Helicopter Rig Control Project

Group 1- Jack Miller & Jack Edwards
Date of submission: 29 May 2024

Section 0: Introduction

The helicopter control rig project is a program written in C for the Tiva Launchpad with an Orbit Boosterpad. The program allows a ‘pilot’ to control a remote control helicopter, which in addition to being controlled by the pilot will also find the correct direction to take off, takes off when the pilot decides, rising to an altitude where stable and maneuverable flight can be maintained. Once in the air, the pilot can control the rotation and altitude of the helicopter using the button peripherals on the Tiva board, setting a ‘target’ to which the helicopter can move towards. At any point, the pilot is able to use the switch peripheral to instruct the helicopter to land at a predetermined ‘home’ position.

Section 1: Tasks

- **PWM signal generation**

On the Tiva Launchpad, the PWM peripherals can be accessed on PC5(main rotor) and PF1(tail rotor). In the context of this project, PWM signals are used to control the speed of the two rotors (main rotor & tail rotor). Using the difference between the last recorded value

and the target value, the PID controller will calculate the duty cycle of the PWM waveform required to move toward the target yaw or altitude.

- ## User control

During flight, the pilot is able to have full control over the helicopter's rotation and altitude using the UP, DOWN, LEFT & RIGHT button peripherals. The buttons do not directly change the PWM signal of the rotors, instead incrementing a target yaw (by 15°) or altitude percent (by 10%), which can then be used to calculate the error in the PID controller by using the difference between the respective measured reading and the target value. In addition to these more specific controls, the user is also able to use the SWITCH1 peripheral (PA7) to initiate the takeoff sequence (When PA7 is HIGH) and initiate the landing sequence when switching it to LOW. SWITCH2 (PA6) can also be used to perform a soft reset of the system. The buttons are run using flag-based polling, as relative to other aspects, buttons aren't used as frequently. The reset switch is run using an interrupt, but upon later reflection, this should have been done using flags, such as the other buttons and switch, as it caused unwanted behavior (This is discussed in more detail in the conclusion in Section 5).

- ## Altitude control

To implement the altitude control, ADC is used. The altitude sensor gives a value of between 1 and 2 volts. 2 volts is the landed or minimum value, and as the helicopter rises to maximum altitude, the ADC voltage will decrease by approximately 1 volt. This process is triggered by the systick timer to ensure readings are taken at a frequent and regular rate. This value is stored in a circular buffer and an average is then taken to ensure smooth flight, whilst maintaining accurate measurements. These measurements are then converted into a percentage of the helicopter's total possible altitude and then are able to be used by the control functions. When the pilot is able to control the altitude, the target altitude is able to be controlled with the button peripherals(PE0 & PD2), at which point the PID function is called, which controls the main rotor's PWM signal, thus adjusting the height.

- ## Yaw control

In order to control the rotation of the helicopter, similar to the altitude, the pilot is able to use the button peripherals (PF0 & PF4) in order to increment the target yaw. A PID controller is then called to calculate the updated PWM signal of which the tail rotor should be run at, similar to the main rotor. The quadrature encoder detects any change in rotation, and then triggers the quadrature encoder interrupt which, depending on the states of the two encoders (determined by the direction of the rotation), sets the correct phase (This can be seen in listing 1). Depending on phase yaw, a global variable is either incremented or decremented. This value is then converted to degrees for use by the display and control functions. To ensure accurate measurement of yaw, a reference yaw interrupt is used. This interrupt sets the yaw to zero whenever the yaw reaches the reference point.

```
if (!encoder_A && !encoder_B) {
    cur_phase = 1;
} else if(!encoder_A && encoder_B) {
    cur_phase = 2;
} else if(encoder_A && encoder_B) {
    cur_phase = 3;
} else if(encoder_A && !encoder_B) {
    cur_phase = 4;
}
```

