Załącznik nr 3

Kod źródłowy pliku proj.h

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include "Arduino.h"
#include "Button/Button.h"
#include "max6675.h"

class Button;


// TEMPERATURA
class Temperature
{
  private:
    float val; //obecna wartość temperatury

  public:
              Temperature(float=0); //konstruktor
    float     getTempValue(MAX6675);
    float     value();
    float     set(float); //ustawia podaną temperaturę i zwraca jej wartość
};



// TIMER
class Timer
{
  private:
    unsigned long   millisecNow;
    unsigned long   millisecStart;
  public:
                Timer();
    long int    threshold;
    boolean     stepTimer(unsigned long); //zwraca true cyklicznie co określoną ilość milisekund,
pozwala na wykonanie jakiejś akcji co pewien czas
};

// POWER SUPPLY - MANSON 2405
class Manson2405
{
  public:
    String sendCommand(SoftwareSerial, String); //wysyła komendę do zasilacza
    String startSession(SoftwareSerial, int); // wyłącza przedni panel
    String endSession(SoftwareSerial, int); // włącza przedni panel i kończy sesję
};
```

```
// Regulator PID
class RegPID
{
  private:
    float k;
    float k_i;
    float k_d;

  public:
    void setK(float);
    void setKI(float);
    void setKD(float);
    RegPID(float =100,float=0.002,float =20);
    float regulator(float , float );
};

//INNE
extern void lcdPrint(LiquidCrystal lcd, String napis, int rzad);
extern void displayTemp(float value, LiquidCrystal lcd, int row);
```

Kod źródłowy pliku proj.cpp

```
#include "proj.h"


/* ------------------------TEMPERATURE---------------------------- */
Temperature::Temperature(float tv) {
  val = tv;
}

float Temperature::getTempValue(MAX6675 thermocouple) {
  val = 0.95*(thermocouple.readCelsius()) - 11.58;
  return val;
}

float Temperature::value() {
  return val;
}


float Temperature::set(float setting) {
  val = setting;
  return val;
}


/*----------------------------TIMER-----------------------------------------*/
Timer::Timer() {
  millisecNow = millis();
  millisecStart = millis();
  threshold = 20;
}

boolean Timer::stepTimer(unsigned long milliseconds) {
  millisecNow = millis();
  if( (millisecNow - millisecStart > milliseconds) && (millisecNow - millisecStart)%milliseconds < threshold ) {
    millisecStart = millis();
    return true;
  }
  else return false;
}
```

```
/*---------------------------MANSON 2405---------------------------------*/

String Manson2405::sendCommand(SoftwareSerial rs, String caption){
  rs.print(caption);
  rs.write(0x0D);
  return caption;
}

String Manson2405::startSession(SoftwareSerial rs, int adress){
  String adressString = (String)adress;
  if(adress<10) adressString = "0" + adressString;

  sendCommand(rs, "SESS"+adressString);
  return adressString;
}

String Manson2405::endSession(SoftwareSerial rs, int adress){
  String adressString = (String)adress;
  if(adress<10) adressString = "0" + adressString;

  sendCommand(rs,"ENDS"+adressString);
  return adressString;
}

/*-------------------------REGULATOR PID-------------------------------*/
void RegPID::setK(float value){
  k = value;
}
void RegPID::setKI(float value){
  k_i = value;
}
void RegPID::setKD(float value){
  k_d = value;
}

RegPID::RegPID(float k_, float k_i_, float k_d_){
  setK(k_);
  setKI(k_i_);
  setKD(k_d_);
}

float RegPID::regulator(float desired, float tval)  {
  //zmienne pomocnicze
  float p, i, d, r;
  float e; //uchyb regulacji
  static float e_p = 0; // poprzedni uchyb
  static float s_e = 0; // suma poprzednich uchybów
  e = desired - tval;
  // wyznaczenie skladnika proporcjonalnego
  p = k * e;
  // wyznaczenie składnika całkowego
  s_e += e; // wyliczenie sumy wszystkich uchybów
  i = k_i * s_e;

  // wyznaczenie składnika różniczkującego
  d = k_d * ( e - e_p);
  e_p = e; // zapisanie chwilowej wartości uchybu w pomocniczej zmiennej

  r = p + i + d; // sygnał wyjściowy regulatora

  if (r < 0) r = 0;
  if (r > 100) r = 100;

  return r;
}
```

```
/*
  INNE
*/
extern void lcdPrint(LiquidCrystal lcd, String napis, int rzad) {
  lcd.setCursor(0,rzad);
  lcd.print(napis);
}

//wyświetlanie wartości liczbowej value, 4 ostatnie znaki w wierszu row na wyświetlaczu lcd
extern void displayTemp(float value, LiquidCrystal lcd, int row) {
  if (value < 10) { //liczby jednocyfrowe
    lcd.setCursor(12,row);
    lcd.print(" ");
    lcd.setCursor(13,row);
    lcd.print((String)value);
  }
  else if (value >= 10 && value <100 ) { //liczby dwucyfrowe
    lcd.setCursor(12,row);
    lcd.print((String)value);
  } else{ //liczby powyżej 100 niewyświetlane
    lcd.setCursor(13,row);
    lcd.print("err");
  }
}
```