HELHa Charleroi - Informatique Industrielle - 3BINI - 2022/2023

Internet des objets (IoT) – TP03 – TOFFOLO Nicolas

1. CAHIER DES CHARGES

Réalisation d'un montage et programmation du processus suivant :

Affichage d'une procédure sur un écran LCD :

1. L'écran LCD passe en mode veille si aucune activité n'est détectée durant 5s

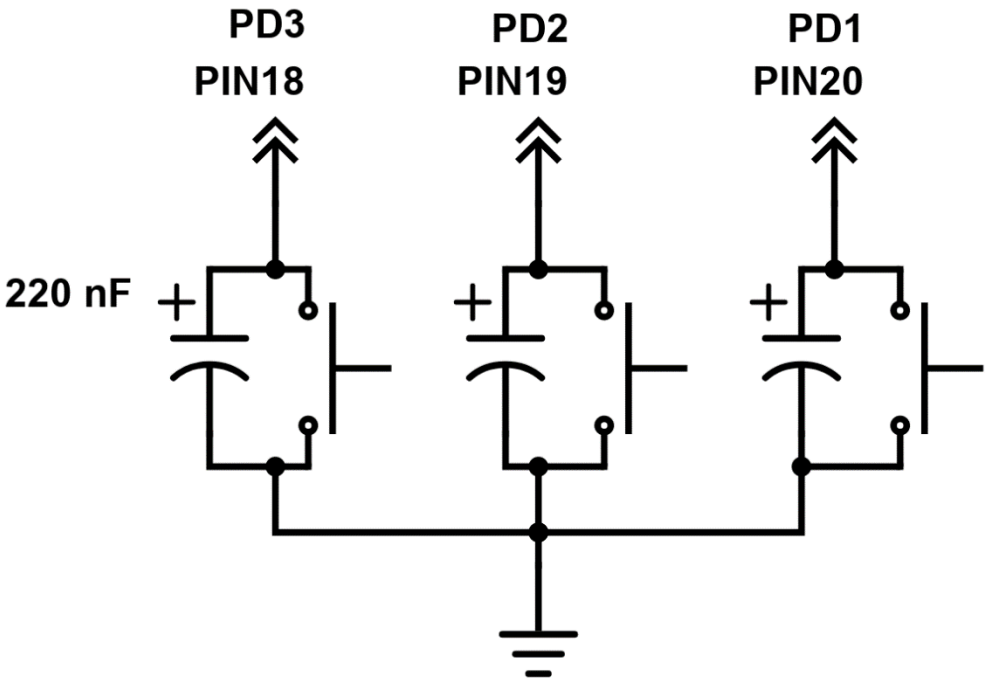
2. L'appui sur le bouton 1 active l'écran LCD

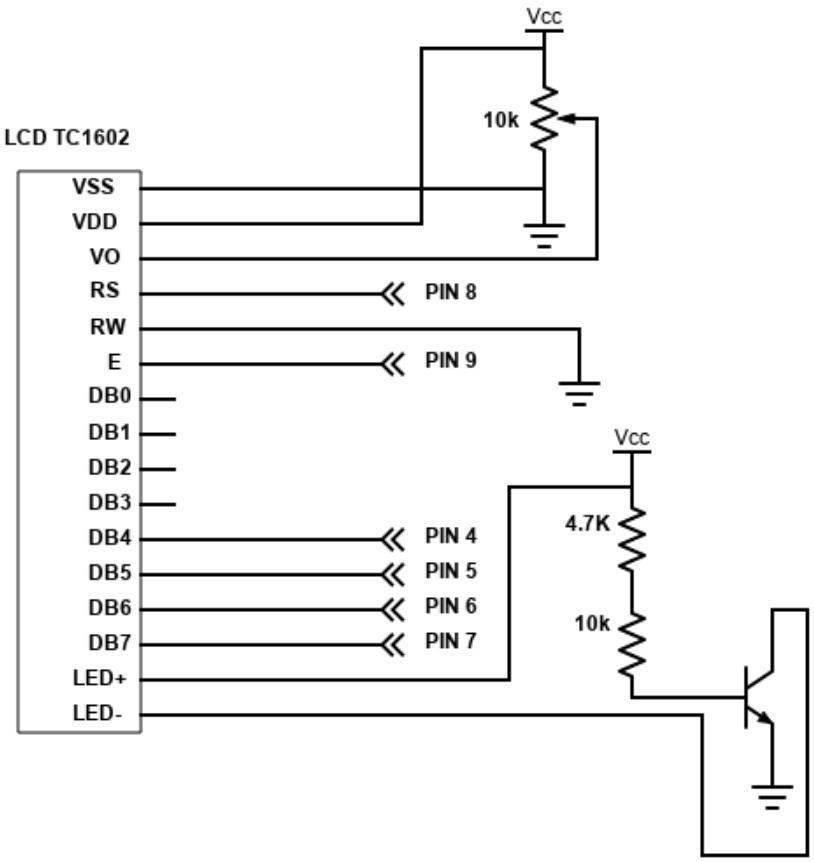
3. En gardant l'appui sur le bouton 1, chaque appui sur le bouton 2 entraine alternativement l'affichage sur l'écran LCD de « Mode Manuel » et l'allumage de la LED1 ou l'affichage de « Mode Automatique » et l'allumage de la LED 2

