Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

2021/2022

Architecture des ordinateurs

Rapport 04

Travail pratique 4 : Codeur rotatif et écran LCD

Ecole	Haute école d'ingénierie et d'architecture de Fribourg (HEIA-FR)
Branche	Architecture des ordinateurs
Etudiants	Barras Simon, Terreaux Nicolas
Groupe	B01
Classe	ISC-IL-2d
Professeur	Supcik Jacques
GitLab	https://gitlab.forge.hefr.ch/ado/2021-2022/classe-supcik/groupe-B-01/tp04
Version/Date	V1 du 20.12.2021

1 Introduction

Ce TP consiste à réaliser un compteur avec le codeur rotatif et le joystick et l'affichage de ce compteur se fait sur le 7-segment, l'écran LCD et les LEDs qui se trouvent autour du codeur rotatif. Le premier point à réaliser est de rendre notre Discolib accessible par le CI/CD de notre projet TP04. Il faut ensuite reprendre le code de notre tp03, l'adapter et ajouter les nouvelles fonctionnalités pour arriver à faire le mini-projet comme indiqué dans la consigne.

2 Résumés

2.1 Non acquis

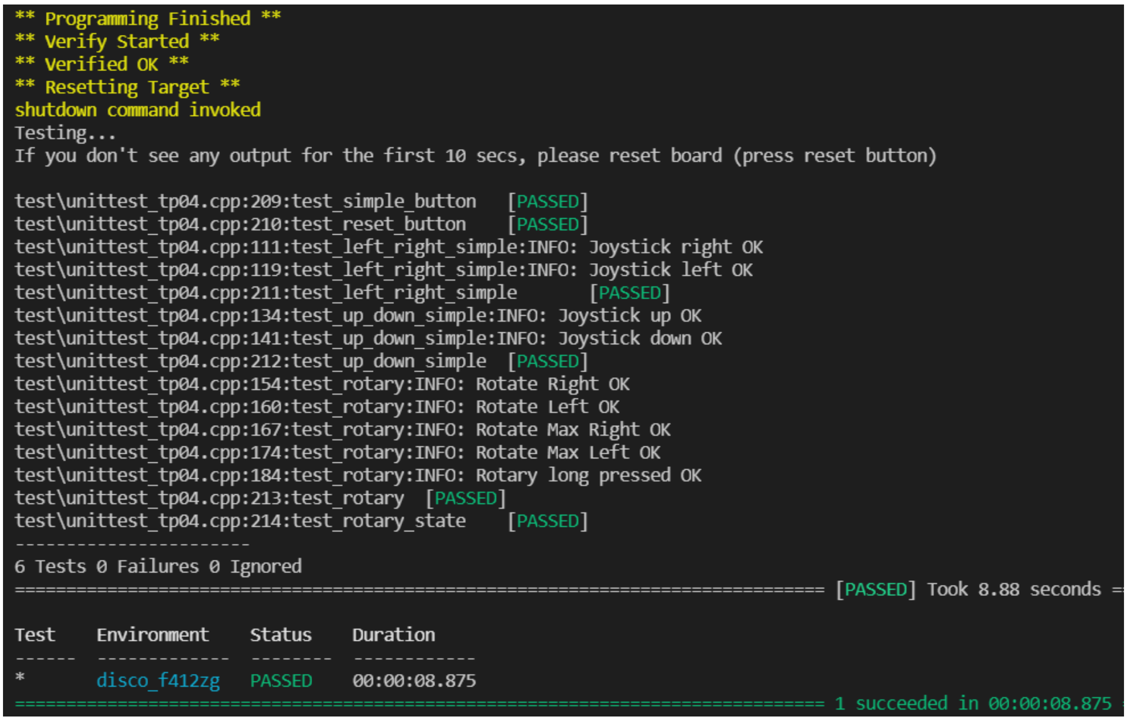
Warning -Wreorder

Durant toute l'implémentation de notre projet nous avons eu des warnings de type -Wreorder. Nous avons réussi à les résoudre mais nous avons procédé à tâtons ce qui ne devrait pas être le cas pour un ingénieur… Nous allons passer du temps sur le sujet pour que ça n'arrive plus (mais nous n'avons pas eu le temps de le faire durant ce TP).

