/\* Includes ------------------------------------------------------------------\*/

**#include** "main.h"

/\* Private includes ----------------------------------------------------------\*/

**#include** "FreeRTOS.h"

**#include** "task.h"

**#include** "queue.h"

**#include** "semphr.h"

**#include** "stm32f3xx.h"

/\* Private variables ---------------------------------------------------------\*/

**UART\_HandleTypeDef** huart2;

**#define** **FIRE\_ALARM\_PRIORITY** 3

**#define** **SECURITY\_BREACH\_PRIORITY** 2

**#define** **LOW\_BATTERY\_PRIORITY** 1

**QueueHandle\_t** alarmQueue;

**typedef** **enum** {

*FIRE\_ALARM*,

*SECURITY\_BREACH*,

*LOW\_BATTERY*

} **AlarmType**;

// Structure for Alarm Data

**typedef** **struct** {

**AlarmType** type;

**uint32\_t** timestamp;

} **AlarmEvent**;

/\* Function Prototypes \*/

**void** **vAlarmTask**(**void** \*pvParameters);

**void** **vHandleFireAlarm**(**void**);

**void** **vHandleSecurityBreach**(**void**);

**void** **vHandleLowBattery**(**void**);

**void** **EXTI0\_IRQHandler**(**void**);

**void** **EXTI1\_IRQHandler**(**void**);

**void** **EXTI2\_IRQHandler**(**void**);

**void** **SystemClock\_Config**(**void**);

**void** **MX\_GPIO\_Init**(**void**);

**void** **MX\_USART2\_UART\_Init**(**void**);

/\* Main Function \*/

**int** **main**(**void**)

{

**HAL\_Init**();

**SystemClock\_Config**();

**MX\_GPIO\_Init**();

**MX\_USART2\_UART\_Init**();

alarmQueue = xQueueCreate(5, **sizeof**(**AlarmEvent**));

// Create Tasks

**xTaskCreate**(**vAlarmTask**, "AlarmTask", 256, NULL, 2, NULL);

// Start Scheduler

**vTaskStartScheduler**();

**while** (1)

{

// Main loop - Should never reach here if FreeRTOS is running correctly

}

}

/\* GPIO Initialization \*/

**void** **MX\_GPIO\_Init**(**void**)

{

**GPIO\_InitTypeDef** GPIO\_InitStruct = {0};

/\* Enable GPIO Clocks \*/

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\* Configure GPIO pin : B1\_Pin \*/

GPIO\_InitStruct.Pin = B1\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_FALLING;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

**HAL\_GPIO\_Init**(B1\_GPIO\_Port, &GPIO\_InitStruct);

/\* Configure GPIO pin : LD2\_Pin \*/

GPIO\_InitStruct.Pin = LD2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

**HAL\_GPIO\_Init**(LD2\_GPIO\_Port, &GPIO\_InitStruct);

}

/\* USART2 Initialization \*/

**void** **MX\_USART2\_UART\_Init**(**void**)

{

huart2.Instance = USART2;

huart2.Init.BaudRate = 38400;

huart2.Init.WordLength = UART\_WORDLENGTH\_8B;

huart2.Init.StopBits = UART\_STOPBITS\_1;

huart2.Init.Parity = UART\_PARITY\_NONE;

huart2.Init.Mode = UART\_MODE\_TX\_RX;

huart2.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart2.Init.OverSampling = UART\_OVERSAMPLING\_16;

huart2.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart2.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

**if** (**HAL\_UART\_Init**(&huart2) != *HAL\_OK*)

{

**Error\_Handler**();

}

}

/\* Task for Alarm Processing \*/

**void** **vAlarmTask**(**void** \*pvParameters)

{

**AlarmEvent** receivedAlarm;

**while**(1)

{

**if** (**xQueueReceive**(alarmQueue, &receivedAlarm, portMAX\_DELAY) == pdPASS)

{

**switch**(receivedAlarm.type)

{

**case** *FIRE\_ALARM*:

**vHandleFireAlarm**();

**break**;

**case** *SECURITY\_BREACH*:

**vHandleSecurityBreach**();

**break**;

**case** *LOW\_BATTERY*:

**vHandleLowBattery**();

**break**;

}

}

}

}

/\* ISR for Fire Alarm \*/

**void** **EXTI0\_IRQHandler**(**void**)

{

**AlarmEvent** fireAlarm = {*FIRE\_ALARM*, **xTaskGetTickCount**()};

xQueueSendFromISR(alarmQueue, &fireAlarm, NULL);

EXTI->PR |= (1 << 0); // Clear interrupt flag

}

/\* ISR for Security Breach \*/

**void** **EXTI1\_IRQHandler**(**void**)

{

**AlarmEvent** securityBreach = {*SECURITY\_BREACH*, **xTaskGetTickCount**()};

xQueueSendFromISR(alarmQueue, &securityBreach, NULL);

EXTI->PR |= (1 << 1); // Clear interrupt flag

}

/\* ISR for Low Battery \*/

**void** **EXTI2\_IRQHandler**(**void**)

{

**AlarmEvent** lowBattery = {*LOW\_BATTERY*, **xTaskGetTickCount**()};

xQueueSendFromISR(alarmQueue, &lowBattery, NULL);

EXTI->PR |= (1 << 2); // Clear interrupt flag

}

/\* Alarm Handlers \*/

**void** **vHandleFireAlarm**(**void**)

{

// Code to handle Fire Alarm

}

**void** **vHandleSecurityBreach**(**void**)

{

// Code to handle Security Breach

}

**void** **vHandleLowBattery**(**void**)

{

// Code to handle Low Battery Warning

}

/\* Timer Interrupt Callback \*/

**void** **HAL\_TIM\_PeriodElapsedCallback**(**TIM\_HandleTypeDef** \*htim)

{

**if** (htim->Instance == TIM3)

{

**HAL\_IncTick**();

}

}

/\* Error Handler \*/

**void** **Error\_Handler**(**void**)

{

**\_\_disable\_irq**();

**while** (1) {}

}

**void** **SystemClock\_Config**(**void**) {

**RCC\_OscInitTypeDef** RCC\_OscInitStruct = {0};

**RCC\_ClkInitTypeDef** RCC\_ClkInitStruct = {0};

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI;

RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL16;

**if** (**HAL\_RCC\_OscConfig**(&RCC\_OscInitStruct) != *HAL\_OK*) {

**Error\_Handler**();

}

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

**if** (**HAL\_RCC\_ClockConfig**(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != *HAL\_OK*) {

**Error\_Handler**();

}

}