4. En gardant l'appui sur le bouton 1et uniquement en « Mode Automatique » un 1er appui sur le bouton 3 entraine l'affichage sur la deuxième ligne de l'écran LCD de « PGM 01» et un 2ème appui l'affichage de « PGM 02 » et ainsi de suite jusque « PGM12 »

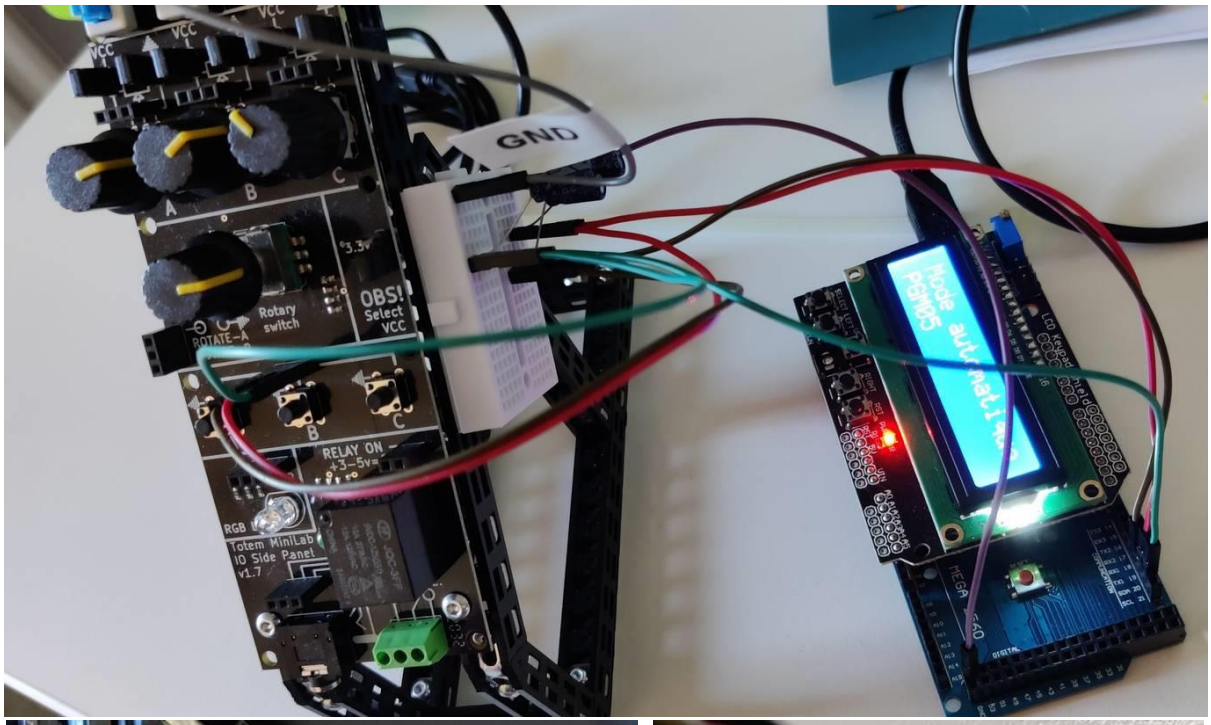
Obligation d'utiliser les registres disponibles pour les entrées et sorties.

