1. Project Overview[1]

Groups of 3 students are to design, implement, test and analyse a common Tiva MCU module for control on firstly a Heli-emulator, and secondly on a Heli-Rig. This project assignment extends your ENCE361 HeliRig project requirement in several ways. First, FreeRTOS must be used, as distinct to a simple round-robin or time-slice scheduler. Second, a new HeliRig mode of operation will be required for demonstration. Third, a discussion (in your report) will be required to extend the functionality of our current HeliRigs, and this could include for example, sound generation or sound reproduction.

During Stage 1 of your project (Weeks 1-3) your controller module will use FreeRTOS on a Tiva MCU board (re-used from ENCE361) on a STM32F072 microcontroller module, pre-programmed with a HeliRig implementation, coupled with variable rig and environmental controls, via a PC connected by USB. Our ECE technician, Daniel Hopkins, has recently designed and made available a graphical interface, which will allow your emulated rig controlled by your Tiva board to appear to "fly" in a graphical window on a monitor. Instructions on how to connect his interface are on Learn.

For Stage 2 (Weeks 4-6) your FreeRTOS controller will be uploaded to a booked (by your group) HeliRig for real-time "flight", testing, and calibration. HeliRig input signals will response to virtual buttons connected to an on-board Tiva control module, programmed using your uploaded FreeRTOS control implementation. In addition to user commands, tasks managed under the *FreeRTOS* scheduler will update height and yaw settings under the control of your PID (or PI) algorithm, which will maintain stability. Two additional modes using sequenced button presses will provide for automated mid-flight altitude adjustment and turns. A block diagram of both stages of your Term-3 project is shown in Figure 1.

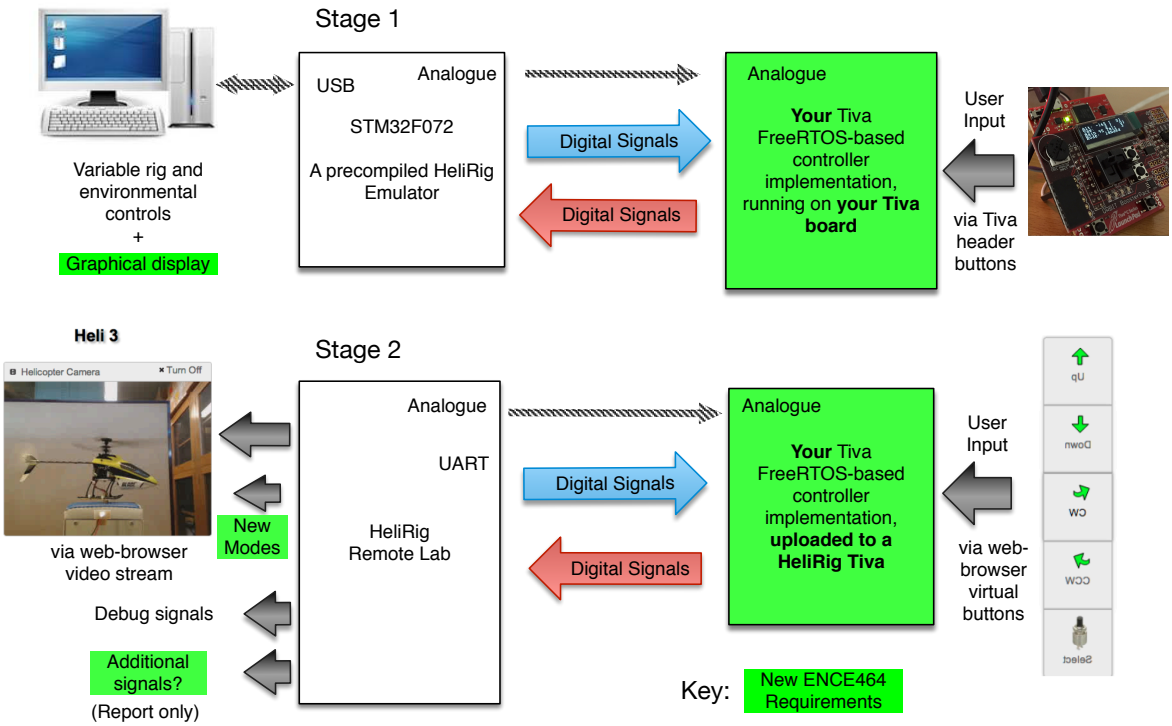

Figure 1: Block diagram of 3-student group HeliRig emulator and real-time system modules.

