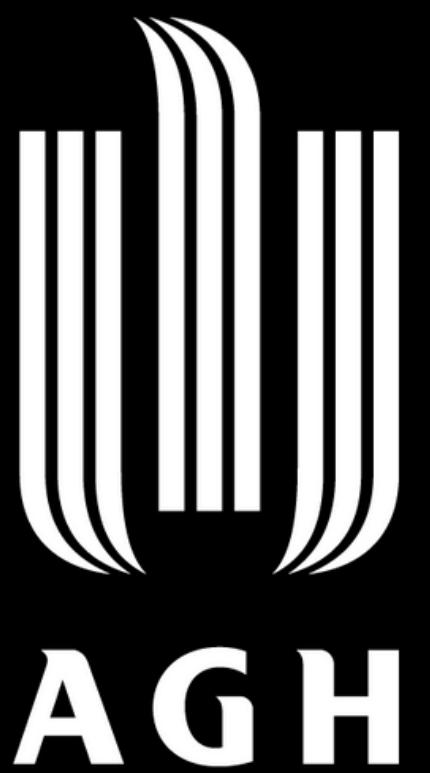


# **STACJA LUTOWNICZA - POMIAR TERMOPARY I TERMISTORA**

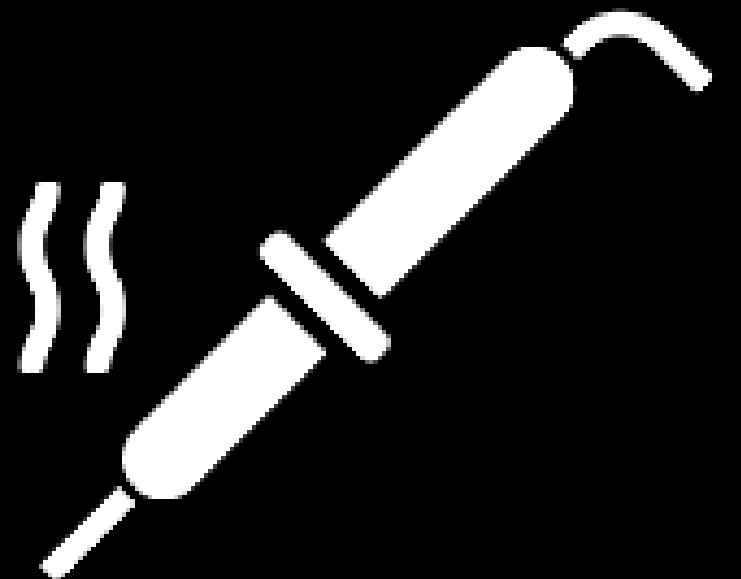


Iwona Suda 419283

Jarosław Przerywacz 419315

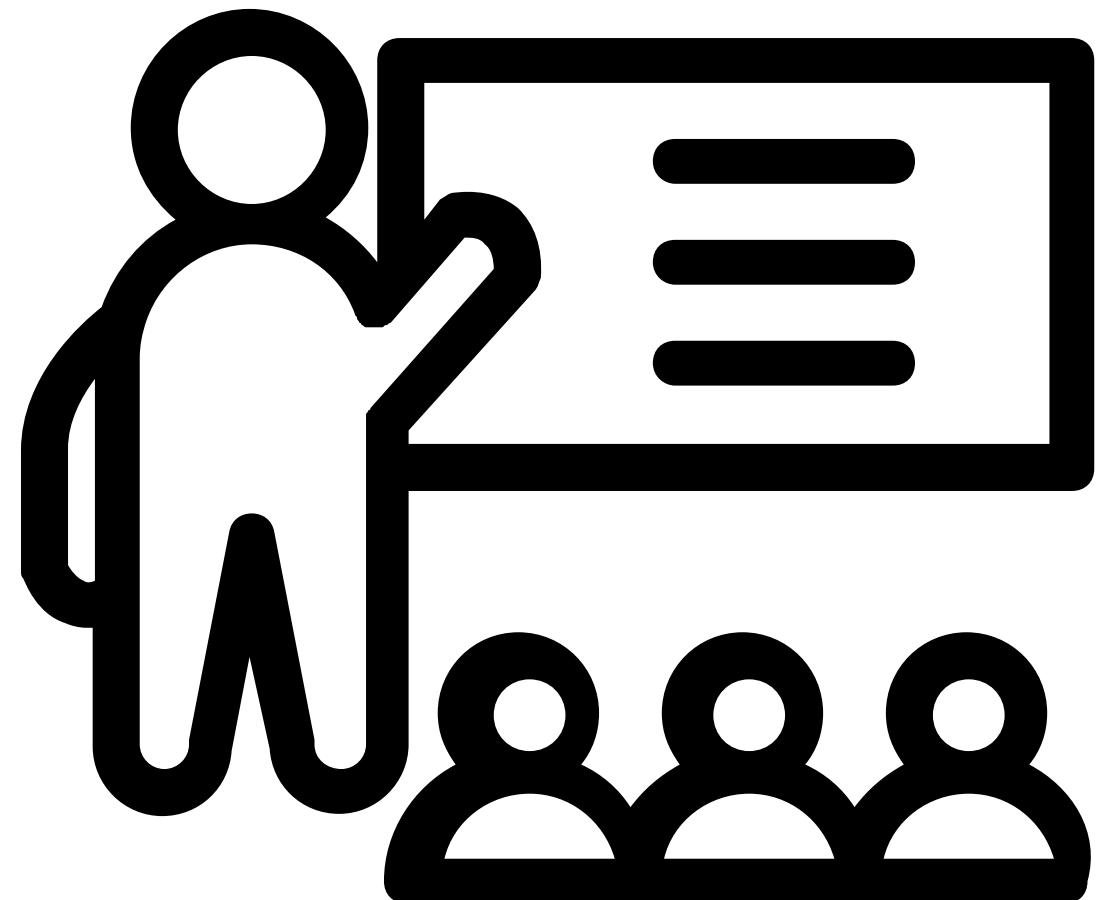
Radosław Borczuch 419282

Kamil Godek 419299



# Plan prezentacji

1. Cel pracy.
2. Podział zadań.
3. Główny element wykonawczy.
4. Schematy, PCB.
5. Konfiguracja, stany pracy.
6. Napotkane problemy.
7. Uruchomienie.



# Cel pracy

Celem pracy jest projekt, budowa i wykonanie stacji lutowniczej sterowanej mikroprocesorowo w oparciu o groty T12.



# Podział zadań

## Imię i nazwisko

## Zadania do wykonania

**Radosław Borczuch**

Hardware, płytka PCB i schemat ideowy.  