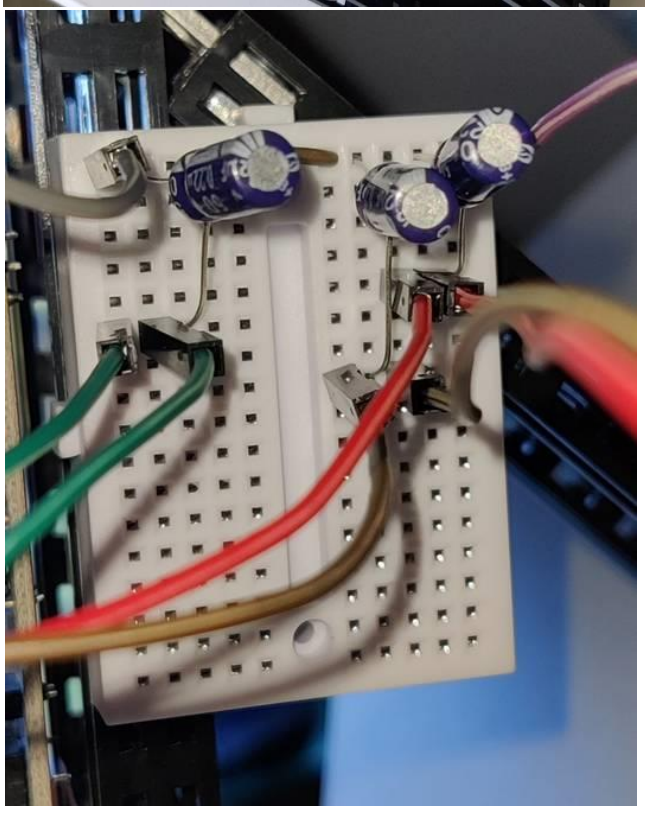
2. SCHÉMA ÉLECTRONIQUE





3. MONTAGE


GND
Rotary switch
OBS!
Select VCC
RELAY ON
+3-5vs
Totem MiniLab
IO Side Panel
v1.7
Mode auto/manuel
PGM05




COMMUNCATION
RESET
DIGITAL
MEGA 2560
GND

4. MATÉRIEL

4.1. Pins d'interruption

Hormis les pins 2 et 3 que tous les types d'Arduino peuvent utiliser en interruption, les Arduino Mega en ont 4 de plus :

J'ai utilisé les pins 18 à 20, car elles avaient l'avantage de rester accessibles malgré le shield LCD.

4.2. Condensateur 220 nF

Les condensateurs présents dans le montage permettent un anti-rebonds des boutons. En effet, ils doivent se charger après chaque appui, ce qui prend un certain temps et évite les doubles appuis fantôme.

Sachant que les résistances de l'Arduino sont « garanties d'être entre 20kΩ and 50kΩ », le temps pour que le condensateur se recharge complètement est compris entre **22 ms** et **55 ms** (5*RC) :

$$5 . R . C = 5 . 20\,000 . 220 . 10^{-9} = 0{,}022 \text{ sec}$$
$$= 5 . 50\,000 . 220 . 10^{-9} = 0{,}055 \text{ sec}$$

5. DÉCLARATION DES VARIABLES

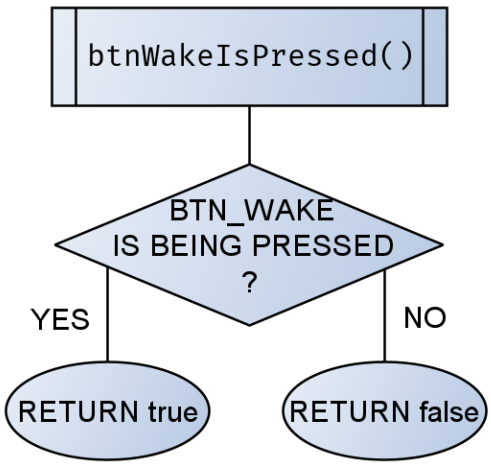
Global

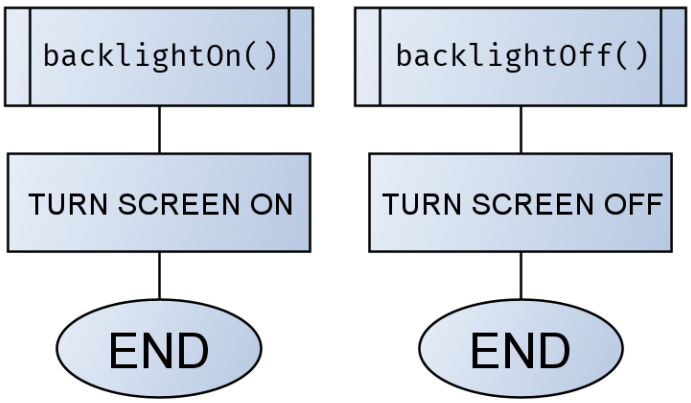
```
const byte BTN_WAKE, BTN_MODE, BTN_MODE_PIN, BTN_PGM, BTN_PGM_PIN, BACKLIGHT
const byte RS, ENABLE, D4, D5, D6, D7, COLS, ROWS
LiquidCrystal lcd
const int SLEEP_DELAY
unsigned long prevTime
const char* MODES[2]
selMode_t selMode ← enum selMode_t { MANUAL, AUTO }
byte selPgm
```