[1] During Term-3, small changes to this project requirement and specification handout are possible.

2. Background

The 2nd Pro helicopter rig (HeliRig) project has been offered as a capstone ENCE361 project over the last 6 years. Since first conceived as an Honours project [1], enhancements to the actual HeliRig have been numerous. Each enhancement has resulted in a more reliable platform and has subsequently reduced rig maintenance.

The Tiva board modules shown in Figure 1 are implemented in the form of embedded system modules. Historically, a Stellaris board from Texas Instruments (TI) was used but 4 years ago a Tiva board replaced this. ENCE361 students work in teams of 2 on a Tiva board to develop software to control a “flying” model helicopter within a closed-loop system. This year’s ENCE464 project aims to provide enhancements to extend this remote lab’s student experience.

3. Project Details & Requirements

For this Term-3 project students will sign-up during Week-1 (by 9 AM Friday 17 July) to a 3-person project group. As mentioned in §1, this 2-stage project will firstly require students to complete a FreeRTOS implementation on a Tiva board, most likely reusing a previously purchased ENCE361 Tiva board (or a loaned board from ECE), and connecting this to a precompiled and programmed HeliRig emulator, implemented by an ST Micro STM32F072 microcontroller. This is shown in the top portion of Figure 1.

An enhancement this year is a Helirig *visualiser,* which has been extended from the STM32 serial output that normally runs through a terminal emulation application. Parameters taken from the heliRig emulator can now be passed to a graphical interface application, which shows an animation of the HeliRig “flying”. This font-end visualiser will be available from PCs in the ESL under, `C:\Heli_Visualiser`. An example of the Heli-Visualiser is shown in Figure 2.

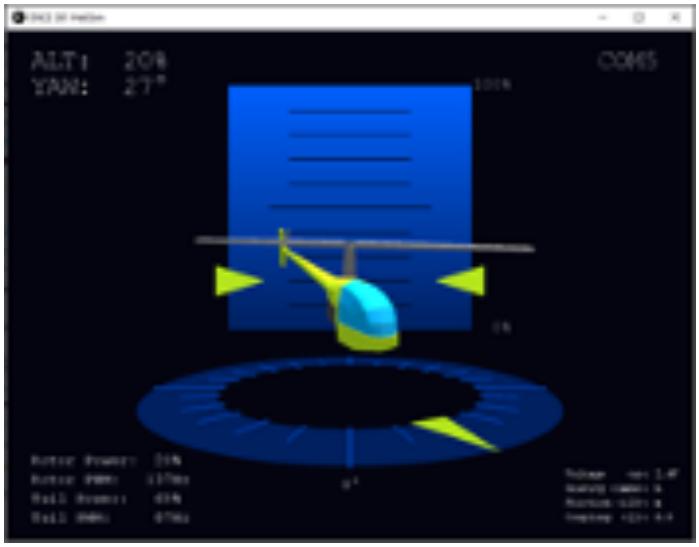

Figure 2: Front-End Visualiser for the HeliRig Emulator.

Groups should start the second part (Stage-2) of their development from Week-3, and continue to work on the requirements until demonstration of their project using one of the real-time HeliRigs for assessment in their Week-6 lab session. A FreeRTOS-based controller will also be required. However, it is expected that most (theoretically, all) of the FreeRTOS

software developed for the emulator will be suitable to use on a real-time HeliRig module. Testing for calibration on an actual HeliRig for demonstration of step control and altitude (similar to ENCE361), in addition to two new modes (see §3.1) may still however, be time consuming. Stage-2 of this project is shown in the lower-portion of Figure 1.

In Stage-1, groups will develop C code using FreeRTOS to “take-off” and control stable HeliRig “flight” from a precompiled and programmed HeliRig emulator. The emulator is limited to basic functionality, however use of a separate PC-based program will allow environmental parameters to be introduced, which essentially provide small variations between actual rigs. The goal for developers is to provide a robust control system to ensure stability of “flight” over two linked subsystems, i.e., the main rotor output with user ± input to select altitude (height) and tail rotor output with ±user input to select direction (yaw).