Softwarowa obsługa pomiaru temperatury termopary i prądu.  
Montaż podzespołów na płytce i pomiary układu  
Algorytm PID.  
Wyznaczenie charakterystyki termopary grotu i symulacja toru analogowego.

**Iwona Suda**

Sterownik do obsługi wyświetlacza.  
Biblioteka do obsługi menu.  
Integracja i debugowanie softu.

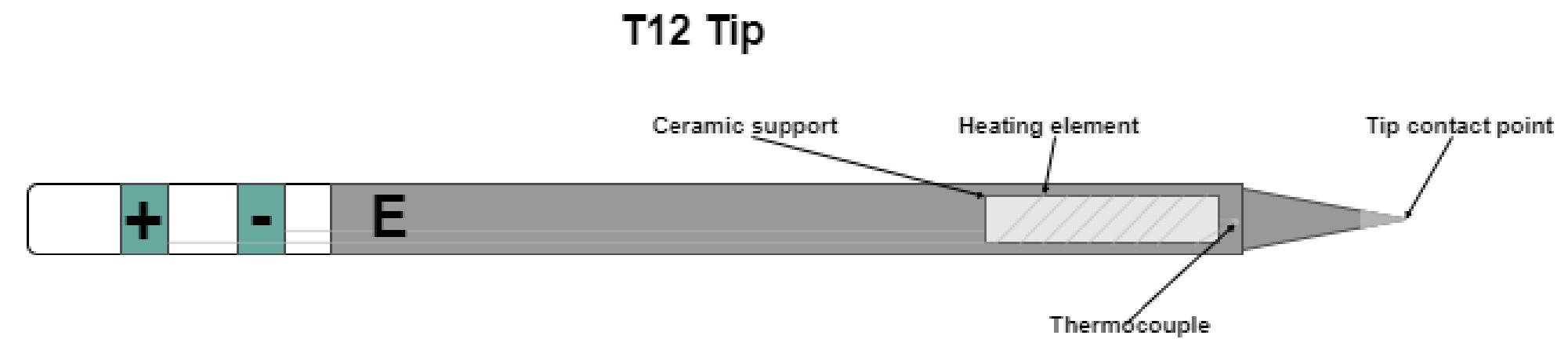
**Kamil Godek**

Softwarowa obsługa sterowania grzałką.  
Integracja i debugowanie softu.  
Funkcja pomiaru temperatury termistora.  
Wizualizacja modelu obudowy lutownicy

**Jarosław Przerywacz**

Funkcje obsługi enkodera i przycisków  
Biblioteka do obsługi menu.  
Integracja i debugowanie softu.

