1. Wprowadzenie | Introduction

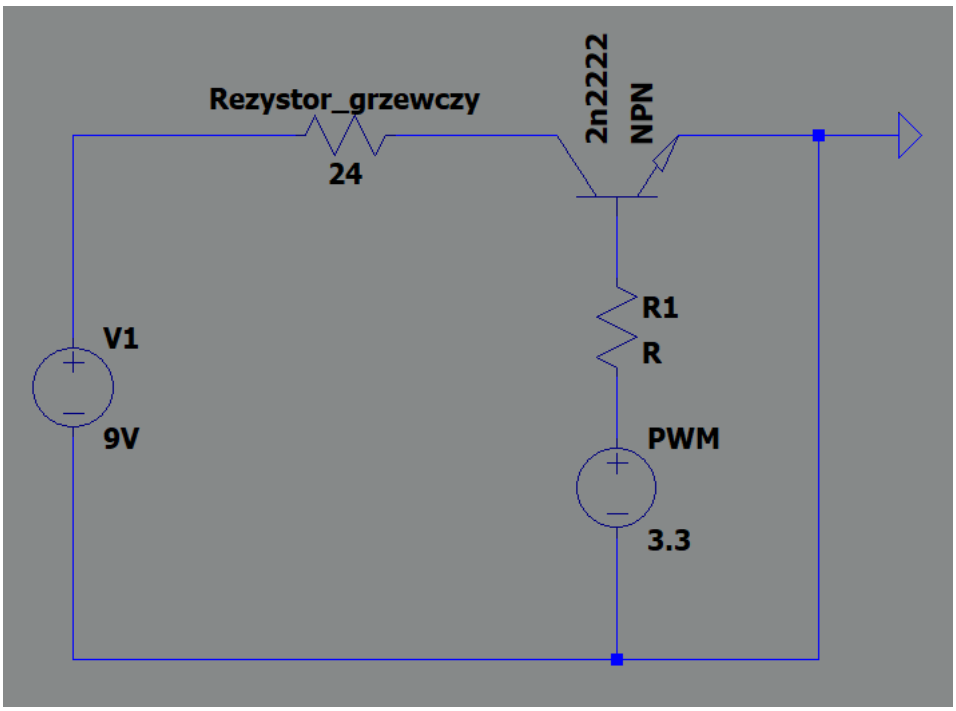
   Za zadanie projektowe określiliśmy regulowanie temperatury rezystora cieplnego za pomocą układu automatycznej regulacji z wykorzystaniem regulatora PID. Temperatura została zmierzona za pomocą czujnika temperatury na układzie BMP280 podłączonego do płytki Nucleo i przez nią zasilany. Do układu mierzącego temperaturę jest przyłożony rezystor grzewczy, który nagrzewa się poprzez tranzystor NPN “2N2222”. Przez tranzystor przechodzi sygnał PWM, który steruje wartością temperatury zadanej, do której ma się nagrzać rezystor cieplny.

   As a design task, we defined the regulation of the temperature of the thermal resistor by means of an automatic control system with the use of a PID controller. The temperature was measured with a temperature sensor on the BMP280 chip connected to the Nucleo board and powered by it. A heating resistor is attached to the temperature measuring circuit, which heats up through the NPN transistor "2N2222". The PWM signal passes through the transistor, which controls the value of the set temperature to which the thermal resistor is to heat up.

