluated in real-flight test on a small scale helicopter platform.

The rest of the paper is as follows. Section II gives brief overview of the helicopter platform adopted in this study. The hardware selection and description followed by the MATLAB-based embedded programme are presented in section II and IV respectively. The hardware-software integration is presented in section V, while the results of both static and field tests are provided in section VI. The paper is concluded in section VII with future study.

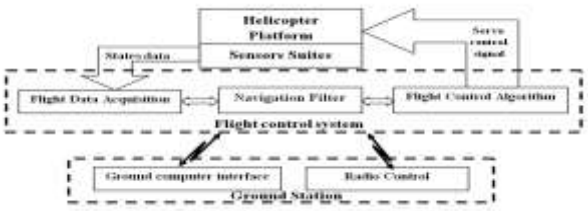

Fig. 1 Blcok diagram of Flight control system components

II. HARDWARE COMPONENTS: SELECTIONS AND DESCRIPTION

The selection of hardware components for the system development is majorly affected by first by the intending application, cost, performance expectation and required development tools (software). The major hardware employed are: RC helicopter platform, inertial measurement units (IMU), global positioning system (GPS), ultrasonic based altimeter, servo switching multiplexer.

A. Helicopter Platform

A 50 class Hirobo RC helicopter (Hirobo SDX50) is selected as the autonomous helicopter platform with a FUTABA ten channels RC transmitter and receiver. Hirobo 50 class SDX is a single rotor hobby helicopter with stabilizer bar capable of both basic and 3D aerobatic flight. The helicopter is shown in Fig. 1, and its general specifications are given in Table 1.

Fig. 2 Hirobo SDX 50

Table 1
General specifications of Hirobo SDX50 Helicopter

Components	Specifications
Full length of fuselage	1220mm
Full width of fuselage	186mm
Height	395mm
Main rotor diameter	1348mm
Tail rotor diameter	258mm
Skid ration	8.7:1:4.71
Dry Weight	3400g
Engine	50 ize,1.9s/17000RPM
payload	2500g

B. Onboard Processor

Selection of this device depends partly on the computing specifications (processing speed, program and data memory), power consumption, weight, and on the required programming development tools. Considering all these factors, a 16-bits dsPIC33fj256g710 (40MHz CPU, 256Kb flash memory, 30Kb SRAM memory) together with a 32-bits, PIC32MMX460F512L (80MHz CPU, 512Kb flash memory, 32Kb SRAM memory) microcontroller boards are selected. The dsPIC is to provide sensors interface and data acquisition while the PIC32 is meant for control algorithm implementation. Both are to be interconnected for full FCS. They are equipped with several peripherals interface for SPI, UART, I^2C, PWM, analog and digital ports communication. Moreover, this device is readily programmable with the

proposed MATLAB-based embedded design tools in this study.

C. Inertial Measurement Unit (IMU)

A MEMs based VN-100 by Vectornav is selected to provide attitude and heading measurement of the RUAV states. The device is small (24x22x2.5mm), light weight (3g), low power consumption (65mA/5volts). Also, it combines 3-axis accelerometer, 3-axis gyros, and 3-axis magnetic sensors as well as a 32-bit processor on a single module with an inbuilt quaternion based drift compensation Kalman filter to produce system states (attitude and heading) at a rate up to 200Hz. The filter can be tuned to meet a specific application and operating condition. Lastly, the device comes with a C-library that facilitates timely interfacing of the device with onboard processor.

D. Global Positioning System (GPS)

A 20 channel EM406A GPS smart antenna receiver is selected to provide position information. The GPS is capable of data output up every 1 second (1Hz). It provides TTL serial interface 4800bps, and operates at 5.0V/44mA. The detailed specification can be obtained in the device datasheet (Globasat, 2007).

E. Wireless Device

Considering the cost, the ease of configuration and the need to operate the system within 1Km distance for research purposes, a very popular member of Zigbee family of wireless devices, 2.4GHz X-bee pro module with wire-antenna is selected to provide bi-directional communication between the ground computer and onboard avionics system up to 1500m outdoor operation. More so, compatibility of several Zigbee devices allows for easy future upgrade to higher distance transmission.

F.Others

Other supporting hardware are ultrasonic sensor by Maxbotic (Maxbotic,) for low altitude measurement, and Cytron servo switching multiplexer (SCM) for manual-auto operation.