# Główny element wykonawczy

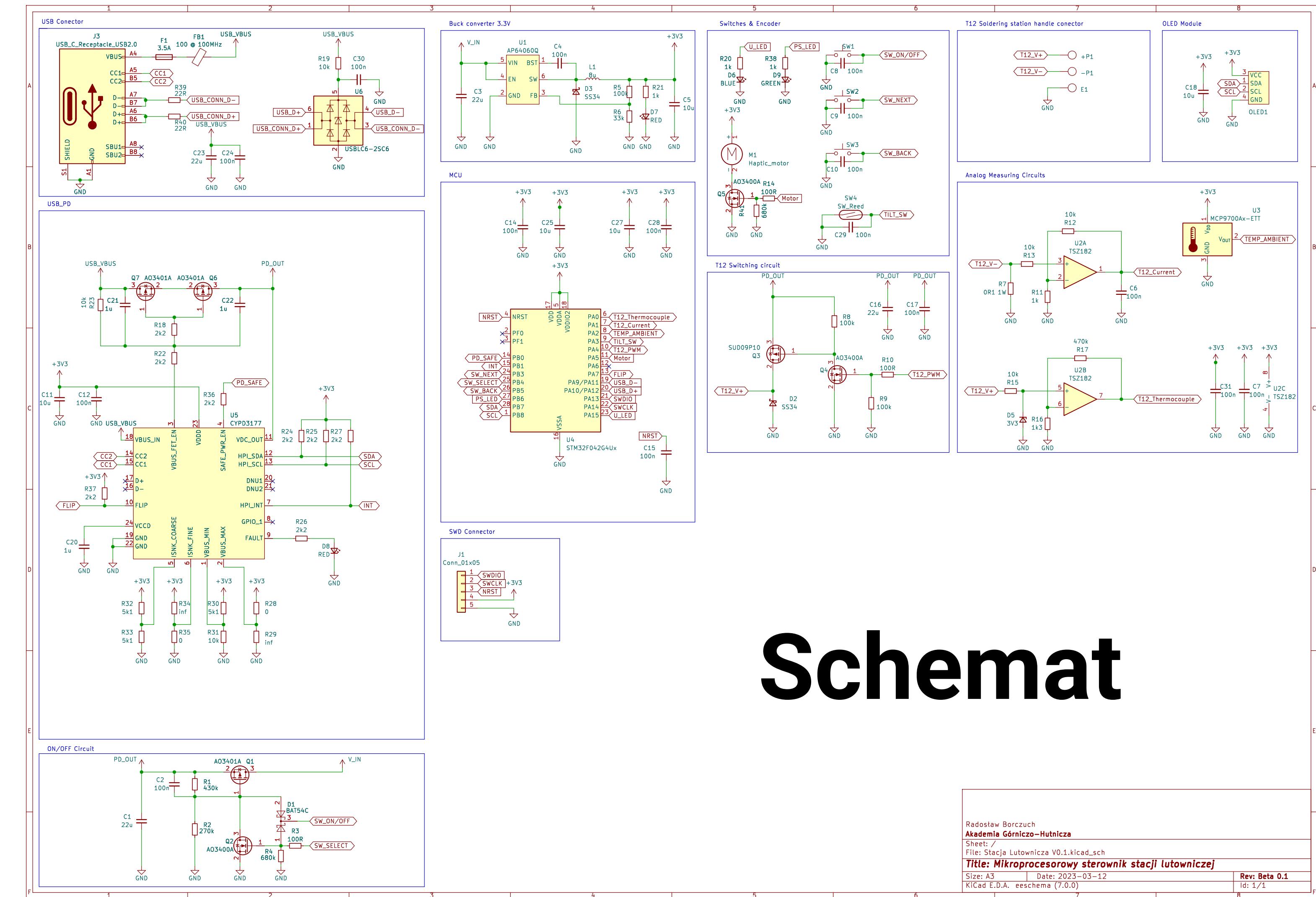


## Grot T12:

- Zintegrowany element grzejny
