

Exercise 9:

```

439 /* USER CODE BEGIN 4 */
440 int startFlag = 1;
441 void StartAccessSoftware()
442 {
443     /* USER CODE BEGIN 5 */
444     /* Infinite loop */
445     for(;;)
446     {
447         if(startFlag == 1)
448         {
449             startFlag = 0;
450         }
451         else
452         {
453             HAL_GPIO_TogglePin(GPIOD, LD6_Pin); //Toggle Blue LED
454         }
455         for(int i = 0; i < 2000000; i++); //Delay ~0.5 sec
456         startFlag = 1;
457         return;
458     }
459     /* USER CODE END 5 */
460 }

470 void StartGreenTask(void const * argument)
471 {
472     /* init code for USB_HOST */
473     MX_USB_HOST_Init();
474     /* USER CODE BEGIN 5 */
475     /* Infinite loop */
476     for(;;)
477     {
478         HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET); //Green LED ON
479
480         osMutexWait(CriticalResourceMutexHandle, osWaitForever); //wait until mutex available
481         StartAccessSoftware(); //Enter simulation of R/W
482         osMutexRelease(CriticalResourceMutexHandle); //once done release it so other tasks can use it
483
484         osDelay(200); //delay 0.2 sec
485         HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET); //Green LED OFF
486
487         osDelay(200); //delay 0.2 sec
488     }
489 }

499 void StartRedTask(void const * argument)
500 {
501     /* USER CODE BEGIN StartRedTask */
502     /* Infinite loop */
503     for(;;)
504     {
505         HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET); //RED LED ON
506
507         osMutexWait(CriticalResourceMutexHandle, osWaitForever); //wait until mutex available
508         StartAccessSoftware(); //Enter simulation of R/W
509         osMutexRelease(CriticalResourceMutexHandle); //once done release it so other tasks can use it
510
511         osDelay(550);
512         HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET); //RED LED OFF
513
514         osDelay(550); //delay 0.55 sec
515     }
516     /* USER CODE END StartRedTask */
517 }

```

Exercise 10:

```

446 int startFlag = 1;
447 void StartAccessSoftware()
448 {
449     /* USER CODE BEGIN 5 */
450     /* Infinite loop */
451     osSemaphoreWait(CriticalSemaphoreHandle, osWaitForever); //wait until semaphore available
452     if(startFlag == 1)
453     {
454         startFlag = 0;
455     }
456     else
457     {
458         HAL_GPIO_TogglePin(GPIOD, LD6_Pin); //Toggle Blue LED
459     }
460     for(int i = 0; i < 2000000; i++); //Delay ~0.5 sec
461     startFlag = 1;
462     osSemaphoreRelease(CriticalSemaphoreHandle); //once done release it so other tasks can use it
463     return;
464     /* USER CODE END 5 */
465 }
466
467 void StartGreenTask(void const * argument)
468 {
469     for(;;)
470     {
471         HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET); //Green LED ON
472
473         StartAccessSoftware(); //Enter simulation of R/W
474
475         osDelay(200); //delay 0.2 sec
476         HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET); //Green LED OFF
477
478         osDelay(200); //delay 0.2 sec
479     }
480     /* USER CODE END 5 */
481 }
482
483 void StartRedTask(void const * argument)
484 {
485     /* USER CODE BEGIN StartRedTask */
486     /* Infinite loop */
487     for(;;)
488     {
489         HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET); //RED LED ON
490
491         //osMutexWait(CriticalResourceMutexHandle, osWaitForever); //wait until mutex available
492         StartAccessSoftware(); //Enter simulation of R/W
493         //osMutexRelease(CriticalResourceMutexHandle); //once done release it so other tasks can use it
494
495         osDelay(550);
496         HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET); //RED LED OFF
497
498         osDelay(550); //delay 0.55 sec
499     }
500     /* USER CODE END StartRedTask */
501 }