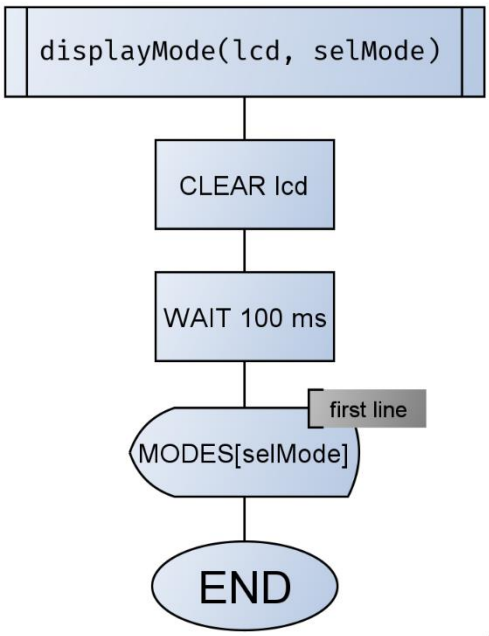
Local

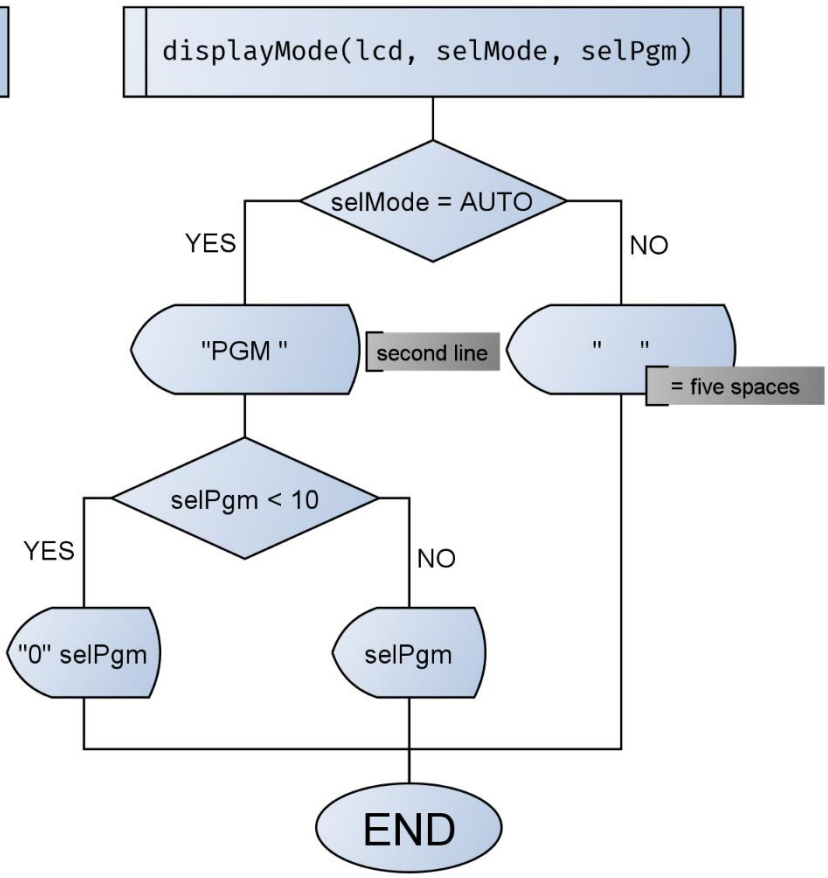
```
const byte BTN_MASK
```

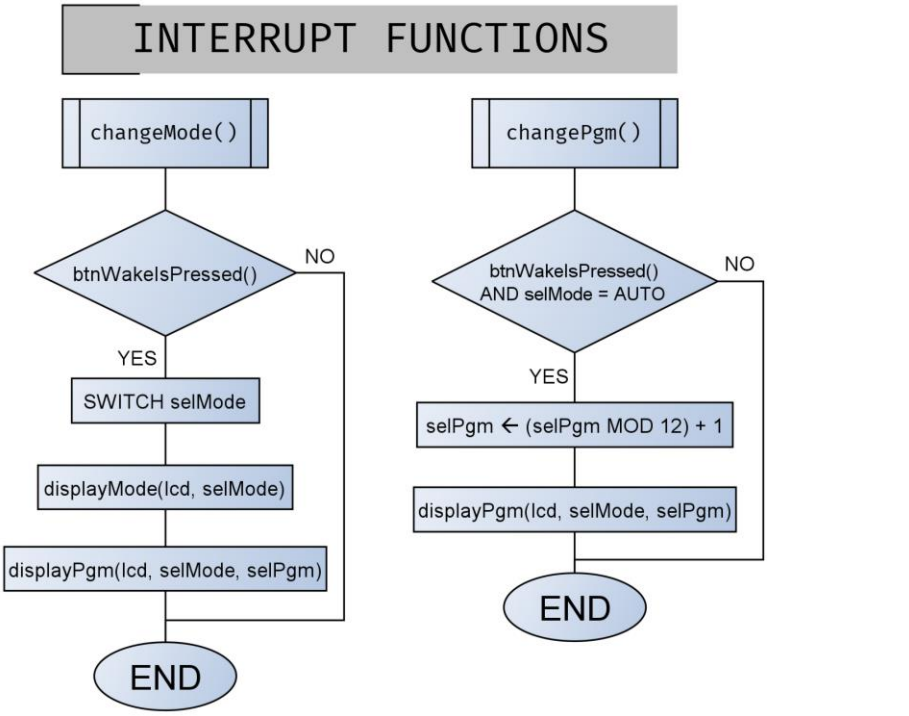
6. ORDINOGRAMME

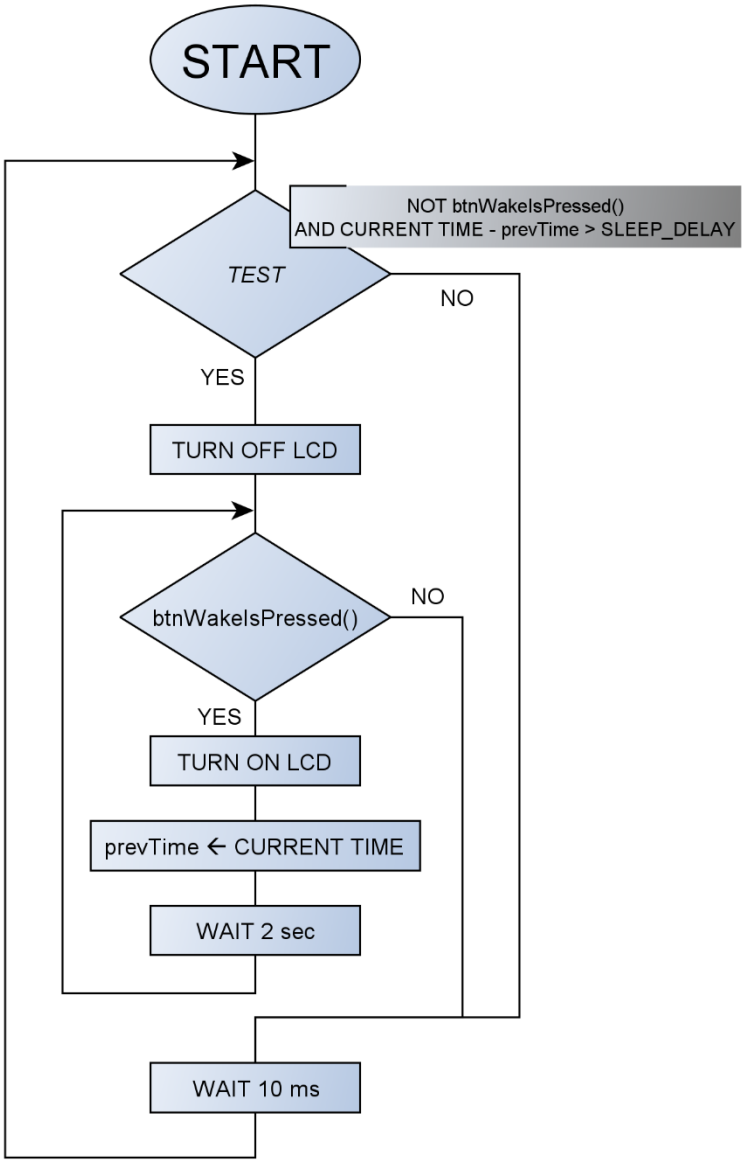









INTERRUPT FUNCTIONS
changeMode()
btnWakeIsPressed()
NO
YES
SWITCH selMode
displayMode(lcd, selMode)
displayPgm(lcd, selMode, selPgm)
END
changePgm()
btnWakeIsPressed()
AND selMode = AUTO
NO
YES
selPgm ← (selPgm MOD 12) + 1
displayPgm(lcd, selMode, selPgm)
END


START
NOT btnWakeIsPressed()
AND CURRENT TIME - prevTime > SLEEP_DELAY
TEST
NO
YES
TURN OFF LCD
btnWakeIsPressed()
NO
YES
TURN ON LCD
prevTime ← CURRENT TIME
WAIT 2 sec
WAIT 10 ms

7. CODE

7.1. Avec registres

```
#include <Arduino.h>
#include <LiquidCrystal.h>

#define BTN_WAKE 1 << PD3
#define btnWakeIsPressed() (PIND & BTN_WAKE) == 0
#define BTN_MODE 1 << PD2
#define BTN_MODE_PIN 19
#define BTN_PGM 1 << PD1
#define BTN_PGM_PIN 20

#define BACKLIGHT 1 << PB4 // Pin 10
#define backlightOn() PORTB |= BACKLIGHT
#define backlightOff() PORTB &= ~BACKLIGHT

const byte RS = 8;