- Zintegrowana termopara typu N



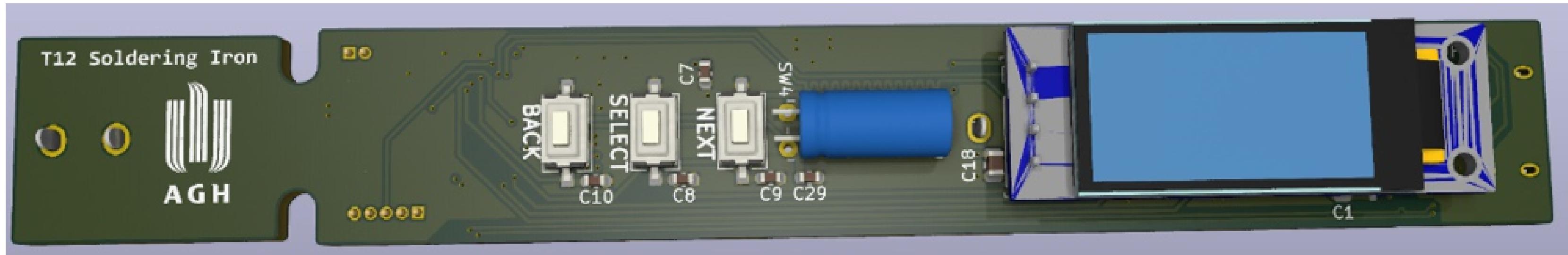
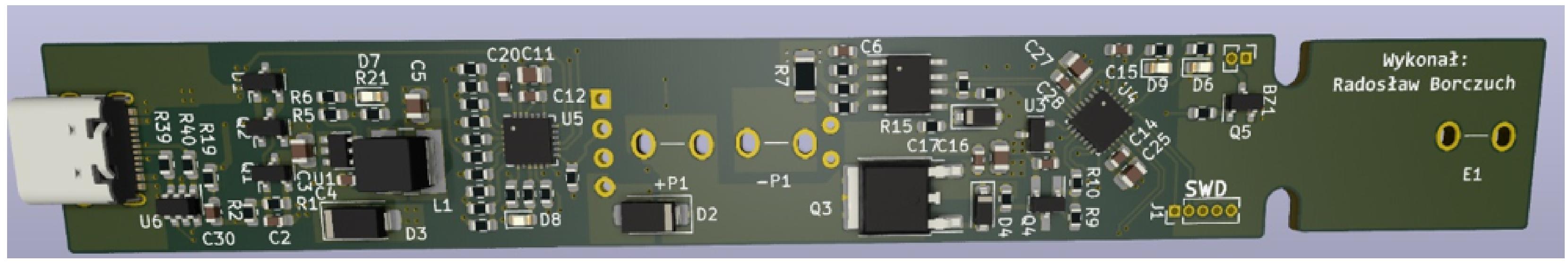
# Schemat