2.2 A exercer

Tests unitaires :

Nous sommes désormais beaucoup plus à l'aise avec les tests unitaires en C++. Nous ne mettons pas ce point dans parfaitement acquis car nous savons que nos tests pourraient être plus précis et plus concis.



```
** Programming Finished **
** Verify Started **
** Verified OK **
** Resetting Target **
shutdown command invoked
Testing...
If you don't see any output for the first 10 secs, please reset board (press reset button)

test\unittest_tp04.cpp:209:test_simple_button   [PASSED]
test\unittest_tp04.cpp:210:test_reset_button    [PASSED]
test\unittest_tp04.cpp:111:test_left_right_simple:INFO: Joystick right OK
test\unittest_tp04.cpp:119:test_left_right_simple:INFO: Joystick left OK
test\unittest_tp04.cpp:211:test_left_right_simple       [PASSED]
test\unittest_tp04.cpp:134:test_up_down_simple:INFO: Joystick up OK
test\unittest_tp04.cpp:141:test_up_down_simple:INFO: Joystick down OK
test\unittest_tp04.cpp:212:test_up_down_simple  [PASSED]
test\unittest_tp04.cpp:154:test_rotary:INFO: Rotate Right OK
test\unittest_tp04.cpp:160:test_rotary:INFO: Rotate Left OK
test\unittest_tp04.cpp:167:test_rotary:INFO: Rotate Max Right OK
test\unittest_tp04.cpp:174:test_rotary:INFO: Rotate Max Left OK
test\unittest_tp04.cpp:184:test_rotary:INFO: Rotary long pressed OK
test\unittest_tp04.cpp:213:test_rotary  [PASSED]
test\unittest_tp04.cpp:214:test_rotary_state    [PASSED]
-----------------------
6 Tests 0 Failures 0 Ignored
=============================================================================== [PASSED] Took 8.88 seconds =

Test    Environment    Status    Duration
------  -------------  --------  ------------
*       disco_f412zg   PASSED    00:00:08.875
=============================================================================== 1 succeeded in 00:00:08.875
```

Liens dans l'application :

Nous avons passé beaucoup de temps à comprendre les liens entre les classes que nous devions faire dans notre programme ainsi que de savoir exactement qui fait quoi. Il nous faut encore de la pratique pour être plus efficace.

Variable (void) :

Nous avons utilisé des variables (void) dans le but de ne plus avoir d'avertissements dans le compilateur mais nous ne sommes pas sûr si nous les avons bien utilisés. Exemple :

```
void OnRepeated(int repeated)
{
    (void)repeated;
    counter_.Decrement();
};
```

2.3 Parfaitement acquis

Rotary :

Avant de commencer ce TP, nous pensions que le codeur rotatif allait nous donner du fil à retordre mais finalement, grâce aux explications durant le début du TP, nous avons rapidement compris comment il fonctionnait.

Lecture des Datasheets :

Grâce au TP03, nous avons trouvé rapidement les différents Pins et Ports que nous avions besoin dans ce TP.

3 Points importants

3.1 Retrouver des infos dans un datasheet

Comme dans le TP03, nous devons rechercher les différents ports et pins que nous aurons besoin.

Codeur rotatif et Arduino shield:

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

Signal	GPIO (socle de gauche)	GPIO (socle de droite)
ENCA_OUT	PD6/PWMA	PD5/PWMB
ENCB_OUT	PC0/AN0	PC1/AN1
SW	PD2/INT0	PD3/INT1
LATCH	PB2/SPI-SS	PB1
SCK	PB5/SPI-SCK	PB5/SPI-SCK
SDO	PB4/SPI-MISO	PB4/SPI-MISO
SDI	PB3/SPI-MOSI	PB3/SPI-MOSI

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)
ENCA_OUT	D6	D5
ENCB_OUT	A0	A1
SW	D2	D3
LATCH	D10	D9
SCK	D13	D13
SDO	D12	D12
SDI	D11	D11

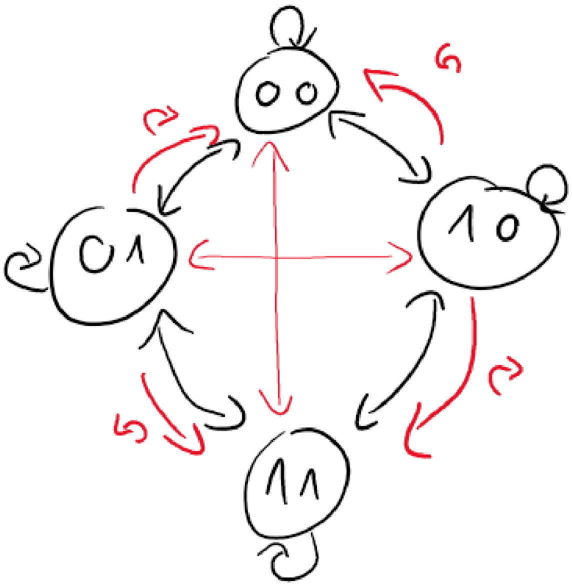
Processeur et Arduino shield:

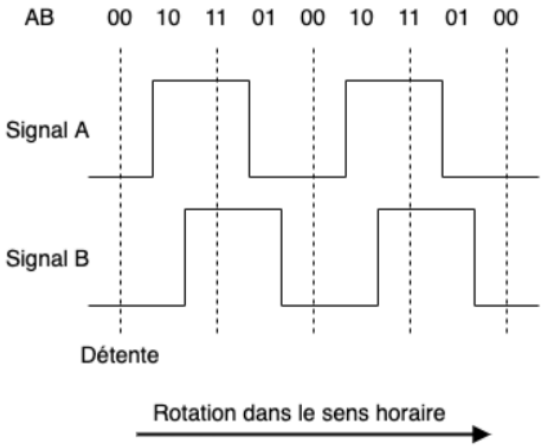
Voici la liste des différents Ports/Pins de la cible qui nous intéresse pour effectuer la programmation :

