Anexe

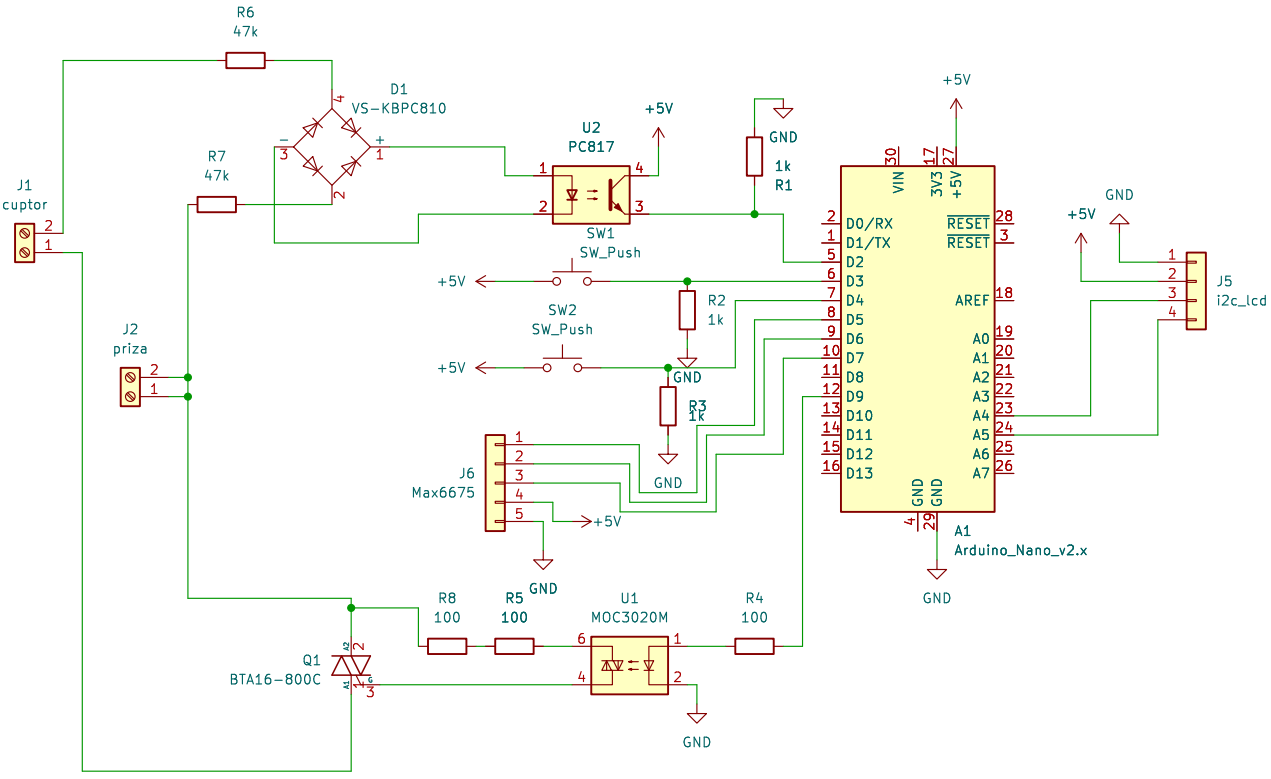
Codul folosit și comentat

```
#include "max6675.h"  // Include biblioteca pentru comunicarea cu senzorul MAX6675
#include <Wire.h>  // Include biblioteca pentru comunicarea I2C
#include <LiquidCrystal_I2C.h>  // Include biblioteca pentru controlul unui LCD I2C

int firing_pin = 9;  // Pinul folosit pentru controlul triacului
int increase_pin = 4;  // Pinul pentru butonul de creștere a setării temperaturii
int decrease_pin = 3;  // Pinul pentru butonul de scădere a setării temperaturii
int zero_cross = 2;  // Pinul pentru detectarea trecerii prin zero a tensiunii
int thermoDO = 5;  // Pinul pentru datele de ieșire ale senzorului termocupla
int thermoCS = 6;  // Pinul pentru selectarea senzorului termocupla
int thermoCLK = 7;  // Pinul pentru semnalul de ceas al senzorului termocupla

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);  // Inițializează obiectul pentru citirea temperaturii

LiquidCrystal_I2C lcd(0x27, 20, 4);  // Inițializează obiectul pentru controlul LCD-ului

bool zero_cross_detected = false;  // Indicator pentru detectarea trecerii prin zero a tensiunii
//////////////////////////////
int firing_delay = 8300;  // Timpul de întârziere pentru declanșarea triacului
int maximum_firing_delay = 8300;  // Valoarea maximă pentru timpul de întârziere

unsigned long previousMillis = 0;  // Momentul în milisecunde al ultimei iterații
unsigned long currentMillis = 0;  // Momentul curent în milisecunde
int temp_read_Delay = 500;  // Intervalul de citire a temperaturii
int real_temperature = 0;  // Valoarea temperaturii măsurate
//////////////////////////////////////////////////////////////////////////////
int setpoint = 40;  // Valoarea țintă pentru temperatură

bool button_increase_state = LOW;  // Starea butonului de creștere a setării
bool button_decrease_state = LOW;  // Starea butonului de scădere a setării

float PID_error = 0;  // Eroarea PID
float previous_error = 0;  // Eroarea PID anterioară
```