III. MATLAB BASED EMBEDDED SOFTWARE DESIGN

The operation of the onboard microcontroller unit (MCU) is guided by the software developed to carry out the necessary tasks for autonomous flight. The major three functional components of the onboard MCU are shown in Fig. 3.

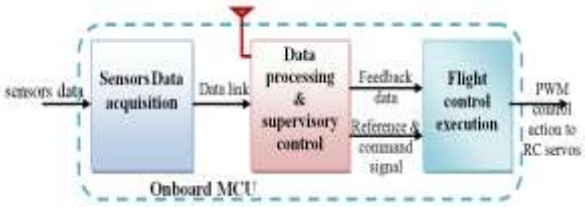

Fig. 3 Functional Block diagram of the MCU

Operationally, the software algorithm is expected to conFig. the MCU, and its peripherals to establish the communication with the sensors through the appropriate interface.. Then, process and transform the acquired data into appropriate format for onward transmission to ground station and to the flight control algorithm as feedback states. Among the data processing tasks to be performed are: (i) filtering, especially the accelerometer and rates gyro data, and (ii) GPS data processing which majorly involves transformation from Latitude, Longitude, and Altitude (LLA) format to Earth Centered, Earth Fixed (ECEF) format and finally to North East Down (NED). The flight algorithm is responsible for the execution of the flight control law designed to autonomously guide the helicopter for the intended mission. The overall mission execution and coordination are handled by the supervisory algorithm unit. This brief operational sequence serves as backbone for the software development task. To this end, the programming task is modularized into: (i) Target MCU configuration, (ii) data acquisition, processing and transmission, (iii) ground station interface and data logging (iv) flight control execution, and (v) mission coordination algorithm. The first three modules are detailed in this study while the last two modules constitute part of the ongoing research activities towards full autonomous helicopter system development.

The integrated embedded design environment comprises of four programming tools: MATLAB toolboxes/SIMULINK, a MATLAB dsPIC blocksets (provides SIMULINK blocks for PIC peripherals interface), C-language, and microchip MPLAB tools. These four tools are integrated within MATLAB environment to achieve a single environment based rapid prototyping of the 16 bits core data acquisition MCU (DAQ-MCU). A single click is required to compile the developed MATLAB SIMULINK based algorithm that is downloadable into the embedded target using microchip 16/32 bits compiler and In-circuit Programming and Debugging (ICD3).

Fig. 4 shows the core embedded SIMULINK blocks employed in the programming. The essential programming sequence is presented here, while the detailed description of these building blocks can be obtained in [15].

MCU configuration: this is achieved with dsPIC master block of Fig. 4a. The target MCU is conFig.d to operate 40MIPS by activating Phase Lock Loop (PLL). The sampling rate is set at 50Hz.

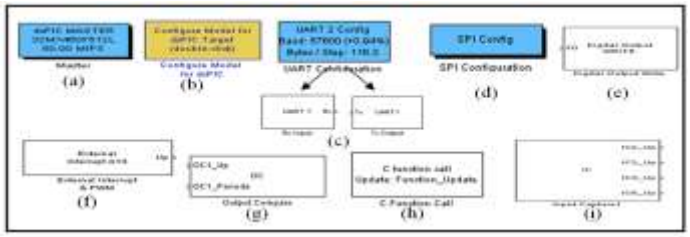

Fig. 4 MATLAB SIMULINK dsPIC/PIC blocksets

Sensor Interfacing: this involves configuration of the MCU peripherals for the sensors' interface, and data transformation

and processing using appropriate custom MATLAB/C custom programmes. The UART ports 1 and 2 are configured using the blocksets of Fig. 4c for GPS (at 4800Kbs) and wireless module (57600Kbs) respectively. The transmission rate of the GPS is specified based on the device default rate of 4800Kbs while that of the wireless device is computed based on total expected data size per sample time. Total of 26 state data comprises of 3 accelerometer, 3 gyro, 3 rates, 3 attitudes/heading, 3 position data, 3 velocity data, 3 GPS raw data, 4 PWM servo control data and 1 altitude are expected to be transmitted. This gives a total of 20800bits data/sample time with each data been sent at 16bits data size. The SPI port is configured to communicate with VN100 IMU sensor using the block of Fig. 4d with 10MHz SPI clock frequency. A custom C-codes is written to communicate with the device library. The codes are integrated into the SIMULINK environment using the blocksets of Fig. 4h. The ultrasonic sensor data is captured using one of the external interrupt ports of the MCU, while the four PWM control signals (cyclic, collective and pedal) are captured with four output compare ports (OC1-OC4). Necessary data transformation and conversion are done using a custom m-files function implemented using MATLAB embedded target SIMULINK block.

