TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATION

Laboratory work 5.1

Subject: Control – ON-OFF Hysteresis

Done by: **Konjevic Alexandra,**
st. gr. FAF-213

Verified by: **Moraru Dumitru,**
university lecturer

CHIȘINĂU, 2024

THE TASKS OF THE LABORATORY WORK

An application based on MCU will be implemented, that will have a control system for:

- Open loop control of a led
- ON/OFF control of a led with a sensor (of temperature, humidity etc.), with hysteresis, and that will have a relay.
- Set point that can be set on one of the following options:
  - Potentiometer
  - Two buttons for up/down
  - Encoder sensor
  - Keypad
  - Serial interface
- Set point and current value will be displayed on an LCD

PROGRESS OF THE WORK

1.1 Description

The provided program is an application designed to run on a microcontroller unit (MCU), focusing on implementing control systems for two specific tasks:

1. Open Loop Control of an LED:

The program allows for controlling the brightness of an LED in an open-loop manner. It reads an analog input signal from a designated pin (LED_SETPOINT_PIN) to determine the desired brightness level. This analog value is then mapped to a percentage scale (0-100%) representing the LED's brightness intensity. Subsequently, the program adjusts the LED's output using pulse width modulation (PWM) to achieve the desired brightness level. Additionally, it periodically displays the LED brightness level on the serial monitor.

2. Temperature or Humidity Control using ON/OFF Method with Hysteresis:

This part of the program focuses on controlling temperature (or potentially humidity) using an ON/OFF control method with hysteresis. It reads the setpoint value from an analog source, such as a potentiometer, which is mapped to a desired temperature range. The temperature is then measured using an LM20 temperature sensor (or a similar sensor) and converted from its raw analog value to a temperature reading. Based on a predefined temperature setpoint and a hysteresis margin, the program determines whether to activate or deactivate a relay to maintain the desired temperature range. The current temperature reading, along with the setpoint value, is periodically displayed on an LCD screen.

Additionally, the program allows for setting the setpoint value from various sources, including a potentiometer, two buttons for incrementing or decrementing the setpoint, an encoder sensor, a keypad, or a serial interface. The program utilizes the LiquidCrystal_I2C library to interface with a 16x2 LCD screen for displaying the current temperature and setpoint values. It employs the Serial communication for debugging purposes and to provide feedback on the LED brightness level.

1.2 Block Diagram

First of all, below is presented the flow chart diagram for the function get_temperature_control. It implements the hysteresis for the on/off control of the second led.

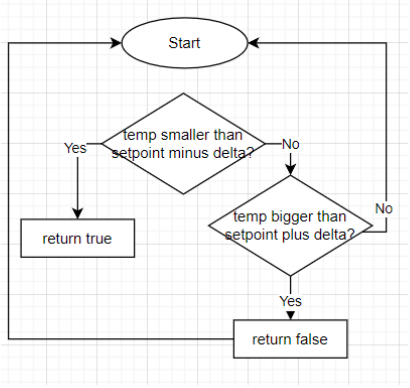

Figure 1. LCDController class

Next, we have the diagram of the loop function: there we can see how the values for both tasks (loop control and on/off control) are read from the potentiometers (using analogRead), and are converted to the proper values.

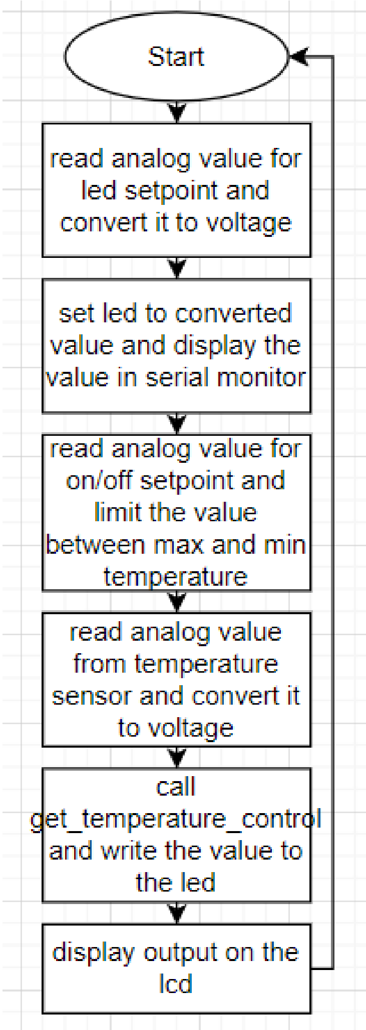

Figure 2. L293DController class

1.3 Electrical Schematic

To simulate the electrical schematic for this project, I began by placing an Arduino Uno board at the center. I then connected the needed devices for the first task: a led (the red one) and the potentiometer needed for it (A1 pin). For the second task I used another potentiometer (A0), another led and a temperature analog sensor. Also, I connected the LCD for displaying some log messages.