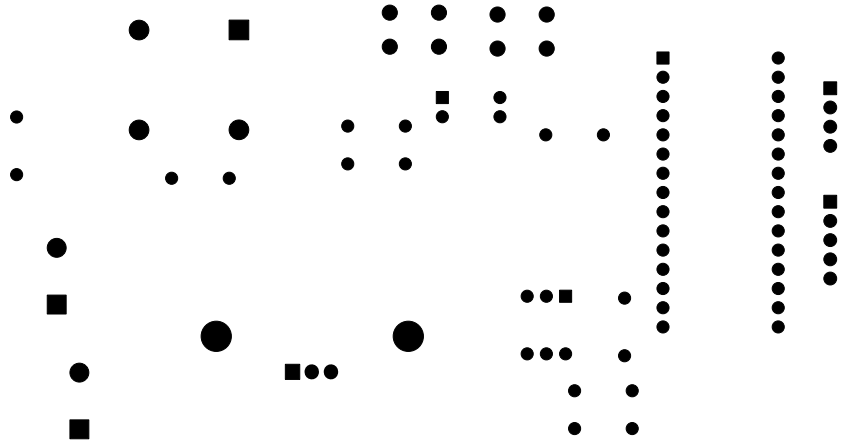
```
float elapsedTime, Time, timePrev;  // Variabile pentru calculul timpului pentru PID
int PID_value = 0;  // Valoarea calculată de PID

////////////////////////////////////////////////////////////////////////////////
int kp = 79;  // Coeficientul proportional al controllerului PID
int ki = 0.98;  // Coeficientul integral al controllerului PID
int kd = 0.49;  // Coeficientul diferențial al controllerului PID

int PID_p = 0;  // Termenul proportional al controllerului PID
int PID_i = 0;  // Termenul integral al controllerului PID
int PID_d = 0;  // Termenul diferențial al controllerului PID

int temperatureBuffer[5];  // Buffer pentru citirile temperaturii
int bufferIndex = 0;  // Indexul pentru buffer

void setup() {
  pinMode(firing_pin, OUTPUT);  // Configurează pinul pentru declanșare ca ieșire
  pinMode(zero_cross, INPUT_PULLUP);  // Configurează pinul pentru detectarea zero crossing ca intrare cu rezistență pull-up
  pinMode(increase_pin, INPUT_PULLUP);  // Configurează pinul pentru butonul de creștere ca intrare cu rezistență pull-up
  pinMode(decrease_pin, INPUT_PULLUP);  // Configurează pinul pentru butonul de scădere ca intrare cu rezistență pull-up

  lcd.init();  // Inițializează LCD-ul
  lcd.backlight();  // Activează iluminarea LCD-ului
}

void loop() {
  currentMillis = millis();  // Actualizează momentul curent

  if (currentMillis - previousMillis >= temp_read_Delay) {
    previousMillis += temp_read_Delay;

    // Citirea temperaturii în buffer și calculul mediei mobile
    temperatureBuffer[bufferIndex] = thermocouple.readCelsius();
    bufferIndex = (bufferIndex + 1) % 5;
    real_temperature = calculateMovingAverage();

    PID_error = setpoint - real_temperature;  // Calculează eroarea PID

    if (PID_error > 20) {
      PID_i = 0;  // Resetarea termenului integral dacă eroarea este mare
    }
```

```
    PID_p = kp * PID_error;  // Calculează termenul proportional
    PID_i = PID_i + (ki * PID_error);  // Calculează termenul integral

    // Calcularea timpului scurs între iterații pentru PID
timePrev = Time;  // Salvarea timpului precedent
Time = millis();  // Actualizarea timpului curent
elapsedTime = (Time - timePrev) / 1000;  // Calculul timpului scurs în secunde
PID_d = kd * ((PID_error - previous_error) / elapsedTime);  // Calculul
termenului diferențial pentru PID
