Ministerul Educaţiei, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Departamentul Ingineria Software și Automatică

RAPORT

Lucrare de laborator Nr.5.1
Disciplina: IoT
Tema: Control - ON-OFF cu histereza

A efectuat: st.gr.TI-212,
Vlasitchi Stefan

A verificat : asist. univ.
Lupan Cristian

Chișinău 2024

Definirea problemei:

1. Sa se realizeze o aplicatie in baza de MCU care va implementa sisteme de control pentru control temperature sau umeditate cu aplicarea metodei de control On-Off cu histeresis cu actionare prin releu.
2. Set point (valoarea de referinta pentru control) se va seta de la una din surse, la alegere - doua butoane pentru UP/Down
3. Valoarea de setpoint si cea curenta se vor afisa la LCD,

Obiective:

1. Implementarea comunicării de la butoane pentru recepționarea comenzilor;
2. Controlul releului pentru acționarea becului electric la satisfacerea conditiilor;
3. Afișarea stării pe LCD.

INTRODUCERE

Controlul ON-OFF cu histerezis reprezintă o metodă esențială de control utilizată în numeroase sisteme automatizate, în special pentru menținerea parametrilor precum temperatura și umiditatea în intervale dorite. Această metodă de control este relativ simplă, dar eficientă, fiind frecvent aplicată în situații unde este necesară o reacție binară de pornire-oprire, cu scopul de a stabiliza condițiile unui mediu controlat, de exemplu în camere frigorifice, în sistemele de climatizare sau în sisteme de umidificare.

Metoda de control ON-OFF cu histerezis funcționează prin activarea sau dezactivarea unui actuator (cum ar fi un releu) în funcție de măsurătorile oferite de senzori și de pragurile de control stabilite. Histerezisul adaugă o zonă de „toleranță” între punctele de pornire și oprire ale sistemului, prevenind comutarea excesivă a actuatorului atunci când măsurătorile sunt aproape de pragul de referință. Aceasta ajută la reducerea uzurii componentelor și la îmbunătățirea stabilității sistemului.

EFECTUAREA LUCRĂRII

Materiale necesare:

- **Microcontroler** (Arduino Mega) folosit ca microcontroler principal pentru controlul releului și afișarea stării pe LCD. Acesta va primi comenzi prin interfața serială și va controla becul;
- **Releu** va fi utilizat pentru a comanda aprinderea și stingerea LED-ului, care simulează becul electric în această configurație;
- **Sensor de temperatura** utilizat pentru a seta o temperatura anumita pentru a putea verifica funcționalitatea aplicației
- **LED** folosit pentru a reprezenta vizual becul aprins/stins. Acesta va fi conectat la terminalul NO (normally open) al releului, ceea ce permite LED-ului să se aprindă doar când releul este activat;
- **Display LCD I2C** pentru a afișa starea becului (aprins sau stins). Interfața I2C simplifică conexiunile necesare pentru LCD, reducând numărul de pini necesari pe Arduino Mega;
- **Butoane** vom utiliza pentru a incrementa si decrementa set pointul;