Signal	GPIO (socle de gauche)	GPIO (socle de droite)
ENCA_OUT	PF3	PF10
ENCB_OUT	PA1	PC1
SW	PG13	PF4
LATCH	PA15	PB8
SCK	PA5	PA5
SDO	PA6	PA6
SDI	PA7	PA7

3.2 Codeur rotatif (machine d’état)

Durant l’analyse nous avons fait la machine d’état du codeur rotatif la voici :





Et nous avons fait le tableau récapitulatif :

	00	01	10	11
00	-	t	t	e
01	↻	-	e	↺
10	↺	e	-	↻
11	e	t	t	-

t = transition / e = error

En résumé : Lorsque les valeurs sont **00** ou **11**, on regarde si l’état précédent était **01** ou **10**. Si c’est le cas, il faut mettre à jour le compteur sinon il ne faut rien faire.

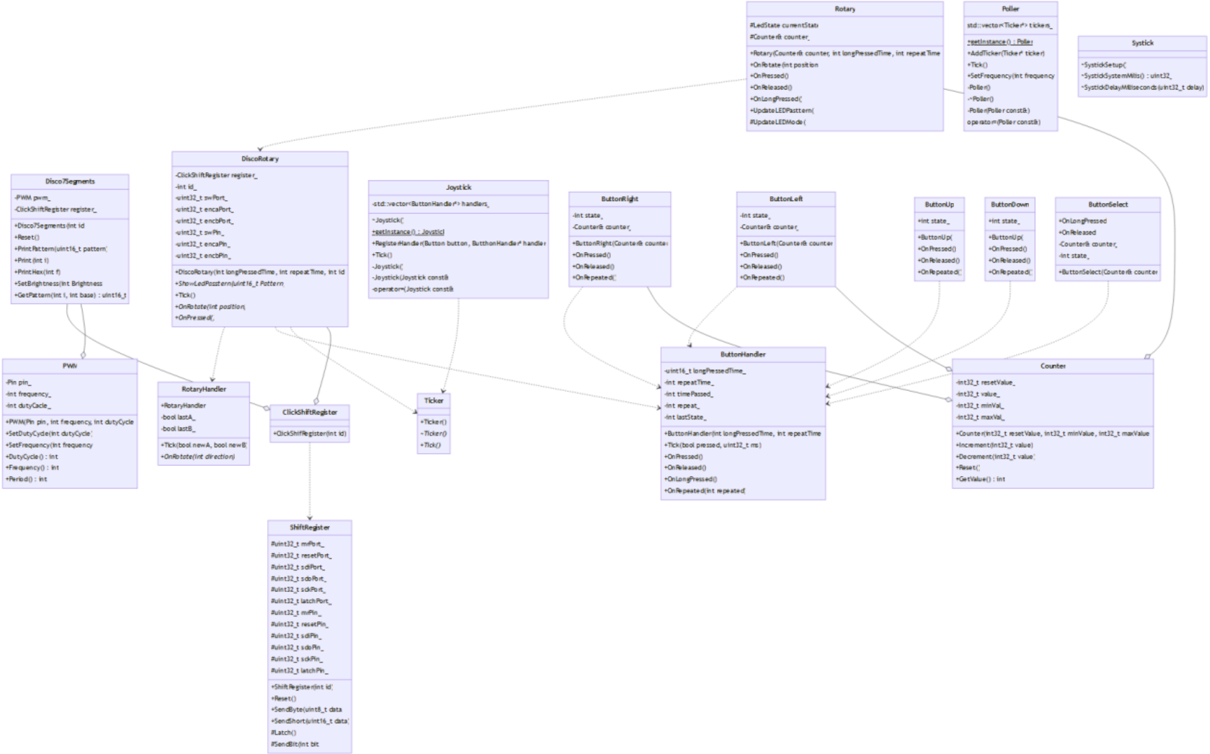
Voici le code que nous avons fait :

```
RotaryHandler::RotaryHandler()
{
    lastA_      = 0;
    lastB_      = 0;
 }

void RotaryHandler::Tick(bool a, bool b)
{
    if (a == b && lastA_ != lastB_) {
        if (lastA_ != a && lastB_ == b) {
            OnRotate(-1);
        } else {
            OnRotate(1);
        }
    }
    lastA_ = a;
    lastB_ = b;
}
```

3.3 Diagramme de classe

Le diagramme de classe suivant a été réalisé en fonction des indications dans la consigne.