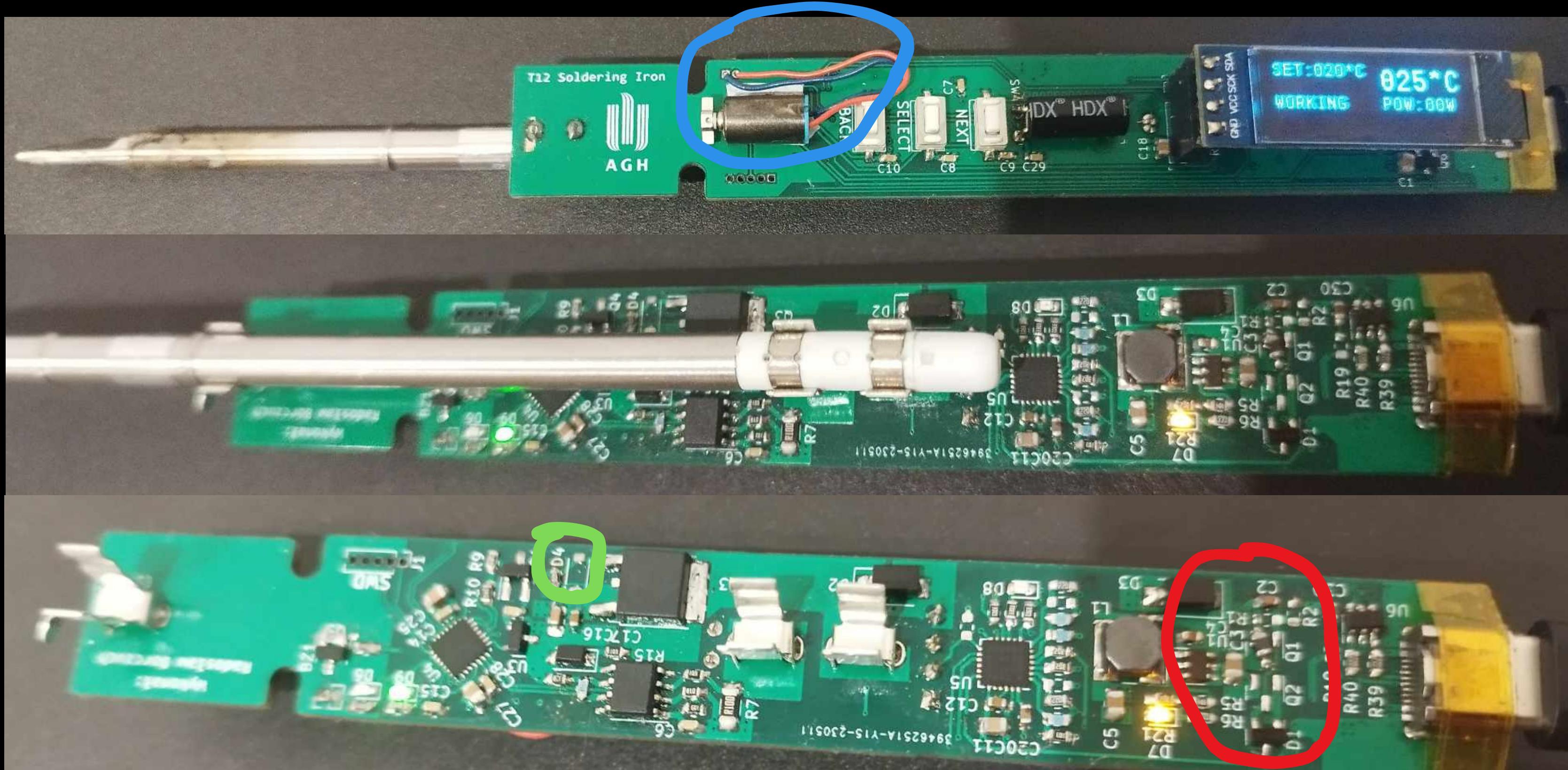
# Widok na każdą warstwę PCB



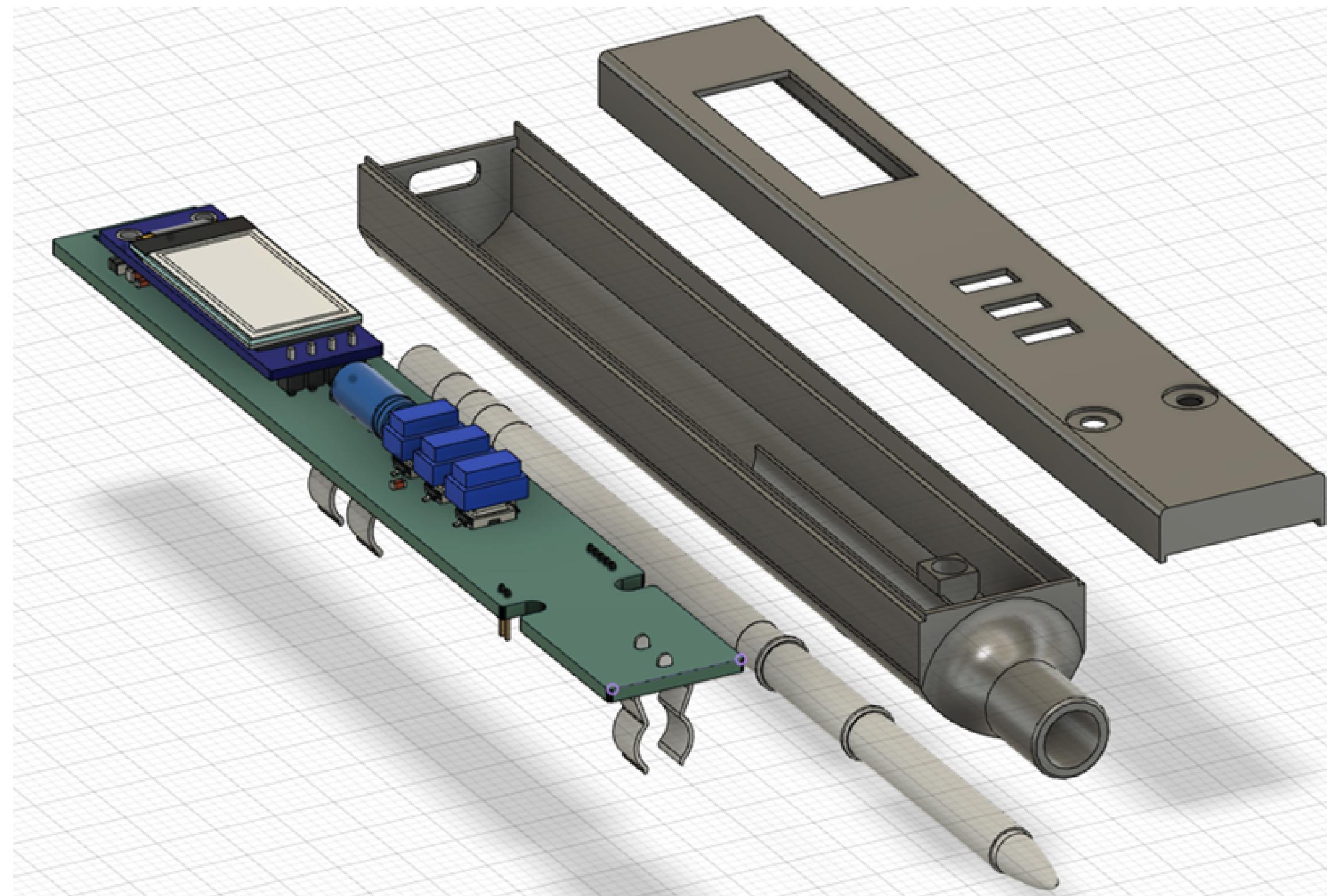
# Wizualizacja płytki



# Zdjęcia urządzenia



# Wizualizacja obudowy



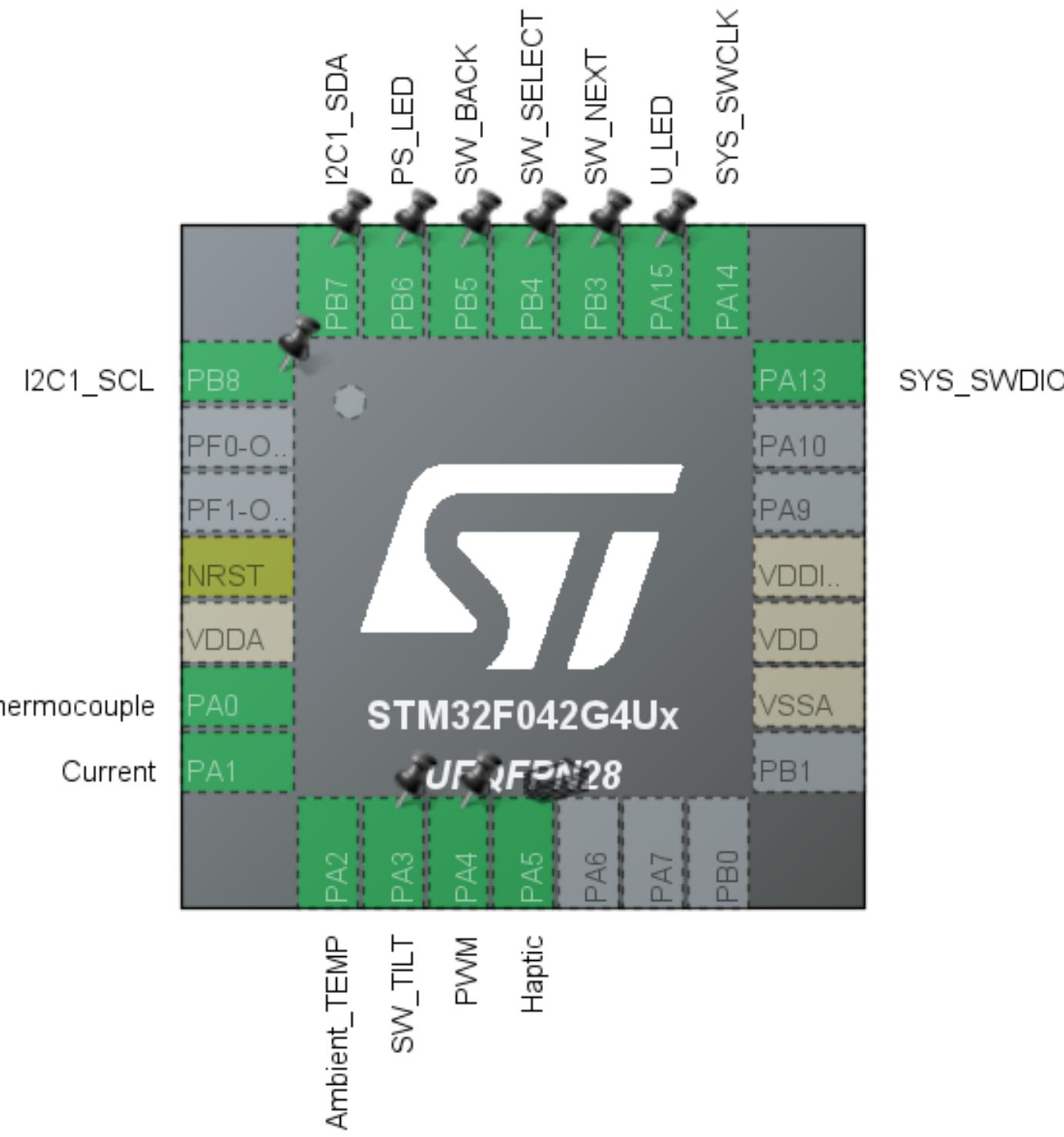
# Zmienne globalne

- **uint16\_t setTemperature** - Zmienna przechowuje temperaturę zadaną grotu,
- **uint16\_t currentTemperature** - Temperatura odczytana z termopary grotu,
- **bool screenState** - Zmienna przechowująca status ekranu, jeżeli FALSE ekran wyświetla ekran główny, jeżeli TRUE ekran wyświetla menu,
- **volatile uint8\_t work\_status** - Zmienna przechowuje aktualny stan pracy (working, ready).

# Stany pracy lutownicy

- **working** - stan, w którym grot utrzymuje temperaturę zadaną, ekran ma pełną jasność.
- **ready to work** - stan, w którym temperatura grotu jest obniżona o wartość zdefiniowaną przez użytkownika, menu ma obniżoną jasność, przerwanie od sensora vibracji kolby lutownicy lub któregokolwiek z przycisków spowoduje przejście do stanu working.

# Konfiguracja w STM32Cube



# Sterowanie grzałką

Grzałka sterowana przy pomocy sygnału PWM generowanym sprzętowo przez układu Timera. Do sterowania został wykorzystany algorytm PID.

```
void PWM_generation(uint8_t percentage){
