Project: Helicopter Rig Control

The project will be performed by groups of 2 students each. See the Learn page for confirmation of your group and to check your group number. Please include the group number on all source files, your laboratory book and your project report. It also identifies your Git repository. Each group must have *at least* one Tiva/Orbit unit (purchased from the Copy Centre); it is however recommended that each student have a board.

The aim for each group is to produce a program to control a model helicopter, as detailed below. Each group will demonstrate their completed real-time control program on a remotely accessed helicopter during student group laboratory sessions over the last week of the semester.

There are **two** milestones scheduled during the project which are designed to help groups to make progress towards the final goal. **Milestone 1** is due during **Week 5**; **Milestone 2** is due during **Week 8**. A final group report on the project is due on **Sunday 3rd June**. The code within the group's ***Git*** repository will be checked out at **8 p.m. Sunday 3rd June**.

At least 3 tutorials will be dedicated to support achieving the milestones and the final program.

1. Goal & Final Demonstration

The goal of this project is for student groups to "fly" a sortie, where a "pilot" can: i) find a direction in which to take-off from, ii) rise to an altitude were stable, maneuverable flight can be maintained, iii) rotate around a fixed position and over several incremental steps, and lastly, iv) land back at base, "parking" your helicopter in a home position.

Every group will be given the opportunity to demonstrate the above goal during their lab session in the last week of the semester. As mentioned, two Milestones have been set to achieve this overall goal, and to help student groups maintain a more manageable workload.

2. Description

The Tiva board is to be programmed with a small real-time kernel and interrupt-driven control program designed to control the lift off, hover, heading and landing of a small model helicopter mounted on a rig. The power to achieve this is provided by two motors, one for the main rotor and the other for the stabilising tail rotor. Both motors are controlled by pulse width modulated (PWM) outputs. Feedback to the Tiva board is available in the form of altitude, yaw, and reference orientation signals.

One of the helicopter rigs is shown in Figure 1. It comprises the helicopter (with main and tail rotors) mounted on a cantilevered arm, plus altitude and yaw sensors with associated electronics. *virtual signals* are sent to one of our helicopter rigs over an internet-based remote laboratory interface[1].

[1] Weddell SJ, Bones PJ, & Waring NM, *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

Altitude is monitored by means of an optical distance transducer and simple electronic interface which produces an analogue signal between ~2V (down) and ~1V (up), corresponding to the approximately 70 mm altitude range. The yaw is monitored by means of a quadrature optical angle encoder which generates two binary signals at voltage levels compatible with the Tiva board. There are 112 slots around the circumference of the angle encoder (i.e., 112 over 360 degrees). The yaw movement isn't limited or restricted in any way, i.e. the helicopter can rotate freely. A reference signal is asserted (**LOW**) when the helicopter faces directly toward the camera.

Figure 1. The helicopter mounting rig.

The motors of the helicopter are powered from bench power supplies via a power electronics interface supplied with the helicopter mounting rig. The main rotor motor takes up to 6 amps at full power. The power to each of the motors is controlled by an output PWM signal from a Tiva board, which is identical to the one that students have purchased. The main rotor rotates clockwise (viewed from above). The tail rotor generates a torque which opposes the torque of the main rotor.

3. **Program requirements** (*Version 1.0*)

The program to be developed must have the following features:

3.1 The 2-channel quadrature signal for the helicopter yaw is processed via interrupt-driven or polled decoding. (***There is a built-in Quadrature Encoder Interface (QEI) unit on the Tiva board however this is NOT used.)*** A display of the helicopter yaw in ±degrees is maintained on the Orbit display.

3.2 An independent yaw reference signal is available to the Tiva board. This allows the yaw to be calibrated from a known orientation of the helicopter. Initially, the helicopter needs to be rotated to locate the reference position (see §3.5).

3.3 The analogue signal for helicopter altitude is sampled and converted by the Tiva ADC (module 0, ch 9); a display of the helicopter height expressed as a percentage (100% is fully up, 0% fully down) is maintained on the Tiva's OLED display. The fully down voltage is measured when the helicopter is known to be 'landed'. To reduce the effect of electrical noise, several samples must be averaged to estimate a reliable height.

3.4 Two output PWM signals are generated, one for the main rotor motor (Module 0, PWM 7), one for the tail rotor motor (Module 1, PWM 5). The frequency for the PWM signals is set in the range 150 Hz to 300 Hz. A display of the PWM duty cycle for each motor as a percentage is maintained on the Orbit display.

3.5 The SW1 slider switch on the Orbit daughter board is programmed to control the controller mode (MODE). The down position indicates that the helicopter is landed already or that it should land from its current position. The landing sequence causes the motors to reduce in speed in a way which brings the helicopter to a smooth landing in the reference orientation, and to then make the motors stop. Any subsequent change of the slider is ignored until the landing sequence is completed. Sliding the switch up (provided the helicopter is landed) causes the motors to start and the helicopter to rotate to the reference orientation (i.e., facing directly ahead). On start-up of the program, the motors are off. If the slider is up at start-up or after a 'reset' operation (either physically on the Orbit board or virtually on the remote interface), the helicopter remains in the 'landed' state. The change from 'landed' to 'flying' state only occurs after a change from down to up of the

slider. Note that this may mean two pushes of the virtual button are required to make the helicopter fly again after re-programming or a reset.

