A Control System for an Unmanned Micro Aerial Vehicle

L. Huang[1] and C. Murton[2]

[1]School of Engineering, Auckland University of Technology, New Zealand
loulin.huang@aut.ac.nz

[2] Opus International Consultants Ltd, New Zealand
Callum.Murton@opus.co.nz

Abstract – This paper presents the development of a PC-based control system for an unmanned micro aerial vehicle (UAV) that has been converted from a helicopter model. The hand-controller of the helicopter model is modified to be a wireless link between a ground control computer (GCC) and the UAV, which carries sensors including an inertia measurement unit (IMU) and the actuators for the main rotor and pitch angle controls. A programming environment, written with the C# programming language, is set up for the implementation of control algorithms for the UAV. Manual and automatic modes of controlling the UAV are achieved. Implementation issues, such as the reliability of the IMU data for closed-loop control, are also discussed.

Keywords – Control; Micro Aerial Vehicle.

1 INTRODUCTION

Research into unmanned aerial vehicles (UAVs) has been active in recent decades due to its huge potential in applications such as search and rescue, surveillance, and precision agriculture where ground based vehicles are unsuitable. Many research projects are aimed at producing fully autonomous UAVs where the communication path between the GCC and the UAV is mainly used for sending motion commands, and the real-time signal processing and control are executed by the flight control units within the vehicle. This ensures that the UAV has a fast response and is able to quickly react to disturbances [1][2]. Manual operation with a hand-held controller is also kept in the system in the event of an emergency. The limited computing power of onboard devices, however, can make it difficult to implement complex and demanding control tasks.

In [3], a UAV is partially controlled by the GCC. The vehicle's state is obtained via a camera tracking system and fed back to the computer. A position detection server is programmed to recognise a small square marker on the vehicle. By comparing the apparent size of the marker with stored knowledge of its true size, the server is able to compute the distance of the UAV from the camera, and therefore its position. The servers are connected via a gigabit Ethernet which significantly reduces the time taken for data transmission.

Another type of UAV, presented in [4], is manually operated with sensory input provided to the human operator via a computer monitor. Cameras mounted on the UAV provide the outlook, and onboard sensors provide the avionics data. The human operator interacts with the GCC in much the same way as they would with a flight simulator. The GCC reads the manual controls from the hand-controller, applies any control algorithms, transmits the commands to the UAV, and receives and displays the video and avionics data from the UAV. The onboard electronics are mainly used for wireless communication, brushless motor control, servo control, and sensor reading. This system is also used as a platform to compare the performances of manual and feedback augmented control where the inputs from both the human operator and the data from the onboard IMU are used to maintain the stability of the UAV during navigation. In [5], a UAV is converted from an ESky Big Lama helicopter model for competition in the 2009 International Aerial Robotics Competition. While the flight control algorithms are executed with the onboard micro-processors, XBee wireless modules are used to send the navigation data to the GCC.

A common feature of the various UAVs is that both manual and automatic modes exist within the system; each of which has a specific role to play. In some systems the automatic mode is dominant, whereas in others the two modes carry the same level of importance. Though there are a lot of publications on UAVs, it is difficult to find any that provide detail regarding how the UAV has been designed and constructed.

This paper presents the development of a dual-mode UAV system for testing UAV control algorithms. The steps taken to convert the helicopter model to a dual mode (manual and automatic) UAV are explained in terms of hardware and software design. The initial testing results are also presented.

The paper is organized as follows. Section 2 describes the system structure, Section 3 describes the computer program development, Section 4 describes the testing results, and a conclusion is provided in Section 5.

2 SYSTEM CONFIGURATION

Fig. 1 shows the structure of the UAV that has been converted from an ESky Lama V4 helicopter model. The system consists of two main parts – the ground station and the helicopter. The helicopter has been fitted with an inertia measurement unit (IMU), XBee Pro wireless transceiver, and related circuitry such as power supplies. As a result, the payload of the helicopter has increased by

approximately 60 grams. The main part of the ground control station is a ground control computer (GCC) which processes data sent from the sensors, and generates motion commands.

