**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**

**CHIŞINĂU, 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

# 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.

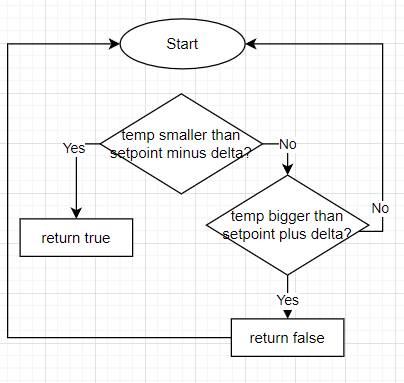
1. Temperature or Humidity Control using ON/OFF Method with Hysteresis:

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.

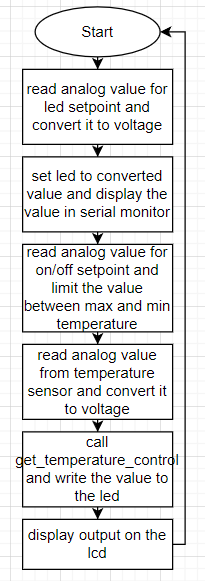
# 1.2 Block Diagram

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**

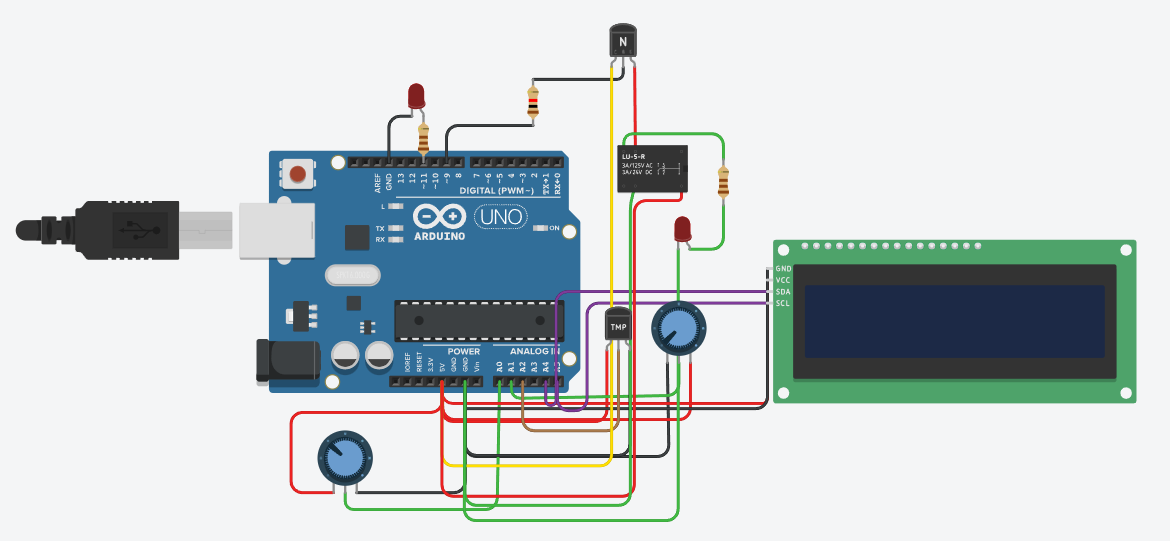
Next, we have the diagram of the loop function: there we can see how the values for both tasks (loop control and on/off control) are read from the potentiometers (using analogRead), and are converted to the proper values.



**Figure 2. L293DController class**

# 1.3 Electrical Schematic

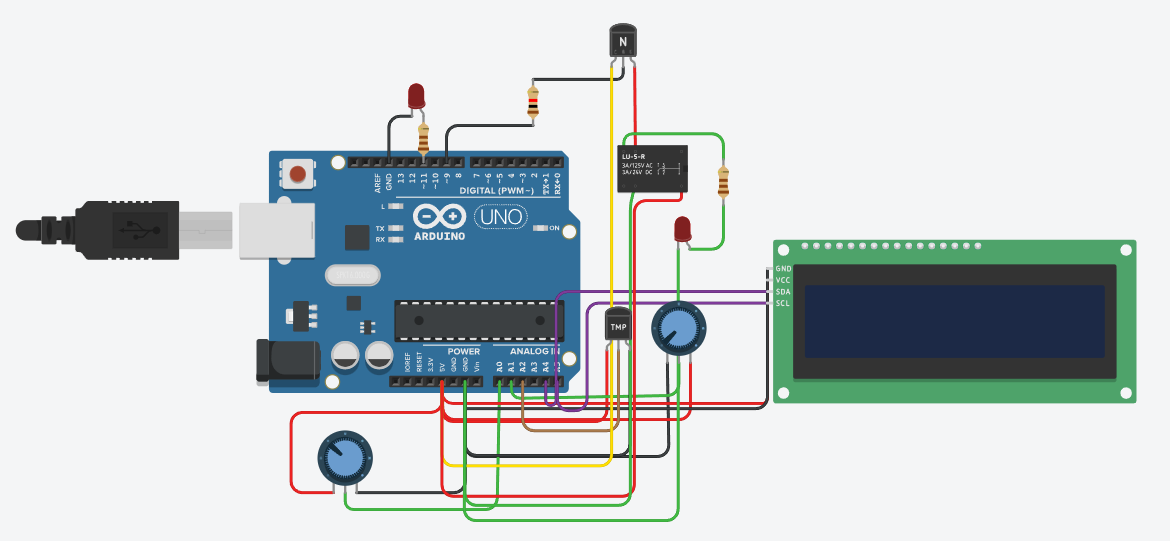
To simulate the electrical schematic for this project, I began by placing an Arduino Uno board at the center. I then connected the needed devices for the first task: a led (the red one) and the potentiometer needed for it (A1 pin). For the second task I used another potentiometer (A0), another led and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