Physical variations exist over the actual (real) HeliRigs currently in use. ENCE361 students typically find running their control algorithm on different rigs quite challenging, especially in terms of maintaining stability when height or yaw changes are initiated on *any* of the the available rigs. Variations in spring tensions for balancing each rig, combined with non-linearised sensors and other factors, place a lot of emphasis on tuning control parameters to maintain stability. By using a remote lab emulation environment, which allows variations to basic rig and environmental properties, program control parameters can be refined to achieve a satisfactory result. This program, once loaded on any available (non-emulator) HeliRig, should allow a user to update height and yaw settings, whilst maintaining reasonable stable “flight”.

Most students who take ENCE361 purchase a Tiva board to complete a set of exercises and the HeliRig team project. The user interface, together with a PI or PID control algorithm running on the Cortex 4 microcontroller, allows a model helicopter to “fly” to any altitude and direction by setting specific height and yaw parameters via 4 buttons. These parameters drive main and tail motors by varying the duty cycle of corresponding PWM outputs. Feedback is provided by the rig in the form of an analogue height value and pulses from an encoder wheel allow a yaw position to be acquired. A proportional-integral-differential (PID) algorithm, or just PI, is used as a control method to maintain stability when take-off, landing or altitude/yaw updates are actioned.

These points and several more are detailed in the 2019 ENCE361 Helicopter Rig Control Project reference handout [2]. Table 1 provides a wiring guide for the Tiva control board and connections to the STM32 board, and where specific colour-coded wiring is indicted.

3.1. New Flight Modes

As mentioned in §1, two new flight modes will be required for this project. The basic aim is to provide users with more HeliRig functionality through a set of sequenced button presses. Two examples are as follows: 1) enabling an automated mid-flight altitude adjustment, and 2) initiating a 180 degree turn.

An example of set of sequenced, virtual button presses to enable mid-altitude level-flight (Mode-1), based on a simple calculation, ROUND[min_alt + (max_alt - min_alt)/2], may be initiated on a rig by using the following sequence:

`UP -> UP -> DWN -> DWN,` (*with reasonably quick succession button presses*)

A sequenced series of button presses to initiate a 180 degree turn (Mode-2), is left for student group discussion.

3.2. Generation or Reproduction of New HeliRig Signals

Also mentioned in §1, the generation of new HeliRig signals is required in an effort to add more realism to this project and provide a better user experience. An example is sound, where signals could be either:

i) generated from encoded signals sent from the rig to the client, or

ii) acquired using a microphone on the rig, and then reproduced at the client-end through an amplifier.

Unlike, however, the aforementioned requirements that should be coded into your TIVA controller, a description only of possible signals to extend functionality, and description of how this information might be transmitted to the client, is limited to your group report.

4. Mapping the Helicopter Emulator Board

As shown in the top portion (Stage-1) of Figure 1, the Tiva 1 board will be used to emulate the helicopter, which will receive digital signals corresponding to those from the controller board, e.g., button activations, such as *UP*, *DOWN,* etc., and PWM signals from controller outputs to drive tail and main helicopter motors. Note that the *Altitude* input, listed in Table 1 as Tiva input pin PE4, is an analogue signal. As a result, the ADC from your TIVA board will need to connect to the Heli Emulator DAC output on pin A2 of the Nucleo board. Note that the colours of the last 6 rows of Table 1 correspond to the wire colours from the Nucleo board. Please do not change or rewire these signal wires on our emulator boards.

Tiva pin	Tiva function	in/ Out	Helicopter signal	Nucleo board pins	Notes
J1-10	PA7	in	MODE	NA	Slider switch (HIGH = up) & virtual signal
J2-03	PE0	in	UP	NA	Active HIGH; virtual signal pluses HIGH on operation.
J3-05	PD2	in	DOWN	NA	Active HIGH; virtual signal pluses HIGH on operation.
J4-10	PF4	in	CCW	NA	Active LOW; virtual signal pluses LOW on operation.
J2-04	PF0	in	CW	NA	Active LOW; virtual signal pluses LOW on operation.
J1-09	PA6	in	RESET	NA	Active LOW; virtual signal pluses LOW on operation.
J3-10	PF1	Out	Tail rotor motor	D8 (in)	2% <= duty cycle <=98%, otherwise off
J1-03	PB0	in	Yaw channel A	D7 (out)	Ch. B leads Ch. A when yaw increases clockwise
J1-04	PB1	in	Yaw channel B	D6 (out)	
J4-04	PC4	in	Yaw reference	D2 (out)	**Low** indicates helicopter is at the reference position
J4-05	PC5	Out	Main rotor motor	D9 (in)	2% <= duty cycle <= 98%, otherwise off
J1-05	PE4	in	Altitude analogue	A2 (out)	Approx. range 1-2 V (decrease with increase altitude)