3.6 The four user-programmable buttons of the Tiva/Orbit unit are programmed to interactively control the altitude and yaw of the helicopter. Each operation of the UP button on the Orbit causes the helicopter to rise in altitude by 10% of the total range; each operation of the DOWN button on the Orbit causes the helicopter to lower in height by 10%, while retaining current yaw in both cases. Each operation of the CCW (left) button causes the helicopter to rotate 15 degrees counterclockwise (viewed from above); each operation of the CW (right) button causes the helicopter to rotate 15 degrees clockwise (viewed from above), while retaining the current height in both cases.

3.7 If the helicopter is 'flying', once the desired altitude has reached either of the limits of the altitude range, further operations of the button associated with that limit are ignored.

3.8 The actions of the slider switch and the four pushbuttons, as described in 3.5, 3.6 and 3.7, are emulated by *virtual signals* which are sent to one of our helicopter rigs over an internet-based remote laboratory interface. In the case of the pushbuttons, the signal is asserted for 200 ms for each "push"; only one can be operated at a time. The virtual signal for the slider (MODE) is HIGH in the up position.

3.9 Operation of the virtual RESET button (active LOW) invokes a call to the API function `SysCtlReset()` (prototype in `driverlib/sysctl.h`).

3.10 The program has a real-time foreground/background kernel operating on a round-robin basis for background tasks. Robust behavior is maintained at all times. A *PID controller* design is recommended to maintain controlled "flight".

3.11 The program uses the pins and on-board resources according to the data in **Table 1**:

Tiva pin	Tiva function	In/out	Helicopter signal	Notes
J1-10	PA7	In	MODE	Slider switch (HIGH = up) & virtual signal
J2-03	PE0	In	UP	Active HIGH; virtual signal pulses HIGH on operation.
J3-05	PD2	In	DOWN	Active HIGH; virtual signal pulses HIGH on operation.
J4-10	PF4	In	CCW	Active LOW; virtual signal pulses LOW on operation.
J2-04	PF0	In	CW	Active LOW; virtual signal pulses LOW on operation.
J1-09	PA6	In	RESET	Active LOW; virtual signal pulses LOW on operation.
J3-10	PF1 (M1PWM5)	Out	Tail rotor motor	2% <= duty cycle <= 98%, otherwise off
J1-03	PB0	In	Yaw channel A	Channel B leads channel A when yaw increases clockwise
J1-04	PB1	In	Yaw channel B	
J4-04	PC4	In	Yaw reference	LOW indicates helicopter is at the reference position
J4-05	PC5 (M0PWM7)	Out	Main rotor motor	2% <= duty cycle <= 98%, otherwise off
J1-05	PE4 (M0AIN9)	In	Altitude (analogue)	Approx. range 1 – 2 V (decreases with increasing altitude)

Ground (0V) is available on Tiva pins J2-01, J3-02

3.12 The serial link via UART0 and the Tiva board USB is used to output information on the status of the helicopter (9600 Baud, 1 stop bit, no parity). Regular updates are provided of the desired and actual yaw (degrees), the desired and actual altitude (%), the duty cycle for each of the main and tail motors (%, with 0 meaning off) and the current operating mode. The information needs to be in a concise, but easily readable, and format compatible with the scrolling display on the remote lab web page. The set of information must be communicated to the remote host at regular intervals (no less often than 4 times per second).

4. Source Code

On Learn, Section 1 contains a **Laboratory Source Code** folder, and in Section 2 (Real Time Processing), an **Example code** folder. Combined, these contain all of the examples discussed in lectures and tutorials by Phil Bones and your Term-2 lecturers. In addition, you may use the TivaWare examples. *Note well*: If you use or adapt code written by your lecturer or any other person, you must take responsibility for that code. That responsibility includes changing the comments to reflect your use of the code; you will lose marks for not following this guidance.

5. Serial port

During development, groups may choose to output *debug information in addition to that specified in 3.12 on Page 3*, i.e., the serial port. Such debug information **should not** be present in the final version. It is strongly recommended that groups use compiler directives to control the transmitting of such debug information. For example, a section of code which should not be compiled can be bracketed as follows:

```
#ifdef DEBUG_ONLY
// Code for debugging here; only compiled if DEBUG_ONLY is defined non zero.
#endif
```

6. Milestones

The milestones (*specified in separate documents*) are to encourage you and your team to work on the project and to give you some guidance towards code, which may contribute to your final helicopter control program.

7. Final report

The final report is to be a design document for your implementation of the helicopter control program. I would like to see evidence of a methodical design process undertaken to satisfy the specification. In particular, there should be sections which: a) identify the *tasks* to be performed by the program, b) identify the frequency of execution required for each task, c) describe the design of the round-robin scheduler, d) describe the method or methods used for inter-task communication, e) describe the process used to set the parameters of the PID controllers, f) describe any other special features of your program design and g) provide some conclusions about what worked and (possibly) what could have worked better.

The report should be structured with numbered sections, but it does not need to keep to any strictly formal template. The report should be no longer than 6 pages (11-point font with single line spacing). Do not include code listings, but small extracts from the source code can be used to illustrate points made in the report.

Submit the report as a **PDF** via the upload facility on the Learn page under Project.

8. Marking

The total mark of 35 (contributing 35% towards the final ENCE361 grade) comprises:

- 3 marks for Milestone 1.
- 6 marks for Milestone 2.
- 10 marks for the demonstration of the program, i.e. how closely it meets the specification.
- 16 marks for the final report and code (as checked out of the Group's repository on **June 3rd**).