Introduction to IoT

School Year 2023-2024

Valsalice

Course Structure

1	Introduction and Basics	
2	Basic Data Types and Operators	
3	Control Structures Pt. 1	
4	Control Structures Pt. 2	
5	Functions and Scope	
6	Arrays	
7	Introduction to Contiki-NG and nRF52840	
8	Sensing and Actuating with Contiki-NG	DEC 5
9	Basic Communication and Networking	DEC 12
10	Preprocessor and Macros	DEC 19
11	Custom Data Types	JAN 9
12	Introduction to RPL and Network Routing	JAN 16
13	Challenges in Wireless Communication	JAN 23
14	Advanced Protocols: TSCH and 6TiSCH	JAN 30
15	Advanced Topics in Wireless Communication	FEB 6
16	Reliable Data Transfer Challenge	FEB 20, FEB 27, MAR 5

= Core Topics = Optional Topics

Open your Virtual Machines

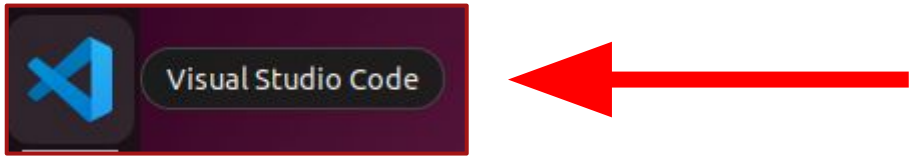
1. Turn on your Laptops
2. Login to Windows using “User”
3. Open the **Virtual Box** program
4. Add a new Virtual Machine (**Ctrl + A**)
5. Open the **VirtualBox** folder ⚠ (**NOT** the .VirtualBox)
6. Select the **nRF52840LAB** file
7. Click **Start**

Prepare the Coding Environment

1. Start the Virtual Machine **nRF52840LAB**
2. Log-in using credentials:

   Username: **ubuntu**

   Password: **ubuntu**
3. Open **Visual Studio Code** (use the App bar on the left)

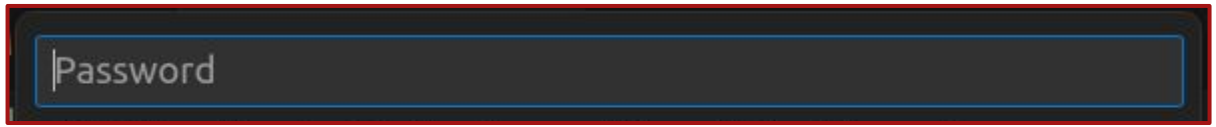


Prepare the Coding Environment

4. From the Terminal:

```
make setup
```

5. 

6. 

7. 

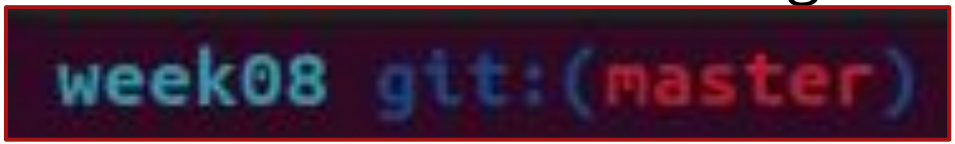
If you see **any (yellow) errors** input the credentials again

Prepare the Coding Environment

7. Open the **`week08`** folder in the terminal

   Right click on the left + "**Open in Integrated terminal**"

8. You should see the following in the terminal:



Recap: Data Types

C has a number of primitive data types:

int `42` `1200` `1_200` `-3`

float `3.14` `0.00001` `-2.1`

char `'A'` `'@'` `'\n'`

bool `true` `false`

Strings are *NOT* a primitive data type, and have special syntax.

strings `"Hello"` `"A"` `"I am a full sentence!"`

Recap: Variables

A variable is a named container that stores data or values.

```
int x = 42;
float y = -0.12;
char w = 'A';
char z[50] = "Full sentence";
```

Booleans require a custom include statement:

```
#include <stdbool.h>
bool hello = true;
```

Recap: Boolean Operators

Greater than	>
Greater or equal than	>=
Less than	<
Less or equal than	<=
Equals	==
Not equals	!=
Not	!

Recap: Chaining Comparisons

- **and** (both must be true)

```
true && false
```

```
(5 < 6) && (5 < 10)
```

- **or** (either must be true)

```
true || false
```

```
(5 < 3) || (5 < 10)
```

- **not** (negation)

```
!true
```

```
!(5 < 3)
```

Recap: If-Statement chaining

You can chain multiple conditions with **`else if`**.