    HAL_TIM_PWM_Start(&PWM_timer, PWM_Chanel);
    if (percentage > 100) percentage=100;
    TIM14->CCR1 = percentage*655;
}
```

```
#ifndef INC_PID_H_
#define INC_PID_H_

#ifndef PID_CONTROLLER_H
#define PID_CONTROLLER_H

typedef struct {

    /* Controller gains */
    float Kp;
    float Ki;
    float Kd;

    /* Derivative low-pass filter time constant */
    float tau;

    /* Output limits */
    float limMin;
    float limMax;

    /* Integrator limits */
    float limMinInt;
    float limMaxInt;

    /* Sample time (in seconds) */
    float T;

    /* Controller "memory" */
    float integrator;           /* Required for integrator */
    float prevError;            /* Required for integrator */
    float differentiator;       /* Required for differentiator */
    float prevMeasurement;      /* Required for differentiator */

    /* Controller output */
    float out;

} PIDController;

void PIDController_Init(PIDController *pid);
float PIDController_Update(PIDController *pid, float setpoint, float measurement);

#endif

#endif /* INC_PID_H_ */
```

# Obsługa ADC

Przetwornik ADC obsługuje trzy kanały które odpowiadają za:

- Pomiar temperatury termopary.
- Pomiar prądu przepływającego poprzez grzałkę.