const byte ENABLE = 9;
const byte D4 = 4;
const byte D5 = 5;
const byte D6 = 6;
const byte D7 = 7;

LiquidCrystal lcd(RS, ENABLE, D4, D5, D6, D7);

const byte COLS = 16;
const byte ROWS = 2;

unsigned long prevTime = 0;
const int SLEEP_DELAY = 5000;

// Variables pour les modes et les programmes
const char *MODES[] = {"Mode manuel", "Mode automatique"};

enum selMode_t
{
    MANUAL, // 0
    AUTO    // 1
};

byte selPgm = 1;

selMode_t selMode = MANUAL;



void displayMode(LiquidCrystal lcd, selMode_t selMode)
{
    lcd.clear();
    delay(100);
    lcd.setCursor(0, 0);
    lcd.print(MODES[selMode]);
}
```

```
void displayPgm(LiquidCrystal lcd, selMode_t selMode, int selPgm)
{
    if (selMode == AUTO)
    {
        lcd.setCursor(0, 1);
        lcd.write("PGM");
        selPgm < 10 && lcd.write("0");
        lcd.print(selPgm);
    }
    else
    {
        lcd.setCursor(0, 1);
        lcd.write("     ");
    }
}

void changeMode()
{
    if (btnWakeIsPressed())
    {
        selMode = selMode == AUTO ? MANUAL : AUTO;
        displayMode(lcd, selMode);