What is the difference between these two snippets of code?

```c
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
} else if (num < 10) {
    printf("Medium number\n");
}
```

```c
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
}
if (num < 10) {
    printf("Medium number\n");
}
```

Recap: While-Loops

Repeat parts of your code!

```c
int num;
printf("Input a number greater than 100: ");
scanf("%d", &num);

while (num <= 100) {
    printf("Wrong number, try again: ");
    scanf("%d", &num);
}

printf("Well done!\n");
```

Recap: For-Loops

Repeat a **specific** amount of times!

```
int x;

for (x = 1; x <= 5; x++) {
    printf("Hello %d\n", x);
}
```

```
int x = 0;

while (x < 5) {
    x += 1;
    printf("Hello %d\n", x);
}
```

Recap: Arrays

Modifiable containers for data.

With **variables**:

```
int num1 = 42;
int num2 = 100;
int num3 = 10;

printf("%d\n", num1);
printf("%d\n", num2);
printf("%d\n", num3);
```

With a **list**:

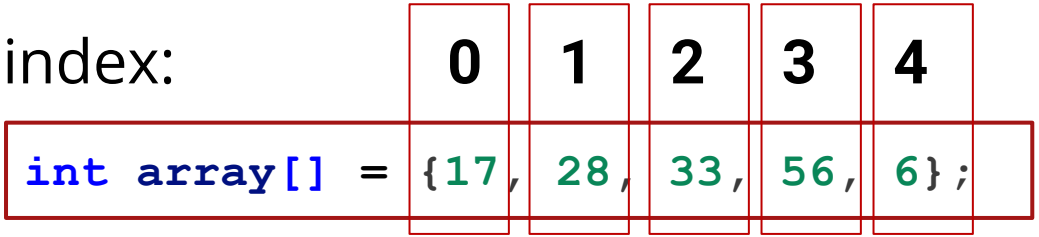
```
int array[] = {42, 100,
10};

for(int i = 0; i < 3; i++)
{
    printf("%d\n",
array[i]);
}
```

Recap: Accessing Array Elements

To access array elements you can use the **[index]** operator.

NOTE: List indices start from **0**




```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```

Assigning Array Elements

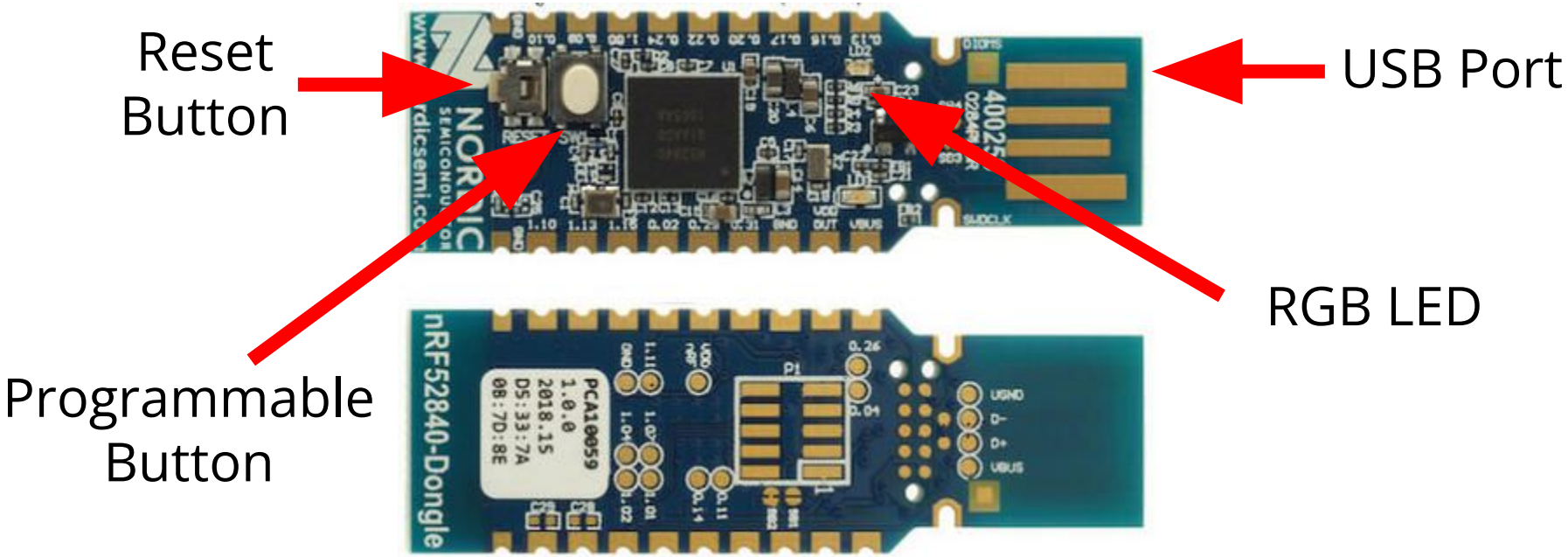
To assign array elements you can use the **[index]** operator on the left-hand-side of a statement (like a variable)