****

**Figure 5. Electric circuit**

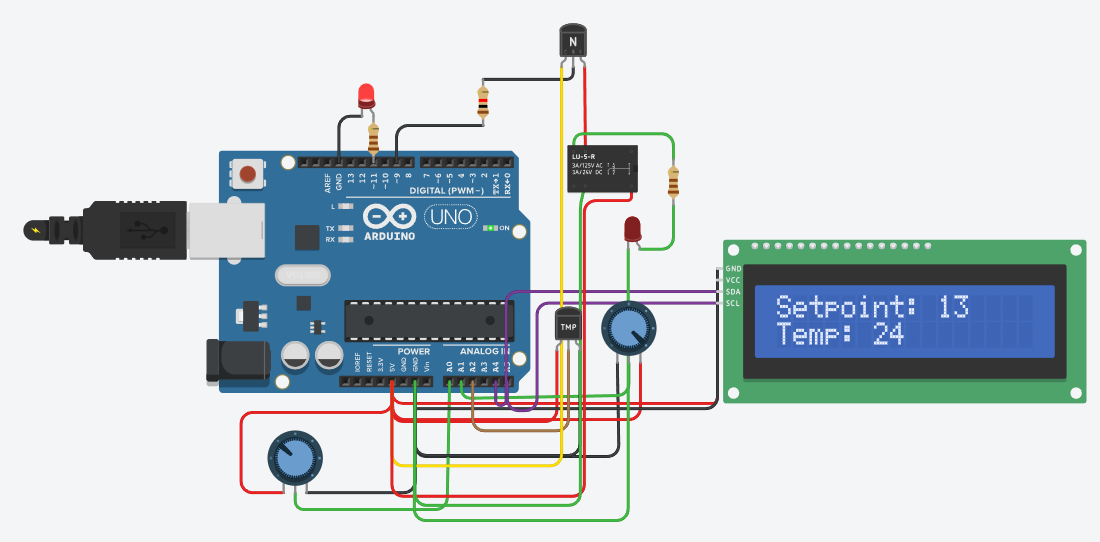
# Running Simulation

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.



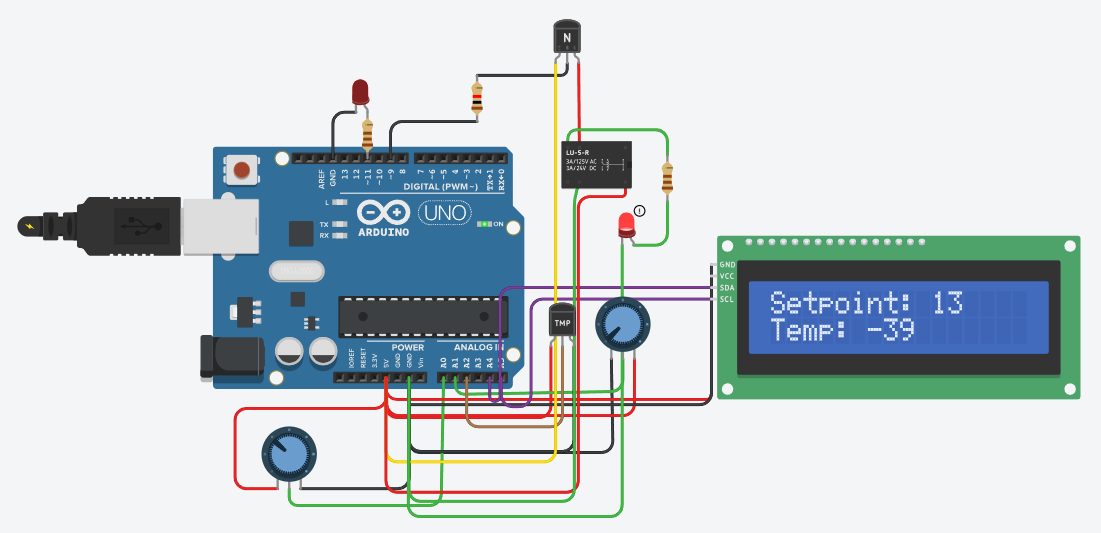
**Figure 6. Electric circuit**

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/>