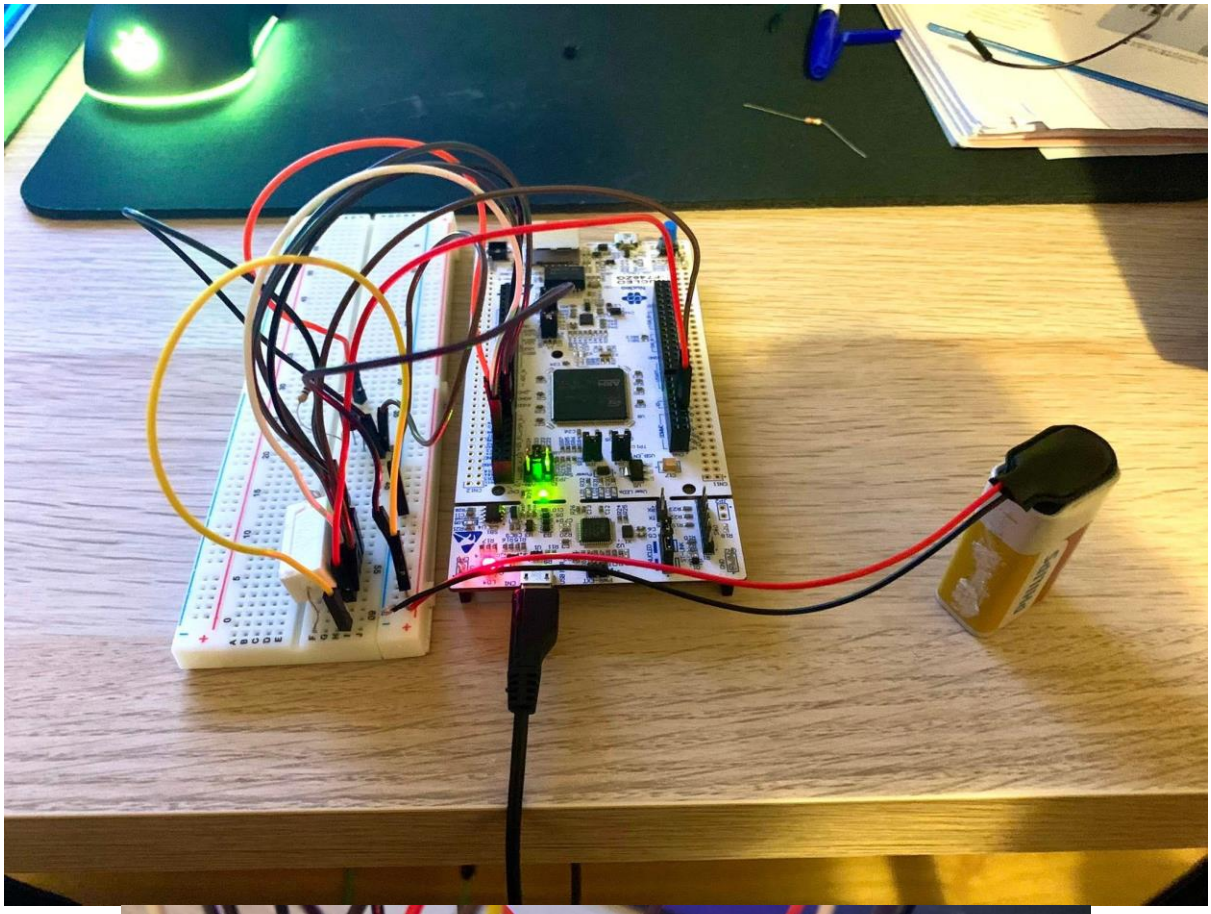
2. Spis użytych elementów | List of used elements
   - BMP280
   - Tranzystor | Transistor NPN “2N2222”
   - Rezystor cieplny | Heat resistor 5W24OhmJ
   - Rezystor | Resistor 100 Ohm
   - Płytka Nucleo | Nucleo plate F746ZG
   - Płytka stykowa | Breadboard
   - Zasilanie zewnętrzne | External power supply 9V