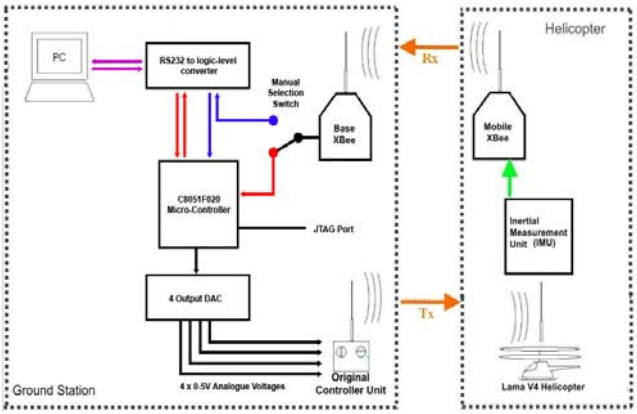

Fig. 1 Structure of the UAV system

The GCC communicates with the helicopter through a communication channel consisting of an XBee Pro wireless module, a C8051 micro-controller, the helicopter's hand-controller, and interfacing circuitry. Data from the inertia measurement unit (IMU), via the XBee Pro modules, can be sent to the universal asynchronous receiver transmitter (UART) port of the micro-controller, or to the PC via an asynchronous serial port, allowing control algorithms to be run on the PC, the micro-controller, or both. A four-channel, 10-bit DAC chip is used to convert digital outputs from the micro-controller into analogue signals compatible with those produced by the operation of the control sticks of the hand-controller. This utilises the existing receiver on the helicopter and allows it to be switched to manual control in the event of an emergency. It also allows for the helicopter to be easily replaced with a similar four-channel model by exchanging the transmitter crystals.

The IMU provides up to nine measurements consisting of three linear accelerations, three magnetic directions, and three gyroscopic angular velocities, that reflect the states of the helicopter in motion. The format of the data frame can be either purely ASCII format, or mixed ASCII and binary formats. The length of the data frame has been reduced by excluding the magnetic directions, which have been found to be affected by the electromagnetic fields of the motors. The data frame consists of data for the six measurements, a count number, and characters signifying its beginning and end. The baud rate of the IMU is 115200 bps, resulting in a transmission time for one data frame of approximately 3.4 ms. The communication path consists of two asynchronous serial connections between the IMU and the PC. It also includes the wireless connection between the XBee Pro modules which have a data rate of 250 kbps, corresponding to a transmission time of approximately 1.6 ms. Therefore the communication delay from the IMU to the PC is approximately 8.4 ms.

The complete communication path begins at the IMU, which digitises the acceleration and velocity measurements. The IMU transmits the measurements in an ASCII data frame to the XBee Pro on the avionics circuit board, which then transmits the data frame to the XBee Pro at the ground control station (base XBee Pro). The base XBee Pro can then either send the data to the micro-controller or to the GCC via an RS232 serial port. The GCC program uses the received data to display the UAV's states in the graphic user interface, and produces control commands which are sent to the micro-controller via the RS232 serial port. The micro-controller converts the received commands to analogue signals via DACs which are then in-putted to the helicopter's hand-controller. The hand-controller's onboard micro-controller then digitises the analogue voltages and transmits the corresponding commands to the helicopter through the existing transmitter and receiver.

A manual switch is installed to allow the user to select either the hand-controller's control sticks or the DACs as the source of the analogue signals to the hand-controller. A JTAG port has been provided in the system to allow for reprogramming the micro-controller. While two 12-bit DAC modules are available in the micro-controller, they have not been used as there is not a sufficient number to handle all of the control channels, and their maximum output voltage is 3.3V rather than the required 5V.

3 SOFTWARE DEVELOPMENT

The software development includes the programming of the micro-controller and the GCC.

The micro-controller is the core of the communication path for the UAV system. It is capable of receiving data from the IMU and the GCC, and sending commands to the hand-controller. In the final communication path, the IMU data is sent directly from the base XBee Pro to the GCC, and the micro-controller provides the interface between the GCC and the hand-controller. Fig. 2 shows the modules of the micro-controller program designed to complete this task. The data frames from the GCC consist of a "T", "P", "R", or "Y", representing the control channels for *throttle*, *pitch*, *roll* and *yaw* motions respectively, followed by the data for the control commands, and finally a "Z" character, signifying the end of the data frame. The entire data frame is encoded using ASCII characters.