```
int array[] = {17, 28, 33, 56, 6};
array[3] = 100;
array[2] = -7;
```

```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```

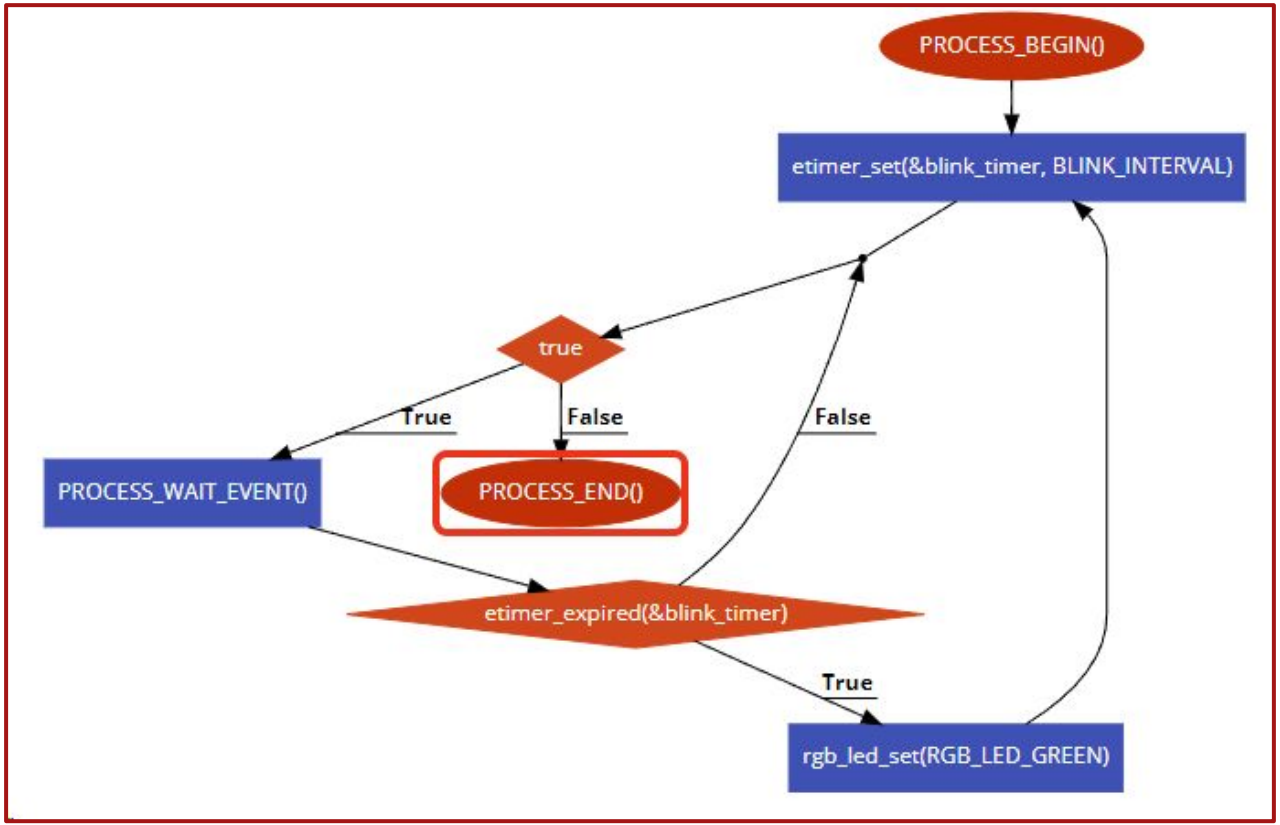
What is the nRF52840?