- Pomiar temperatury otoczenia.

```
uint32_t adc_read(uint32_t channel)
{
    uint32_t value;
    ADC_ChannelConfTypeDef adc_ch;
    adc_ch.Channel = channel;
    adc_ch.Rank = 1;
    adc_ch.SamplingTime = ADC_SAMPLETIME_28CYCLES_5;
    HAL_ADC_ConfigChannel(&hadc, &adc_ch);

    HAL_ADC_Start(&hadc);
    HAL_ADC_PollForConversion(&hadc, 10);
    value = HAL_ADC_GetValue(&hadc);
    HAL_ADC_Stop(&hadc);
    return value;
}
```

# Początki problemów

- Pisanie kodu na innych mikrokontrolerach,
- Mikrokontroler o bardzo małej pamięci Flash 16KB oraz SRAM 6KB.
- Usunięcie FreeRTOS z obsługi menu
- Mikrokontroler STM32F0 brak możliwości pomiaru ADC dla trzech różnych kanałów.

| lutowica.elf - /lutowica/Debug - Jun 22, 2023, 8:32:24 PM |                |             |       |        |         |           |
|---|----------------|-------------|-------|--------|---------|-----------|
| Memory Regions  | Memory Details |             |       |        |         |           |
| Region  | Start address  | End address | Size  | Free   | Used    | Usage (%) |
| RAM   | 0x20000000     | 0x200017ff  | 6 KB  | 3,7 KB | 2,3 KB  | 38.28%    |
| FLASH   | 0x08000000     | 0x08003fff  | 16 KB | 612 B  | 15,4 KB | 96.26%    |

