Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

TECHNIKA MIKROPROCESOROWA

“Projekt stacji mierzenia temperatury oraz poziomu natężenia światła z wizualizacją”

Autorzy:

Arkadiusz Orzeł
Dawid Lisek
Bartosz Żak

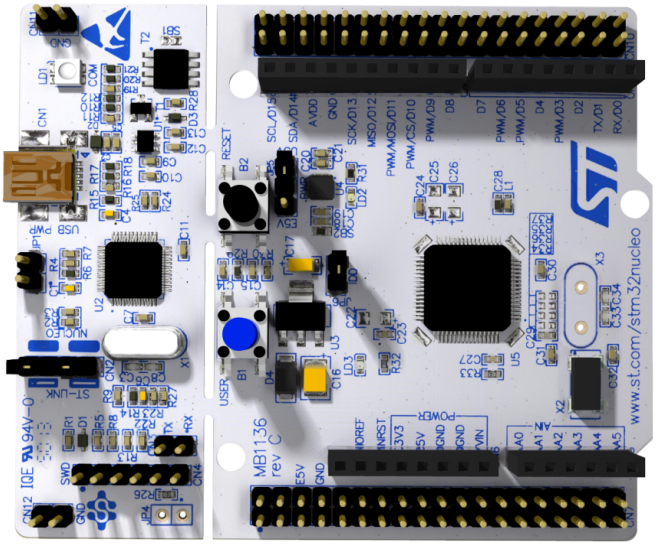
Kierunek studiów: **Automatyka i Robotyka**
Grupa: 3B

1. Wstęp

Założeniem projektu była budowa stacji do pomiaru temperatury oraz poziomu natężenia światła. Całość została zrealizowana na płytce Nucleo STM32F411 oraz dedykowanej nakładce KA-Nucleo-Multisensor. Podstawowym założeniem był odczyt wartości oraz wyświetlenie na wyświetlaczy siedmio-segmentowym opcją przełączania między odczytywanymi wartościami za pomocą przycisków. Aktualna pozycja “menu” wyświetlana jest na diodach LED. Dodatkowo zostało zrealizowane wysyłanie danych przez UART, a następnie analiza oraz wyświetlenie przy pomocy programu napisanego w języku *PYTHON*.

2. Użyte komponenty

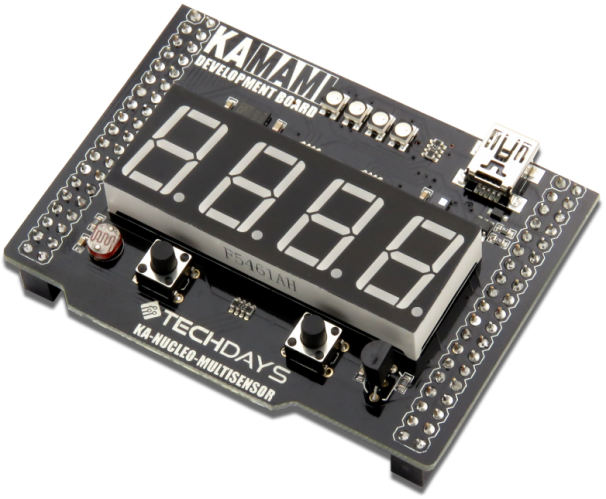
a) Płytka Nucleo STM32F411RE



Rys. 1. Nucleo STM32F411RE

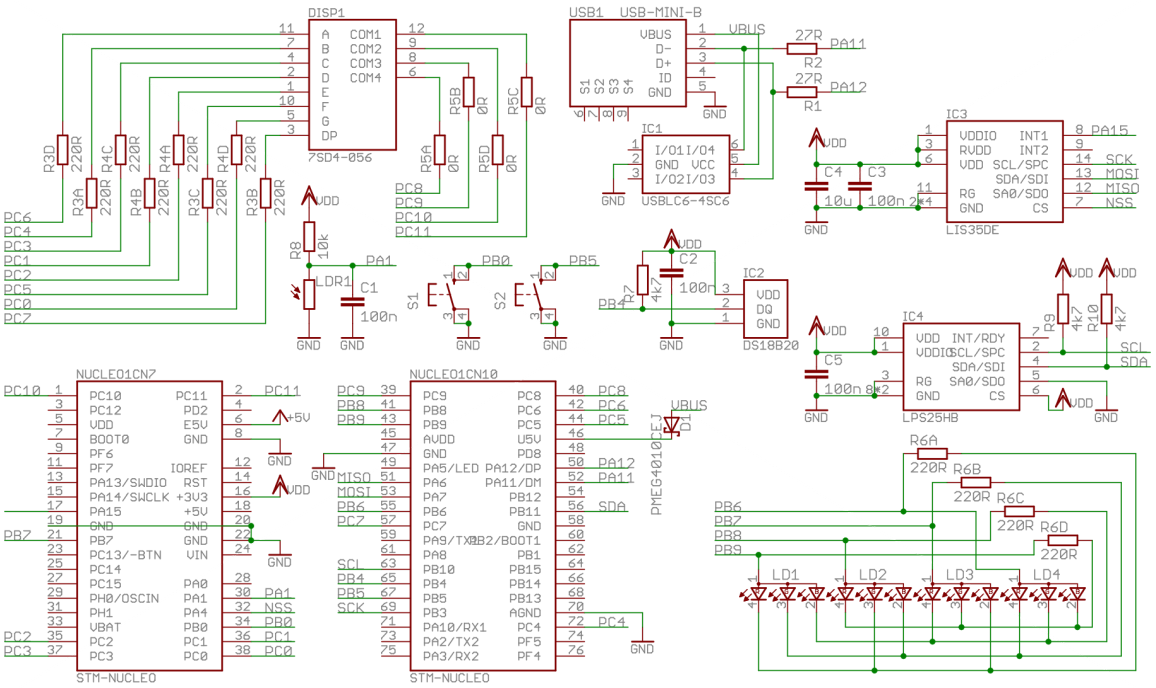
Płytka STM32 Nucleo-64 zapewnia użytkownikom niedrogi i elastyczny sposób na wypróbowanie nowych koncepcji i budowę prototypów poprzez wybór spośród różnych kombinacji cech wydajności i zużycia energii, zapewnianych przez mikrokontroler STM32.

