



2021/2022

# Architecture des ordinateurs

*Rapport 03*

## Travail pratique 3 : 7-Segments et PWM

<b>Ecole</b>	Haute école d'ingénierie et d'architecture de Fribourg (HEIA-FR)
<b>Branche</b>	Architecture des ordinateurs
<b>Etudiants</b>	Barras Simon, Terreaux Nicolas
<b>Groupe</b>	B01
<b>Classe</b>	ISC-IL-2d
<b>Professeur</b>	Supcik Jacques
<b>GitLab</b>	<a href="https://gitlab.forge.hefr.ch/ado/2021-2022/classe-supcik/groupe-B-01/tp03">https://gitlab.forge.hefr.ch/ado/2021-2022/classe-supcik/groupe-B-01/tp03</a>
<b>Version/Date</b>	V1 du 02.11.2021

<b>1</b>	<b>INTRODUCTION.....</b>	<b>2</b>
<b>2</b>	<b>RÉSUMÉS .....</b>	<b>2</b>
2.1	NON ACQUIS.....	2
2.2	A EXERCER .....	2
2.3	PARFAITEMENT ACQUIS.....	2
<b>3</b>	<b>POINTS IMPORTANTS .....</b>	<b>3</b>
3.1	7-SEGMENT .....	3
3.2	PWM (PULSE WIDTH MODULATION) .....	3
3.3	RETROUVER DES INFOS DANS UN DATASHEET .....	4
<b>4</b>	<b>CONCLUSION .....</b>	<b>5</b>

# 1 Introduction

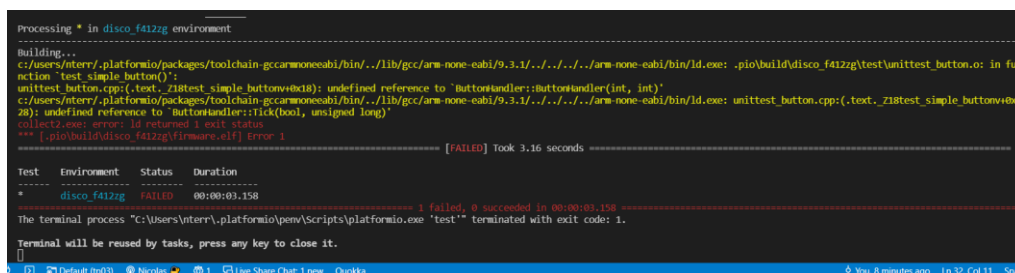
Ce 3<sup>ème</sup> TP a pour but de nous faire utiliser les 7-segments ainsi que d'apprendre les PVM. Les 7-segments sont des éléments visuels qui nous permettent d'avoir un feedback rapide. Dans ce TP nous les utilisons comme un compteur que nous pouvons incrémenter. Lors de l'utilisation de ces compteurs, nous avons dû faire tout le chemin des différentes connexions afin de savoir quels pins et quels ports nous devons utiliser. Le PVM est un outil qui permet de définir une intensité ou une valeur avec un seul pin. On utilise ce pin avec une horloge pour fournir un signal rectangulaire avec une fréquence et une amplitude fixe. Pour modifier la valeur, on change que le « Duty-cycle » qui est le pourcentage de temps qu'un pin est UP. Par exemple : pour un duty-cycle de 100%, le signal sera toujours à 1. Dans le cas contraire, 0% signifie tout le temps à 0. Nous avons l'habitude de voir des signaux avec un duty-cycle à 50%. Ces signaux sont autant de temps en haut que en bas : „|”|”|”.

## 2 Résumés

Voici la liste des différentes notions théoriques que nous avons dû utiliser dans ce TP. Elles sont organisées par niveau de compréhension.