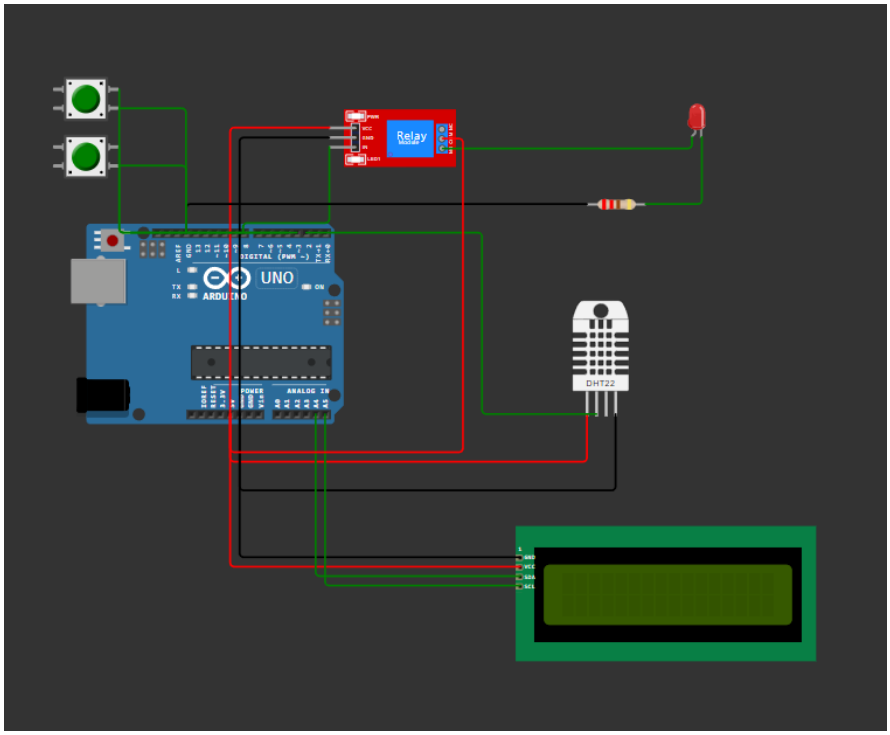


Figura 1 – Ansamblarea circuitului virtual

Modul de lucru

Utilizând această aplicație în bază de MCU, putem gestiona starea unui LED (aprins/stins), utilizând un releu comandat de microcontroler prin menținerea temperaturii pana la un set point dat**.**

```
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define RELAY_PIN 8
#define DHT_PIN 2
#define DHT_TYPE DHT22
#define HYSTERESIS 2.0

#define UP_BUTTON_PIN 3
#define DOWN_BUTTON_PIN 4

float setpoint;
float temp;
bool relayState = false;  // Track relay state (on/off)

DHT dht(DHT_PIN, DHT_TYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(9600);
  dht.begin();
  lcd.init();
  lcd.backlight();

  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);  // Initialize relay as off

  pinMode(UP_BUTTON_PIN, INPUT_PULLUP);
  pinMode(DOWN_BUTTON_PIN, INPUT_PULLUP);

  // Initialize setpoint from potentiometer
  int potValue = analogRead(A0);
