1. Project Overview and Scope[1]

Groups of 4 students are to design real-time software to run on a Tiva microcontroller (MCU), which will interact and control a supplied helicopter emulator program running a separate microcontroller. The Tiva will provide a user interface (UI) to allow user height and yaw changes, as well as stable flight through a set of control tasks. A key requirement is to use the FreeRTOS operating system, i.e., as distinct to a round-robin or simple time-slice scheduler, to implement user defined tasks, including a control algorithm, that will *fly* an emulated helicopter program. To achieve this, the project has been divided into two stages, where completion of five milestones will be required to complete the project.

For Stage 1 of your project, you will develop a set of FreeRTOS tasks on your Tiva MCUs that will be used and tested on mechanical rigs (HeliRig pedestals), supplied and supported in our embedded systems lab (ESL). These mechanical rigs have been developed over several years for our embedded systems course, ENCE361, and were last used in 2020 for student laboratories. The key features of these test rigs include the ability for a developer to change height and yaw settings through the use of a mechanical spindle or shaft, which supplies analogue data and digital signals to analogue-to-digital (ADC) and GPIO ports on the Tiva board, respectively. This provides a basis for software task development using FreeRTOS.

During Stage 2 of your project (Weeks 4 and 5) your UI and controller modules, running on your Tiva MCU using FreeRTOS tasks developed during Stage 1, will be refined for testing on a STM32F072 microcontroller module. This emulation module has been pre-programmed with a HeliRig implementation, and has been coupled with variable rig and environmental controls, via a PC connected by USB. Our ECE technician, Daniel Hopkins, has designed a graphical interface, which will allow your emulated rig, when controlled by your Tiva board, to appear to “fly” in a graphical window on a monitor. Instructions to use this are on Learn.

In addition to user commands, such as altitude (height) and yaw changes, implemented through 4 buttons on your Tiva board, tasks managed under the *FreeRTOS* scheduler will update respective settings under the control of a PID (or PI) algorithm. This will maintain stability of “flight”. The block diagram in Figure 1 shows both stages of your project.

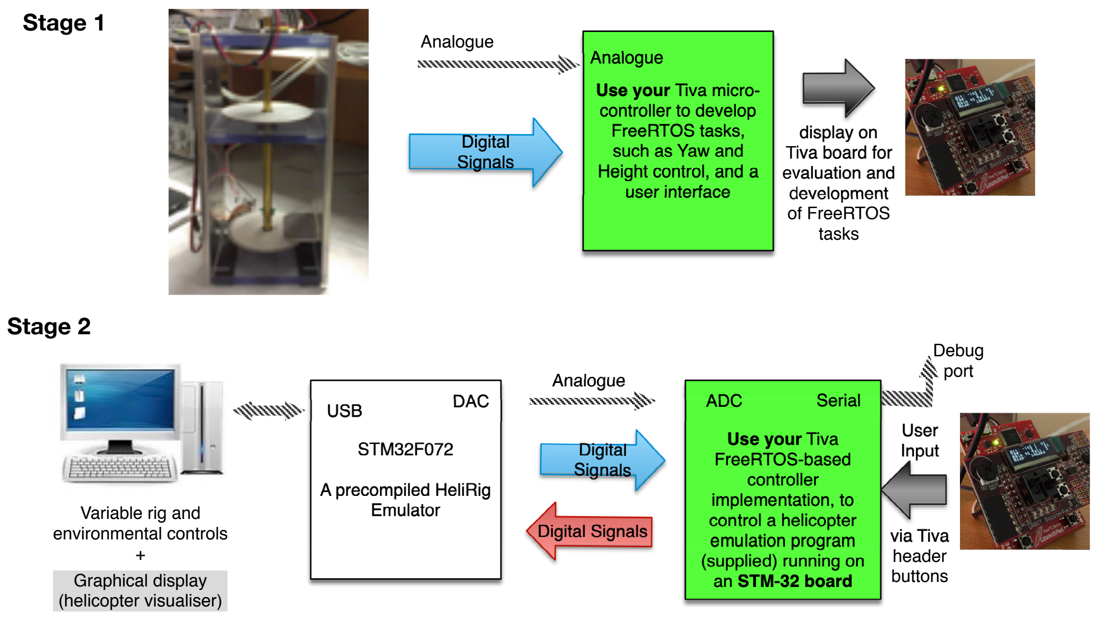
Stage 1
Analogue
Digital Signals
Analogue
Use your Tiva micro-controller to develop FreeRTOS tasks, such as Yaw and Height control, and a user interface
display on Tiva board for evaluation and development of FreeRTOS tasks
Stage 2
Variable rig and environmental controls
+
Graphical display (helicopter visualiser)
USB
DAC
STM32F072
A precompiled HeliRig Emulator
Analogue
Digital Signals
Digital Signals
ADC
Serial
Use your Tiva FreeRTOS-based controller implementation, to control a helicopter emulation program (supplied) running on an STM-32 board
Debug port
User Input
via Tiva header buttons

Figure 1: Block diagram of 4-student group Helicopter emulator and system modules.