### 2.1 Non acquis

Les tests unitaires : Les tests unitaires sont un outil que nous connaissons et nous savons les réaliser. Cependant, nous n'avons pas été capable de les mettre place dans ce projet. En effet, nous avons tester de créer un bouton afin de tester son fonctionnement artificiellement mais nous nous sommes confrontés à une erreur de référence.



```
Processing * in disco_f412zg environment
Building...
c:/users/nterr/.platformio/packages/toolchain-gccarmnoneabi/bin/../lib/gcc/arm-none-eabi/9.3.1/../../../../arm-none-eabi/bin/ld.exe: .pio/build/disco_f412zg/test/unittest_button.o: in function 'test_simple_button()':
unittest_button.cpp:(.text+0x18): undefined reference to 'buttonHandler::buttonHandler(int, int)'
c:/users/nterr/.platformio/packages/toolchain-gccarmnoneabi/bin/../lib/gcc/arm-none-eabi/9.3.1/../../../../arm-none-eabi/bin/ld.exe: unittest_button.cpp:(.text+0x28): undefined reference to 'buttonHandler::Tick(bool, unsigned long)'
collect2.exe: error: ld returned 1 exit status
*** [.pio/build/disco_f412zg/firmware.elf] error 1

===== [FAILED] Took 3.16 seconds =====

Test Environment Status Duration
-----
* disco_f412zg FAILED 00:00:03.158

===== 1 failed, 0 succeeded in 00:00:03.158 =====
The terminal process "C:\Users\nterr\.platformio\penv\Scripts\platformio.exe "test"" terminated with exit code: 1.
Terminal will be reused by tasks, press any key to close it.
```

### 2.2 A exercer

Nous avons encore de la peine à comprendre le lien entre la consigne avec les éléments physique et le code. En effet, la manière de programmer est différente de celle que nous avons connu jusqu'à maintenant nous n'avons jamais programmé des choses en lien direct avec le Hardware. Lors de nos réflexions nous nous mettons des barrières alors que les choses sont tout à fait réalisables comme quelquefois nous imaginons des actions alors qu'elles sont irréalisables. Comme par exemple, au début, nous voulions directement allumer le 7 segments sans passer par les registres.

Le second point à exercer est les références. En effet, ce n'est pas encore naturel pour nous de savoir quand il faut utiliser un objet, une référence ou un appel avec « :: ».

### 2.3 Parfaitement acquis

Nous avons bien compris le fonctionnement et l'implémentation du 7-segments.

Nous avons fait le chemin à travers les pins grâce aux différentes datasheets dans le but de connaître tous les ports/pins utile pour passer à l'implémentation. Nous avons ensuite utilisé les TP précédents comme exemple et nous avons rapidement réussi à afficher quelques chose (comme quoi à force on

arrive...). De plus nous avons compris comment les valeurs sont transmises au hardware grâce au cours sur les périphériques.

### 3 Points importants

Les points importants de ce TP sont répertoriés ici. Ceci permet d'avoir une vue d'ensemble des nouvelles notions théoriques et leur explication pour les futurs TP.

#### 3.1 7-segment

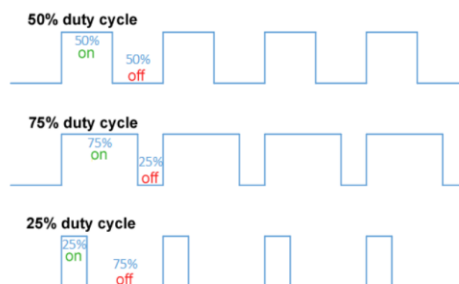
L'utilisation du 7-segments est importantes pour représenter des nombres entiers

Voici la liste des patterns que nous avons utilisé pour la réalisation du projet :

```
const uint8_t kPatterns[] = {
    0b01111110, // 0
    0b01010000, // 1
    0b01101101, // 2
    0b01111001, // 3
    0b01010011, // 4
    0b00111011, // 5
    0b00111111, // 6
    0b01110000, // 7
    0b01111111, // 8
    0b01111011, // 9
    0b01110111, // A
    0b00011111, // b
    0b00101110, // C
    0b01011101, // d
    0b00101111, // E
    0b00100111, // F
    // 0b point top-right top bottom-right bottom bottom-left
    top-left mid
};
```

#### 3.2 PWM (Pulse Width Modulation)

Le PWM est un outil qui permet de définir une valeur en continu. Dans notre cas, il alimente le 7-segments. Comme nous modifions le up-time du signal, nous changeons la luminosité. Il est facilement imaginable que si nous donnons seulement 50% de 1, la luminosité sera diminuée de moitié.



Source : <https://ado.pages.forge.hefr.ch/2021-2022/assignments/tp03/>

Ce système n'est pas seulement utilisé pour la luminosité d'une LED, par exemple les moteurs des métros peuvent aussi utiliser ce système.