```

Exercise 11.1)

```

459 void StartAccessSoftware()
460 {
461     int startFlag = 0;
462     osSemaphoreWait(CriticalSemaphoreHandle, osWaitForever); //wait until semaphore available
463     if(startFlag == 1)
464     {
465         startFlag = 0;
466     }
467     else
468     {
469         for(int i = 0; i <= 20; i++)
470         {
471             HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_SET);
472             for(int j = 0; j < 375000; j++);
473             HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_RESET);
474             for(int j = 0; j < 375000; j++);
475         }
476         //HAL_GPIO_TogglePin(GPIOD, LD6_Pin); //Toggle Blue LED
477     }
478     //for(int i = 0; i < 2000000; i++); //Delay ~0.5 sec
479     startFlag = 1;
480     osSemaphoreRelease(CriticalSemaphoreHandle); //once done release it so other tasks can use it
481     return;
482 }
483 }

511 void StartGreenTask(void const * argument)
512 {
513     /* USER CODE BEGIN StartGreenTask */
514     /* Infinite loop */
515     for(;;)
516     {
517         StartAccessSoftware();
518         for(int i = 0; i <= 40; i++)
519         {
520             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET); //Green LED ON
521             for(int j = 0; j < 375000; j++);
522             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET); //Green LED ON
523             for(int j = 0; j < 375000; j++);
524         }
525         vTaskSuspend(NULL);
526     }
527     /* USER CODE END StartGreenTask */
528 }

537 void StartRedTask(void const * argument)
538 {
539     /* USER CODE BEGIN StartRedTask */
540     /* Infinite loop */
541     for(;;)
542     {
543         StartAccessSoftware();
544         for(int i = 0; i <= 40; i++)
545         {
546             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);
547             for(int j = 0; j < 375000; j++);
548             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
549             for(int j = 0; j < 375000; j++);
550         }
551         vTaskSuspend(NULL);
552     }
553     /* USER CODE END StartRedTask */
554 }

563 void StartOrangeTask(void const * argument)
564 {
565     /* USER CODE BEGIN StartOrangeTask */
566     /* Infinite loop */
567     for(;;)
568     {
569         for(int i = 0; i <= 40; i++)
570         {
571             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_SET);
572             for(int j = 0; j < 375000; j++);
573             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
574             for(int j = 0; j < 375000; j++);
575         }
576         vTaskSuspend(NULL);
577     }
578     /* USER CODE END StartOrangeTask */
579 }

```

11.2)

```

511 void StartGreenTask(void const * argument)
512 {
513     /* USER CODE BEGIN StartGreenTask */
514     /* Infinite loop */
515     for(;;)
516     {
517         StartAccessSoftware();
518         for(int i = 0; i <= 40; i++)
519         {
520             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET); //Green LED ON
521             for(int j = 0; j < 375000; j++);
522             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET); //Green LED ON
523             for(int j = 0; j < 375000; j++);
524         }
525         vTaskSuspend(NULL);
526     }
527     /* USER CODE END StartGreenTask */
528 }
529
530 void StartRedTask(void const * argument)
531 {
532     /* USER CODE BEGIN StartRedTask */
533     /* Infinite loop */
534     for(;;)
535     {
536         osDelay(1000);
537         StartAccessSoftware();
538         for(int i = 0; i <= 40; i++)
539         {
540             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);
541             for(int j = 0; j < 375000; j++);
542             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
543             for(int j = 0; j < 375000; j++);
544         }
545         vTaskSuspend(NULL);
546     }
547     /* USER CODE END StartRedTask */
548 }