  setpoint = map(potValue, 0, 1023, 15, 30);
}
```

Mai sus are loc instanțierea stărilor primare a componentelor, configurând microcontrolerul pentru a putea controla releul și LCD-ul si motorul prin releu.

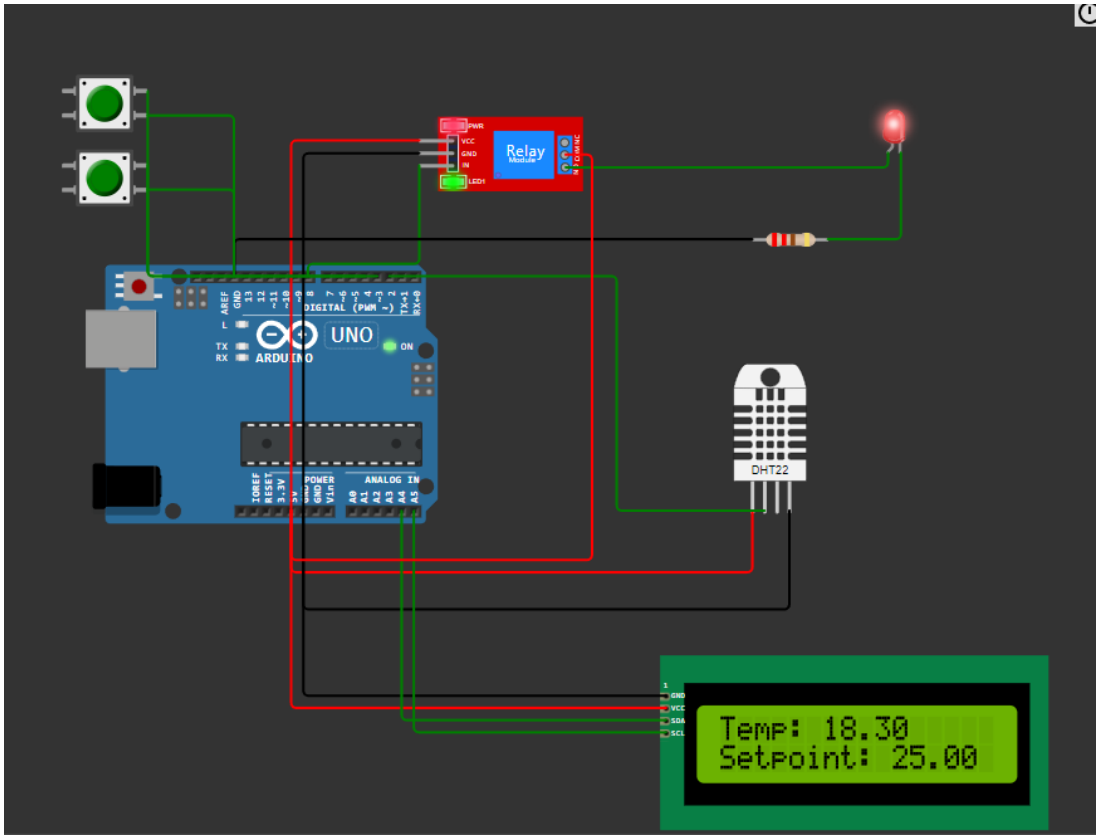
```
void loop() {
  // Read temperature
  temp = dht.readTemperature();

  // Adjust setpoint with buttons
  if (digitalRead(UP_BUTTON_PIN) == LOW) {
```

```
  setpoint += 0.5;
  delay(200);  // Debounce delay
}
if (digitalRead(DOWN_BUTTON_PIN) == LOW) {
  setpoint -= 0.5;
  delay(200);  // Debounce delay
}

// Keep setpoint within bounds
setpoint = constrain(setpoint, 15.0, 30.0);

// On-Off control with hysteresis
if (!relayState && temp < setpoint - HYSTERESIS) {
  relayState = true;  // Turn relay on
  digitalWrite(RELAY_PIN, HIGH);
}
else if (relayState && temp > setpoint + HYSTERESIS) {
  relayState = false;  // Turn relay off
  digitalWrite(RELAY_PIN, LOW);
}
```