### 3.3 Retrouver des infos dans un datasheet

L'utilisation du Datasheet est très importante. Dans notre cas, nous avons recherché les informations sur les différents ports que nous avons eu besoin.

#### 7-segments et Arduino shield:

Voici la liste des différents Ports/Pins du 7 segments pour les connexions sur l'arduino shield:

Signal	GPIO (socle de gauche)	GPIO (socle de droite)
<b>MR#</b>	PC3	PC2
<b>SCK</b>	PB5/SPI-SCK	PB5/SPI-SCK
<b>SDO</b>	PB4/SPI-MISO	PB4/SPI-MISO
<b>SDI</b>	PB3/SPI-MOSI	PB3/SPI-MOSI
<b>PWM</b>	PD6/PWMA	PD5/PWMB
<b>LATCH</b>	PB2/SPI-SS	PB1

#### Arduino shield out et Arduino shield in:

Voici la liste des différents Ports/Pins de l'arduino shield pour les connexions sur la cible :

Signal	GPIO (socle de gauche)	GPIO (socle de droite)
<b>MR#</b>	A3	A2
<b>SCK</b>	D13	D13
<b>SDO</b>	D12	D12
<b>SDI</b>	D11	D11
<b>PWM</b>	D6	D5
<b>LATCH</b>	D10	D9

#### Processeur et Arduino shield:

Voici la liste des différents Ports/Pins de la cible qui nous intéresse pour effectuer la programmation :

Signal	GPIO (socle de gauche)	GPIO (socle de droite)
<b>MR#</b>	PC4	PC3
<b>SCK</b>	PA5	PA5
<b>SDO</b>	PA6	PA6
<b>SDI</b>	PA7	PA7
<b>PWM</b>	PF3	PF10
<b>LATCH</b>	PA15	PB8

## 4 Conclusion

Nous avons eu beaucoup de peine à comprendre le fonctionnement de ce TP. L'affichage d'un nombre sur le 7-segment ainsi que le réglage de la luminosité ne nous a pas posé beaucoup de problèmes mais nous avons dû passer beaucoup plus de temps sur la deuxième partie qui consistait à utiliser les boutons pour faire changer le compteur et la luminosité. Comme cité précédemment, nous avons aussi perdu énormément de temps sur l'élaboration des tests unitaires. Nous avons eu une erreur qui nous a bloqué dès le début...

Nous avons passé chacun environ 8-10h en plus des heures de cours.

## 5 Sources

### 5.1 Main.cpp

```
// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//      http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file main.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
 * <nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Control the 7-segments with a joystick
 *
 * @date 2021-11-16
 * @version 0.1.0
 *****/
#include <Systick.hpp>

#include "Buttons.hpp"

int main()
{
    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);
```

```

Joystick::Setup();

Joystick::RegisterHandler(Joystick::Right, new ButtonRight());
Joystick::RegisterHandler(Joystick::Left, new ButtonLeft());
Joystick::RegisterHandler(Joystick::Up, new ButtonUp());
Joystick::RegisterHandler(Joystick::Down, new ButtonDown());
Joystick::RegisterHandler(Joystick::Select, new ButtonSelect());
auto seg = Disco7Segments();

SystickSetup();
DiscoConsoleSetup();

while (1) {
    seg.SetBrightness(brightness);
    seg.Print(counter);
    asm volatile("nop");
}
}

```

## 5.2 Joystick.hpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//      http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Joystick.hpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Interface for the group of buttons
 *
 * @date 2021-11-16
 * @version 0.1.0
 *****/

```

```

#ifndef JOYSTICK_HPP_
#define JOYSTICK_HPP_
#include "ButtonHandler.hpp"

class Joystick {
public:
    Joystick() = delete; // suppress constructor
    Joystick(const Joystick&) = delete; // suppress copy constructor
    ~Joystick() = delete; // suppress destructor
    enum Button { Select, Down, Left, Right, Up };

    static const int kPollingFrequency_ = 100; // in Hz
    static void Setup();
    static void RegisterHandler(Button button, ButtonHandler* handler);
    static void Tick();

private:
    static ButtonHandler* handlers_[];
};
#endif /* JOYSTICK_HPP_ */

```

### 5.3 Joystick.cpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Joystick.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
 * <nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implementation of the list of buttons
 *
 * @date 2021-11-16
 * @version 0.1.0