Voici le lien pour une meilleure visibilité :

https://gitlab.forge.hefr.ch/ado/2021-2022/classe-supcik/groupe-B-01/tp04/-/blob/main/docs/class-diagramm.svg

4 Conclusion

Nous avons trouvé ce projet très intéressant mais extrêmement long. La partie qui nous a pris le plus de temps est la compréhension des différents liens qu'il devait y avoir entre les classes pour tout faire fonctionner ensemble. De plus, nous avons commencé par rendre Discolib accessible par le CI/CD ce qui nous a fait perdre énormément de temps pour le test de notre programme. Nous n'avions pas tout de suite vu que nous pouvions faire les changement directement dans le dossier « .pio » de notre tp04 et lorsque ça fonctionnait, tout copier-coller dans Discolib.

Nous avons aussi rajouté les longues pressions sur le Joystick (point que nous n'avions pas eu le temps de terminer lors du tp03).

Le moment où tout à fonctionné ensemble (Joystick, codeur rotatif, LEDs, LCD et 7-segment) a été particulièrement satisfaisant. Nous avons passé chacun environ **14** heures en dehors des cours pour terminer ce projet.

5 Sources

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 SIMON BARRAS <simon.barras@edu.hefr.ch>, NICOLAS TERREAUX
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Main file of the project. This project will display animation with
the rotary and show the
 * index.
 *
 * @date 2021-12-20
 * @version 0.1.0
 ***************************************************************************/

/*** INCLUDES **************************************************************/
#include <DiscoConsole.h>
#include <FontsGFX/FreeSans12pt7b.h>
#include <FontsGFX/IBMPlexMonoBold60pt7b.h>
#include <libopencm3/cm3/assert.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 <AdafruitGFX.hpp>
#include <ClickShiftRegister.hpp>
#include <Counter.hpp>
#include <Disco7Segments.hpp>
```

```
#include <DiscoAssert.hpp>
#include <DiscoLcd.hpp>
#include <Joystick.hpp>
#include <PWM.hpp>
#include <Poller.hpp>
#include <Systick.hpp>

#include "Buttons.hpp"
#include "Rotary.hpp"

/*** CONSTANTS *************************************************************/
constexpr uint8_t kMaxNumber = 99;
constexpr uint8_t kMinNumber = 0;

/*** GLOBAL VARIABLES ******************************************************/
Counter counter(kMinNumber, kMinNumber, kMaxNumber);

/*** FUNCTIONS *************************************************************/
/**
 * @brief Main loop of the program. Initialize all the components and update
vue.
 */
int main()
{
    DiscoConsoleSetup();
    SystickSetup();
    DiscoLcdSetup();

    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);
    auto backLight = PWM(PWM::PF5);
    backLight.SetDutyCycle(40);

    cm3_assert(DiscoLcdId() == kLcdST7789H2Id);
    auto gfx = DiscoLcdGFX(kLcdScreenWidth, kLcdScreenHeight);
    gfx.setFont(&IBMPlexMono_Bold60pt7b);
    gfx.setTextColor(0xFFFF);

    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);

    Joystick& joystick = Joystick::getInstance();

    joystick.RegisterHandler(Joystick::Right, new ButtonRight(counter));
    joystick.RegisterHandler(Joystick::Left, new ButtonLeft(counter));
    joystick.RegisterHandler(Joystick::Up, new ButtonUp());
    joystick.RegisterHandler(Joystick::Down, new ButtonDown());
    joystick.RegisterHandler(Joystick::Select, new ButtonSelect(counter));
    auto seg = Disco7Segments(0);
    seg.SwitchOn();
    seg.Print(15);
    Rotary rotary(counter, 2000, 2000);
    rotary.ShowLedPattern(1 << 15);

    Poller::getInstance().AddTicker(&rotary);
    Poller::getInstance().AddTicker(&joystick);

    char buffer[8];
    int oldCounter    = -1;
    int oldBrightness = -1;
    while (1) {
        if (oldBrightness != brightess) {
            seg.SetBrightness(brightess);
            oldBrightness = brightess;
        }

        int c = counter.GetValue();
        if (c != oldCounter) {
            seg.Print(counter.GetValue());
            rotary.UpdateLEDPattern();
            sprintf(buffer, "%02d", c);
            int16_t x, y;
            uint16_t w, h;
            gfx.getTextBounds(buffer, 40, 150, &x, &y, &w, &h);
            gfx.fillRect(x - 5, y - 5, w + 10, h + 10, 0x0000);
            gfx.setCursor(40, 150);
            gfx.write(buffer);
            oldCounter = c;
        }

