Meenakshi Shankar
Sr No. 22400
M.Tech EPD

E3-257 ESD Final Assignment Report

→PART 1

3Primitive flight control system for helicopter

1. Use Two-line LCD display and LED indications to show the system status.
    a. Glow green LED when helicopter is in flight. Blink green when helicopter is ascending or descending.
    b. Blink yellow LED when obstacle is detected (< 30 cm)
    c. Print appropriate log messages on the UART terminal every time the system status changes.

Following code snippets implement the above requirements:

- **Ultrasonic Configuration:**

Echo Pin: PD2
Trig Pin: PD3

```
void HC_SR04_setup(void)
{
    SYSCTL_RCGCGPIO_R |= 0x08;      // Enable clock to GPIO PORTD
    GPIO_PORTD_DIR_R |= 0x04;       // Set PD2 as output (trigger) and PD3 as input (echo)
    GPIO_PORTD_DEN_R |= 0x0C;       // Enable bit 2 and 3 of PORTD
    GPIO_PORTD_AFSEL_R |= 0x08;      // Set alternate function WTIMER3CCP1
    GPIO_PORTD_PCTL_R &= ~0x0000F000;
    GPIO_PORTD_PCTL_R |= 0x00007000;// PORTD bit 3 selected for alternate function

    // Set up timer WT3CCP1
    SYSCTL_RCGCWTIMER_R |= 8;       // Enable clock to to timer 3
    WTIMER3_CTL_R &= ~0x0100;       // Disable timer during setup
    WTIMER3_CFG_R = 0x00000004;     // 32 bit mode
    WTIMER3_TBILR_R = (657894);     // Timer LOAD REG value = 38 ms 657894
    // Configure TIMER_B
    // bits 1:0 = [11] Capture Mode
    // bit 2    = 1    Capture Mode - Edge Time
    // bit 3    = 0    Capture Mode enabled
    // bit 4    = 0    Count down
    WTIMER3_TBMR_R = 0x07;          // Configure TIMER_B
    WTIMER3_CTL_R |= 0x0C00;        // Detect both edges
    WTIMER3_CTL_R |= 0x0100;        // Enable Timer
    WTIMER3_ICR_R = 0x0400;         // Clear event interrupt flags
    WTIMER3_IMR_R |= 0x0400;        // Unmask interrupt for capture event mode

    NVIC_EN3_R |= 0x00000020;       // NVIC enabled (IRQ number = 102)
    NVIC_PRI25_R |= 0x00600000;     // Set priority = 3
}
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

- **Concept behind distance measurement using Ultrasonic:**
  - When we want to measure the distance, we apply a 10us pulse to the trigger input.

```
void send_trigger_pulse(void)
{
    if(trigger_delay >20)
    {
    GPIO_PORTD_DATA_R &= ~0x04;
    delay_us(20);
    GPIO_PORTD_DATA_R |= 0x04;
    delay_us(10);
    GPIO_PORTD_DATA_R &= ~0x04;
    trigger_delay = 0;
    }
}
```

  - Time duration for which the output signal remains high depends on the distance between the ultrasonic sensor and the object under test.

We setup Wide timers to allow us to detect the time between the edges of this pulse that denotes the distance.

```
void WTIMER3B_Handler(void)
{
    if((WTIMER3_MIS_R & 0x0400) && (GPIO_PORTD_DATA_R & 0x08))
    {
        time_value1 = WTIMER3_TBR_R;
    }
    if((WTIMER3_MIS_R & 0x0400) && !(GPIO_PORTD_DATA_R & 0x08))
    {
        time_value2 = (WTIMER3_TBR_R);
        update = 1;
    }
    WTIMER3_ICR_R = 0x0400 ;
}
```

Rising edge and Falling edge interrupts help us capture the time difference between two occurrence.

- From this the distance is measured as follows:
- Speed of sound in air = 340m/sec

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

```c
unsigned int calculate_distance(void)
{
    unsigned int distance;
    int time;
    if(time_value1 < time_value2)
    {
        time = time_value1 + (657894) - time_value2;
    }
    else
    {
        time = time_value1 - time_value2;
    }
    time = time/(16*2);
    distance = 34000*time/1000000;
    //UARTprintf("\033[1;0H%3d", distance);
    return distance;
}
```

- **BONUS PART:**Blinking of Yellow LED when the distance goes below 30cm ;also takes care of the blinking of green LED when the helicopter ascending scenario.