```

```

*****
****/
#include "Joystick.hpp"

#include <DiscoConsole.h>
#include <libopencm3/cm3/nvic.h>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/timer.h>
#include <stdio.h>

#include <Disco7Segments.hpp>

#include "Systick.hpp"

constexpr static struct {
    enum rcc_periph_clken rcc;
    uint32_t port;
    uint16_t pin;
} buttons_config[] = {[Joystick::Select] = {.rcc = RCC_GPIOA, .port =
GPIOA, .pin = GPIO0},
                      [Joystick::Down]   = {.rcc = RCC_GPIOG, .port =
GPIOG, .pin = GPIO0},
                      [Joystick::Left]    = {.rcc = RCC_GPIOF, .port =
GPIOF, .pin = GPIO14},
                      [Joystick::Right]   = {.rcc = RCC_GPIOF, .port =
GPIOF, .pin = GPIO15},
                      [Joystick::Up]      = {.rcc = RCC_GPIOG, .port =
GPIOG, .pin = GPIO1}};

constexpr int kNumberOfButtons = sizeof(buttons_config) /
sizeof(buttons_config[0]);

ButtonHandler* Joystick::handlers_[kNumberOfButtons]{};
Disco7Segments seg = Disco7Segments();
int currentNumber = 0;

void Joystick::Setup()
{
    // Start clocks
    rcc_periph_clock_enable(RCC_GPIOA);
    rcc_periph_clock_enable(RCC_GPIOG);
    rcc_periph_clock_enable(RCC_GPIOF);

    // Configure the GPIOs for the 5 buttons and Reset handlers
    for (int i = 0; i < kNumberOfButtons; i++) {
        gpio_mode_setup(
            buttons_config[i].port, GPIO_MODE_INPUT, GPIO_PUPD_PULLDOWN,
            buttons_config[i].pin);
    }
}

```



```

        Joystick::handlers_[i] = nullptr;
    }

    // Configure Timer2 for periodical polling (like TP02)
    rcc_periph_clock_enable(RCC_TIM2);
    rcc_periph_reset_pulse(RST_TIM2);
    timer_set_period(TIM2, rcc_apb2_frequency / kPollingFrequency_);
    timer_enable_irq(TIM2, TIM_DIER_UIE);
    timer_enable_counter(TIM2);
    nvic_enable_irq(NVIC_TIM2_IRQ);
}

void Joystick::Tick()
{
    uint32_t now = SysTickSystemMillis();
    for (int i = 0; i < kNumberOfButtons; i++) {
        printf("Time %ld\n", now);
        if (handlers_[i] != nullptr) {
            uint16_t value = gpio_get(buttons_config[i].port,
buttons_config[i].pin);
            handlers_[i]->Tick(value != 0, now);
        }
    }
}

void Joystick::RegisterHandler(Button button, ButtonHandler* handler)
{
    handlers_[button] = handler;
}

void tim2_isr(void)
{
    timer_clear_flag(TIM2, TIM_SR_UIF); // acknowledge interrupt
    Joystick::Tick();
}

```

#### 5.4 ButtonHandler.hpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software

```

```

// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
****
* @file ButtonHandler.hpp
* @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
*
* @brief Interface for the button and his action
*
* @date 2021-11-16
* @version 0.1.0
****
****/
#ifndef BUTTONHANDLER_HPP_
#define BUTTONHANDLER_HPP_

#include <stdint.h>
#include <stdio.h>

class ButtonHandler {
public:
    ButtonHandler(int longPressedTime = 1000, int repeatTime = 500);
    void Tick(bool pressed, uint32_t ms);

    virtual void OnPressed(){};
    virtual void OnReleased(){};
    virtual void OnLongPressed(){};
    virtual void OnRepeated(int repeated) { (void)repeated; };

private:
    uint64_t longPressedTime_;
    int repeatTime_;
    int timePassed_;
    int repeat_; // for long press, unused now
};
#endif /* DISC07SEGMENTS_HPP_ */

```

## 5.5 ButtonHandler.cpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.