        asm volatile("nop");
    }
}
```

Buttons.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 SIMON BARRAS <simon.barras@edu.hefr.ch>, NICOLAS TERREAUX
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Contains all implementations for the 5 buttons
 *
 * @date 2021-12-20
 * @version 0.1.0
 ****************************************************************************/

/*** INCLUDES **************************************************************/
#include "ButtonHandler.hpp"
#include "Counter.hpp"

/*** CONSTANTS *************************************************************/
const int MAX_BRIGHTNESS    = 100;
const int MIN_BRIGHTNESS    = 0;
const int MAX_VALUE         = 99;
const int MIN_VALUE         = 0;
const int LONG_PRESSED_TIME = 1000;
const int REPEAT_TIME       = 50;

/*** VARIABLES *************************************************************/
int brightess = 50;

/**
 * @brief Class that defines the behavior of the right button
 *
 */
class ButtonRight : public ButtonHandler {
   public:
    ButtonRight(Counter& counter)
        : ButtonHandler(LONG_PRESSED_TIME, REPEAT_TIME), counter_(counter),
state_{0}
    {
    }
    void OnPressed() override
    {
        if (state_ == 0) counter_.Increment();
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };
    void OnRepeated(int repeated)
    {
        (void)repeated;
        counter_.Increment();
    };
   private:
    Counter& counter_;
    int state_;
};

/**
 * @brief Class that defines the behavior of the left button
 *
 */
class ButtonLeft : public ButtonHandler {
   public:
    ButtonLeft(Counter& counter)
        : ButtonHandler(LONG_PRESSED_TIME, REPEAT_TIME), counter_(counter),
state_{0}
    {
    }
    void OnPressed() override
    {
        if (state_ == 0) counter_.Decrement();
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };
    void OnRepeated(int repeated)
    {
        (void)repeated;
        counter_.Decrement();
    };

   private:
    Counter& counter_;
    int state_;
};

/**
 * @brief Class that defines the behavior of the up button
 *
 */
class ButtonUp : public ButtonHandler {
   public:
    ButtonUp() : ButtonHandler(LONG_PRESSED_TIME, REPEAT_TIME), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && brightess != MAX_BRIGHTNESS) brightess += 5;
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };
    void OnRepeated(int repeated)
    {
        (void)repeated;
        if (brightess != MAX_BRIGHTNESS) brightess += 5;
    };

    int state_;
};

/**
 * @brief Class that defines the behavior of the down button
 *
 */
class ButtonDown : public ButtonHandler {
   public:
    ButtonDown() : ButtonHandler(LONG_PRESSED_TIME, REPEAT_TIME), state_{0} {}
    void OnPressed() override
    {
        if (state_ == 0 && brightess != MIN_BRIGHTNESS) brightess -= 5;
        state_ = 1;
    };
    void OnReleased() override { state_ = 0; };
    void OnRepeated(int repeated)
    {
        (void)repeated;
        if (brightess != MIN_BRIGHTNESS) brightess -= 5;
    };
    int state_;
};

/**
 * @brief Class that defines the behavior of the select button
 *
 */
class ButtonSelect : public ButtonHandler {
   public:
    ButtonSelect(Counter& counter) : ButtonHandler(), counter_(counter),
state_{0}
    {
    }

    void OnLongPressed() override
    {
        if (state_ == 0) {
            counter_.Reset();
            state_ = 1;
        }
    };
    void OnReleased() override { state_ = 0; };

   private:
    Counter& counter_;
    int state_;
};
```

Rotary.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 Rotary.hpp
 * @author SIMON BARRAS <simon.barras@edu.hefr.ch>, NICOLAS TERREAUX
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Contains all implementations for the Rotary
 *
 * @date 2021-11-30
 * @version 0.1.0
 ***************************************************************************/

/*** INCLUDES **************************************************************/
#include <DiscoRotary.hpp>

#include "Counter.hpp"

/*** ENUMS *****************************************************************/
enum LedState { LED_SINGLE, LED_DUAL, LED_QUAD, LED_FILL, LED_INVERTED };

/**
 * @brief class that support the rotary encoder
 *
 */
class Rotary : public DiscoRotary {
   public:
    Rotary(Counter& counter, int longPressedTime, int repeatTime)
        : DiscoRotary(longPressedTime, repeatTime), counter_(counter)
    {
    }
    void OnRotate(int position)
    {
        if (position == -1) {
            counter_.Decrement();
        } else if (position == 1) {
            counter_.Increment();
        }
    }
    void OnPressed()
    {
        printf("Rotary pressed");
        UpdateLEDMode();
    }
    void OnReleased() { UpdateLEDPattern(); }
    void OnLongPressed()
    {
        counter_.Reset();
        currentState = LED_SINGLE;
    }

    void UpdateLEDPattern()
    {
        switch (currentState) {
            case LED_SINGLE:
                ShowLedPattern(1 << (15 - counter_.GetValue() % 16));
                break;
            case LED_DUAL:
                ShowLedPattern(1 << (7 - counter_.GetValue() % 8) |
                               1 << (15 - counter_.GetValue() % 8));
                break;
            case LED_QUAD:
                ShowLedPattern(
                    1 << (3 - counter_.GetValue() % 4) | 1 << (7 -
counter_.GetValue() % 4) |
                    1 << (11 - counter_.GetValue() % 4) | 1 << (15 -
counter_.GetValue() % 4));
                break;
            case LED_INVERTED:
                ShowLedPattern(1 << counter_.GetValue() % 16);
                break;
            default:
                ShowLedPattern(0xffff << (16 - counter_.GetValue() / 6));
        }
    }

