Assignments week 2 – Super Loop Construct with an ISR

2.1 Create a copy of the project based on `CC3220S_leeg_project`, which you used to implement Assignment 1.9 last week, within CSS and rename it to `Assignment2.1`. Read the datasheet to figure out how to use the system timer of the ARM Cortex-M4 processor called SysTick. Configure the SysTick interrupt to run at a frequency of 1 kHz. Make use of the macro `HWREG()` to configure the SysTick registers. You can define and install an ISR (Interrupt Service Routine) in the file `cc3220s_startup_ccs.c`. Make sure the ISR is properly called using the debugger.

2.2 Now use this ISR to replace the functionality of the delay function. You can use a (**`static unsigned int`**) software counter to wait a second and set a **`volatile`** global flag variable to tell the main loop to continue.

2.3 The SysTick can also be configured by using the Driver Library, following these steps:

- Import the example project you can find in: `C:\ti\simplelink_cc32xx_sdk_x_xx_xx_xx\examples\nortos\CC3220S_LAUNCHXL\drivers\gpiointerrupt\ccs`.
- Right-click on the project name and choose *Rename...*. Rename the project to `leds_CC3220S_LAUNCHXL_nortos_ccs`.
- Rename `gpiointerrupt.c` to `leds.c`.
- Rename `gpiointerrupt.syscfg` to `leds.syscfg`.
- Choose Project > Properties and select (if needed) *CCS General*. Push the button Manage Configurations... and activate the *Debug configuration*.

Next, delete the *MCU+Image configuration*[1]. Finally press OK and Aply and Close.

- Remove `main_nortos.c`, `Board.html`, `image.syscfg`, `MCU+Image`, `README.html`, and `README.md` by right clicking the file and by choosing *Delete*.
- Add support for the yellow led by using the System Configuration tool you used in assignment Assignment 1.10. Name this led `CONFIG_GPIO_LED_2`.
- Remove support for the two buttons by using the System Configuration tool.
- Please note that the System Configuration tool can *not* be used to configure the SysTick.
- Replace the code in `led.c` by the code given in Listing 1.
- Compile and run this program.
- See Systick_api to learn how you can configure the SysTick by using the Driver Library. To use this API you must include `<ti/devices/cc32xx/driverlib/systick.h>`. Now implement the same behavior as Assignments 2.1 and 2.2 by using this API and the led drivers.

2.4 The Cortex-M4 processor can be put in a low power mode by executing the `WFI` assembler instruction. The core resumes execution when it receives an interrupt. Configure the processor to sleep while a led combination is showing, until the next SysTick interrupt. To execute an assembly instruction, you can use the macro:

```
__asm("     ");
```

Make sure to include some space before the instruction itself. E.g.:

```
__asm("     nop");
```

[1] The MCU+Image configuration is needed if we want load our program the Flash memory. In this case we want to load the program into the RAM to debug it (using the Debug configuration).

```c
/*
 * Copyright (c) 2020, Rotterdam University of Applied ↩
 ↪ Sciences
 * All rights reserved.
 */

#include <stdint.h>
#include <stddef.h>

/* Driver Header files */
#include <NoRTOS.h>
#include <ti/drivers/GPIO.h>

/* Driver configuration */
#include "ti_drivers_config.h"

int main(void)
{
    Board_init();
    NoRTOS_start();

    /* Call driver init functions */
    GPIO_init();
    /* Turn on red led */
    GPIO_write(CONFIG_GPIO_LED_0, CONFIG_GPIO_LED_ON);
    /* Turn off green and yellow led */
    GPIO_write(CONFIG_GPIO_LED_1, CONFIG_GPIO_LED_OFF);
    GPIO_write(CONFIG_GPIO_LED_2, CONFIG_GPIO_LED_OFF);

    while (1) {} /* Wait forever */
}
```

Listing 1: Code to drive the leds, see leds.c

2.5 The low power mode can also be entered by using the Driver Library. See PRCM_Power_Reset_Clock_Module_api. Now repeat Assignment 2.4 by using this API.

2.6 Now create a rotation loop which simulates a simple traffic light: Red (5 seconds), Yellow (1 second), Green (4 seconds). Make use of an enumeration construct (`enum`) for the colors and a `switch-case`-statement for the rotation.

You have just created a simple and efficient static scheduler!