Table 1: Tiva MCU digital input/output data signals for the HeliRig control board [2]. Ground (0V) is available on Tiva pins J2-01, J3-02. 5V is available on J3-01.

Lastly, note that a serial data port, continuously streaming status and settings data, is extremely useful for debugging.

5. Documentation

The documentation contained in the *references* section of this project description is expected reading. You will also find details on the FreeRTOS functions on the Learn website. Other information, such as application notes and code examples, obtained from TI and the FreeRTOS websites, may be considered useful reading to help you complete your Term-3 group project.

6. Support

6.1. Purchase of Tiva and Orbit modules

Group members will probably each have a board from ENCE361 purchased in 2019. However, a Tiva board can be booked out through our Electronics lab office during the first week of Term-1. If you don't have a Tiva board, please see either of our Technicians, Randy or Diego in the ECE Electronics Laboratory Office during office hours.

6.2. STM32F072 Boards

A box of precompiled and programmed STM32F072 emulator boards with serial USB cables will be available in the ESL at all times during Term-3. In all fairness to other groups, these boards should not be removed from the ESL under any circumstances. If any board does go missing, we will implement a booking system during office hours. However, we would much prefer to maintain open student access to such resources; please respect this.

6.3. Laboratory sessions

A 2-hour laboratory slot between 1 p,m. to 3 p.m. each Thursday has been allocated for ENCE464 students in the timetable. Your Tutor, Andre Renaud, will be available during the first hour each week to assist students with this project. We will also have a TA to assist in the lab. Your course coordinator and Term-3 lecturer should also be available during these assigned lab sessions, as well as our lab technician, unless otherwise occupied with other duties.

6.4. Group allocations and document control

Since this is strictly a term-based project, deadlines are very tight, and milestones should be met in order to complete your term project. A self-selection group resource will be available on the ENCE464 Term-3 project support Learn page from Tuesday 14 July. Students will have until 09:00 Friday 17 July to sign up to a group. Any student not signed up by 09:00 Friday will be allocated randomly to form a 3 student group, irrespective of prior, informal arrangements with other group members. Your course coordinator reserves the right to add another student to an existing group of 3 students with an associated and correspondently, higher workload.

A git repository will be set up for each group at the start of your second week of Term-3. If you need help with conflicts etc, Dave van Leeuwen will be available for this.

7. Deadlines, deliverables and mark allocations

Each **milestone** listed below is provided as a guide for groups, and as such they are not directly assessed. However, the **deliverables** listed below are considered **hard deadlines**, in that they will be assessed as required on the day prescribed. As such, each deadline assessment will accrue a penalty of 10% per day if submitted late.

Milestone 1 (Week-1): Use of an emulator and defining a few simple acquisition tasks:

By the end of this week, all groups should have a least installed FreeRTOS and executed a few demonstration tasks.

Milestone 2 (Week-2): HeliRig emulator and running a few simple acquisition tasks:

By the end of this week, all groups should be proficient at using a HeliRig emulator with their Tiva board and be able to monitor *Height* data and preferably, yaw data from the controller module using a few FreeRTOS tasks.

Milestone 3 (Week-3): Refining task development using Helirig emulators

Work on the control module should be at the stage were PI or PID tasks can close the control loop, however there may still be underlying stability issues. All team members should now be somewhat confident with the use of FreeRTOS queues, i.e., in terms of streaming data between tasks, and the implementation of mutexes and semaphores, where appropriate. Code for two extra *flight* modes should be underway. By the end of this week groups will need to transition from Helirig emulators to HeliRig real-time modules.