   protected:
    void UpdateLEDMode()
    {
        switch (currentState) {
            case LED_SINGLE:
                currentState = LED_DUAL;
                break;
            case LED_DUAL:
                currentState = LED_QUAD;
                break;
            case LED_QUAD:
                currentState = LED_FILL;
                break;
            case LED_FILL:
                currentState = LED_INVERTED;
                break;
            default:
                currentState = LED_SINGLE;
        }
    }

   protected:
    LedState currentState = LED_SINGLE;

   private:
    Counter& counter_;
};
```

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>

#include <PWM.hpp>

#include "ClickShiftRegister.hpp"

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)
    : register_(id), pwm_(id == 0 ? PWM::Pin::PF3 : PWM::Pin::PF10, 10000,
100)
{
    SetBrightness(100);
}

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

void Disco7Segments::PrintPattern(uint16_t pattern) {
register_.SendShort(pattern); }

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); }
```

DiscoRotary.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 DiscoRotary.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implement the interface for the rotary
 *
 * @date 2021-12-20
 * @version 0.1.0
 ****************************************************************************/

#include "DiscoRotary.hpp"

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

#include <ButtonHandler.hpp>
```

```
#include <ClickShiftRegister.hpp>
#include <Systick.hpp>
DiscoRotary::DiscoRotary(int longPressedTime, int repeatTime, int id)
    : ButtonHandler(longPressedTime, repeatTime), register_(id), id_(id)
{
    // Ports

    encaPort_ = GPIOF;
    encbPin_  = GPIO1;

    // Check if right or left part of the compound, then assign corresponding
value
    if (id_ == 0) {
        swPort_   = GPIOG;
        swPin_    = GPIO13;
        encaPin_  = GPIO3;
        encbPort_ = GPIOA;

        rcc_periph_clock_enable(RCC_GPIOG);

    } else if (id_ == 1) {
        swPort_   = GPIOF;
        swPin_    = GPIO4;
        encaPin_  = GPIO10;
        encbPort_ = GPIOC;

        rcc_periph_clock_enable(RCC_GPIOC);
    }

    rcc_periph_clock_enable(RCC_GPIOF);

    // configure GPIOs
    gpio_clear(swPort_, swPin_);
    gpio_mode_setup(swPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLDOWN, swPin_);

    gpio_clear(encaPort_, encaPin_);
    gpio_mode_setup(encaPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_PULLDOWN,
encaPin_);

    gpio_clear(encbPort_, encbPin_);
    gpio_mode_setup(encbPort_, GPIO_MODE_INPUT, GPIO_PUPD_PULLDOWN, encbPin_);
}
void DiscoRotary::ShowLedPattern(uint16_t pattern) {
register_.SendShort(pattern); }

void DiscoRotary::Tick()
{
    uint32_t now   = SystickSystemMillis();
    uint16_t value = gpio_get(swPort_, swPin_) != 0;
    ButtonHandler::Tick(value, now);

    bool enca_out = gpio_get(encaPort_, encaPin_) != 0;
    bool encb_out = gpio_get(encbPort_, encbPin_) != 0;

    RotaryHandler::Tick(enca_out, encb_out);
}
```

Poller.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 Poller.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implementation of the Poller
 *
 * @date 2021-12-20
 * @version 0.1.0
 ****************************************************************************/

#include "Poller.hpp"

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

#include <Systick.hpp>

static const int kPollingFrequency = 2000;
Poller& instance_                  = Poller::getInstance();

Poller::Poller()
{
    // tickers_;
    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);
    // Initialize Timer
    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);
}

Poller& Poller::getInstance()
{
    static Poller* instance = new Poller;
    return *instance;
}

void Poller::AddTicker(Ticker* ticker) { tickers_.push_back(ticker); }

void Poller::Tick()
{
    for (auto& ticker : tickers_) {
        ticker->Tick();
    }
}

void Poller::SetFrequency(int frequency) { timer_set_period(TIM2,
rcc_apb2_frequency / frequency); }

void tim2_isr(void)
{
    timer_clear_flag(TIM2, TIM_SR_UIF);  // acknowledge interrupt
    Poller::getInstance().Tick();
}
```

RotaryHandler.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 RotaryHandler.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implement the action that will be do when the user rotate the rotary
 *
 * @date 2021-12-20
 * @version 0.1.0
 *************************************************************************/

#include "RotaryHandler.hpp"

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

RotaryHandler::RotaryHandler()
{
    lastA_ = 0;
    lastB_ = 0;
}

void RotaryHandler::Tick(bool a, bool b)
{
    if (a == b && lastA_ != lastB_) {
        if (lastA_ != a && lastB_ == b) {
            OnRotate(-1);
        } else {
            OnRotate(1);
        }
    }
    lastA_ = a;
    lastB_ = b;
}
```

ClickShiftRegister.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 ClickShiftRegister.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Implementation of the ClickShiftRegister
 *
 * @date 2021-12-20
 * @version 0.1.0
 ***************************************************************************/

#include "ClickShiftRegister.hpp"

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

#include "ShiftRegister.hpp"
// ------- Constructor ---------

ClickShiftRegister::ClickShiftRegister(int id) : ShiftRegister(id) {}
```