GPS position/velocity filter: the GPS position information in Latitude Longitude and Altitude (LLA) in Geodetic coordinates is converted to position in north east down (NED) suitable for navigation system. The transformation involves three levels: LLA to Earth Center Earth Fixed (ECEF) coordinate; ECEF to NED. The conversion from LLA, $[\varphi, \lambda, h]$ to ECEF $[X, Y, Z]$ is achieved using the relationship [16]:

$$X = (N + h)\cos\varphi\cos\lambda \tag{1}$$

$$Y = (N + h)\cos\varphi\sin\lambda \tag{2}$$

$$Z = \left(\frac{b^2}{a^2}N + h\right)\sin\varphi \tag{3}$$

where N is the Radius of curvature (meters) defined as:

$$N = \frac{a}{\sqrt{1 - e^2\sin^2\varphi}} \tag{4}$$

and parameters a, b and e are the WGS84 parameters given as:

$$a = 6378137$$
$$b = a(1 - f)$$
$$f = \frac{1}{298.257223563} \tag{5}$$

The transformation from the ECEF to NEU is accomplished with respect to a local reference point, which is taken to be the location of the Ground Station (GS) system. NEU is formed from a plane tangent to the earth's surface and fixed to the CG of the helicopter, hence it is termed a local tangent or local geodetic plane. By convention, the north is labeled x-axis, the east as y-axis and up as the z-axis.

Defined $[X_p\ Y_p\ Z_p]$ as location of the helicopter system with center coincided approximately to the CG of the system, and $[X_r\ Y_r\ Z_r]$ as the location of the GS, then the vector point $[x\ y\ z]$ of the helicopter from the GS location is computed using:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin\varphi_r\cos\lambda_r & \sin\varphi_r\sin\lambda_r & \cos\varphi_r \\ -\sin\lambda_r & \cos\lambda_r & 0 \\ \cos\varphi_r\cos\lambda_r & \cos\varphi_r\sin\lambda_r & -\sin\varphi_r \end{bmatrix} \begin{bmatrix} X_p - X_r \\ Y_p - Y_r \\ Z_p - Z_r \end{bmatrix} \tag{6}$$

where φ_r and λ_r are latitude and longitude angles respectively of the GS location.

Considering the off-set of the GPS device from the location of the IMU which is approximately at the CG of the RUAV, the final position estimation is computed by:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{h} \end{bmatrix} + \begin{bmatrix} x\_offset \\ y\_offset \\ z\_offset \end{bmatrix} \tag{7}$$

where the offset vector

$$[x\_offset \quad y\_offset \quad z\_offset] = [50 \quad 0 \quad 10]$$

Also, the velocity vector in NED coordinates $[v_{north} \quad v_{east}]$, is computed from the GPS reported heading (degree) information otherwise refer to as course of the ground (COG), and speed over the ground (m/s), (SOG) using the expression:

$$v_{north} = SOG * cos(COG) \tag{8}$$

$$v_{east} = SOG * sin(COG) \tag{9}$$

The final output velocity vector $[v_x \quad v_y]$ is computed using the velocity rate filter that updates the velocity information using:

$$v_{i+1} = v_i + (0.5 * \Delta v^{new} + 0.5 * \Delta v^{old}) * T_s \tag{10}$$

where T_s is the application sampling time and, velocity rate is Δv is given by

$$\Delta v = \frac{v_i - v_{i-1}}{T_s} \tag{11}$$

Heading computation: the heading angle as a measured of angle measured clockwise from a true North (earth's polar axis) direction is computed from the measured magnetometer readings by the IMU sensor (VN100). The heading is calculated using the relationship:

$$Heading = tan^{-1}(Y'_m / X'_m) \tag{12}$$

where X'_m and Y'_m are x-axis an y-axis magnetic reading respectively.

