```c
/*
 * ETML-ES
 *
 * 2228_AlarmeFenetreOuverte
 *
 * Main NRF Emetteur
 *
 * Miguel Santos
 * 2023
 */

#include <zephyr/kernel.h>
#include <zephyr/drivers/gpio.h>


/* Main loop sleeping time in ms */
#define SLEEP_TIME_MS   100000

/* Devicetree nodes identifiers */
#define LED0_NODE DT_ALIAS(led0)
#define LED1_NODE DT_ALIAS(led1)
#define LED2_NODE DT_ALIAS(led2)
#define LED3_NODE DT_ALIAS(led3)

#define SW0_NODE DT_ALIAS(sw0)

#define VBAT_OK_NODE DT_ALIAS(vbat_ok)
#define MAG_OUT_NODE DT_ALIAS(mag_out)
#define MAG_EN_NODE DT_ALIAS(mag_en)


/* GPIO specifications */
static const struct gpio_dt_spec led0 = GPIO_DT_SPEC_GET(LED0_NODE, gpios);
static const struct gpio_dt_spec led1 = GPIO_DT_SPEC_GET(LED1_NODE, gpios);
static const struct gpio_dt_spec led2 = GPIO_DT_SPEC_GET(LED2_NODE, gpios);
static const struct gpio_dt_spec led3 = GPIO_DT_SPEC_GET(LED3_NODE, gpios);

static const struct gpio_dt_spec btn0 = GPIO_DT_SPEC_GET(SW0_NODE, gpios);

static const struct gpio_dt_spec batteryOk = GPIO_DT_SPEC_GET(VBAT_OK_NODE, gpios);
static const struct gpio_dt_spec sensorPin = GPIO_DT_SPEC_GET(MAG_OUT_NODE, gpios);
static const struct gpio_dt_spec sensorEnable = GPIO_DT_SPEC_GET(MAG_EN_NODE, gpios);


/* Define a variable of type static struct gpio_callback */
static struct gpio_callback btn0_cb_data;
static struct gpio_callback sensor_cb_data;


/* Check if devices are available
 * if not, programm should be stop */
bool check_devices()
{
    bool device_error = false;

    if (!device_is_ready(led0.port)) {
        device_error = true;
    }

    if (!device_is_ready(led1.port)) {
        device_error = true;
    }

    if (!device_is_ready(led2.port)) {
        device_error = true;
    }

    if (!device_is_ready(led3.port)) {
        device_error = true;
    }

    if (!device_is_ready(btn0.port)) {
        device_error = true;
    }

    if (!device_is_ready(batteryOk.port)) {
        device_error = true;
    }

    if (!device_is_ready(sensorPin.port)) {
        device_error = true;
    }

    if (!device_is_ready(sensorEnable.port)) {
        device_error = true;
    }

    return device_error;
}

/* Configure pins to input or outputs */
bool configure_devices()
{
    bool device_error = false;
    int device_return;

    device_return = gpio_pin_configure_dt(&led0, GPIO_OUTPUT_INACTIVE);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&led1, GPIO_OUTPUT_INACTIVE);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&led2, GPIO_OUTPUT_INACTIVE);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&led3, GPIO_OUTPUT_INACTIVE);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&btn0, GPIO_INPUT);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&batteryOk, GPIO_INPUT);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&sensorPin, GPIO_INPUT);
    if (device_return < 0) {
        device_error = true;
    }

    device_return = gpio_pin_configure_dt(&sensorEnable, GPIO_OUTPUT_INACTIVE);
    if (device_return < 0) {
        device_error = true;
    }

    return device_error;
}

void ISR_btn0(const struct device *dev, struct gpio_callback *cb, uint32_t pins)
{
    if(gpio_pin_get_dt(&btn0))
    {
        gpio_pin_set_dt(&led0, true);
    }
    else
    {
        gpio_pin_set_dt(&led0, false);
    }
}

void ISR_sensor(const struct device *dev, struct gpio_callback *cb, uint32_t pins)
{
    if(gpio_pin_get_dt(&sensorPin))
    {
        gpio_pin_set_dt(&led3, false);
    }
    else
    {
        gpio_pin_set_dt(&led3, true);
    }
}

bool ISR_btn0_configure()
{
    bool int_return;

    int_return = gpio_pin_interrupt_configure_dt(&btn0, GPIO_INT_EDGE_BOTH);

    /* Initialize the static struct gpio_callback variable   */
    gpio_init_callback(&btn0_cb_data, ISR_btn0, BIT(btn0.pin));

    /* Add the callback function by calling gpio_add_callback()   */
    gpio_add_callback(btn0.port, &btn0_cb_data);

    return int_return;
}

bool ISR_sensor_configure()
{
    bool int_return;

    int_return = gpio_pin_interrupt_configure_dt(&sensorPin, GPIO_INT_EDGE_BOTH);

    /* Initialize the static struct gpio_callback variable   */
    gpio_init_callback(&sensor_cb_data, ISR_sensor, BIT(sensorPin.pin));

    /* Add the callback function by calling gpio_add_callback()   */
    gpio_add_callback(sensorPin.port, &sensor_cb_data);

    return int_return;
}


void main(void)
{
    //int gpio_return;
    bool error_return;

    error_return = check_devices();
    if(error_return){
        return;
    }

    error_return = configure_devices();
    if(error_return){
        return;
    }

    ISR_btn0_configure();
    ISR_sensor_configure();

    while (true) {
        k_msleep(SLEEP_TIME_MS);
    }
}
```