ShiftRegister.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 ShiftRegister.cpp
 * @author Barras Simon <simon.barras@edu.hefr.ch>, Terreaux Nicolas
<nicolas.terreaux@edu.hefr.ch>
 *
 * @brief Send bits to the shift register
 *
 * @date 2021-12-20
 * @version 0.1.0
 ***************************************************************************/

#include "ShiftRegister.hpp"

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

ShiftRegister::ShiftRegister(int id)
{
    // Ports
    mrPort_  = GPIOC;
    sckPort_ = GPIOA;
    sdoPort_ = GPIOA;
    sdiPort_ = GPIOA;
    sckPin_  = GPIO5;
    sdoPin_  = GPIO6;
    sdiPin_  = GPIO7;

    // Check if right or left part of the compound, then assign corresponding
value
    if (id == 0) {
        latchPort_ = GPIOA;
        mrPin_     = GPIO4;
        latchPin_  = GPIO15;

    } else if (id == 1) {
        latchPort_ = GPIOB;
        mrPin_     = GPIO3;
        latchPin_  = GPIO8;
        rcc_periph_clock_enable(RCC_GPIOB);
    }

    rcc_periph_clock_enable(RCC_GPIOA);
    rcc_periph_clock_enable(RCC_GPIOC);

    // configure GPIOs
    gpio_set(mrPort_, mrPin_);
    gpio_mode_setup(mrPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, mrPin_);

    gpio_clear(sckPort_, sckPin_);
    gpio_mode_setup(sckPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, sckPin_);

    gpio_clear(sdoPort_, sdoPin_);
    gpio_mode_setup(sdoPort_, GPIO_MODE_INPUT, GPIO_PUPD_NONE, sdoPin_);

    gpio_clear(sdiPort_, sdiPin_);
    gpio_mode_setup(sdiPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, sdiPin_);

    gpio_clear(latchPort_, latchPin_);
    gpio_mode_setup(latchPort_, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, latchPin_);
}

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

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

void ShiftRegister::SendByte(uint8_t data)
{
    for (int i = 0; i < 8; i++) {
        SendBit((data & (1 << i)));
    }
    Latch();
}