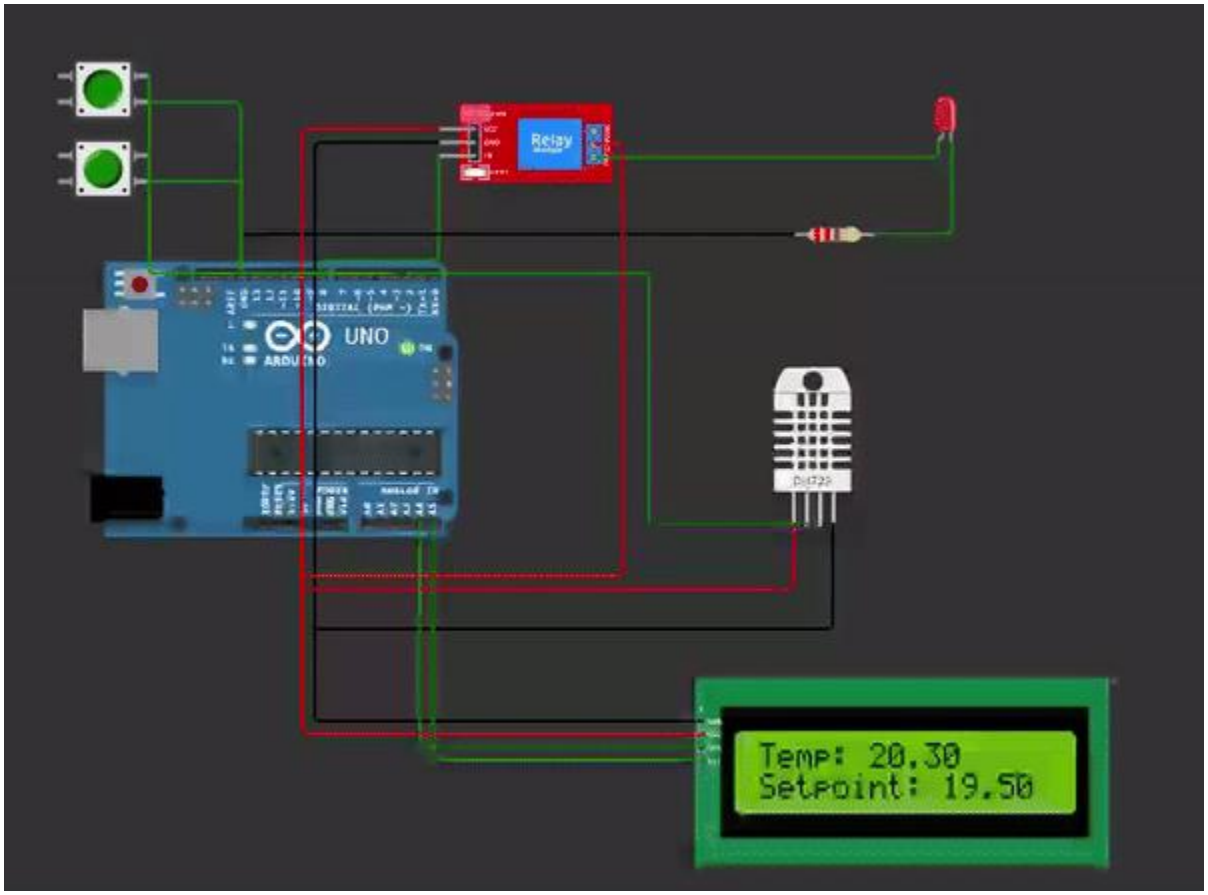


Figura 2 – Simularea aplicației în wokwi

CONCLUZIE

Lucrarea de laborator a demonstrat aplicarea practică a controlului unui actuator simplu (releu) prin intermediul unui microcontroler Arduino Mega. Prin utilizarea interfeței seriale, sistemul primește initial un set point si o temperatura pentru a controla un bec electric, simulând funcționalitatea unui sistem IoT de automatizare astfel ca cand temperatura e mai mare decat set pointul becul se stinge si cand e mai mic se aprinde. Afișajul LCD a oferit o modalitate eficientă de feedback vizual, permițând utilizatorului să vizualizeze starea actuală a releului în timp real.

Anexa 1

Fișierul stdinout.h

```
#ifndef _STDINOUT_H
#define _STDINOUT_H

// no need to make an instance of this yourself
class initializeSTDINOUT
{
        static size_t initnum;
public:
        // Constructor
        initializeSTDINOUT();
};

// Call the constructor in each compiled file this header is included in
// static means the names won't collide
static initializeSTDINOUT initializeSTDINOUT_obj;

#endif
```

Fișierul stdinout.cpp

```
#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif
#include <stdio.h>
#include "stdinout.h"

// Function that printf and related will use to print
static int serial_putchar(char c, FILE *f)
{
  if (c == '\n') {
    serial_putchar('\r', f);
  }

  return Serial.write(c) == 1 ? 0 : 1;
}
// Function that scanf and related will use to read
static int serial_getchar(FILE *)
{
  // Wait until character is avilable
  while (Serial.available() <= 0) { ; }

  return Serial.read();
}

static FILE serial_stdinout;

static void setup_stdin_stdout()
```

```
{
  // Set up stdout and stdin
  fdev_setup_stream(&serial_stdinout, serial_putchar, serial_getchar, _FDEV_SETUP_RW);
  stdout = &serial_stdinout;
  stdin  = &serial_stdinout;
  stderr = &serial_stdinout;
}

// Initialize the static variable to 0
size_t initializeSTDINOUT::initnum = 0;

// Constructor that calls the function to set up stdin and stdout
initializeSTDINOUT::initializeSTDINOUT()
{
  if (initnum++ == 0) {
    setup_stdin_stdout();
  }
}
```