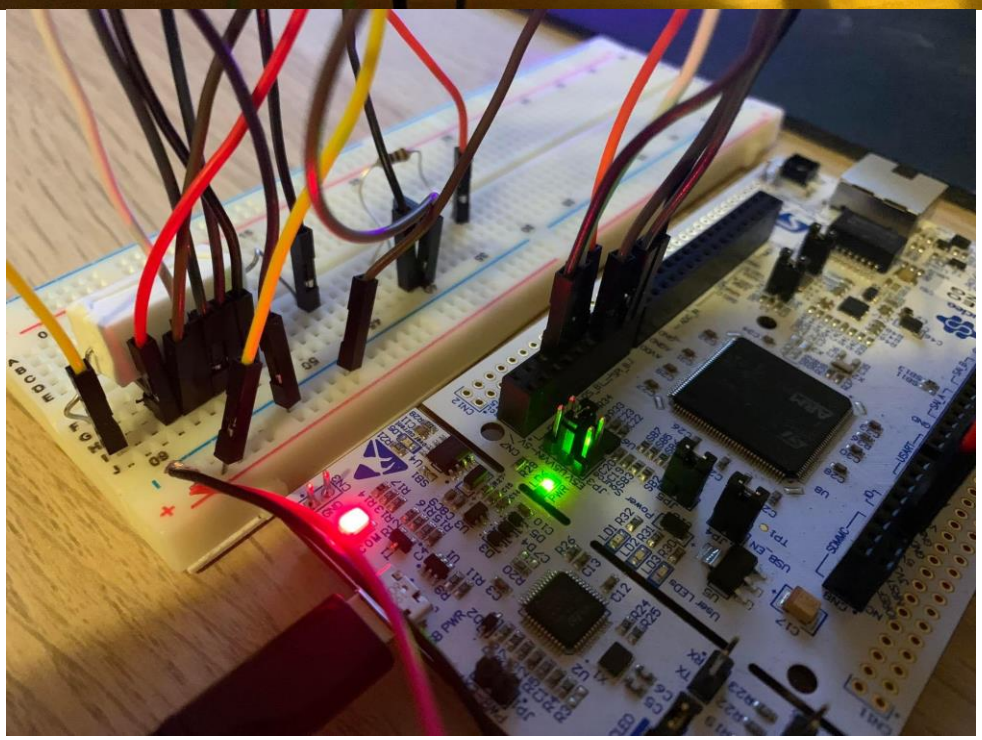
3. Schemat układu | Layout scheme

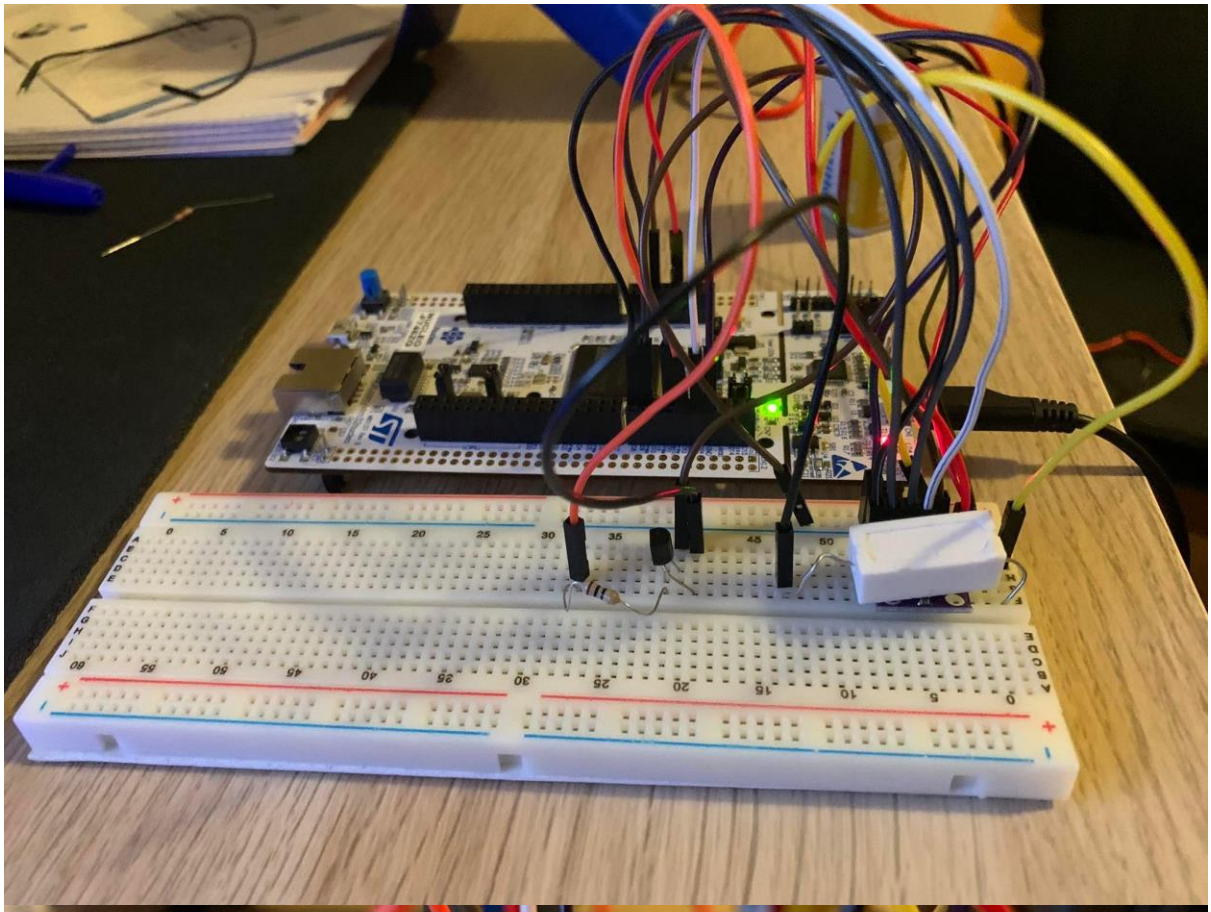
   - Schemat układu w LTSpice | Layout scheme in LTSpice