Anatomy of a Contiki-NG Program

```
1  PROCESS_THREAD(simple_led_program, ev, data) {
2        static struct etimer blink_timer;
3        PROCESS_BEGIN();
4        etimer_set(&blink_timer, BLINK_INTERVAL);
5        while (true) {
6              PROCESS_WAIT_EVENT();
7              if (etimer_expired(&blink_timer)) {
8                    rgb_led_set(RGB_LED_GREEN);
9                    etimer_set(&blink_timer, BLINK_INTERVAL);
               }
         }
10       PROCESS_END();
   }
```

Anatomy of a Contiki-NG Program



Make the LED blink

1. Attach the **nRF52840** chip to your laptops

⚠ Ensure the device is in **bootloader mode** (blinking red light)

Reset Button

3. Program the firmware

```
make blinker.dfu-upload
```

Recap Exercise

Change the function in (**`blinker.c`**):

- `void use_rgb(int counter)`
  Currently the function turns the LED on/off.
  Change it so that it changes across the following states:
  1. ***GREEN***
  2. ***YELLOW***
  3. ***RED***
  4. ***OFF***

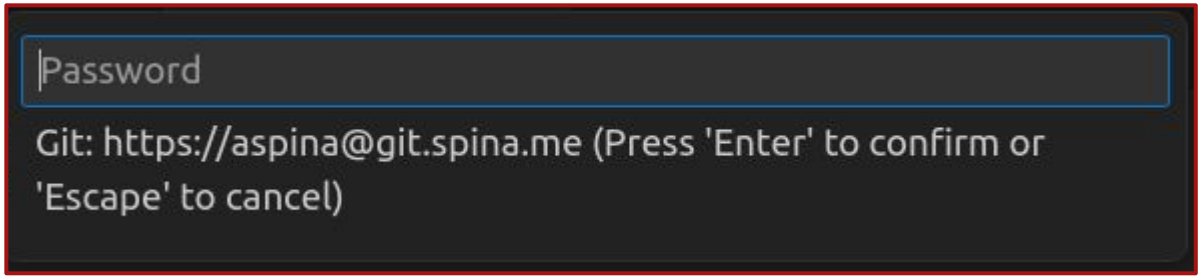
To flash: `make blinker.dfu-upload`

Exercise Solution

```
void use_rgb(int counter) {
      if (counter % 4 == 0) {
            rgb_led_off();
      } else if (counter % 4 == 1) {
            rgb_led_set(RGB_LED_GREEN);
      } else if (counter % 4 == 2) {
            rgb_led_set(RGB_LED_YELLOW);
      } else if (counter % 4 == 3) {
            rgb_led_set(RGB_LED_RED);
      }
}
```

Save remotely your Changes

1. `make save`

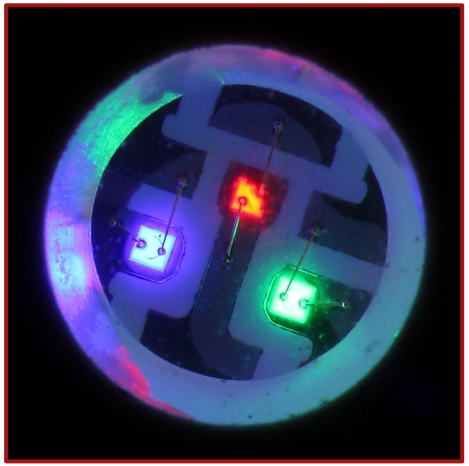
2. 

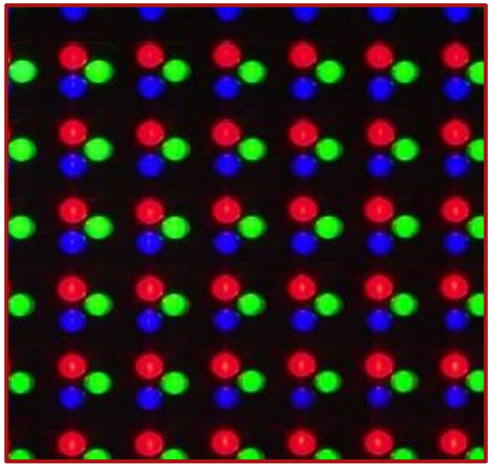
3. 