```

```

// You may obtain a copy of the License at
//
//      http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file ButtonHandler.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implement the tick for all five buttons
 *
 * @date 2021-11-16
 * @version 0.1.0
 *****/
#include "ButtonHandler.hpp"

#include <DiscoConsole.h>
#include <stdint.h>
#include <stdio.h>

ButtonHandler::ButtonHandler(int longPressedTime, int repeatTime)
    : longPressedTime_(longPressedTime), repeatTime_(repeatTime)
{
}

void ButtonHandler::Tick(bool pressed, uint32_t ms)
{
    if (pressed == false) {
        OnReleased();
        timePassed_ = 0;
    }
    if (pressed) {
        OnPressed();

        if (timePassed_ == 0) timePassed_ = ms;
        // Pressed more than 1 second
        if (longPressedTime_ <= (ms - timePassed_)) {
            OnLongPressed();
        }
    }
}

```

```
}
```

## 5.6 Button.hpp

```
// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//      http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Buttons.hpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
 * <nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Contains all implementations for the 5 buttons
 *
 * @date 2021-11-16
 * @version 0.1.0
 *****/
#include <DiscoConsole.h>
#include <assert.h>
#include <libopencm3/stm32/rcc.h>
#include <stdio.h>

#include <Disco7Segments.hpp>
#include <DiscoAssert.hpp>

#include "ButtonHandler.hpp"
#include "Joystick.hpp"

const int MAX_BRIGHTNESS = 100;
const int MIN_BRIGHTNESS = 0;
const int MAX_VALUE      = 99;
const int MIN_VALUE      = 0;

int counter    = 0;
```

```

int brightness = 50;

class ButtonRight : public ButtonHandler {
public:
    ButtonRight() : ButtonHandler(), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && counter != MAX_VALUE) counter++;
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };

    int state_;
};

class ButtonLeft : public ButtonHandler {
public:
    ButtonLeft() : ButtonHandler(), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && counter != MIN_VALUE) counter--;
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };

    int state_;
};

class ButtonUp : public ButtonHandler {
public:
    ButtonUp() : ButtonHandler(), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && brightness != MAX_BRIGHTNESS) brightness += 5;
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };

    int state_;
};

class ButtonDown : public ButtonHandler {
public:
    ButtonDown() : ButtonHandler(), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && brightness != MIN_BRIGHTNESS) brightness -= 5;
        state_ = 1;
    };
};

```

```

    void OnReleased() override { state_ = 0; };

    int state_;
};

class ButtonSelect : public ButtonHandler {
public:
    ButtonSelect() : ButtonHandler(), state_{0} {}

    void OnLongPressed() override
    {
        if (state_ == 0) {
            counter = MIN_VALUE;
            state_ = 1;
        }
    };
    void OnReleased() override { state_ = 0; };

    int state_;
};

```

## 5.7 Disco7Segments.hpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Disco7Segments.hpp
 * @author Jacques Supcik, Simon Barras <simon.barras@edu.hefr.ch>,
Nicolas Terreaux
 * <nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Defines the interface of the 7 segments.
 *
 * @date 2021-11-02
 * @version 0.0.1

```

```

*****
****/
#ifndef DISCO7SEGMENTS_HPP_
#define DISCO7SEGMENTS_HPP_

#include <stdint.h>

#include "PWM.hpp"

class Disco7Segments {
public:
    Disco7Segments(int id = 0);
    void Reset();
    void PrintPattern(uint16_t pattern);
    void Print(int i);
    void PrintHex(int i);

    void SwitchOn();
    void SwitchOff();
    void SetBrightness(int brightness);

private:
    uint32_t resetPort_, resetPin_;
    uint32_t pwmPort_, pwmPin_;
    uint32_t sdiPort_, sdiPin_;
    uint32_t sckPort_, sckPin_;
    uint32_t latchPort_, latchPin_;
    void Latch();
    void SendBit(int bit);
    uint16_t GetPattern(int i, int base);
    PWM pwm_;
};

#endif /* DISCO7SEGMENTS_HPP_ */

```

## 5.8 Disco7Segments.cpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,