To account for the orientation of the system, X'_m and Y'_m are computed from actual three axis magnetic readings (X_m, Y_m, Z_m) using the mathematical expression

$$X'_m = X_m \cos\phi + Y_m \sin\theta \sin\phi - Z_m \cos\theta \sin\phi \quad (13)$$

$$Y'_m = Y_m \cos\theta + Z_m \sin\theta \quad (14)$$

where ϕ and θ are the roll and pitch angles describing the orientation of the system. Note that where is system is horizontally flat, $X'_m = X_m$ and $Y'_m = Y_m$.

The sign of heading computation is determined using the logical expression as follows [17]:

$$Heading = \begin{cases} 180 - \tan^{-1}(Y'_m / X'_m) & if\ (X'_m < 0 \\ -\tan^{-1}(Y'_m / X'_m) & if\ (X'_m > 0, Y'_m < 0) \\ 360 - \tan^{-1}(Y'_m / X'_m) & if\ (X'_m > 0, Y'_m > 0) \\ 90 & if\ (X'_m = 0, Y'_m < 0) \\ 270 & if\ (X'_m = 0, Y'_m > 0) \end{cases} \quad (15)$$

The effect of magnetic declination angle was considered by subtracting the country declination angle. The angle of declination of the country magnetic field was obtained from the National Geophysical Center (NOAA) [NOAA] as -0.1075 degree. This was accounted for in the final heading computation using:

Heading_final = heading_computed - (-0.1075) (16)

A MATLAB m-file functions and relevant SIMULINK blocks were used to compute both the position/velocity filters and heading computation using the expression (1-15). This was then implemented using MATLAB embedded function Simulink block.

IV. SYSTEM INTEGRATION

The overall system integration components is shown in Fig. 5 and comprises mainly of hardware set-up, embedded MATLAB based programming, and integration of hardware and software for deployment on the autonomous helicopter platform.

The overview of the cost, weight and current rating (where applicable) for the hardware components are given in Table 2. The cost is without the shipping cost which is relative to location. The overall cost of the system is approximately estimated at 1500 US dollars, which is quite within low cost system. The total weight estimated at 1010g shows that only around 40% total payload has been utilized by the current set-up leaving the rest 60% for future upgrade. However, during the flight test, dummy loads were added to balance the system, and also as provision for other items (camera, control MCU, extra battery) to be added in the nearest future.

I. PERFORMANCE EVALUATION AND RESULTS

Series of tests were carried out to evaluate the performance of the developed autopilot on the autonomous helicopter system.

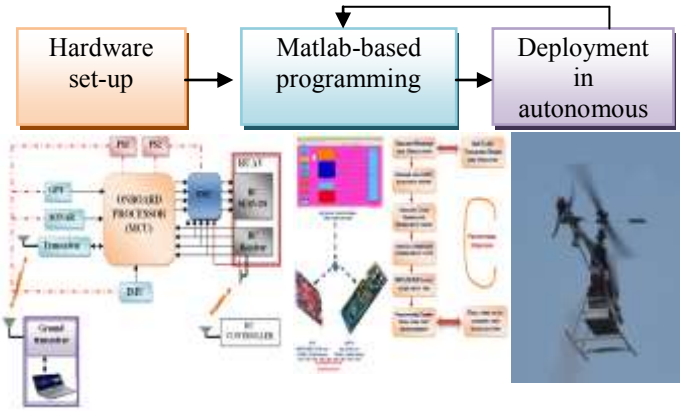

Fig. 5 Complete system integration stages

Table 2: Estimated cost, weight and current rating

Devices	weight (g)	current (mA)	cost (US dollars)
MCU	80	50	250
IMU	40	65	800
GPS	15	44	60
Sonar	50	3.4	89
Xbee Pro	15	215	145 (2pcs)
Fuel	300	-	-
Skip and boxes	400	-	-
Battery	150	(2700mAH)	64.5
Total	1050	377.4	1498.5

These include static test (on ground test), and flight test. The attitudes, heading, acceleration and rates reading of the IMU sensor are logged for up to 4minutes (200s) at rate of 20ms. This was repeated five times, then the mean and standard deviation of the static readings are computed. The results obtained were within the accuracy of the sensor performance specification. By starting from a direction approximately closed to true North using a compass device, the helicopter system was rotated clock-wisely and back to the original point. The heading information computed by the onboard system is shown in Fig. 6.