RGB LEDs

LEDs are **actuators**, they allow the device to act on the outside world. RGB LEDs have **three configurable color** channels:

1. **Red**
2. **Green**
3. **Blue**





LED displays (such as those of PCs) work the same way

The LED Library

```
#define RGB_LED_RED     1
#define RGB_LED_GREEN   2
#define RGB_LED_BLUE    4
#define RGB_LED_MAGENTA (RGB_LED_RED  | RGB_LED_BLUE)
#define RGB_LED_YELLOW  (RGB_LED_RED  | RGB_LED_GREEN)
#define RGB_LED_CYAN    (RGB_LED_GREEN  | RGB_LED_BLUE )
#define RGB_LED_WHITE   (RGB_LED_RED  | RGB_LED_GREEN  |
RGB_LED_BLUE)
/*-------------------------------------------------------------*
/
void rgb_led_off(void);
void rgb_led_set(uint8_t colour);
```

Exercise

Change the function in (**`blinker.c`**):

- `void use_rgb(int counter)`
  Change it so that it changes across the following states:
  1. ***WHITE***
  2. ***CYAN***
  3. ***OFF***

NOTE: You must use the **number value** of the color!

To flash: `make blinker.dfu-upload`

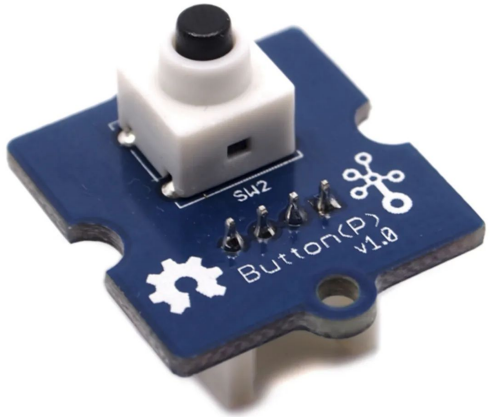
Exercise Solution

```
void use_rgb(int counter) {
      if (counter % 3 == 0) {
            rgb_led_off();
      } else if (counter % 3 == 1) {
            rgb_led_set(7);
      } else if (counter % 3 == 2) {
            rgb_led_set(6);
      }
}
```

Buttons

Buttons allow the device to **“sense”** the world around them.

The button allows the device to **receive input and react** to actions in the world around them.



Button Library

```
/* Event generated when a button gets pressed */
extern process_event_t button_hal_press_event;
/* Event generated when a button gets released */
extern process_event_t button_hal_release_event;
/* Event generated every second the button is kept pressed */
extern process_event_t button_hal_periodic_event;
/*---------------------------------------------------------------*
/
#define BUTTON_HAL_STATE_RELEASED  0
#define BUTTON_HAL_STATE_PRESSED   1
/*---------------------------------------------------------------*
/
void button_hal_init(void);
```

Detecting Button Presses

```
PROCESS_THREAD(button_hal_example, ev, data) {
  PROCESS_BEGIN();
  while (1) {
    PROCESS_YIELD();

    if (ev == button_hal_press_event) {
      printf("Button pressed!\n");
    }
  }
  PROCESS_END();
}
```

1

To flash: `make button.dfu-upload`

For console: `make login`

Exercise

Change the code in (**`button.c`**):

1. Turn the LED to color **GREEN** when the button is pressed

To flash: `make button.dfu-upload` **For console:** `make login`

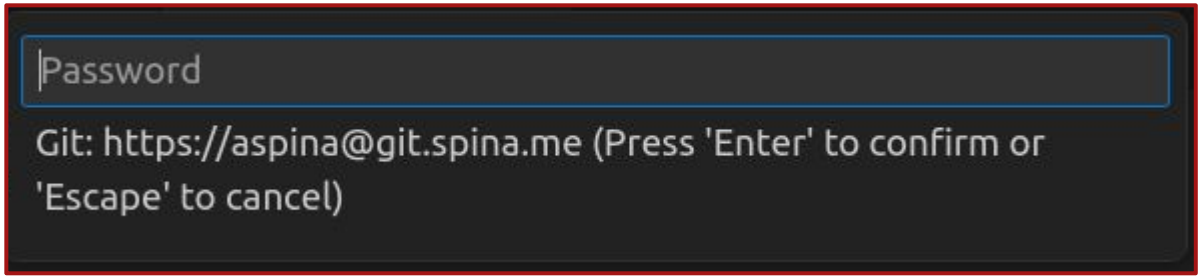
Exercise Solution

```
while (1) {
  PROCESS_YIELD();

  if (ev == button_hal_press_event) {
    printf("Button pressed!\n");
    rgb_led_set(RGB_LED_GREEN);
  }
}
```

Save remotely your Changes

1. `make save`

2. 