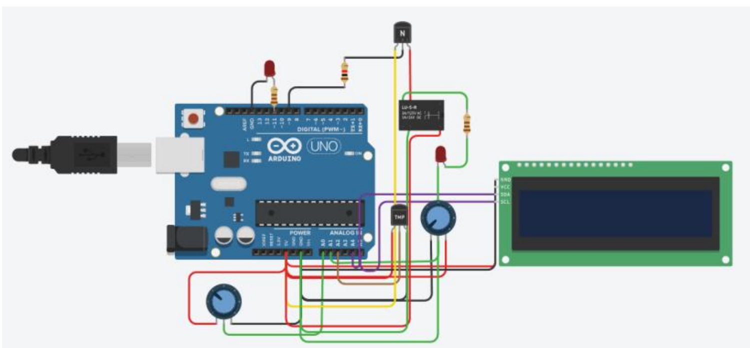

Figure 5. Electric circuit

1.4 Running Simulation

In the image below we can see how the simulation works for the open loop control task: I gave the potentiometer the value that is corresponding to the level 17 of led brightness, and the led turned off. Also, this value is displayed on the serial monitor.

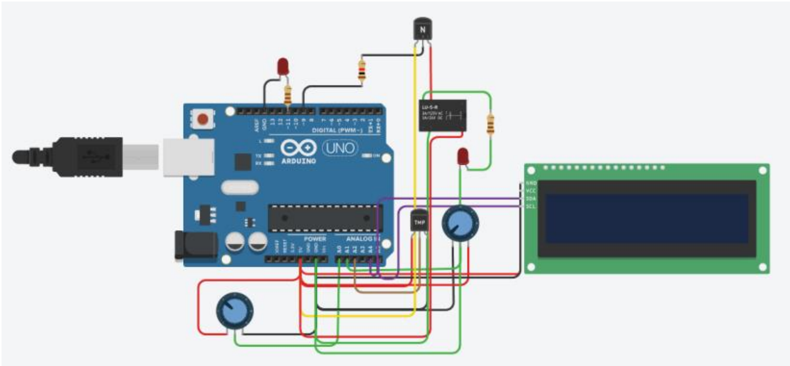

Figure 6. Electric circuit

In the following image there is presented the simulation when the setpoint is set to 76 (via the corresponding potentiometer) and the temperature on the sensor is set to 83. That means that the temperature is higher than the set point, which means that the led turns off.

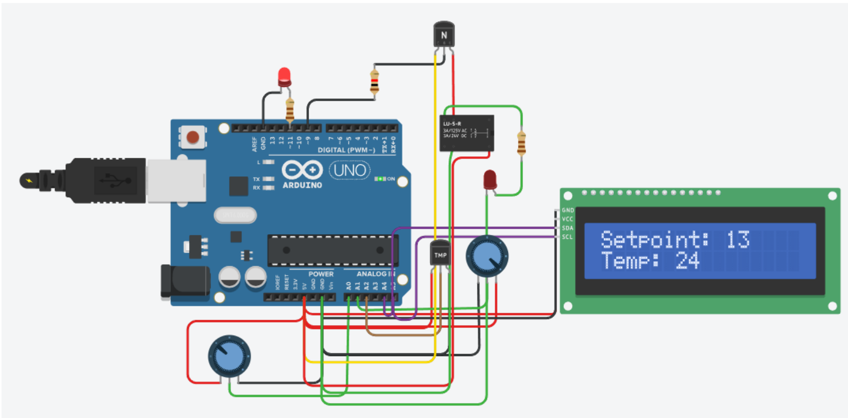

Figure 6. Simulation with the first led turned on

Next, there is the second scenario for this task: when the temperature is lower than the setpoint: it is -32. In this case, we can see how the led turns on.