```

```

// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Disco7Segments.hpp
 * @author Simon Barras <simon.barras@edu.hefr.ch>, Nicolas Terreaux
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implementation of the 7 segments.
 *
 * @date 2021-11-02
 * @version 0.0.1
 *****/
#include "Disco7Segments.hpp"

#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>

// ----- Constants -----
constexpr auto kMrPort      = GPIOC;
constexpr auto kSPort      = GPIOA;
constexpr auto kPwmPort    = GPIOF;
constexpr auto kLatch1Port = GPIOA;
constexpr auto kLatch2Port = GPIOB;
constexpr auto kMr1Pin     = GPIO4;
constexpr auto kMr2Pin     = GPIO3;
constexpr auto kSckPin     = GPIO5;
constexpr auto kSdoPin     = GPIO6;
constexpr auto kSdiPin     = GPIO7;
constexpr auto kPwm1Pin    = GPIO3;
constexpr auto kPwm2Pin    = GPIO10;
constexpr auto kLatch1Pin  = GPIO15;
constexpr auto kLatch2Pin  = GPIO8;

const uint8_t kPatterns[] = {
    0b01111110, // 0
    0b01010000, // 1
    0b01101101, // 2
    0b01111001, // 3
    0b01010011, // 4
    0b00111011, // 5
    0b00111111, // 6
    0b01110000, // 7
    0b01111111, // 8
    0b01111011, // 9

```



```

    0b01110111, // A
    0b00011111, // b
    0b00101110, // C
    0b01011101, // d
    0b00101111, // E
    0b00100111, // F
        // 0b point top-right top bottom-right bottom bottom-
left top-left mid
};

// ----- Constructor -----

Disco7Segments::Disco7Segments(int id) : pwm_(id == 0 ? PWM::Pin::PF3 :
PWM::Pin::PF10, 10000, 100)
{
    // start clock
    rcc_periph_clock_enable(RCC_GPIOA);
    if (id == 1) {
        rcc_periph_clock_enable(RCC_GPIOB);
    }

    rcc_periph_clock_enable(RCC_GPIOC);
    rcc_periph_clock_enable(RCC_GPIOD);

    // initialize port and pin
    resetPort_ = kMrPort;
    sckPort_   = kSPort;
    // sdoPort_ = kSPort;
    sdiPort_   = kSPort;
    pwmPort_   = kPwmPort;
    sckPin_     = kSckPin;
    // sdoPin_   = kSdoPin;
    sdiPin_     = kSdiPin;

    if (id != 0) {
        latchPort_ = kLatch2Port;
        resetPin_   = kMr2Pin;
        pwmPin_     = kPwm2Pin;
        latchPin_   = kLatch2Pin;

    } else {
        latchPort_ = kLatch1Port;
        resetPin_   = kMr1Pin;
        pwmPin_     = kPwm1Pin;
        latchPin_   = kLatch1Pin;
    }

    // enable port and pin

```

```

        gpio_mode_setup(resetPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
resetPin_);
        gpio_mode_setup(sdiPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
sdiPin_);
        gpio_mode_setup(sckPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
sckPin_);
        gpio_mode_setup(latchPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE,
latchPin_);
        Reset();
    }

// ----- Private methods -----

void Disco7Segments::Latch()
{
    gpio_set(latchPort_, latchPin_);
    gpio_clear(latchPort_, latchPin_);
}

void Disco7Segments::SendBit(int bit)
{
    if (bit != 0) {
        gpio_set(sdiPort_, sdiPin_);
    } else {
        gpio_clear(sdiPort_, sdiPin_);
    }
    gpio_set(sckPort_, sckPin_);
    gpio_clear(sckPort_, sckPin_);
    gpio_clear(sdiPort_, sdiPin_);
}

// ----- Public methods -----

void Disco7Segments::Reset()
{
    gpio_clear(resetPort_, resetPin_);
    gpio_set(resetPort_, resetPin_);
}

void Disco7Segments::PrintPattern(uint16_t pattern)
{
    for (int i = 0; i < 16; i++) {
        SendBit(pattern & (1 << i));
    }
    Latch();
}

uint16_t Disco7Segments::GetPattern(int i, int base)
{