void ShiftRegister::SendShort(uint16_t data)
{
    for (int i = 0; i < 16; i++) {
        SendBit((data & (1 << i)));
    }
    Latch();
}
```

Unittest_tp04.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.
 *
 * @date 2021-12-20
 * @version 0.1.0
 ***************************************************************************/
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/timer.h>
#include <unity.h>

#include <Counter.hpp>
#include <Rotary.hpp>

#include "ButtonHandler.hpp"
#include "Buttons.hpp"
#include "Joystick.hpp"
#include "Systick.hpp"
constexpr uint8_t kMaxNumber = 99;
constexpr uint8_t kMinNumber = 0;
static void setup(void)
{
    // Enable clock
    rcc_clock_setup_pll(&rcc_hse_8mhz_3v3[RCC_CLOCK_3V3_84MHZ]);
    SystickSetup();
}

class TestRotary : public Rotary {
   public:
    TestRotary(Counter& counter) : Rotary(counter, 2000, 2000) {}
    LedState getState() { return currentState; }
};

class SimpleButton : public ButtonHandler {
   public:
    SimpleButton() : ButtonHandler(), state_{0} {};
    void OnPressed() override { state_ = 1; };
    int state_;
};

void test_simple_button(void)
{
    SimpleButton b{};
    uint32_t timer = 0;
    b.Tick(false, timer += 10);  // First tick is needed to initialize the button
    TEST_ASSERT_EQUAL(b.state_, 0);
    b.Tick(false, timer += 10);  // Still released
    TEST_ASSERT_EQUAL(b.state_, 0);
    b.Tick(true, timer += 10);  // Pressed
    TEST_ASSERT_EQUAL(b.state_, 1);
    b.Tick(true, timer += 60000);  // Long pressed
    TEST_ASSERT_EQUAL(b.state_, 1);
    b.Tick(false, timer += 10);  // Released
    TEST_ASSERT_EQUAL(b.state_, 1);
}

void wait_2_sec()
{
    for (int i = 0; i < 54000000; i++) {
        asm volatile("nop");
    }
}
void test_reset_button(void)
{
    Counter counter(kMinNumber, kMinNumber, kMaxNumber);
    uint32_t timer     = 0;
    ButtonRight right  = ButtonRight(counter);
    ButtonSelect reset = ButtonSelect(counter);
    right.Tick(false, timer += 10);
    right.Tick(true, timer += 10);
    right.Tick(false, timer += 10);
    right.Tick(true, timer += 10);
    reset.Tick(false, timer += 10);
    TEST_ASSERT_EQUAL(2, counter.GetValue());
    reset.Tick(true, timer += 10);
    reset.Tick(true, timer += 1001);
    reset.Tick(false, timer += 10);
    TEST_ASSERT_EQUAL(0, counter.GetValue());
```

```
}

void test_left_right_simple(void)
{
    Counter counter(kMinNumber, kMinNumber, kMaxNumber);
    uint32_t timer    = 0;
    ButtonRight right = ButtonRight(counter);
    ButtonLeft left   = ButtonLeft(counter);
    right.Tick(false, timer += 10);
    right.Tick(true, timer += 10);
    right.Tick(false, timer += 10);
    right.Tick(true, timer += 10);
    right.Tick(false, timer += 10);

    TEST_ASSERT_EQUAL(2, counter.GetValue());
    TEST_MESSAGE("Joystick right OK");

    left.Tick(false, timer += 10);
    left.Tick(true, timer += 10);
    left.Tick(false, timer += 10);
    left.Tick(true, timer += 10);
    left.Tick(false, timer += 10);
    TEST_ASSERT_EQUAL(0, counter.GetValue());
    TEST_MESSAGE("Joystick left OK");
}

void test_up_down_simple(void)
{
    uint32_t timer  = 0;
    ButtonDown down = ButtonDown();
    ButtonUp up     = ButtonUp();
    down.Tick(true, timer += 10);
    down.Tick(false, timer += 10);
    down.Tick(true, timer += 10);
    down.Tick(false, timer += 10);
    down.Tick(true, timer += 10);
    down.Tick(false, timer += 10);
    TEST_ASSERT_EQUAL(35, brightess);
    TEST_MESSAGE("Joystick up OK");

    up.Tick(false, timer += 10);
    up.Tick(true, timer += 10);
    up.Tick(false, timer += 10);
    up.Tick(true, timer += 10);
    TEST_ASSERT_EQUAL(45, brightess);
    TEST_MESSAGE("Joystick down OK");
}

void test_rotary(void)
```

```
{
    Counter counter(kMinNumber, kMinNumber, kMaxNumber);
    Rotary rotary = Rotary(counter, 2000, 2000);

    // Turn the rotary ten times to the right
    for (int i = 0; i < 10; i++) {
        rotary.OnRotate(1);
    }
    TEST_ASSERT_EQUAL(10, counter.GetValue());
    TEST_MESSAGE("Rotate Right OK");

    // Turn the rotary twice to the left
    rotary.OnRotate(-1);
    rotary.OnRotate(-1);
    TEST_ASSERT_EQUAL(8, counter.GetValue());
    TEST_MESSAGE("Rotate Left OK");

    // Turn the rotary 100 times to the right
    for (int i = 0; i < 100; i++) {
        rotary.OnRotate(1);
    }
    TEST_ASSERT_EQUAL(99, counter.GetValue());
    TEST_MESSAGE("Rotate Max Right OK");

    // Turn the rotary 111 times to the left
    for (int i = 0; i < 111; i++) {
        rotary.OnRotate(-1);
    }
    TEST_ASSERT_EQUAL(0, counter.GetValue());
    TEST_MESSAGE("Rotate Max Left OK");

    // Turn the rotary 2 times to the right
    rotary.OnRotate(1);
    rotary.OnRotate(1);
    TEST_ASSERT_EQUAL(2, counter.GetValue());

    // Long press on the rotary
    rotary.OnLongPressed();
    TEST_ASSERT_EQUAL(0, counter.GetValue());
    TEST_MESSAGE("Rotary long pressed OK");
}

void test_rotary_state(void)
{
    Counter counter(kMinNumber, kMinNumber, kMaxNumber);
    TestRotary rotary(counter);
    TEST_ASSERT_EQUAL(LedState::LED_SINGLE, rotary.getState());
    rotary.OnPressed();
    TEST_ASSERT_EQUAL(LedState::LED_DUAL, rotary.getState());
    rotary.OnPressed();
    TEST_ASSERT_EQUAL(LedState::LED_QUAD, rotary.getState());
    rotary.OnPressed();
    TEST_ASSERT_EQUAL(LedState::LED_FILL, rotary.getState());
    rotary.OnPressed();
    TEST_ASSERT_EQUAL(LedState::LED_INVERTED, rotary.getState());
    rotary.OnPressed();
    TEST_ASSERT_EQUAL(LedState::LED_SINGLE, rotary.getState());
}

int main()
{
    setup();
    UNITY_BEGIN();
    wait_2_sec();
    RUN_TEST(test_simple_button);
    RUN_TEST(test_reset_button);
    RUN_TEST(test_left_right_simple);
    RUN_TEST(test_up_down_simple);
    RUN_TEST(test_rotary);
    RUN_TEST(test_rotary_state);
    UNITY_END();
    return 0;
}
```