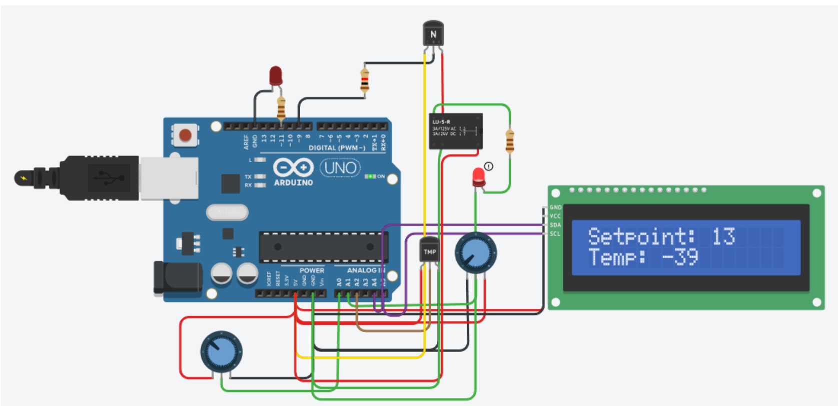

Figure 7. Simulation with the second led turned on

CONCLUSION

In conclusion, the laboratory work aimed to implement an application on a microcontroller unit (MCU) that incorporated control systems for open-loop control of an LED and ON/OFF control of another LED based on sensor readings, with hysteresis. The setpoint for these control systems could be adjusted using various input methods like potentiometers, buttons, encoder sensors, keypads, or a serial interface. Additionally, both the setpoint and current values were displayed on an LCD screen.

The progress of the work involved a detailed description of the program's functionalities, including the implementation of control algorithms for LED brightness and temperature regulation. This included explanations of how PWM was used for LED brightness control and how ON/OFF control with hysteresis was implemented for temperature regulation. Furthermore, the report included block diagrams illustrating the program's flow and electrical schematics depicting the circuit connections for simulation.

Simulation results demonstrated the program's functionality in both open-loop LED control and temperature-based ON/OFF LED control scenarios. These simulations showcased how the program responded to changes in setpoints and sensor readings, effectively controlling the LEDs based on the specified conditions.

Overall, the laboratory work successfully demonstrated the implementation of basic control systems on an MCU platform, providing valuable insights into LED brightness adjustment and temperature regulation techniques using digital control methods.