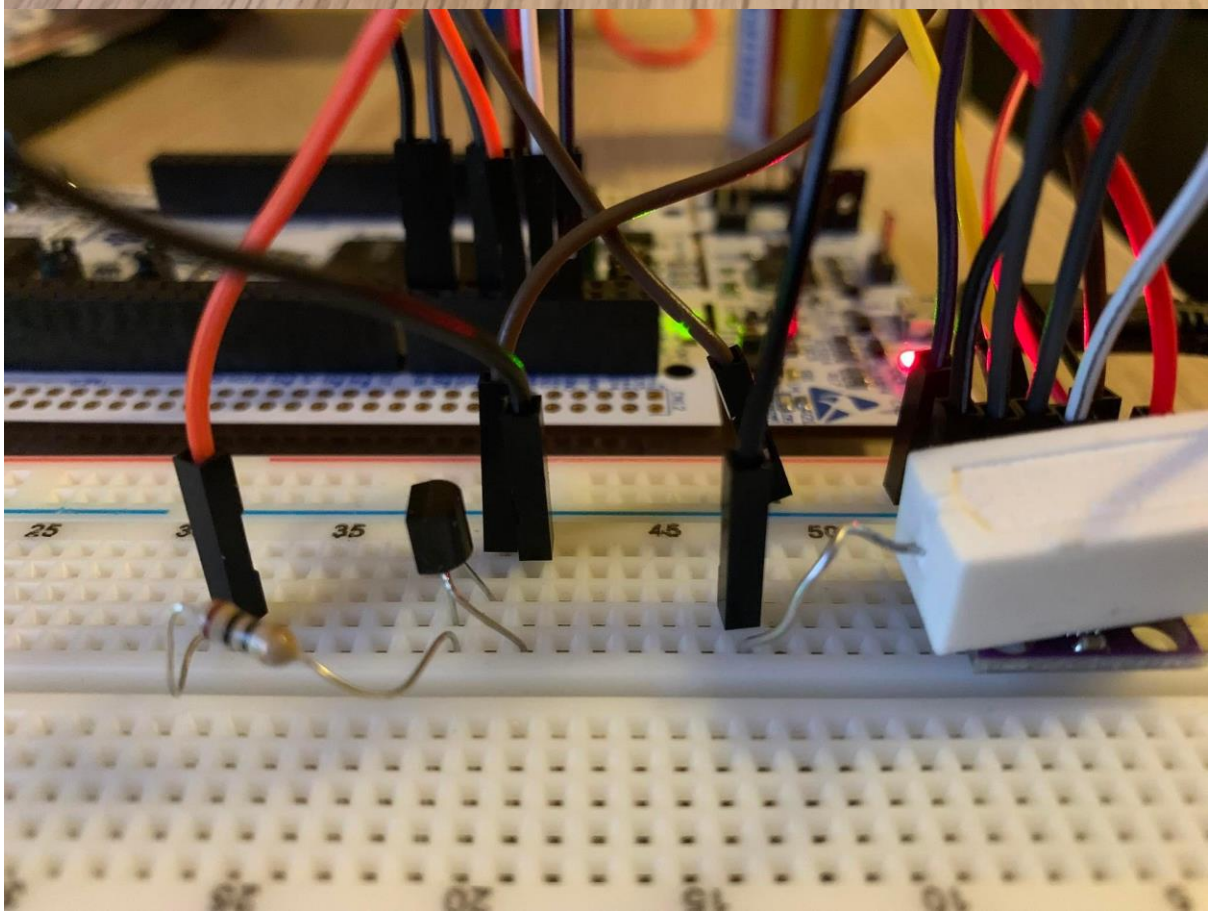


- Zbudowany układ przy użyciu płytki Nucleo | Built circuit using Nucleo PCB









4. Najważniejsze elementy programu | The most important elements of the program

```
66 float32_t calculate_discrete_pid(pid_t* pid, float32_t setpoint, float32_t measured){
67     float32_t u=0, P, I, D, error, integral, derivative;
68
69     error = setpoint-measured;
70
71     //proportional part
72     P = pid->p.Kp * error;
73
74     //integral part
75     integral = pid->previous_integral + (error+pid->previous_error) ; //numerical integrator without anti-windup
76     pid->previous_integral = integral;
77     I = pid->p.Ki*integral*(pid->p.dt/2.0);
78
79     //derivative part
80     derivative = (error - pid->previous_error)/pid->p.dt; //numerical derivative without filter
81     pid->previous_error = error;
82     D = pid->p.Kd*derivative;
83
84     //sum of all parts
85     u = P  + I + D; //without saturation
86
87     return u;
88 }
89 /* USER CODE END PV */
90
```

Rys 1. implementacja regulatora PID w programie | PID implementation in CubeIDE

```
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
    float32_t temperature;
    float32_t pwm_ctrl = 0;
    float32_t pwm;
    int32_t pressure;
    float32_t target;
    float32_t limit;
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

      pid_t pid1 = { .p.Kp=0.3, .p.Ki=0.000365, .p.Kd=3, .p.dt=10, .previous_error=0, .previous_integral=0};
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/
```

Rys 2. dobranie nastaw regulatora oraz deklaracja zmiennych | selection of controller settings and declaration of variables

```
/* USER CODE BEGIN 2 */
    BMP280_Init(&hspi1, BMP280_TEMPERATURE_16BIT, BMP280_STANDARD, BMP280_FORCEDMODE);
    HAL_TIM_PWM_Start (&htim2, TIM_CHANNEL_4);
/* USER CODE END 2 */
```

Rys 3. inicjalizacja timera oraz czujnika temperatury | timer and temperature sensor initialization

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

    BMP280_ReadTemperatureAndPressure(&temperature, &pressure);