```c
if(distance < 30 || asc_des)
{
   if(asc_des)
       GPIO_PORTF_DATA_R ^= 0X08;
   else
       GPIO_PORTF_DATA_R ^= 0X0A;
}
else
{
    GPIO_PORTF_DATA_R = 0x00;
}
```

- # PART 2

b. Z-axis movement controls speed of the main rotor. Minimal base speed is maintained at ground level.

   i. Adjust the tail rotor speed to counter the torque from the main rotor and keep the helicopter stable.

   ii. Increase the speed of the main rotor to generate lift, allowing the helicopter to climb altitude.

c. Pitch angle controls the thrust direction and magnitude, indicated using the tilt on the servo motor.

   i. Forward/Backward direction: by tilting the main rotor blade slightly forward or backward using the servo, thrust can be created in the desired direction, simulating forward or backward movement.

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

- **Motor Configuration:**
  The main motor, tail motor and servo motor are controlled using 3 timers setup in the same manner the PWM for all three are generated as follows:

- Timer Configuration for the Motors
  We are creating software PWM to run the motor based on timer handlers set.

```
void timerA_setup(void)    // for main motor
{
            SYSCTL_RCGCTIMER_R |= 1<<0;
            TIMER0_CTL_R &= (~(1<<0));
            TIMER0_CFG_R = 0X4;
            TIMER0_TAMR_R = 0X2;
            TIMER0_TAPR_R |= 0XFF;
            TIMER0_TAILR_R = 1570/2;
            TIMER0_ICR_R |= 1<<0;
            TIMER0_CTL_R |= 1<<0;
            TIMER0_IMR_R |= 1<<0;
            NVIC_EN0_R |= 0X80000; // INTERRUPT 19
}



void timerB_setup(void)     // for tail motor
{
        SYSCTL_RCGCTIMER_R |= 1<<1;
        TIMER1_CTL_R &= (~(1<<0));
        TIMER1_CFG_R = 0X4;
        TIMER1_TAMR_R = 0X2;
        TIMER1_TAPR_R |= 0XFF;
        TIMER1_TAILR_R = 1570/2;
        TIMER1_ICR_R |= 1<<0;
        TIMER1_CTL_R |= 1<<0;
        TIMER1_IMR_R |= 1<<0;
        NVIC_EN0_R |= 0X200000; // INTERRUPT 21

}
```

→ PWM generators for each motor

```
void Timer0_Handler(void)
{
    GPIO_PORTA_DATA_R |= 0X10;
    TIMER0_ICR_R   |= 1<<0;
    timer0_a_off;
}

void Timer1_Handler(void)
{
    GPIO_PORTA_DATA_R |= 0X08;
    TIMER1_ICR_R   |= 1<<0;
    timer1_a_off;

}
void Timer2_Handler(void)
{
    GPIO_PORTA_DATA_R |= 0X20;
    TIMER2_ICR_R   |= 1<<0;
       timer2_a_off;
}
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

→Running Motors based of the MPU data:

- Using the DCM library to get the Euler matrix to extract the pitch, yaw,roll in radians and further convert them to degrees.

```
CompDCMComputeEulers(&g_sCompDCMInst, pfEulers, pfEulers + 1,
                     pfEulers + 2);


CompDCMComputeQuaternion(&g_sCompDCMInst, pfQuaternion);

//
// convert mag data to micro-tesla for better human interpretation
//
pfMag[0] *= 1e6;
pfMag[1] *= 1e6;
pfMag[2] *= 1e6;

//
// Convert Eulers to degrees. 180/PI = 57.29...
// Convert Yaw to 0 to 360 to approximate compass headings.
//
pfEulers[0] *= 57.295779513082320876798154814105f;
pfEulers[1] *= 57.295779513082320876798154814105f;
pfEulers[2] *= 57.295779513082320876798154814105f;
if(pfEulers[2] < 0)
{
    pfEulers[2] += 360.0f;
}
```

- Similarly we also get the x,y,z accelerometer values from the mpu9150 library.

```
MPU9150DataAccelGetFloat(&g_sMPU9150Inst, pfAccel, pfAccel + 1,
                         pfAccel + 2);

//
// Get floating point version of angular velocities in rad/sec
//
MPU9150DataGyroGetFloat(&g_sMPU9150Inst, pfGyro, pfGyro + 1,
                        pfGyro + 2);

//
// Get floating point version of magnetic fields strength in tesla
//
MPU9150DataMagnetoGetFloat(&g_sMPU9150Inst, pfMag, pfMag + 1,
                           pfMag + 2);
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

```
present_x_val = pfAccel[0];
present_y_val = pfAccel[1];
present_z_val = pfAccel[2];
present_pitch = pfEulers[0];
present_yaw = pfEulers[2];
```

- Setting thresholds to determine the motor operation
  1. We rotate the main motor for the variation in z value read from accelerometer.
     The speed is adjusted based on the mapping of the difference to the timer value needed for 10ms interrupt.

