```
 85 int TaskCount = 1;
 86
 87 void RendezvousBarrier(void)
 88 {
 89     switch(TaskCount)
 90     {
 91     case 1: TaskCount++;
 92             osSemaphoreWait(SynchSemaAHandle, osWaitForever); // Wait Semaphore SynchSemaA
 93             break;
 94     case 2: TaskCount++;
 95             osSemaphoreWait(SynchSemaBHandle, osWaitForever); // Wait Semaphore SynchSemaB
 96             break;
 97     case 3: osSemaphoreRelease(SynchSemaAHandle); // Release Semaphore SynchSemaA
 98             osSemaphoreRelease(SynchSemaBHandle); // Release Semaphore SynchSemaB
 99             TaskCount = 1; // Reset to 1
100             break;
101     }
102     return;
103 }
104 /* USER CODE END 0 */
169   /* USER CODE BEGIN RTOS_QUEUES */
170   /* add queues, ... */
171   osSemaphoreWait(SynchSemaAHandle, 0);
172   osSemaphoreWait(SynchSemaBHandle, 0);
173  // osSemaphoreWait(SynchSemaCHandle, 0);
540 void StartRedTask(void const * argument)
541 {
542   /* USER CODE BEGIN StartRedTask */
543   /* Infinite loop */
544   for(;;)
545   {
546         flashLed(RedLed_Pin, 20, 10);//20sec 10Hz
547         RendezvousBarrier();
548         flashLed(RedLed_Pin, 8, 1);//8sec 1Hz
549         RendezvousBarrier();
550         flashLed(RedLed_Pin, 20, 5);//20sec 5Hz
551         RendezvousBarrier();
552
553   }
554   /* USER CODE END StartRedTask */
555 }
```

```
564 void StartGreenTask(void const * argument)
565 {
566   /* USER CODE BEGIN StartGreenTask */
567   /* Infinite loop */
568   for(;;)
569   {
570         flashLed(GreenLed_Pin, 12, 10);//12sec 10Hz
571         RendezvousBarrier();
572         flashLed(GreenLed_Pin, 20, 1);//20sec 1Hz
573         RendezvousBarrier();
574         flashLed(GreenLed_Pin, 8, 5);//8sec 5Hz
575         RendezvousBarrier();
576
577   }
578   /* USER CODE END StartGreenTask */
579 }
588 void StartBlueTask(void const * argument)
589 {
590   /* USER CODE BEGIN StartBlueTask */
591   /* Infinite loop */
592   for(;;)
593   {
594         flashLed(BlueLed_Pin, 8, 10);//8sec 10Hz
595         RendezvousBarrier();
596         flashLed(BlueLed_Pin, 12, 1);//12sec 1Hz
597         RendezvousBarrier();
598         flashLed(BlueLed_Pin, 12, 5);//12sec 5Hz
599         RendezvousBarrier();
600
601   }
602   /* USER CODE END StartBlueTask */
603 }
```

18)

```c
void Start_GREEN_LED(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
    InitSemaphores();
  for(;;)
  {
     // Checks GreenTaskFlashRate
     int x = GetGreenTaskFlashRate();
     if (x == a)
     {
     // Flash GREEN LED
        for (int i=0; i<100; i++) // cycles = 10Hz * 10sec = 100cycles
        {
            HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_SET);
            osDelay(50); // ~50 mSec - T/2
            HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_RESET);
            osDelay(50); // ~50 mSec - T/2
        }
     }
        else if (x == b) {
        // Flash GREEN LED
        for (int i=0; i<10; i++) // cycles = 1Hz * 10sec = 10cycles
        {
            HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_SET);
            osDelay(500); // ~0.5 mSec - T/2
            HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_RESET);
            osDelay(500); // ~0.5 mSec - T/2
        }
    }

   }
  /* USER CODE END 5 */
}
```

```c
void Start_RED_LED(void const * argument)
{
  /* USER CODE BEGIN Start_RED_LED */
  /* Infinite loop */
  for(;;)
  {
    // Checks RedTaskFlashRate
    int x = GetRedTaskFlashRate();
    if (x == b) {
        // Flash RED LED
        for (int i=0; i<6; i++) // cycles = 1Hz * 6sec = 6cycles
        {
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_SET);
            osDelay(500); // ~0.5 mSec - T/2
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_RESET);
            osDelay(500); // ~0.5 mSec - T/2
        }
    }
        else if (x == a){
        // Flash RED LED
        for (int i=0; i<60; i++) // cycles = 10Hz * 6sec = 60cycles
        {
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_SET);
            osDelay(50); // ~50 mSec - T/2
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_RESET);
            osDelay(50); // ~50 mSec - T/2
        }
        }
    }
```

```c
void Start_Flashing_Rate(void const * argument)
{
  /* USER CODE BEGIN Start_Flashing_Rate */
  /* Infinite loop */
  for(;;)
  {
   osDelay(8000); // 8sec delay
   SetGreenTaskFlashRate(b); // Set GreenTaskFlashRate to 1 Hz
   SetRedTaskFlashRate(a);   // Set RedTaskFlashRate to 10 Hz
   osDelay(8000); // 8sec delay
   SetGreenTaskFlashRate(a); // Set GreenTaskFlashRate to 10 Hz
   SetRedTaskFlashRate(b);   // Set RedTaskFlashRate to 1 Hz
  }
  /* USER CODE END Start_Flashing_Rate */
}
```