3. 

Detecting Button Releases

```
PROCESS_THREAD(button_hal_example, ev, data) {
  PROCESS_BEGIN();
  while (1) {
    PROCESS_YIELD();
    if (ev == button_hal_press_event) {
      printf("Button pressed!\n");
    } else if (ev == button_hal_release_event) {
      printf("Button released!\n");
    }
  }
  PROCESS_END();
}
```

1

Exercise

Change the code in (**`button.c`**):

1. Turn the LED to color **GREEN** when the button is pressed
2. Turn the LED **OFF** when the button is released

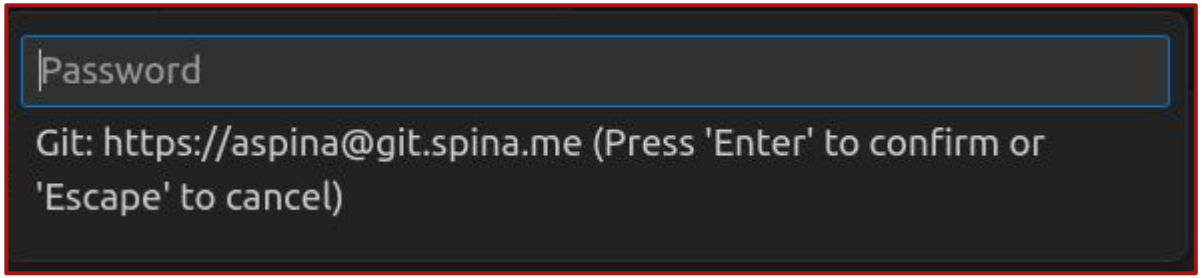
To flash: `make button.dfu-upload` **For console:** `make login`

Exercise Solution

```
while (1) {
  PROCESS_YIELD();
  if (ev == button_hal_press_event) {
    printf("Button pressed!\n");
    rgb_led_set(RGB_LED_GREEN);
  }
  else if (ev == button_hal_release_event) {
    printf("Button released!\n");
    rgb_led_off();
  }
}
```

Save remotely your Changes

1. `make save`

2. 

3. 

Detecting Continued Button Press

1

```
static int press_seconds = 0;
while (1) {
  PROCESS_YIELD();
  if (ev == button_hal_press_event) {
    printf("Button pressed!\n");
  } else if (ev == button_hal_release_event) {
    printf("Button released!\n");
  }
  if (ev == button_hal_periodic_event) {
    press_seconds = press_seconds + 1;
    printf("Button pressed for %d seconds!\n", press_seconds);
  } else {
    press_seconds = 0;
  }
}
```

2 3 4 5 6 7

Exercise

Change the code in (**`button.c`**):

1. Turn the LED to color **GREEN** when the button is pressed
2. Turn the LED **OFF** when the button is released
3. Turn the LED **CYAN** when the button is kept pressed for more than five seconds

To flash: `make button.dfu-upload` **For console:** `make login`

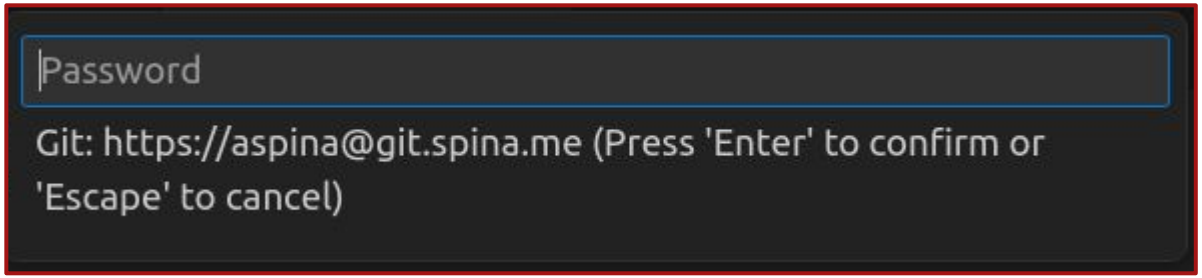
Exercise Solution

```
    if (ev == button_hal_periodic_event) {
      press_seconds = press_seconds + 1;
      printf("Button pressed for %d seconds!\n",
press_seconds);

      if (press_seconds >= 5) {
        rgb_led_set(RGB_LED_CYAN);
      }
    }
    else {
      press_seconds = 0;
    }
```

Save remotely your Changes

1. `make save`

2. 

3. 

End of Class

See you all next week!