PID_value = PID_p + PID_i + PID_d;  // Calculul valorii finale a controlului PID

if (PID_value < 0) {
  PID_value = 0;  // Asigurarea că valoarea PID este între 0 și
maximum_firing_delay
}//////////////////////////////////
if (PID_value > 8300) {
  PID_value = 8300;  // Limitarea valorii PID la maximum_firing_delay
}

// Actualizarea afișajului LCD cu valorile PID și setpoint-ului
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Set: ");
lcd.setCursor(5, 0);
lcd.print(setpoint);
lcd.setCursor(0, 1);
lcd.print("Temperatura: ");
lcd.setCursor(12, 1);
lcd.print(real_temperature);
previous_error = PID_error;  // Actualizarea eroarei anterioare pentru termenul
diferențial al PID
}

// Verificarea stării butonului de creștere și scădere a setării temperaturii
if (digitalRead(increase_pin) == LOW) {  // Verificarea dacă butonul de creștere
este apăsat
  if (!button_increase_state) {  // Verificarea dacă butonul tocmai a fost apăsat
    setpoint += 5;  // Incrementarea valorii țintă a temperaturii
    button_increase_state = true;  // Actualizarea stării butonului
  }
} else {
  button_increase_state = false;  // Resetarea stării butonului
}
```

```
if (digitalRead(decrease_pin) == LOW) {  // Verificarea dacă butonul de scădere
este apăsat
  if (!button_decrease_state) {  // Verificarea dacă butonul tocmai a fost apăsat
    setpoint -= 5;  // Decrementarea valorii țintă a temperaturii
    button_decrease_state = true;  // Actualizarea stării butonului
  }
} else {
  button_decrease_state = false;  // Resetarea stării butonului
}

// Detectarea trecerii prin zero
if (digitalRead(zero_cross) == HIGH) {
  zero_cross_detected = true;  // Indică detectarea trecerii prin zero
}

// Controlul declanșării triacului
if (zero_cross_detected) {
  delayMicroseconds(maximum_firing_delay - PID_value);  // Aplică întârzierea
pentru declanșare
  digitalWrite(firing_pin, HIGH);  // Activează tiristorul / triacul
  delayMicroseconds(100);  // Pauză mică pentru a asigura declanșarea corectă
  digitalWrite(firing_pin, LOW);  // Dezactivează tiristorul / triacul
  zero_cross_detected = false;  // Resetează indicatorul de trecere prin zero
}
}
// Funcția pentru calcularea mediei mobile
int calculateMovingAverage() {
  int sum = 0;
  for (int i = 0; i < 5; i++) {
    sum += temperatureBuffer[i];  // Adaugă valoarea temperaturii în suma totală
  }
  return sum / 5;  // Calculează și returnează media mobilă
}
```