Milestone 4 (Week-4): Porting code to a real-time HeliRig

By mid-week, queues, semaphores, and mutex operations should be complete and the testing of new flight modes should also be nearing completion. Code should be ported to the real-time HeliRigs and tests performed to ensure that the demands of a real-time system, as distinct to running on an emulator, are achievable. This would be an ideal time to compare and record changes between an emulation and real-time system.

Milestone 5 (Week-5): Refining task module on the real-time HeliRig

Adequate time should be allocated for system testing. It is most important that your group assign measurable test criteria so you can secure testable results. Given the tight timeframe of the project it is impossible to provide a comprehensive set of all test conditions, however, basic test procedures, such as taking off, incremental height adjustments, and CW and CCW yaw changes, should be operational. Testing of at least one of these basic operations will be required as part of your demonstration and one of the two new flight modes of operation should also be ready for demonstration in Week-6; which one is used will be up to the discretion of the assessor at the time of your inspection.

Deliverable 1 (Thursday, Week 6): Demonstration (33% of your assignment mark)

13:00 - 15:00, 20 August: Each group will be asked to give a short 6-minute demonstration of their assessment in one of two parallel sessions held in scheduled ENCE464 lab time during the last week of Term-3. Two real-time Helirigs will exclusively be used (by assessors) over the duration of the session, in addition to preparation time that will be undertaken over the evening prior to inspections. Given this, it is very important that groups note that access to available HeliRigs for last minute tuning and testing will be severely limited from 5 pm Wednesday 19 August to 4 pm Thursday 20 August. Booking times for demonstrations will be available (posted on the Embedded Systems Lab notice-board or door) from 9 am Friday 14 August. One group member from each group will need to sign up for a demo slot. Demonstration on a HeliRig emulator + graphical display may be allowed. However, this would be at the peril of losing possible demonstration marks for this part of the assessment.

Demonstration of a “take off”, a basic attitude or yaw change operation, and one of the mode functions should be expected. Lastly, use of at least one semaphore, one mutex, and one or more queues in your source code should be highlighted during your demonstration.

Deliverable 2 (Friday, Week 6): Code (33% of your assignment mark)

18:00 Thursday 20 Aug.: Your final source code should be uploaded to your git repository. Modularisation, coding style, OOD[2] where appropriate, documentation, and other factors will be assessed.

Deliverable 3: Critique (ALL groups) (10% of your assignment mark)

23:55, Sunday 23 August: A confidential, and individual, 1-page critique assessing your performance, and those of your fellow group members, should be submitted in a separate dropbox to your group report. This will be similar to your ENCE461 assignment.

Deliverable 4: Report (ALL groups) (24% of your assignment mark)

23:00, Sunday 23 August: A formal group report will be required for this project. The core of your report should be no more than 10 but no less than 6 pages, where a title page, ToC, appendices, and references are required but not included in this max/min page count. At least one page of your report should discuss how the HeliRig can be modified to accommodate sound. Another page should outline code variations required when porting your design and code from a HeliRig emulator to a HeliRig real-time units, and performance issues encountered when testing between platforms.

Your report will include at least 2 appendices (more are allowed, but these may not be assessed). The first appendix should detail results from one or more black-box tests conducted during your development. The second appendix should provide a list of task descriptions and CPU load analysis for your controller Tiva, given by your FreeRTOS scheduler. Last, but far from least, grammar, punctuation and spelling, in addition to technical writing style, will form a portion of your report assessment.

References

[1] Wareing NM., Bones PJ. and Weddell SJ. (2013) *A remote lab implementation using SAHARA.* Auckland, New Zealand: Electronic New Zealand Conference (ENZCon), 5-6 Sep 2013. In ENZCon Conference Proceedings: 51-56.

[2] Weddell, S.J., Bones, P.J. and Wareing, N.M. (2014) *A Remote Laboratory for learning embedded systems and control.* Wellington, New Zealand: 25th Conference of the Australasian Association for Engineering Education (AAEE2014), 8-10 Dec 2014.

[3] Moore, C. and Yang, L., ENCE361 Embedded Systems 1 Project Notes : Helicopter Rig Control, May 2019.

[4] P. Hof, *ENCE361 Heli Emulator Documentation*, 15 July 2019.

[5] www.freertos.org - FreeRTOS/Demo/CORTEX M3/M4 Series

[2] Object orientated design