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

Note: For this project, student groups will need to demonstrate adjustment (of a set-point) and control (at a new setpoint) for **either** height, **or** yaw, **not both**. However, groups who demonstrate adjustment and control of both height and yaw will be eligible for a bonus mark.

2. Background

The ECE 300-level helicopter rig (HeliRig) project has been offered as a capstone project over the last 6 years. Since first conceived as an Honours and then Summer Student 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 module, highlighted green in Figure 1, is an embedded system that are programmed by students in various ECE undergraduate courses to implement a UI and control system for a model helicopter. More recently, a real-time HeliRig emulator has been developed to allow students to develop a UI and helirig control system, which can be used in parallel with our six powered helicopters. Essentially, software is written and tested on student Tiva boards to control and actually fly a model helicopter within a closed-loop system. This year, your ENCE464 project will instead use one of our set of ECE Helirig emulators, in addition to a heli-visualiser, shown in Figure 2, to extend their knowledge of embedded software using FreeRTOS. To achieve this, student pre-owned Tiva boards will be required.

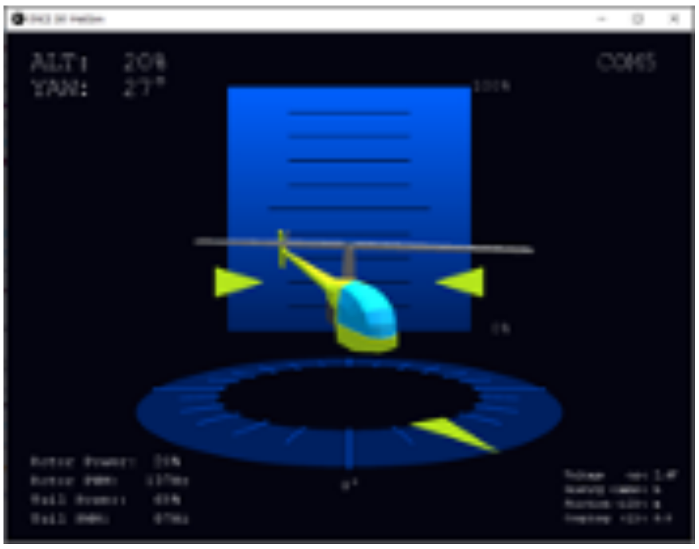

Figure 2: Front-End Visualiser for the HeliRig Emulator.

3. Project Details & Requirements

For this Term-3 project students will sign-up during Week-1 (by 12:00 Friday 23 July) to a 4-person project group. As mentioned in §1.0, this 2-stage project will firstly require student groups to complete a FreeRTOS implementation on a Tiva board using one of our ESL supplied mechanical modules, in conjunction with a previously purchased ENCE361 Tiva board. Students who no longer own a Tiva board can loan a board from ECE.

3.1. The User Interface (UI)

A user interface, together with a PI or PID control algorithm running on the Tiva microcontroller, allows a model helicopter to “fly” to a maximum altitude and over any 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 emulator 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.

3.2. Stage 1 Description

Essentially, for the first 3 weeks of this project Tiva boards will be connected to an ESL helirig pedestal to acquire analogue and digital data from transducers. The pedestals were used last year in ENCE361 labs (3 and 4) to provide analogue and GPIO data for ADC conversion and PWM generation. Pedestals provide a mechanical spindle that is connected to disk and encoder wheel, all contained within a perspex frame. An example of these units is shown on the left of the Stage 1 section (top) of Figure 1.

FreeRTOS routines will be developed to acquire and use data supplied by these mechanical rig modules, and these signals are designed to be connected to ADC and GPIO ports on Tiva modules which can then be used to process the acquired analogue and digital data. A list of the relevant ports are shown in Table 1.

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 pulses HIGH on operation.
J3-05	PD2	in	DOWN	NA	Active HIGH; virtual signal pulses HIGH on operation.
J4-10	PF4	in	CCW	NA	Active LOW; virtual signal pulses LOW on operation.
J2-04	PF0	in	CW	NA	Active LOW; virtual signal pulses LOW on operation.
J1-09	PA6	in	RESET	NA	Active LOW; virtual signal pulses 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.