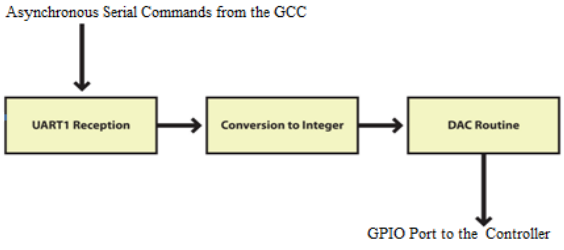

Fig. 2 Modules of the microcontroller program

The control commands are string-based numbers between 0 and 1023. The numbers are converted into integers, which are then processed by the DAC routine. As the DAC chip provides only eight data inputs, the DAC routine separates each integer into high and low bytes that are sent to the DAC chip individually. This process is executed in sequence for each of the four channels.

The core of the software system for the UAV is the GCC program, which oversees all operations of the UAV. The GCC program is created within the Microsoft Visual C# environment which provides a quick and powerful means of designing and implementing graphic user interface based programs for the Microsoft Windows operating system. Though the lack of determinism and schedulability is an obvious shortcoming of C# in the design and implementation of so called "hard" real-time control systems, it should be suitable for "firm" and "soft" real-time applications [6]. Since the sampling frequency of the IMU is controlled by hardware, and due to the fact that each data frame contains a count number, the correct current time can be estimated. This takes care of the schedulability issue provided that the GCC program is capable of processing the control routine within the IMU sampling period.

To perform the tasks required by the UAV system, the structure of the GCC program is designed as shown in Fig 3.

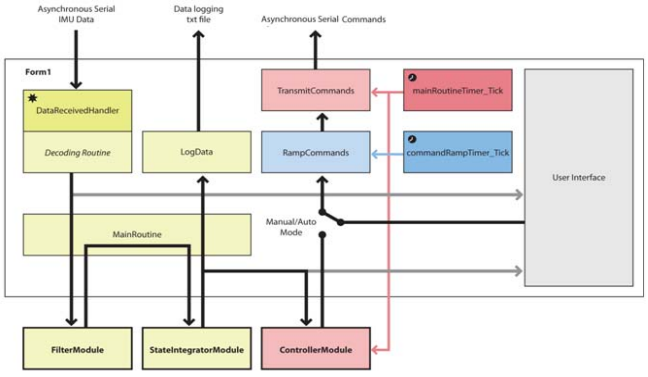

Fig. 3 GCC program structure

The program consists of four classes named *Form1*, *FilterModule, StateIntegratorModule* and *ControllerModule*. The *Form1* class is the main class of the program including all of the communication, command execution, sequencing and timing routines, as well as the graphic user interface (GUI) and its associated routines. The other classes are external classes, and as the names suggest, are designed as modules to perform the tasks of signal filtering, state estimation and the generation of control commands. Each of them provides a specified means of data input and output in the form of arrays, and allow for the modification or replacement of the routines.

The *FilterModule* class implements the digital filtering of the raw data from the IMU. The filter routine runs a 5-pole Butterworth filter with the cut-off frequency set to 20% of the sampling frequency. A Butterworth filter is chosen due to the fact that it is an infinite impulse response (IIR) filter which uses both previous input and output values to compute the impulse response. Furthermore, a Butterworth filter has a maximally flat pass band which reduces the inaccuracy of the positions obtained by integrating the filtered data in the *StateIntegratorModule* class.

The *StateIntegratorModule* class processes the filtered data by numerically integrating it over the sampling period, thereby producing the state vector of the UAV. This consists of the linear and angular velocities, and linear and angular positions. The sampling period for the integration is defined within the GCC program and must match the actual sampling period set within the IMU. Failure to do so would result in a scaling error with respect to the integrated velocities and positions.

The *ControllerModule* class takes the state array as the input, process it, and provides the control commands. This is the main space for control algorithm development. For testing purposes, an open-loop control routine is programmed that consists of a sequence of time-based control commands. The pitch, roll, and yaw are set for vertical flight and the throttle is set to climb for a certain period of time, hover momentarily, and then descend. Closed-loop control algorithms can also be programmed within the class, though the success of these algorithms will be dependent on the successful estimation of the states within the *StateIntegratorModule* class. When the GCC program is in the manual command mode, the control command values are taken directly from the manual user controls provided within the graphic user interface (GUI).

In the *RampCommands* routine, the command ramp timer is introduced to smoothly increase or decrease the magnitude of the control signals. This protects the UAV from a severe increase in motor speed, which demands too much torque from the motors and may cause damage to the UAV. Following the *RampCommands* routine is the *TransmitCommands* routine where the four control channel commands are transmitted to the micro-controller via the RS232 serial port. The frequency of command transmissions is controlled by the main routine timer.