Listing 1: Quadrature encoder interrupt handler

- ## Flight states

In order to increase ease and safety of piloting of the helicopter, the takeoff sequence is automated. When the user sets SWITCH1 to HIGH, the helicopter will transition from LANDED to TAKEOFF. During takeoff, the helicopter will rise to 10% of the maximum altitude and check to see if it is in the direction of the reference yaw. If it is not, it will continue rotating until the reference yaw interrupt is triggered before entering the FLYING state, and allowing the user to have control of yaw and height, using the button peripherals, as discussed in "User Controls". If, during flight, the user sets the target altitude to 0%, or sets SWITCH1 to low, the system will enter LANDING state. During LANDING state, much like TAKEOFF, the helicopter will hover at 10% altitude, and will rotate towards the reference yaw direction, once at the reference yaw the helicopter will reduce its altitude until it reaches 0%. Once at 0%, the state will be set to LANDED, and the rotors will be turned off, by setting the PWM output state to false. A switch is used to call the functions which control each state to increase the modularity and ease of debugging of the code, as shown in listing 2.

```
if (currentState == LANDED) {
    updateLandedState();

} else if (currentState == TAKEOFF) {
    updateTakeoffState();

} else if (currentState == FLYING) {
    updateFlyingState();

} else if (currentState == LANDING) {
    updateLandingState();
}
```

Listing 2: Flight state switch

- **OLED Display and Serial Communication**

In order to get relevant information to the pilot, the Orbit BoosterPack's OLED display is used. The pilot is able to see which flight state the helicopter is in, the yaw, and the percent of total altitude the helicopter is at. Yaw is measured from 0 at the reference point to 180° clockwise, and -180° anticlockwise. When the helicopter rotates past 180° clockwise, the reading will go to -179°. In addition to using the OLED to display information, a serial link from UART0 is used to display even more relevant information such as measured altitude, target altitude, measured yaw, target yaw, duty cycle of main and tail rotors, and the flight state. A slow tick (essentially a condition which is fulfilled every 50 ticks) is used to ensure these methods of displaying information are updated frequently and at consistent intervals (This is covered more in depth in section 2).

Section 2: Task Frequency

As briefly covered in "*Section 1: OLED Display and Serial Communication*", a slow tick is used to ensure the information isn't updated too quickly. This prevents the UART information from being printed in the console so quickly that it affects readability, but conversely not so slowly that the information is outdated. To implement this slowtick, the System Tick Interrupt handler has been modified such that for every 50 ticks, the display function is called to update the information. The system tick has a rate of 100Hz, and thus the slow tick has a frequency of 2Hz (The implementation of this method can be found in listing 3, below).

The ADC interrupt must be run at frequent and consistent intervals, and as such is called for every systick interrupt. Because the altitude measurements are taken as an average of the contents of the circular buffer, the ADC interrupt must be taken at a consistent rate for the averages taken to be consistent and accurate. User input is much less frequent than most other components of the system, and as such uses polling for checking the buttons' states, as well as the switch peripherals. To keep responses to user input quick, without compromising efficiency, the buttons are polled once every time the main loop is run. Similarly, the flight states are checked in each iteration of the main loop because state changes occur relatively infrequently.
To ensure responsive control for the pilot during flight, the control functions essentially run like an ISR within the main loop. This allows states that modify the PWM signal (such as TAKEOFF, FLYING, and LANDING) to adjust in real-time without waiting for other processes while completing its tasks. This modularisation of the states also means that the PID control functions are only called when needed, improving overall efficiency. Despite this pseudo-ISR behavior, actual interrupts, such as systick and reference yaw interrupts, still take precedence to maintain consistency and accuracy for things like ADC interrupts.

```
if (++tickCount >= ticksPerSlow)
{                       // Signal a slow tick
    tickCount = 0;
    slowTick = true;
}
```

Listing 3: Slow tick implementation within the SysTickIntHandler function

Section 3: Scheduler Design

Section 4: PID Implementation

Section 5: Special Features

Section 6: Conclusion

Overall, this project can be seen as a success. The code is well modularised, and efficient, providing for easy readability, which was helpful for reflection and explanation of core design choices. The helicopter was able to maintain mostly smooth flight (except in some edge cases), and with responsive control. It was able to take off, be piloted, and then landed with ease in the correct position. Upon reflection and a more in-depth look at the program having taken a step from it, however, there are some improvements that have become apparent. First and foremost, from a planning point of view, the project specifications should have been reviewed more thoroughly, which becomes apparent when a magnifying glass is taken to the code. For example, the slow tick operates at 2Hz, rather than the specified 4Hz, and the yaw and altitude measurements are displayed in degrees and percent respectively, rather than the specified duty cycle percentages. These, however, are minor imperfections from a largely successful project overall.