The resulting processed data, which will require scaling, buffering and filtering, can then be displayed on the Tiva display for testing. For example, ADC data from a supplied analogue signal voltage is used to display the height or altitude of the helirig, which can be tested by the raising or lowing of a connected mechanical rig. Concomitantly, 2-channel encoder data can be used to display yaw position by turning the spindle. FreeRTOS tasks will be defined and used to manage the data acquisition and signal conditioning, i.e., buffering, filtering, and scaling of data. Completion of this work will result in satisfying the Week-3 project milestone.

3.3. Stage 2 Description

For the remaining weeks of your Term-3 project, your Tiva board will be connected to a precompiled and programmed HeliRig emulator, implemented by an ST Micro STM32F072 microcontroller. This part of your development is defined as as *Stage 2*, and is shown in the lower portion of Figure 1. Student groups will use this emulator, *as is*. As a result, it cannot and should not be reprogrammed by students. The same ports (Table 1) used for the mechanical pedestal used in Stage 1 will however be used on your heli-rig emulator. As mentioned, implementation of PID controller tasks will also be required to stabilise the height and yaw of the emulated HeliRig. A general C example of a PID controller will be provided.

An enhancement to the emulators 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.

Essentially, during Stage-2 groups will develop C code using FreeRTOS on a Tiva board to "take-off" and control stable HeliRig "flight" as their program tasks interact with a precompiled HeliRig emulator running on an STM 32 microcontroller. The emulator is limited to basic functionality, however use of a separate PC terminal-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 via PWM, with user $\pm$ input to change altitude, (the height set-point) and tail rotor position with $\pm$user input to select direction (the yaw set-point).

4. Project Workflow & Assumed Skills

Groups should start the second part (Stage-2) of their development from Week-4, and continue to work on the requirements until demonstration of their project using a HeliRig emulator for assessment during their Week-6 lab. The emphasis here, which is also true of your milestones, is that student groups should show how FreeRTOS can be used as a more effective means of implementing program control. Lecture material will support this.

Most students who take ENCE361 purchase a Tiva board to complete a set of exercises and a team-based project. Therefore, some prior knowledge of the Tiva board, modules, and software development using the Tiviaware API and code composer studio (CCS), is assumed. Furthermore, knowledge from ENCE361 laboratories, and corresponding milestones in 2020, is assumed. Conversion of a round-robin scheduler to FreeRTOS tasks, will however be required.

The user interface, together with a PI or PID control algorithm running on the Tiva micro-controller, allows a model helicopter to "fly" to a maximum altitude and over any 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 emulator 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. To launch this project, a special project tutorial will be given during your ENCE464 lab in Week-1.

5. Mapping the Helicopter Emulator Board

As shown in the bottom portion (Stage-2) of Figure 1, the Tiva board is used to control the "helicopter" that runs on the STM32 (Nucelo board). The Tiva receives digital and analogue signals from the "helicopter", and then transmits PWM power signals to a "helicopter" main and tail "rotors" running on the STM32 emulator to maintain stable "flight". Tiva button activations, such as *UP* or *DOWN* for height, and clockwise (CW) and counter-clockwise (CCW) for yaw, increase or decrease PWM duty cycle signals to the main, and tail rotors, respectively. Such actions initiate a change of height or yaw by the "pilot" (user). 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 on the Nucleo (STM32) board, as shown in Figure 1. 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**.

Lastly, note that a serial data port, continuously streaming status and settings data, is extremely useful for debugging. This will need to be developed and supplied in Stage 2 of your project.

6. 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.

7. Support

7.1. Purchase of Tiva and Orbit modules

We expect that most group members will each still have a board from ENCE361, which would have been purchased in 2020. 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.

7.2. 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.

7.3. Laboratory sessions

A 2-hour laboratory slot between 1 p,m. to 3 p.m. each Tuesday has been allocated for ENCE464 students in the timetable. Your Technical Tutor, Dr Ben Mitchell, will be available during each week to assist students with this project. We will also have a TA (Abdullah Naeem) to assist in the lab. Your course coordinator should also be available during *some* of your assigned lab sessions, as well as our lab technician, unless otherwise occupied with other duties. Our Assistant Lecturer, Dr. Saloni Pal, will be responsible for the management of this term project, i.e., group formulation, demonstrations, and general project coordination.

7.4. 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 20 July. Students will have until 12:00 Friday 23 July to sign up to a group. Any student not signed up by 12:00 Friday 23 July will be allocated randomly to form a 4 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 4 students, where project work would be extended (in addition work outlined in this document) and would result in a correspondently, higher workload. If you have any issues regarding your group or group allocation or management, you should contact Dr. Saloni Pal.