The various routines within the program are triggered by the data received event, and the ticks of the main routine timer and the command ramp timer. The data received event is activated by the serial data entering the asynchronous serial port buffer; which consists of the data frames sent by the IMU. Once the data is considered valid, the main routines including filtering, state estimation and data logging are executed in sequence. The main routine timer is used to execute the *ControllerModule* and the *TransmitCommands* routines.

Fig. 4 shows a screenshot of the graphic user interface (GUI) of the GCC program. The top right panel of the GUI contains the indicators for the raw IMU data. Each of the gauge indicators represents a sliding scale of 0 at the

bottom and 1023 at the top. The heights of the red bars are directly proportional to the values of the IMU data channels for which they represent.

The bottom right panel contains the indicators for the integrated velocities and positions of the UAV. The velocities and positions for all six degrees of freedom are represented. The bottom left panel includes the manual controls for the UAV, which correspond to the control levers of the hand-controller, and work in exactly the same way. The top left panel contains the "Program Status" indicator, mode selection controls, data logging controls, and execution timer indicator.

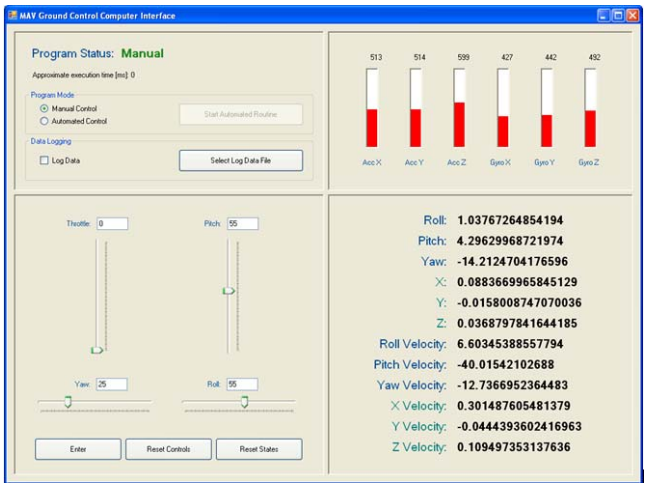

Fig. 4 Graphic user interface of the GCC program

4 RESULTS AND DISCUSSIONS

The complete UAV system is shown in Fig. 5. Test flights of the UAV are conducted to check if all the modules of the system work well. One scene of the test flights is shown in Fig. 6.

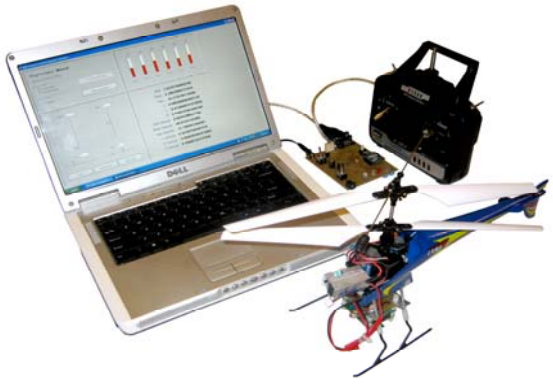

Fig. 5 The UAV system

In one flight test, the UAV is controlled to perform a vertical takeoff and landing. The trajectories of the acceleration, velocity and position of the UAV in the Z (altitude) direction are recorded and shown from Fig. 7 to Fig 9. The system as a whole has proven successful.