The reliability and accuracy of the GPS module position information logging were evaluated. This was achieved by mounting the system on a ground vehicle, and drove around the university campus while the position information in LLA format is been logged into the accomplished ground computer station. The reported position information was processed and compared with Google map using a free software package (GPS visualizer) available online [18]. Fig. 7 shows the comparative view of the reported GPS information and Google map. As shown in Fig. 7, the GPS accurately tracked the Google Earth way-points

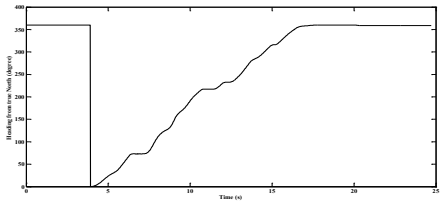

Fig. 6 Sample test result of Heading from true North

Fig. 7 Comparison of GPS and Google map position information

Field test: The system is commanded manually to perform various flight operations such as take-off and landing, hovering, and cruise flight at varying speed. For a complete flight operation test, the pilot takes-off the helicopter off the ground effect, hovering, then climb vertically, perform hovering and excite the attitudes and heading with sweep input signal, then navigate around the flight field. The objective of this test is to evaluate the ability and reliability of the developed system and its onboard electronics to perform the intended operation and withstanding severe operation condition. Fig. 8-Fig.11 show samples of control inputs and state responses of the system for various flight modes. These flight data constituted the information required for system modeling and analysis in other various autonomous helicopter system developments. The sample of position information in NED coordinate is shown in Fig. 12 while the Google map view of a flight pattern is shown in Fig. 13.

The results of both ground and field tests show the effectiveness of the developed autopilot to provide necessary flight data acquisition and state estimation for the helicopter system analysis.

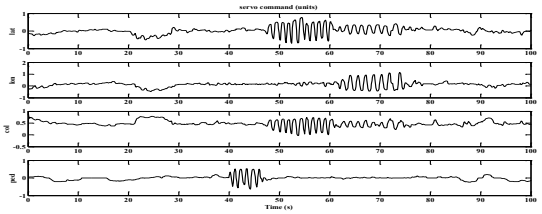

Fig. 8 servo command signal

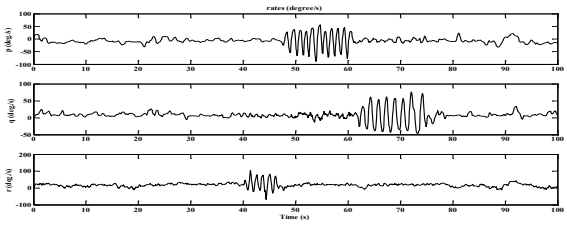

Fig. 9 measured angular rates responses

VI CONCLUSION

The development of a MATLAB-based autopilot system for autonomous helicopter system has been reported in this paper. The deployment is presentedbased on an integrated MATLAB environment (MATLAB-IDE) have been presented in this chapter. Performance evaluation results for both static and in-flight tests of the developed system show the success and ability of the autopilot system to provide effective flight data estimation and acquisition required in the autonomous helicopter system design and deployment. This proposed integrated design enviroment is expected to simplified the future upgrade and reconfiguration of system functionalities. Future study involves upgrade of the flight control system for flight control algorithm implementation.