```
if((present_z_val - last_z_val > 0.8 || last_z_val - present_z_val > 0.8) && (z_var == 0))
{
    if(present_z_val - last_z_val > 0.8)
    lcd_msg_1[4] = 'D';
    else if(last_z_val - present_z_val > 0.8)
    lcd_msg_1[4] = 'A';
    timer0_a_load_value = map(((present_z_val - last_z_val)*10), -30, 30, 10, 1560);
      lcd_show();
    asc_des = 1;
}
```

  2. Similarly for the Tail motor from the yaw values we are determining Left or Right

```
    asc_des = 1;
}

if((present_yaw - last_yaw > 5 || last_yaw - present_yaw > 5)&&(pitch_var == 0))
{
    if(present_yaw - last_yaw > 5)
        lcd_msg_2[4] = 'R';
    else if(last_yaw - present_yaw > 5)
    lcd_msg_2[4] = 'L';
      lcd_show();

    timer1_a_load_value = map((present_yaw), 0 , 360 , 10, 1560);
}
```

  3. For the Servo motor the counter movement to stabilize is derived from the pitch values

```
if((present_pitch - last_pitch > 10 || last_pitch - present_pitch > 10))
    {
            if(present_pitch - last_pitch>10)
                lcd_msg_1[10] = 'F';
            else if(last_pitch - present_pitch > 10)
            lcd_msg_1[10] = 'B';
              lcd_show();

        timer2_a_load_value = map((present_pitch), -90, 90, 1200, 1450);
        timer1_a_load_value = 1450;
          timer0_a_load_value = 1450;
    }
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

a. Base speed of the main rotor and tail motor can be set using potentiometer on the Edu ARM board. Tail motor rotates in anti-clockwise direction to stabilize the helicopter. Main motor rotates in clockwise direction.

→Base speed setting using adc and pot is done as below:

- Setting of ADC

```
void adc_setup(void)
{

    SYSCTL_RCGCGPIO_R |= 0x00000010;
    SYSCTL_RCGCADC_R |= 0x00000001;
    GPIO_PORTE_AFSEL_R |= 8;
    GPIO_PORTE_DEN_R &= ~8;
    GPIO_PORTE_AMSEL_R |= 8;

    // ADC0 configurations
    // safe practice to disable ADC Sam
    // Using Sample Sequencer 3 (One 12

    ADC0_ACTSS_R &= ~8;
    ADC0_EMUX_R &= ~0x0000F000;
    ADC0_SSMUX3_R = 0;
    ADC0_SSCTL3_R |= 6;
    ADC0_ACTSS_R |= 8;

}
```

- Reading Pot Value:

```
void get_adc_data(void)
{
    ADC0_PSSI_R |= 8; /* start a conversion sequence 3 */
                  while((ADC0_RIS_R & 8) == 0)
                  {
                  }
                   /* wait for conversion complete */
                  result = ADC0_SSFIFO3_R; /* read conversion result */
                  ADC0_ISC_R = 8; /* clear completion flag */
}
```

- Mapping Pot Value to Speed of Motor by assigning the Time load value.

```
        get_adc_data();
        if(result-last_result > 50 || last_result-result > 50)
        {
            timer1_a_load_value = map(result, 0, 4095, 1560, 10);
            timer0_a_load_value = map(result, 0, 4095, 1560, 10);

            last_result = result;
        }
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

→BONUS :

3. Bonus points:

    a. Modular and robust design with reliable code

```
gpio_init();
ConfigureUART();
i2c_init();
mpu_init();
LCD_Initialization();
ClearLCD();
HC_SR04_setup();
systick_setup();
timer0_a_setup();
timer1_a_setup();
timer2_a_setup();
adc_setup();
main_motor_check();
tail_motor_check();
servo_motor_check();
ClearLCD();
```

Code is written in a modular format.