HAUTE ÉCOLE LOUVAIN EN HAINAUT

Micro-contrôleur – Groupe B

Projet

3 janvier 2026

Bataille Achille
Deroy Julien
Papier Nattan
Rombaut Romain

Année académique 2025–2026

HELHa Haute École Louvain en Hainaut

Cti Commission des titres d'ingénieur

Différentes tâches du projet

Ce document rassemble et explique toutes les modifications et ajouts réalisés dans le projet de micro-contrôleur pour la gestion de l'éclairage. Il couvre les aspects suivants :

- Creation d'une machine d'état pour la gestion des lampes
- Creation d'une fonction PWM pour faire varier la luminosité des lampes
- Gestion du GPIO Expander (MCP23017),
- Integration d'une SRAM (23LC1024),
- Mise en place de l'eclaire en fonction de l'heure (RTC)

1 Machine d'état pour la gestion des lampes

Le fichier **menu.h** contient la définition de la machine d'état utilisée pour gérer les interactions utilisateur via le menu.

```
#ifndef PGM_P
#define PGM_P const rom char *
#endif

// Définition d'une transition d'état
typedef struct PROGMEM{
    unsigned char state;
    unsigned char input;
    unsigned char nextstate;
} MENU_NEXTSTATE;

// Structure d'un état de menu
typedef struct PROGMEM{
    unsigned char state;
    PGM_P pText;
    char (*pFunc)(char input);
} MENU_STATE;

// Table de transitions stockée en PROGMEM
const MENU_NEXTSTATE menu_nextstate[] PROGMEM = {
    {ST_TXT_START, BTN_ENTER_SHORT,  ST_TXT_SWITCH},
    {ST_TXT_START, BTN_ENTER_DOUBLE, ST_TXT_ALL_OFF},
    {ST_TXT_START, BTN_ENTER_LONG,   ST_TXT_T_UP},
    // ... autres transitions ...
};


// Table des états (texte + fonction associée)
const MENU_STATE Menu_State[] PROGMEM = {
    {ST_TXT_START,   txt_start,   NULL},
    {ST_TXT_SWITCH,  txt_switch,  &Light_Switch},
    {ST_TXT_ALL_OFF, txt_alloff,  &Double_Push_Action},
    {ST_TXT_T_UP,    txt_t_up,    &Light_Trimming_Up},
    // ...
};
```

Listing 1 – Extrait simplifié de l'implémentation de la machine d'état (menu.h)

2 PWM pour la variation de luminosité

Un PWM a été intégré pour la gestion de la luminosité des lampes. L'implémentation principale se trouve dans **main.c**.

Les principales composantes sont :

- Callbacks `Switch_LED_DIM_ON` et `Switch_LED_DIM_OFF` : utilisés pour créer la modulation de largeur d'impulsion en allumant et éteignant la lampe à des intervalles spécifiques.
- Trimming manuel : permet à l'utilisateur d'ajuster la luminosité via des appuis longs sur les boutons, avec sauvegarde de la valeur en SRAM.
- Fonction `Auto_PWM_Control()` : callback périodique qui ajuste la luminosité de **Lamp2** en fonction d'une valeur stockée en SRAM.
- Stockage en SRAM : les valeurs de luminosité et le mode automatique sont persistés en SRAM pour conserver l'état entre les redémarrages.

Voici un extrait de la fonction de base du PWM :

```
void PWM_update(void){
    static float value_dim_float;
    char buffer[10];
    Usart0_Tx_String("PWM_UPDATE");
    Usart0_Tx(0X0D);

    Callbacks_Remove_Timer(IDCB_Switch_LED_DIM_ON);
    IDCB_Switch_LED_DIM_ON = Callbacks_Record_Timer(Switch_LED_DIM_ON, value_dim);

    itoa(value_dim,buffer,10);
    Usart0_Tx_String(buffer);
    Usart0_Tx(0X0D);

    if(value_dim_float <= 100.0){
        if(value_dim_float <= 10.0){
            value_dim_float += 2.0;
        }

        else if((10.0 < value_dim_float) && (value_dim_float < 100.0)) {
            value_dim_float += 10.0;
        }
    }
    else{
        value_dim_float= 1.0;
    }

    value_dim = (int)value_dim_float;
    {
        unsigned int vd = (unsigned int)value_dim;
        unsigned char percent = 0;
        if (vd >= 1 && vd <= 200)
        {
            int p = 101 - (int)vd; // inverse mapping of period -> percent
            if (p < 0) p = 0;
            if (p > 100) p = 100;
            percent = (unsigned char)p;
        }
        LAMP2_PWM_Value = percent;
        LAMP2_PWM_VALUE_WRITE;
    }

}
```

Listing 2 – Extrait simplifié de l'implémentation du PWM (main.c)

3 Gestion du GPIO Expander (MCP23017)

Le GPIO Expander MCP23017 est géré via de l'I2C implémenté dans **EXPANDER_MCP23017.c**. Ce fichier permet de contrôler les broches supplémentaires fournies par le MCP23017, qui sont utilisées pour gérer les lampes.