# Uruchomienie



**RIGOL****STOP**

H 10.0ms

10.0MSa/s  
6.00M pts**D**

208.800000ms

**T**

0.00V

Horizontal



Period



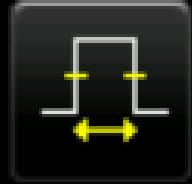
Freq



Rise Time



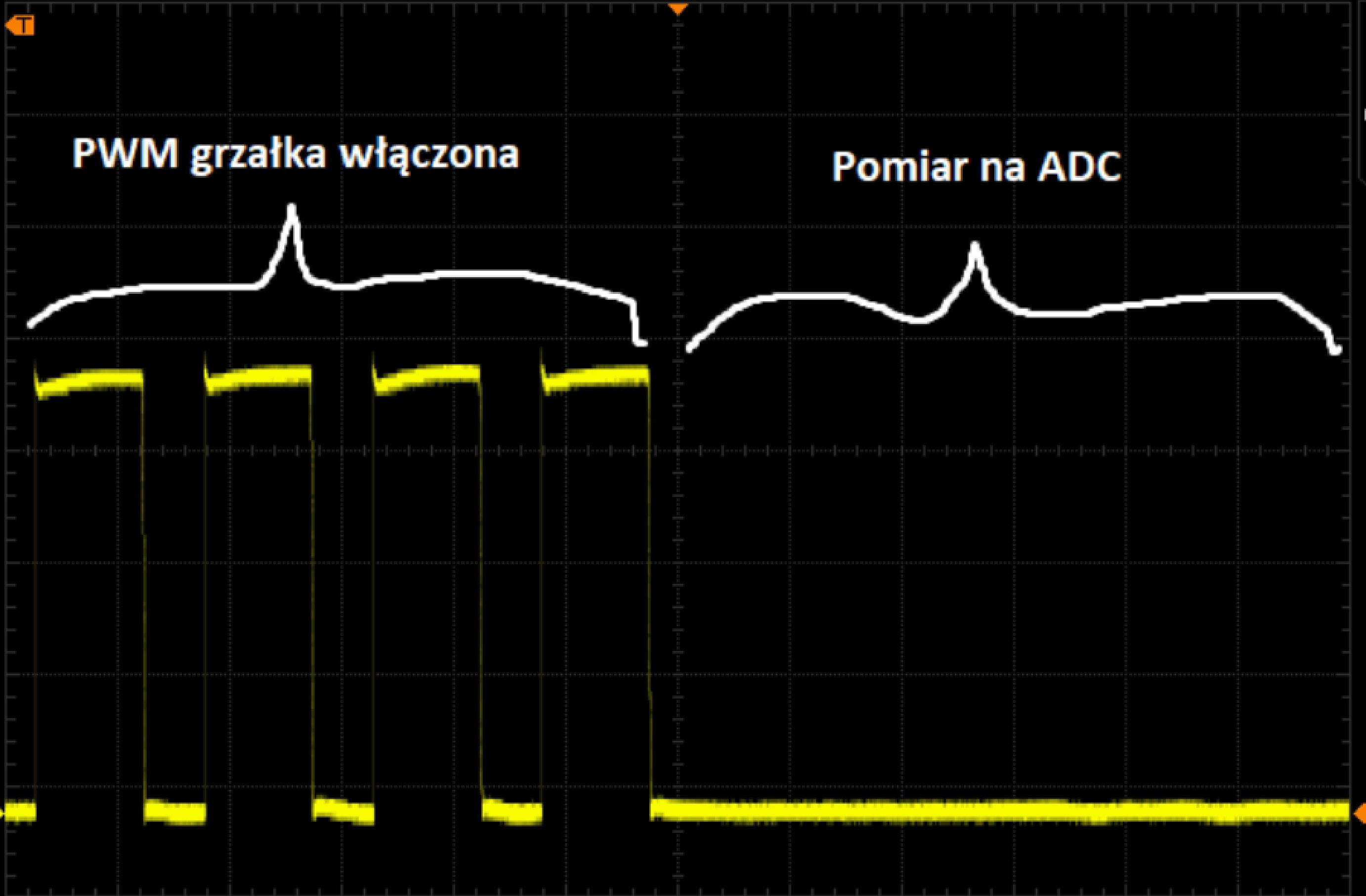
Fall Time



+Width



-Width



Save

Save

New File

NewFolder

Delete



Top=19.1 V

Vpp=21.2 V

Max=20.6 V

Min=-600mV

Rms=10.9 V

1 = 5.00 V

2 = 100uV

3 = 1.00 V

4 = 1.00 V





Dziękujemy za  
uwagę