        displayPgm(lcd, selMode, selPgm);
    }
}

void changePgm()
{
    if (btnWakeIsPressed() && selMode == AUTO)
    {
        selPgm = (selPgm % 12) + 1; // Min 1, max 12
        displayPgm(lcd, selMode, selPgm);
    }
}


void setup()
{
    const byte BTN_MASK = (BTN_WAKE | BTN_MODE | BTN_PGM); // == 0b11100000

    DDRD |= BTN_MASK;  // Mis à "output"
    PORTD |= BTN_MASK; // pullup

    DDRB |= BACKLIGHT; // Backlight en output

    Serial.begin(115200);
    attachInterrupt(digitalPinToInterrupt(BTN_MODE_PIN), changeMode,
FALLING);
    attachInterrupt(digitalPinToInterrupt(BTN_PGM_PIN), changePgm, FALLING);

    lcd.begin(COLS, ROWS);
    lcd.write("TP3 - IoT");
    backlightOn();
    delay(2000);
    displayMode(lcd, selMode);
}
```

```
void loop()
{
    // Éteins l'écran après 5 secondes d'inactivité
    if (!btnWakeIsPressed() && millis() - prevTime > SLEEP_DELAY)
    {
        backlightOff();
    }

    while (btnWakeIsPressed())
    {
        backlightOn();
        prevTime = millis();
        delay(2000);
    }

    delay(10);
}
```