```

11.3)

```
537 void StartRedTask(void const * argument)
538 {
539     /* USER CODE BEGIN StartRedTask */
540     /* Infinite loop */
541     for(;;)
542     {
543         osDelay(1000);
544         StartAccessSoftware();
545         for(int i = 0; i <= 40; i++)
546         {
547             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);
548             for(int j = 0; j<375000;j++);
549             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
550             for(int j = 0; j<375000;j++);
551         }
552         vTaskSuspend(NULL);
553     }
554     /* USER CODE END StartRedTask */
555 }
556
564 void StartOrangeTask(void const * argument)
565 {
566     /* USER CODE BEGIN StartOrangeTask */
567     /* Infinite loop */
568
569     for(;;)
570     {
571         osDelay(1000);
572         for(int i = 0; i <= 40; i++)
573         {
574             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_SET);
575             for(int j = 0; j<375000;j++);
576             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
577             for(int j = 0; j<375000;j++);
578         }
579         vTaskSuspend(NULL);
580     }
581     /* USER CODE END StartOrangeTask */
582 }
```

Exercise 12)

```

465 void StartAccessSoftware()
466 {
467     int startFlag = 0;
468     //osSemaphoreWait(CriticalSemaphoreHandle, osWaitForever); //wait until semaphore available
469     osMutexWait(CriticalMutexResourcesHandle, osWaitForever);
470     if(startFlag == 1)
471     {
472         startFlag = 0;
473     }
474     else
475     {
476         for(int i = 0; i <= 20; i++)
477         {
478             HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_SET);
479             for(int j = 0; j < 375000; j++);
480             HAL_GPIO_WritePin(GPIOD, LD6_Pin, GPIO_PIN_RESET);
481             for(int j = 0; j < 375000; j++);
482         }
483         //HAL_GPIO_TogglePin(GPIOD, LD6_Pin); //Toggle Blue LED
484     }
485     //for(int i = 0; i < 2000000; i++); //Delay ~0.5 sec
486     startFlag = 1;
487     //osSemaphoreRelease(CriticalSemaphoreHandle); //once done release it so other tasks can use it
488     osMutexRelease(CriticalMutexResourcesHandle);
489     return;
490 }
491 }

---
519 void StartGreenTask(void const * argument)
520 {
521     /* USER CODE BEGIN StartGreenTask */
522     /* Infinite loop */
523     for(;;)
524     {
525         StartAccessSoftware();
526         for(int i = 0; i <= 40; i++)
527         {
528             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_SET); //Green LED ON
529             for(int j = 0; j < 375000; j++);
530             HAL_GPIO_WritePin(GPIOD, LD4_Pin, GPIO_PIN_RESET); //Green LED OFF
531             for(int j = 0; j < 375000; j++);
532         }
533         vTaskSuspend(NULL);
534     }
535     /* USER CODE END StartGreenTask */
536 }

```

```
545 void StartRedTask(void const * argument)
546 {
547     /* USER CODE BEGIN StartRedTask */
548     /* Infinite loop */
549     for(;;)
550     {
551         osDelay(1000);
552         StartAccessSoftware();
553         for(int i = 0; i <= 40; i++)
554         {
555             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_SET);
556             for(int j = 0; j<375000;j++);
557             HAL_GPIO_WritePin(GPIOD, LD5_Pin, GPIO_PIN_RESET);
558             for(int j = 0; j<375000;j++);
559         }
560         vTaskSuspend(NULL);
561     }
562     /* USER CODE END StartRedTask */
563 }
564
565 /* USER CODE BEGIN Header_StartOrangeTask */
566 /**
567  * @brief Function implementing the OrangeTask thread.
568  * @param argument: Not used
569  * @retval None
570  */
571 /* USER CODE END Header_StartOrangeTask */
572 void StartOrangeTask(void const * argument)
573 {
574     /* USER CODE BEGIN StartOrangeTask */
575     /* Infinite loop */
576
577     for(;;)
578     {
579         osDelay(1000);
580         for(int i = 0; i <= 40; i++)
581         {
582             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_SET);
583             for(int j = 0; j<375000;j++);
584             HAL_GPIO_WritePin(GPIOD, LD3_Pin, GPIO_PIN_RESET);
585             for(int j = 0; j<375000;j++);
586         }
587         vTaskSuspend(NULL);
588     }
589     /* USER CODE END StartOrangeTask */
590 }
```