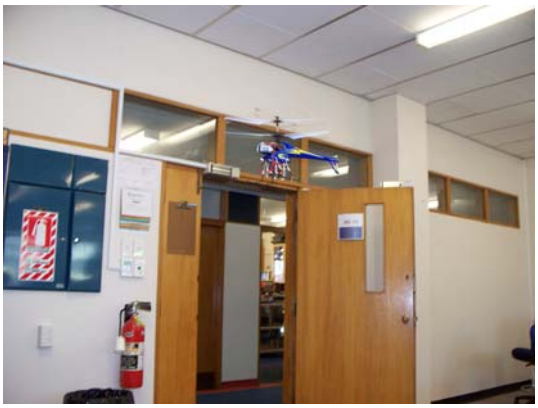

Fig. 6 The UAV under control

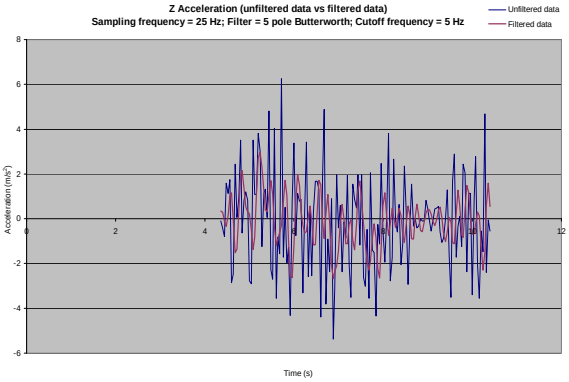

Fig. 7 Acceleration in Z direction

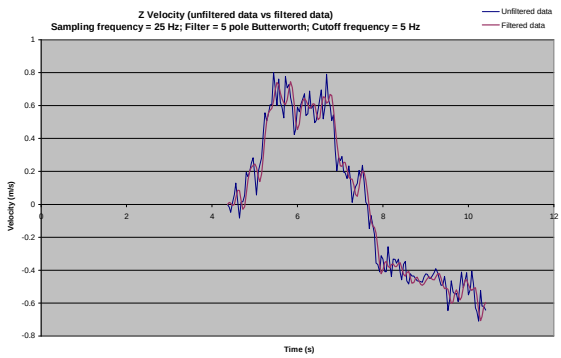

Fig. 8 : Velocity in Z direction

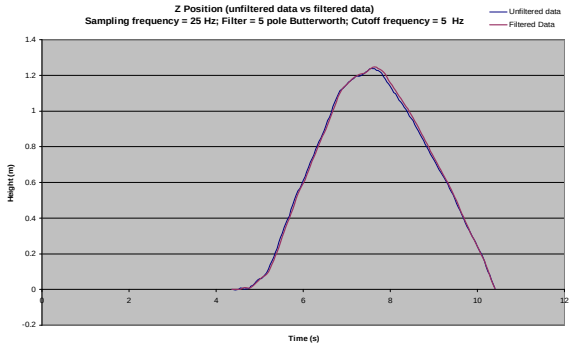

Figure 9: Position in Z direction

In the test flights, it is noted that the zero point of the IMU drifts irregularly, which compromises the reliability of the data received. Possible causes of this irregularity include the input voltage fluctuating during different stages of operation, and signal aliases occurring as a result of the sampling frequency of the IMU. Future work is needed to accurately identify and eliminate the cause of the drift. Closed-loop control can then be implemented with the support of the hardware and software environments of the UAV system.

5 CONCLUSION

This paper presents the hardware and software development for a PC-based control system for an unmanned micro air vehicle converted from a helicopter model. Manual and automatic modes of operation of the system have been tested and proved to be a success. A comprehensive open platform has been provided for future work on issues such as sensing and real time control, to ultimately achieve autonomous closed-loop control of the UAV.

ACKNOWLEDGEMENT

Most of the work reported in this paper was completed when the authors were at the School of Engineering and Advanced Technology, Massey University. We would like to thank Mr Malcolm Watts, Mr Mike Turner, Mr Poh Ng and Mr Neiko Altenburg for their help.

REFERENCES

[1] R. Jinjun, Jun, L., X. Shaorong, and G. Zhenbang, "Subminiature unmanned surveillance aircraft and its ground control station for security", *Proceedings of the 2005 IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 116-119, Kobe, Japan, 2005.

[2] Y. Ge, and M. Zhu, "A control system of the miniature helicopter with stereo-vision and time delay predictor", *Proceedings of the 2007 International Conference on Information Acquisition*, pp. 608-613, Jeju City, Korea, 2007.

[3] M. Hirata, O. Miyazawa, K. Nonami, G. Song and W. Wang, " Autonomous control for micro-flying robot and small wireless helicopter X.R.B", *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2906–2911, Beijing, China, 2006.

[4] J. Andersh, B. Mettler, and N.Papanikolopoulos, "Experimental investigation of teleoperation performance for miniature rotorcraft", *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 6005-6010, Shanghai, China, 2009.

[5] G. Chowdhary, H. C. Christmann, E. N., Johnson, M. S. Kimbrell, E. Salaün, and D. M. Sobers, *Georgia Tech Aerial Robotics Team, 2009 International Aerial Robotics Competition Entry.* Georgia Institute of Technology.

[6] P. A. Laplante, and M. H. Lutz, "C# and the .NET framework: ready for real time?", Software, *IEEE, 20*(1),70-80.