Universitatea tehnica Cluj Napoca		
Sheet: /		
File: licenta.kicad_sch		
Title: Circuit pentru controlul temperaturii		
Size: A4	Date: 2023-08-04	**Rev: Iosif Vitalis**
KiCad E.D.A. kicad 7.0.6		Id: 1/1



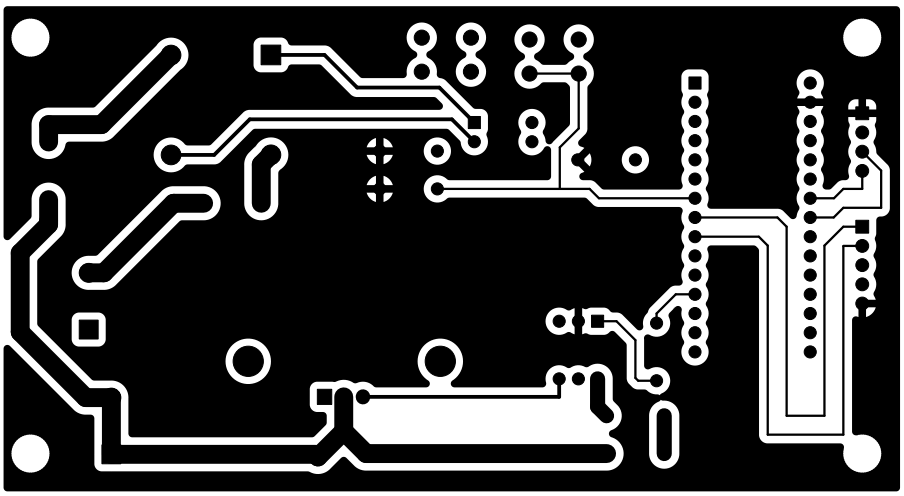
Universitatea tehnica Cluj Napoca		
Sheet: Găurile pentru componente		
File: licenta.kicad_pcb		
Title: Circuit pentru controlul temperaturii		
Size: A4	Date: 2023-08-04	**Rev: Iosif Vitalis**
KiCad E.D.A. kicad 7.0.6		Id: 1/1

Universitatea tehnica Cluj Napoca

Sheet: Marginile
File: licenta.kicad_pcb

Title: Circuit pentru controlul temperaturii

Size: A4	Date: 2023-08-04	**Rev: Iosif Vitalis**
KiCad E.D.A. kicad 7.0.6		Id: 1/1

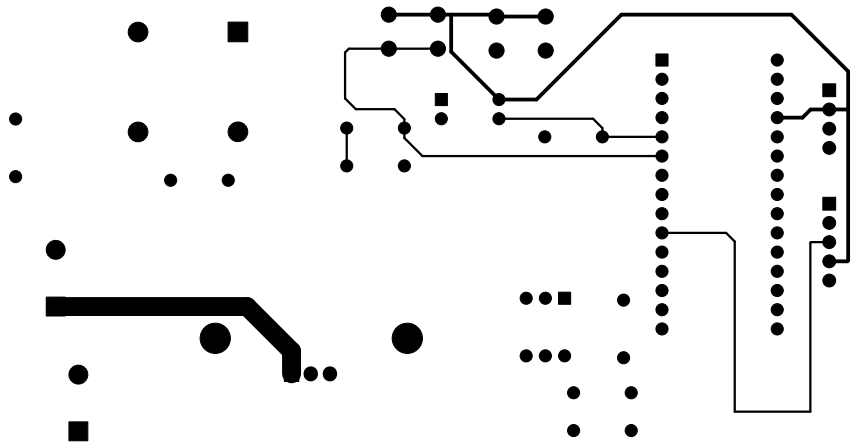


Universitatea tehnica Cluj Napoca

Sheet: Primul start de cupru
File: licenta.kicad_pcb

Title: Circuit pentru controlul temperaturii

Size: A4	Date: 2023-08-04	**Rev: Iosif Vitalis**
KiCad E.D.A. kicad 7.0.6		Id: 1/1