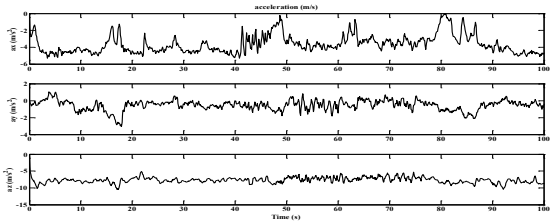

Fig. 10 measured acceleration

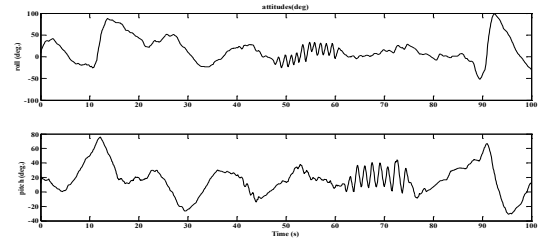

Fig. 11 measured attitudes

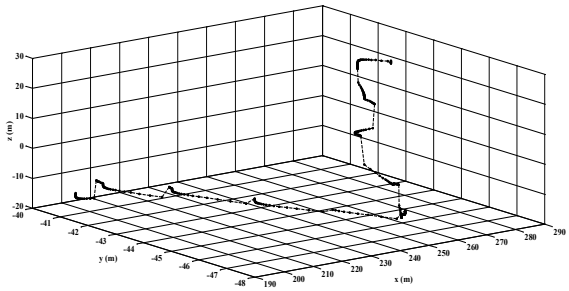

Fig. 12 Sample position estimation

Fig. 13 Sample flight test route within the campus flying field

ACKNOWLEDGMENT

This research was supported by the RMGS (Research Matching Grant Scheme), Research Management Center, IIUM, Malaysia.RMGS-09-02

REFERENCES

[1] Timothy H. Cox, Christopher J. Nagy,Mark A. Skoog, and Ivan A. Somers, " Civil UAV Capability Assessment", prepared for Lawrence Camacho UAV Vehicle Sector Manager, Vehicle Systems Program, NASA Aeronautics Research Mission Directorate 2004..

[2] Gavrilets V., A. Shterenberg, M. A. Dahleh, E. Feron, 'Avionics System For A Small Unmanned Helicopter Performing Agressive Maneuvers", Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th

[3] Guowei Cai, Lin Feng, Ben M. Chen, Tong H. Lee, 'Systematic design methodology and construction of UAV helicopters', Mechatronics 18 (2008) 545–558, Elsevier, 2008

[4] David Vissiere, Pierre-Jean B., Alain Pierre and Nicolas Petit, "Experimental autonomous flight of a small-scaled helicopter using accurate dynamics model and low-cost sensors", in proceeding of the 17th World congress, IFAC, Seoul, Korea, July 6-11,2008

[5] B. Vaglienti, R. Hoag, and M. Niculescu, "Piccolo System User's Guide". Cloud Cap Technology, Hood River Oregon, 2005.

[6] MicroPilot, "Micropilot: World leader in miniature uav autopilots." http://www.micropilot.com, 2009.

[7] Procerus Technologies, "Kestrel autopilot." http://www.procerusuav.com/productsKestrelAutopilot.php, 2009.

[8] Chris Anderson, "ArduPilot", http://diydrones.ning.com/profiles/blog/show?id=705844%3ABlogPost%3A35640, 2008

[9] P. Brisset, A. Drouin, M. Gorraz, P. Huard, and J. Tyler, "The Paparazzi Solution," 2nd US-European Competition and Workshop on Micro Air Vehicles, November 2006.

[10] HaiYang Chao, YongCan Cao, and YangQuan Chen. "Autopilots for Small Unmanned Aerial Vehicles: A Survey", International Journal of Control, Automation, and Systems (2010) 8(1):36-44,Springer.

[11] M. I. Lizarraga, "Autonomous landing system for a UAV," Master's thesis, Naval Postgraduate School, Monterey, CA, USA., March 2004.

[12] Byoung-Jin_Lee, Seung-Jun_Lee and Prof. Sangkyung Sung Rotary UAV Autopilot - Navigation and Autopilot System Development of RUAV based on Virtual Instrumentation Platform (Onboard Avionics with Labview + NI Devices)" available online at https://decibel.ni.com/content/docs/DOC-16504 accessed date June 27, 2012

[13] Daniel Ernst & Kimon Valavanis & Richard Garcia & Jeff Craighead, "Unmanned Vehicle Controller Design, Evaluation and Implementation: From MATLAB to Printed Circuit Board", J Intell Robot Syst (2007) 49:85–108

[14] The Mathworks, Inc., "NASA's X-43A Scramjet Achieves Record-Breaking Mach 10 Speed Using MathWorks Tools for Model-Based Design." User Story, 2005.

[15] L. Kerhuel, "Matlab-simulink device driver blockset for PIC/dsPIC microcontrollers." http://www.kerhuel.eu/wiki/index.php5.

[16] Micro-blox, "Dum Transformations of GPS positions", application note, www.u-blox.ch

[17] Honeywell,"Compass heading using magnetometers" http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/Magnetic__Literature_Application_notes-documents/AN203_Compass_Heading_Using_Magnetometers.pdf

[18] Adam Schneider, "GPS visualizer", http://www.gpsvisualizer.com/

[19] [9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface,"