19)

```
void StartSensorMonitoringTask(void const * argument)
{
  /* USER CODE BEGIN 5 */
    int SensorState;
    osStatus SensorSendState;
    int Push_Button;
    typedef enum {not_pressed_no_alarm, not_pressed_alarm, pressed_no_alarm, pressed_alarm} state_t;
    state_t state = not_pressed_no_alarm; // default state
  /* Infinite loop */
  for(;;)
  {
    Push_Button = HAL_GPIO_ReadPin(GPIOA, PB);
    if (state == not_pressed_no_alarm) {
        if(Push_Button == 1){ // verify
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_SET);     // Button is pressed
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_RESET); // Alarm is off
            SensorSendState = osMessagePut(Q2SAHandle, 1, 0);
            state = pressed_no_alarm;
        }
    }
    else if (state == pressed_no_alarm){
        if(Push_Button == 0){ // verify
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_RESET); // Button is NOT pressed
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_SET); // Alarm is on
            SensorSendState = osMessagePut(Q2SAHandle, 1, 0);
            state = not_pressed_alarm;
        }
    }
    else if (state == not_pressed_alarm){
        if(Push_Button == 1){ // verify
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_SET);   // Button is pressed
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_SET); // Alarm is on
            SensorSendState = osMessagePut(Q2SAHandle, 0, 0);
            state = pressed_alarm;
        }
    }
    else if (state == not_pressed_alarm){
        if(Push_Button == 1){ // verify
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_SET);   // Button is pressed
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_SET); // Alarm is on
            SensorSendState = osMessagePut(Q2SAHandle, 0, 0);
            state = pressed_alarm;
        }
    }
    else if (state == pressed_alarm){
        if(Push_Button == 0){ // verify
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_RESET);   // Button is NOT pressed
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_RESET); // Alarm is off
            SensorSendState = osMessagePut(Q2SAHandle, 0, 0);
            state = not_pressed_no_alarm;
        }
    }
  }
    osDelay(250); // Delay 250 mSec
  }
```

```
void StartAlarmingTask(void const * argument)
{
  /* USER CODE BEGIN StartAlarmingTask */
    osEvent QueReadState;
    int SensorState;
  /* Infinite loop */
  for(;;)
  {
    QueReadState = osMessageGet(Q2SAHandle, 0);
    if (QueReadState.status == osEventMessage)
    {
        SensorState = QueReadState.value.v;
        if (SensorState == 1)
        {
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_SET);
        }
        else if (SensorState == 0)
        {
            HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_RESET);
        }
    }
    osDelay(500); // Delay 500 mSec
  }
  /* USER CODE END StartAlarmingTask */
}
```

20)

```
void StartCommand(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      // Flash RED LED
        for (int i=0; i<=10; i++) // cycles = 1Hz * 10sec = 10cycles; T = 1/1 = 1; 1/2 = 0.5 = 500mSec
              {
                  HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_SET);
                  osDelay(500); // ~0.5 mSec - T/2
                  HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_RESET);
                  osDelay(500); // ~0.5 mSec - T/2
              }
       // Post command to the mailbox
          Post(SYNC1);
       // Flash RED LED
           for (int i=0; i<=50; i++) // cycles = 10Hz * 5sec = 50cycles; T = 1/10 = 0.1; 0.1/2 = 0.05 = 50mSec
               {
                   HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_SET);
                   osDelay(50); // ~0.05 mSec - T/2
                   HAL_GPIO_WritePin(GPIOD, RED_LED, GPIO_PIN_RESET);
                   osDelay(50); // ~0.05 mSec - T/2
               }

        // Post command to the the mailbox
          Post(SYNC2);
  }
  /* USER CODE END 5 */
}
```

```
void Start_LED_FLASHING_TASK(void const * argument)
{
  /* USER CODE BEGIN Start_LED_FLASHING_TASK */
  /* Infinite loop */
  for(;;)
  {
      // Flash GREEN LED
         for (int i=0; i<=50; i++) // cycles = 10Hz * 5sec = 50cycles; T = 1/10 = 0.1; 0.1/2 = 0.05 = 50mSec
                {
                    HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_SET);
                    osDelay(50); // ~0.05 mSec - T/2
                    HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_RESET);
                    osDelay(50); // ~0.05 mSec - T/2
                }
     // Pend on mailbox
        temp = Pend();
        // Flash GREEN LED
           for (int i=0; i<=10; i++) // cycles = 1Hz * 10sec = 100cycles; T = 1/1 = 1; 1/2 = 0.5 = 500mSec
                {
                    HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_SET);
                    osDelay(500); // ~0.5 mSec - T/2
                    HAL_GPIO_WritePin(GPIOD, GREEN_LED, GPIO_PIN_RESET);
                    osDelay(500); // ~0.5 mSec - T/2
                }
    // Pend on mailbox
       temp = Pend();
  }
  /* USER CODE END Start_LED_FLASHING_TASK */
}

void Start_ActOnMessage(void const * argument)
{
  /* USER CODE BEGIN Start_ActOnMessage */
  /* Infinite loop */
  for(;;)
  {
      // Act on message
      if(temp == SYNC1){
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_SET); // ORANGE LED is on
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_RESET); // BLUE LED is off
        }
         else if (temp == SYNC2){
            HAL_GPIO_WritePin(GPIOD, ORANGE_LED, GPIO_PIN_RESET); // ORANGE LED is off
            HAL_GPIO_WritePin(GPIOD, BLUE_LED, GPIO_PIN_SET);     // BLUE LED is on
        }
  }
  /* USER CODE END Start_ActOnMessage */
}
```