    HAL_Delay(1000);

    target = 40.0;

    pwm_ctrl = calculate_discrete_pid(&pid1, target, temperature);
    limit = (target-temperature)/pwm_ctrl;

    if(limit < 0)
        {
            limit=-limit;
        }



        if(pwm_ctrl>limit){
            pwm = 10000;
        }
        else if(limit == 0 )
        {
            pwm = -pwm;

        }
        else
        {
            pwm = 1000*pwm_ctrl;
            pwm=1000+pwm;
        }


    __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_4, pwm);

  }
  /* USER CODE END 3 */
}
```

Rys 4. główna pętla programu | main program loop

5. Opis działania programu | Description of how the program works

pwm_ctrl jest to sygnał sterujący z regulatora PID, gdy jest on większy od naszego limitu, (czyli różnicy temperatury zadanej od aktualnej podzielonej przez wartość sygnału sterującego) program ustawia wypełnienie sygnału pwm na maksymalne. W każdym innym przypadku wypełnienie jest zależne od odpowiednio powiększonego sygnału sterującego z regulatora. Else if zawierający warunek gdy nasz limit będzie równy 0, jest zabezpieczeniem, aby w chwili osiągnięcia wartości zadanej , sygnał pwm nie ustawił się na maksymalne wypełnienie, jak wynika z warunku pierwszego. Sygnał pwm jest ustawiony na timer2 kanał 4, na płytce pin PA3.

pwm_ctrl it is a control signal from the PID controller, when it is greater than our limit (i.e. the difference between the set temperature and the current temperature divided by the value of the control signal) the program sets the pwm signal duty to maximum. In all other cases, the duty cycle is dependent on a suitably magnified control signal from the controller. Else if containing the condition when our limit is equal to 0, is a protection that when the set value is reached, the pwm signal does not set to its maximum fulfillment, as results from the first condition. The PWM signal is set to timer2 channel 4, pin PA3 on the board.

6. Link do filmu | Link to the video : https://youtu.be/egyyovmzyIA