```

```

    uint16_t pattern = 0;
    for (auto k = 0; k < 2; k++) {
        pattern |= kPatterns[i % base] << 8 * k;
        i /= base;
    }
    return pattern;
}

void Disco7Segments::Print(int i) { PrintPattern(GetPattern(i, 10)); }

void Disco7Segments::PrintHex(int i) { PrintPattern(GetPattern(i, 16)); }

void Disco7Segments::SwitchOn() { SetBrightness(100); }

void Disco7Segments::SwitchOff() { SetBrightness(0); }

void Disco7Segments::SetBrightness(int brightness) {
    pwm_.SetDutyCycle(brightness); }

```

## 5.9 PWM.hpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
 * @file Disco7Segments.hpp
 * @author Jacques Supcik, Simon Barras <simon.barras@edu.hefr.ch>,
Nicolas Terreaux
 * <nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Defines the interface of the PWM.
 *
 * @date 2021-11-16
 * @version 0.0.1

```

```

*****
****/

class PWM {
public:
    // clang-format off
    enum Pin {
        PA0, PA1, PA2, PA3, PA6, PA7,
        PB0, PB1, PB4, PB5, PB6, PB7, PB8, PB9,
        PC6, PC7, PC8, PC9,
        PD12, PD13, PD14, PD15,
        PF3, PF4, PF5, PF10
    };
    // clang-format on

    PWM(Pin pin, int frequency = 10000, int dutyCycle = 10);
    void SetDutyCycle(int dutyCycle);
    void SetFrequency(int frequency);

    int DutyCycle() const;
    int Frequency() const;
    int Period() const;

private:
    Pin pin_;
    int frequency_;
    int dutyCycle_;
};

```

## 5.10 PWM.cpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

/*****
*****

```

```

* @file PWM.cpp
* @author Simon Barras <simon.barras@edu.hefr.ch>, Nicolas Terreaux
<nicolas.terreaux@edu.hefr.ch>
*
* @brief Implementation for the PWM
*
* @date 2021-11-16
* @version 0.1.0
*****
****/
#include "PWM.hpp"

#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/timer.h>
#include <stdio.h>

constexpr static struct {
    enum rcc_periph_clken gpioRcc;
    uint32_t port;
    uint16_t pin;
    enum rcc_periph_clken timerRcc;
    uint32_t timer;
    enum tim_oc_id channel;
} pinsConfig[] = {
    // clang-format off
    [PWM::PA0] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO0,
    .timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC1},
    [PWM::PA1] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO1,
    .timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC2},
    [PWM::PA2] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO2,
    .timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC3},
    [PWM::PA3] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO3,
    .timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC4},
    [PWM::PA6] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO6,
    .timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC1},
    [PWM::PA7] = {.gpioRcc = RCC_GPIOA, .port = GPIOA, .pin = GPIO7,
    .timerRcc = RCC_TIM1, .timer = TIM1, .channel = TIM_OC2},
    [PWM::PB0] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO0,
    .timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC3},
    [PWM::PB1] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO1,
    .timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC4},
    [PWM::PB4] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO4,
    .timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC1},
    [PWM::PB5] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO5,
    .timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC2},
    [PWM::PB6] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO6,
    .timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC1},