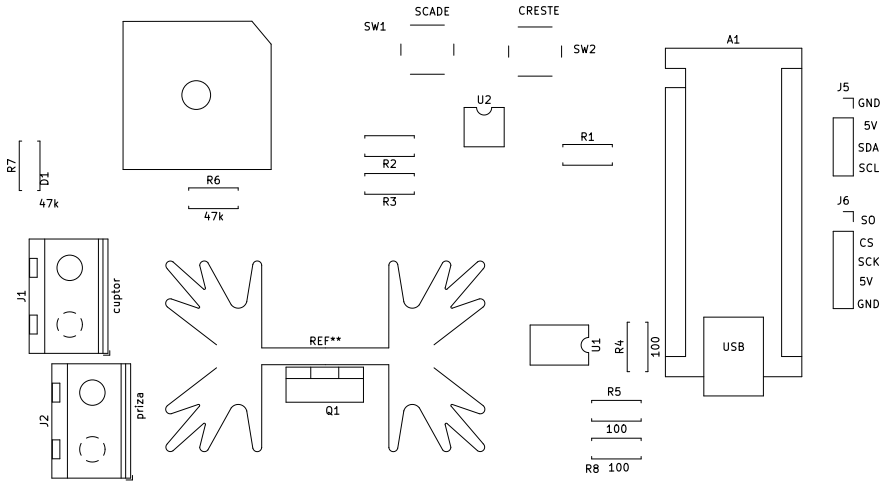


Universitatea tehnica Cluj Napoca

Sheet:Stratul al doilea de cupriu
File: licenta.kicad_pcb

Title: Circuit pentru controlul temperaturii

Size: A4	Date: 2023-08-04	Rev: Iosif Vitalis
KiCad E.D.A. kicad 7.0.6		Id: 1/1



Universitatea tehnica Cluj Napoca		
Sheet: Silkscreen File: licenta.kicad_pcb		
Title: Circuit pentru controlul temperaturii		
Size: A4	Date: 2023-08-04	**Rev: Iosif Vitalis**
KiCad E.D.A. kicad 7.0.6		Id: 1/1

Tabel de referință

Referință	Valoare
A1	Arduino_Nano_v2.x
D1	VS-KBPC810
J1	cuptor
J2	priza
J5	i2c_lcd
J6	Max6675
Q1	BTA16-800C
R1	1k
R2	1k
R3	1k
R4	100
R5	100
R6	47k
R7	47k
R8	100
REF**	Heatsink_Fischer_SK129-STS_42x25mm_2xDrill2.5mm
SW1	SW_Push
SW2	SW_Push
U1	MOC3020M
U2	PC817

Numele amprentei	Poziția X	Poziția Y	Rotație	Latură
Arduino_Nano	182.88	-63.5	0	top
Diode_Bridge_Vishay_KBPC6	119.688	-66.644	-90	top
TerminalBlock_RND_205-00067_1x02_P7.50mm_Horizontal	102.7	-96.1	90	top
TerminalBlock_RND_205-00067_1x02_P7.50mm_Horizontal	105.7	-112.6	90	top
PinSocket_1x04_P2.54mm_Vertical	205	-67.5	0	top
PinSocket_1x05_P2.54mm_Vertical	205	-82.5	0	top
TO-220-3_Vertical	133.89	-105	0	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	167.38	-73.66	0	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	148.81	-72.5	180	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	148.81	-77.5	180	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	177.8	-102.87	90	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	178.81	-107.5	180	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	117.9	-79.4	0	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	97.5	-79.28	90	top
R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	171.19	-112.5	0	top
Heatsink_Fischer_SK129-STS_42x25mm_2xDrill2.5mm	136.5	-100.3	0	top
SW_PUSH_6mm_H9.5mm	146.75	-57.5	0	top
SW_PUSH_6mm_H9.5mm	161	-57.75	0	top
DIP-6_W7.62mm	170	-95	-90	top
DIP-4_W7.62mm	153.7	-68.725	0	top