3.1 Correction du fichier EXPANDER_MCP23017.c

La fonction `Expander_Gpio_Ctrl()` a été corrigée pour assurer un contrôle correct des broches GPIO du MCP23017. La version corrigée de la fonction est la suivante :

```
void Expander_Gpio_Ctrl_test(unsigned char GPIOPort, unsigned char GPIOPin, unsigned char GPIOPortState)
{
    unsigned char currentState;
    unsigned char mask = (1 << GPIOPin);
    unsigned char latchAddress;

    if (GPIOPort == GPIOA) {
        latchAddress = OLATA;
    } else if (GPIOPort == GPIOB) {
        latchAddress = OLATB;
    } else {
        return;
    }

    (latchAddress);

    if (GPIOPortState == HIGH)
    {
        currentState |= mask;
    }
    else if (GPIOPortState == LOW)
    {
        currentState &= ~mask;
    }

    Expander_Write(GPIOPort, currentState);
}
```

Listing 3 – Correction de la fonction Expander_Gpio_Ctrl (EXPANDER_MCP23017.c)

Cette amélioration garantit que les broches GPIO sont correctement contrôlées en fonction de l'état souhaité. (La fonction originale avait des erreurs dans la manipulation des bits et l'adresse du registre de latch selon la datasheet.)

3.2 Implémentation dans le projet

Le MCP23017 est initialisé dans **main.c** pour configurer les broches en entrée/sortie selon les besoins du projet. Les fonctions de contrôle des lampes utilisent ensuite le MCP23017 pour allumer/éteindre les lampes via les broches étendues.

Afin de faciliter la gestion des lampes, un fichier d'en-tête **Light_Control.h** a été créé. Ce fichier encapsule les fonctions de contrôle des lampes, rendant le code plus modulaire et facile à maintenir.

Dans **Light_Control.h**, des cross-definitions et des prototypes de fonctions sont définis pour gérer les lampes.

Extrait de **Light_Control.h** :

```

#define LAMP1_PIN     EXP_GPIOB4
#define LAMP2_PIN     EXP_GPIOB5
...

#define BTN1_PIN      EXP_GPIOB0
#define BTN2_PIN      EXP_GPIOB1
...

#define LAMP1_ON      Expander_Gpio_Ctrl_test(GPIOB, LAMP1_PIN, HIGH);
#define LAMP1_OFF     Expander_Gpio_Ctrl_test(GPIOB, LAMP1_PIN, LOW);
#define LAMP2_ON      Expander_Gpio_Ctrl_test(GPIOB, LAMP2_PIN, HIGH);
#define LAMP2_OFF     Expander_Gpio_Ctrl_test(GPIOB, LAMP2_PIN, LOW);
...
```

Listing 4 – Exemples de cross-definitions dans Light_Control.h

4 Intégration de la SRAM

L'integration de la SRAM 23LC1024 a été implémentée via les bibliothèques fournies **SRAM23LC1024.c** et **SRAM23LC1024.h**.

L'assignation des adresses ainsi que des cross-definitions a été réalisée dans **SRAMConf.h** pour faciliter la lecture.

Cette assignation ce fait comme suis :

```

#define LAMP1_Address                 0x00
#define LAMP2_Address                 0x04
#define LAMP3_Address                 0x08
...

#define LAMP1WRITE                SRAM_Write(LAMP1_Address, LAMP1_State)
#define LAMP2WRITE                SRAM_Write(LAMP2_Address, LAMP2_State)
#define LAMP3WRITE                SRAM_Write(LAMP3_Address, LAMP3_State)
...

#define LAMP1READ                 SRAM_Read(LAMP1_Address)
#define LAMP2READ                 SRAM_Read(LAMP2_Address)
#define LAMP3READ                 SRAM_Read(LAMP3_Address)
...
```

Listing 5 – Extrait de SRAMConf.h

Le changement d'état des lampes est d'abord sauvegardé dans la SRAM avant d'être appliqué a chaque ticks du programme dans la boucle principale via la Callbacks `Lamp_SRAM_Update()`.

5 Mise en place de l'éclairage en fonction de l'heure (RTC)

Pour gérer l'éclairage en fonction de l'heure, le module RTC DS1307 est utilisé. L'implémentation a été réalisée par un autre groupe qui nous fournira une valeur de poucentage de luminosité en fonction de l'heure actuelle. Cette valeur est ensuite utilisée dans la fonction `Auto_PWM_Control()` pour ajuster automatiquement la luminosité de **Lamp2**.

Cette fonction est semblable au `PWM_update()` mais elle lit la valeur de luminosité depuis la SRAM et ajuste la luminosité en conséquence.

Conclusion

Ce document a présenté les différentes tâches réalisées dans le cadre du projet de micro-contrôleur pour la gestion de l'éclairage. Chaque section a détaillé les modifications apportées, les fichiers concernés, et les extraits de code pertinents pour illustrer les implémentations.