```

```

    [PWM::PB7] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO7,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC2},
    [PWM::PB8] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO8,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC3},
    [PWM::PB9] = {.gpioRcc = RCC_GPIOB, .port = GPIOB, .pin = GPIO9,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC4},
    [PWM::PC6] = {.gpioRcc = RCC_GPIOC, .port = GPIOC, .pin = GPIO6,
.timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC1},
    [PWM::PC7] = {.gpioRcc = RCC_GPIOC, .port = GPIOC, .pin = GPIO7,
.timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC2},
    [PWM::PC8] = {.gpioRcc = RCC_GPIOC, .port = GPIOC, .pin = GPIO8,
.timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC3},
    [PWM::PC9] = {.gpioRcc = RCC_GPIOC, .port = GPIOC, .pin = GPIO9,
.timerRcc = RCC_TIM3, .timer = TIM3, .channel = TIM_OC4},
    [PWM::PD12] = {.gpioRcc = RCC_GPIOD, .port = GPIOD, .pin = GPIO12,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC1},
    [PWM::PD13] = {.gpioRcc = RCC_GPIOD, .port = GPIOD, .pin = GPIO13,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC2},
    [PWM::PD14] = {.gpioRcc = RCC_GPIOD, .port = GPIOD, .pin = GPIO14,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC3},
    [PWM::PD15] = {.gpioRcc = RCC_GPIOD, .port = GPIOD, .pin = GPIO15,
.timerRcc = RCC_TIM4, .timer = TIM4, .channel = TIM_OC4},
    [PWM::PF3] = {.gpioRcc = RCC_GPIOF, .port = GPIOF, .pin = GPIO3,
.timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC1},
    [PWM::PF4] = {.gpioRcc = RCC_GPIOF, .port = GPIOF, .pin = GPIO4,
.timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC2},
    [PWM::PF5] = {.gpioRcc = RCC_GPIOF, .port = GPIOF, .pin = GPIO5,
.timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC3},
    [PWM::PF10] = {.gpioRcc = RCC_GPIOF, .port = GPIOF, .pin = GPIO10,
.timerRcc = RCC_TIM5, .timer = TIM5, .channel = TIM_OC4},
    // clang-format on
};

PWM::PWM(Pin pin, int frequency, int dutyCycle)
: pin_(pin), frequency_(frequency), dutyCycle_(dutyCycle)
{
    // Configure GPIO
    rcc_periph_clock_enable(pinsConfig[pin].gpioRcc);
    gpio_mode_setup(pinsConfig[pin].port, GPIO_MODE_AF, GPIO_PUPD_NONE,
pinsConfig[pin].pin);
    gpio_set_af(pinsConfig[pin].port, GPIO_AF2, pinsConfig[pin].pin);

    // Configure timer
    rcc_periph_clock_enable(pinsConfig[pin].timerRcc);
    timer_set_mode(pinsConfig[pin].timer, TIM_CR1_CKD_CK_INT,
TIM_CR1_CMS_EDGE, TIM_CR1_DIR_UP);
    SetFrequency(frequency);
    timer_set_oc_mode(pinsConfig[pin].timer, pinsConfig[pin].channel,
TIM_OCM_PWM2);

```

```

        SetDutyCycle(dutyCycle);
        timer_enable_oc_output(pinsConfig[pin].timer,
pinsConfig[pin].channel);
        timer_enable_counter(pinsConfig[pin].timer);
    };

// Configure the duty cycle between 0% and 100%.
void PWM::SetDutyCycle(int dutyCycle)
{
    dutyCycle_ = dutyCycle;
    timer_set_oc_value(
        pinsConfig[pin_].timer, pinsConfig[pin_].channel, Period() *
(100 - dutyCycle_) / 100);
}

void PWM::SetFrequency(int frequency)
{
    frequency_ = frequency;
    timer_set_counter(pinsConfig[pin_].timer, 0);
    timer_set_period(pinsConfig[pin_].timer, rcc_apb2_frequency /
frequency);
}

int PWM::DutyCycle() const { return dutyCycle_; }

int PWM::Frequency() const { return frequency_; }

int PWM::Period() const { return rcc_apb2_frequency / frequency_; }

```

## 5.11 Unittest\_button.cpp

```

// Copyright 2021 Haute école d'ingénierie et d'architecture de Fribourg
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
// implied.
// See the License for the specific language governing permissions and
// limitations under the License.

```

```

/*****
 * @file unittest_button.hpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Unit test for the button class. Don't work
 *
 * @date 2021-11-22
 * @version 0.1.0
 *****/
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/timer.h>
#include <unity.h>

#include "ButtonHandler.hpp"
#include "Buttons.hpp"
#include "Joystick.hpp"
#include "Systick.hpp"
static void setup(void)
{
    // Enable clock
    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);
    SystickSetup();
}

void test_simple_button(void)
{
    counter = 0;
    // Disco7Segments seg = Disco7Segments();

    ButtonHandler* b = new ButtonRight();
    uint32_t timer = 0;
    b->Tick(false, timer += 10); // First tick is needed to initialize
the button
    TEST_ASSERT_EQUAL(0, counter);
    TEST_PASS_MESSAGE("Right button OK");
}

int main()
{
    setup();
    UNITY_BEGIN();
    UNITY_BEGIN();
    RUN_TEST(test_simple_button);
    return 0;
}

```