APPENDIX A: Source code

```
#include <LiquidCrystal_I2C.h>

// https://forum.arduino.cc/t/printf-on-arduino/888528/3
FILE f_out;

int sput(char c, __attribute__((unused)) FILE *f)
{
    if (c == '\n')
    {
        return !Serial.write("\r\n");
    }
    return !Serial.write(c);
}

void redirect_stdout()
{
    //  https://www.nongnu.org/avr-libc/user-
manual/group__avr__stdio.html#gaf41f158c022cbb6203ccd87d27301226
    fdev_setup_stream(&f_out, sput, nullptr, _FDEV_SETUP_WRITE);
    stdout = &f_out;
}

#define TEMP_SETPOINT_PIN A0
#define LED_SETPOINT_PIN A1
#define LM_20_PIN A2

#define LED_OUT_PIN 11
#define TEMP_OUT_PIN 9

// ADC - analog to digital converter
#define ADC_MIN 0
#define ADC_MAX 1023   // highest possible analog input voltage
#define ADC_V_MIN 0    // (mV) minimum voltage level that the ADC input can
measure
#define ADC_V_MAX 5000 // (mV) maximum voltage level that the ADC input can
measure

#define LED_SETPOINT_MIN 0
#define LED_SETPOINT_MAX 100

// PWM - pulse width modulation
#define PWM_MIN 0
#define PWM_MAX 255 // maximum duty cycle value for the PWM signal

#define POTENT_TEMP_MIN (-40)
```

```
#define POTENT_TEMP_MAX (125)

// Temperature sensor
#define LM20_TMIN (-30)
#define LM20_V_TMIN 206
#define LM20_TMAX (125)
#define LM20_V_TMAX 1745

#define TEMP_SETPOINT_MIN 15
#define TEMP_SETPOINT_MAX 30

#define TEMP_DELTA 1

int led_setpoint_analogue = 0;
int outputValue = 0;

int temp_setpoint = 0;

LiquidCrystal_I2C lcd(0x27, 16, 2);

long current_led_millis = 0;
long lastLedDisplayTime = 0;
long ledDisplayInterval = 1000;

long currentLcdMillis = 0;
long lastLcdDisplayTime = 0;
long lcdDisplayInterval = 500;

bool get_temperature_control(int temperature)
{
    if (temperature < temp_setpoint - TEMP_DELTA)
    {
        return HIGH;
    }
    else if (temperature > temp_setpoint + TEMP_DELTA)
    {
        return LOW;
    }
    return LOW;
}

void setup()
{
    lcd.init();
    lcd.backlight();
```

```
    pinMode(LED_SETPOINT_PIN, INPUT);
    pinMode(LED_OUT_PIN, OUTPUT);
    pinMode(TEMP_OUT_PIN, OUTPUT);
    redirect_stdout();
    Serial.begin(9600);
}

void loop()
{
    // read setpoint
    led_setpoint_analogue = analogRead(LED_SETPOINT_PIN);

    // convert to level 0...100%
    int led_brightness = map(led_setpoint_analogue, ADC_MIN, ADC_MAX,
LED_SETPOINT_MIN, LED_SETPOINT_MAX);

    // led intensity
    int led_out = map(led_brightness, LED_SETPOINT_MIN, LED_SETPOINT_MAX,
PWM_MIN, PWM_MAX);

    // apply the brightness to the LED
    analogWrite(LED_OUT_PIN, led_out);

    // print the results to the serial monitor:
    current_led_millis = millis();
    if (current_led_millis - lastLedDisplayTime >= ledDisplayInterval)
    {
        lastLedDisplayTime = current_led_millis;
        printf("LED Brightness: %d\n", led_brightness);
    }

    // ON OFF control
    // read setpoint
    int potentiometerValue = analogRead(TEMP_SETPOINT_PIN);

    // Map the potentiometer value to the setpoint range
    int newSetpoint = map(potentiometerValue, ADC_MIN, ADC_MAX,
POTENT_TEMP_MAX, POTENT_TEMP_MIN);

    // Limit the value of newSetpoint within POTENT_TEMP_MIN and
POTENT_TEMP_MAX.
    temp_setpoint = constrain(newSetpoint, POTENT_TEMP_MIN, POTENT_TEMP_MAX);

    // Get temperature RAW
```

```
    int lm20_analogue = analogRead(LM_20_PIN);

    // Convert raw adc to vold
    int lm20_voltage = map(lm20_analogue, ADC_MIN, ADC_MAX, ADC_V_MIN,
ADC_V_MAX);

    // Convert voltage to temperature
    int lm20_temperature = map(lm20_voltage, LM20_V_TMIN, LM20_V_TMAX,
LM20_TMIN, LM20_TMAX);

    // ON OFF HIST
    bool temperature_control = get_temperature_control(lm20_temperature);
    digitalWrite(TEMP_OUT_PIN, temperature_control);

    currentLcdMillis = millis();
    if (currentLcdMillis - lastLcdDisplayTime >= lcdDisplayInterval)
    {
        lastLcdDisplayTime = currentLcdMillis;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Setpoint: ");
        lcd.print(temp_setpoint);
        lcd.setCursor(0, 1);
        lcd.print("Temp: ");
        lcd.print(lm20_temperature);
    }
}
```

BIBLIOGRAPHY

[1] Actuators, Arduino - Sensors and Actuators, [Quoted: 10.03.2024], access link: https://www.instructables.com/Arduino-Sensors-and-Actuators/

[2] Relays, Arduino 4 Relays Shield Basics, [Quoted: 02.02.2023], access link: https://docs.arduino.cc/tutorials/4-relays-shield/4-relay-shield-basics/

[3] DC Motors, Arduino DC motor, [Quoted: 2024], access link: https://www.javatpoint.com/arduino-dc-motor

[4] L293 motor driver, Arduino DC motor, [Quoted: 2024], access link: https://www.javatpoint.com/arduino-dc-motor

[5] Example of circuit with bulb, Arduino relay activated lamp [Quoted: 02.11.2020] access link: https://www.tinkercad.com/things/kCHoQ9WGGFY-arduino-relay-activated-lamp

[6] Example of circuit with hobby gearmotor, Obstacle Avoiding Robot Using Tinkercad [Quoted: 2024], access link: https://www.instructables.com/Obstacle-Avoiding-Robot-Using-Tinkercad-1/

[7]