Anexa 2

main.cpp

```
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define RELAY_PIN 8
#define DHT_PIN 2
#define DHT_TYPE DHT22
#define HYSTERESIS 2.0

#define UP_BUTTON_PIN 3
#define DOWN_BUTTON_PIN 4

float setpoint;
float temp;
bool relayState = false;  // Track relay state (on/off)

DHT dht(DHT_PIN, DHT_TYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(9600);
  dht.begin();
  lcd.init();
  lcd.backlight();

  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);  // Initialize relay as off

  pinMode(UP_BUTTON_PIN, INPUT_PULLUP);
  pinMode(DOWN_BUTTON_PIN, INPUT_PULLUP);

  // Initialize setpoint from potentiometer
```

```
  int potValue = analogRead(A0);
  setpoint = map(potValue, 0, 1023, 15, 30);
}

void loop() {
  // Read temperature
  temp = dht.readTemperature();

  // Adjust setpoint with buttons
  if (digitalRead(UP_BUTTON_PIN) == LOW) {
    setpoint += 0.5;
    delay(200);  // Debounce delay
  }
  if (digitalRead(DOWN_BUTTON_PIN) == LOW) {
    setpoint -= 0.5;
    delay(200);  // Debounce delay
  }

  // Keep setpoint within bounds
  setpoint = constrain(setpoint, 15.0, 30.0);

  // On-Off control with hysteresis
  if (!relayState && temp < setpoint - HYSTERESIS) {
    relayState = true;  // Turn relay on
    digitalWrite(RELAY_PIN, HIGH);
  }
  else if (relayState && temp > setpoint + HYSTERESIS) {
    relayState = false;  // Turn relay off
    digitalWrite(RELAY_PIN, LOW);
  }

  // Update LCD display
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.setCursor(0, 1);
  lcd.print("Setpoint: ");
  lcd.print(setpoint);

  // Send data to Serial Plotter
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(", Setpoint: ");
  Serial.print(setpoint);
  Serial.print(", Relay State: ");
  Serial.println(relayState ? "ON" : "OFF");

  delay(1000);
}
```

ANEXA 3

diagram.json

```
{
  "version": 1,
  "author": "Ștefan Vlașițchi",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-arduino-uno", "id": "uno", "top": -18.6, "left": -29.4, "attrs": {}
},
    {
      "type": "wokwi-dht22",
      "id": "dht1",
      "top": 57.9,
      "left": 474.6,
      "attrs": { "temperature": "18.3", "humidity": "37.5" }
    },
    {
      "type": "wokwi-lcd1602",
      "id": "lcd1",
      "top": 284.8,
      "left": 418.4,
      "attrs": { "pins": "i2c" }
    },
   { "type": "wokwi-relay-module", "id": "relay1", "top": -134.2, "left": 230.4, "attrs":
{} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -147.6,
      "left": 580.2,
      "attrs": { "color": "red", "flip": "1" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r1",
      "top": -44.05,
      "left": 489.6,
      "attrs": { "value": "220" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": -166.6,
      "left": -48,
      "attrs": { "color": "green", "bounce": "0" }
    },
    {
```

```
      "type": "wokwi-pushbutton",
      "id": "btn2",
      "top": -109,
      "left": -48,
      "attrs": { "color": "green" }
    }
  ],
  "connections": [
    [ "dht1:VCC", "uno:5V", "red", [ "v48", "h-282.2" ] ],
    [ "dht1:GND", "uno:GND.2", "black", [ "v76.8", "h-301.5" ] ],
    [ "dht1:SDA", "uno:2", "green", [ "h-115.1", "v-182.4" ] ],
    [ "lcd1:VCC", "uno:5V", "red", [ "h0" ] ],
    [ "lcd1:GND", "uno:GND.2", "black", [ "h0" ] ],
    [ "lcd1:SDA", "uno:A4", "green", [ "h0" ] ],
    [ "lcd1:SCL", "uno:A5", "green", [ "h0" ] ],
    [ "relay1:IN", "uno:8", "green", [ "v76.6", "h-211.6" ] ],
    [ "relay1:GND", "uno:GND.2", "black", [ "h0" ] ],
    [ "relay1:VCC", "uno:5V", "red", [ "h0" ] ],
    [ "led1:A", "relay1:NO", "green", [ "v0" ] ],
    [ "led1:C", "r1:2", "green", [ "v0" ] ],
    [ "r1:1", "uno:GND.1", "black", [ "h-403.2", "v28.8" ] ],
    [ "relay1:COM", "uno:5V", "red", [ "h20.4", "v315.4", "h-234.2" ] ],
    [ "btn2:2.r", "uno:GND.1", "green", [ "h0" ] ],
    [ "btn1:2.r", "uno:GND.1", "green", [ "h0" ] ],
    [ "btn1:1.r", "uno:3", "green", [ "v0" ] ],
    [ "btn2:1.r", "uno:4", "green", [ "v0" ] ]
  ],
  "dependencies": {}
}
```