Lastly, 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, please contact Dave van Leeuwen for such matters.

8. 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 deliverable assessment will accrue a penalty of 10% per day if submitted late.

Milestone 1 (Week-1): **Milestone 1 (Week-1)**:

(a) Go through the FreeRTOS tutorials. Start a FreeRTOS project using Mat Pike’s example. Review your old lecture notes and ENCE361-20S1 labs (Wk-3 & Wk-4 are on Learn) and define these requirements in terms of employing FreeRTOS tasks. By the end of this week, all groups should have a least installed FreeRTOS, can run a project, and check that you can get some data from a pedestal rig.

Milestone 2 (Week-2): Use the lab emulator and develop the height and yaw tasks.

Extend your work in Week-1 by:

(b) getting more experience with the pedestal rigs; connect the signals to your Tiva board and start checking the receipt of data samples (from the pedestal rig), to the FreeRTOS tasks running on your Tiva boards, and then,

(c) make sure data is displayed on your Tiva display using your UI routine, which will also need to be implemented as a FreeRTOS task.

By the end of this week, all groups should be proficient at using a pedestal rig with their Tiva board and be able to acquire *Height* data and yaw data from a lab rig using a few FreeRTOS tasks. Data from processed signals should be showing on the orbit display, and moving the shaft from a lab rig should result in changes to height measurements to show on the display data can be added to a height buffer using the allocated Tiva “up” button, and subtracted using the appropriate Tiva “down” button. In addition groups should aim to acquire GPIO data from rotating the lab rig shaft. A simple finite state machine or Ben’s XOR code can convert the acquired GPIO data into encoder counts, and the yaw should be showing in degrees on the display.

Milestone 3 (Week-3): Design a task structure and start development using lab emulators.

Software design considerations will be required for this project, i.e., in terms of a task structure. For example, a serial port should be implemented to allow the monitoring of the helirig for debugging, but how do you protect data written to the port from multiple tasks? Is there a FreeRTOS control resource you can use for this?

Work should also be started on a control module to stabilise **either** the height and yaw, i.e., when changing the setpoint on either to a new setting. An individual FreeRTOS control task will be required to implement either a PI or PID algorithm, which will be effectively used to close the control loop. 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, for appropriate use. By the end of this week groups will need to transition from lab pedestal rigs to lab HeliRig emulator modules.

Milestone 4 (Week-4): Porting all code to a HeliRig emulator.

By mid-week, queues, semaphores, and mutex operations should be complete and the testing of control routines should be underway. Code should be ported to a HeliRig emulator and tests performed to ensure that the demands of a "real-time" emulated system are achievable. The serial port task should be fully operational, and reporting regular status updates on the status of your helirig controller. Depending on your design, key resources should be protected and managed accordingly using appropriate FreeRTOS functions and controls.

Milestone 5 (Week-5): Refining task modules on a HeliRig emulator

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, or 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 in Week-6.

Deliverable 1 (Tue. Week 6): Demo (33% of your assignment mark; bonus 10%, if < max.[2])

13:00 - 15:00, 24 August: Each group will be asked to give a 8- to 10-minute demonstration of their assessment in one of two parallel sessions held in scheduled ENCE464 lab time during the last week of Term-3. Helirig emulators will exclusively be used over the duration of the session. A booking sheet for demonstrations will be available (posted on the Embedded Systems Lab notice-board or door) from 9 am Tuesday 17 August. One group member from each group will need to sign up for a demo slot.

Request from two examiners for a demonstration of a "take off", a basic attitude **or** yaw change operation, or both if you are going for the bonus, using the visualiser and debug information, should be expected. Lastly, but most importantly, 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 (Tue. Week 6): Code (33% of your assignment mark; bonus 10% if < max.[2])

By 13:00 Tuesday 24 Aug: Your final source code should be uploaded to your git repository.

Modularisation, coding style, OOD[3] where appropriate, documentation, and possible extensions related to ENCE464 embedded systems content, should be assessed.

[2] It will not be possible to score more than 100% of this assessment component

[3] Object orientated design

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

23:55 Friday 27 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)

23:55, Friday 27 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, and no more than 8 pages, where a title page, ToC, appendices, and references are required but not included in this max/min page count. Another page should describe how FreeRTOS tasks, queues, semaphores and mutexes were used in your software design when implementing your HeliRig emulator. Measured performance and any issues encountered when testing your software on the heliRig emulator should also be described.

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