b) Nakładka KA-Nucleo-Multisensor



Rys. 2. Nakładka KA-Nucleo-Multisensor

KA-Nucleo-Multisensor to ekspander (nakładka) rozszerzający funkcjonalność płytki uruchomieniowej serii STM32 Nucleo. Shield ma złącza ST Morpho i wyposażony został m.in. w czteroznakowy wyświetlacz 7-segmentowy, cztery diody RGB, akcelerometr LIS35DE, czujnik ciśnienia LPS25HB, a także termometr cyfrowy DS18B20.

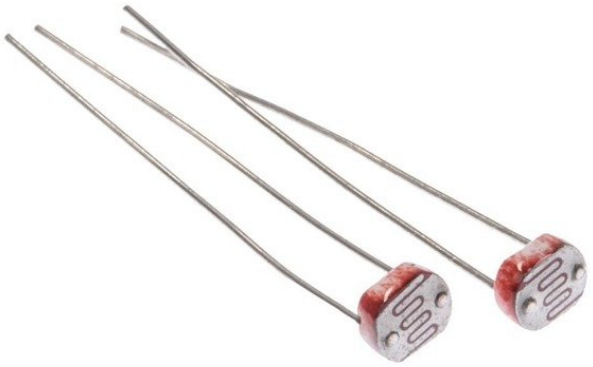


Rys. 3. Schemat elektryczny połączeń komponentów

3. Realizacja programowa

a) Odczyt z fotorezystora

Realizacja odczytu wartości natężenia światła został zrealizowany przez fotorezystor oraz przetwornik ADC. STM32F411 wyposażony jest w przetwornik 12-bitowy przetwornik ADC. Warto podkreślić, że przetworniki ADC wbudowane do naszego mikrokontrolera to zaawansowane peryferia, które dają nam bardzo duże możliwości. Sytuacja jest znacznie bardziej skomplikowana niż np. w przypadku Arduino UNO, gdzie mamy jeden przetwornik i 6 wejść analogowych.



Rys. 4. Fotorezystor

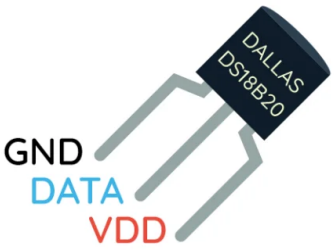
Zaczynamy od wstępnego projektu z mikrokontrolerem STM32F411, który pracuje z częstotliwością 100 MHz. Ustawiamy pin PA1 jako ADC1_init. Wcześniej został skonfigurowany zegar, a następnie przerwanie dzięki któremu mogliśmy odczytywać wartości z przetwornika ADC z odpowiednią częstotliwością.

```
switch (menu) {
case 0:
    // pomiar oswietlenia
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    ADCRes = HAL_ADC_GetValue(&hadc1);                    // 0-4095 == 0-3.3V
    ADCRes = (uint16_t)(3.3 * (double)ADCRes / 4.095);    // [mV]
    displayed_value = ADCRes;
    printf("L=%lu\n", ADCRes);
    fflush(stdout);
    HAL_Delay(100);
    break;
```

Pierwszym poleceniem jest wystartowanie przetwornika ADC, następnie odczytanie wartości, a następnie przekonwertowanie do odpowiedniej wartości. Wartości odczytane z fotorezystora są przekonwertowane na mV.

b) Odczyt czujnika temperatury DS18B20

Czujnik DS18B20, może mierzyć temperaturę w zakresie od −55°C do 125°C. Główną zaletą tych czujników jest to, że korzystają z protokołu 1-wire. W praktyce oznacza to, że potrzebna jest tylko jedna linia danych, do której możemy podłączyć wiele czujników, a wyniki bez problemu przyporządkujemy później do konkretnych sensorów – dzięki temu, że mają one unikalne adresy.



Rys. 5. Czujnik temperatury DS18B20

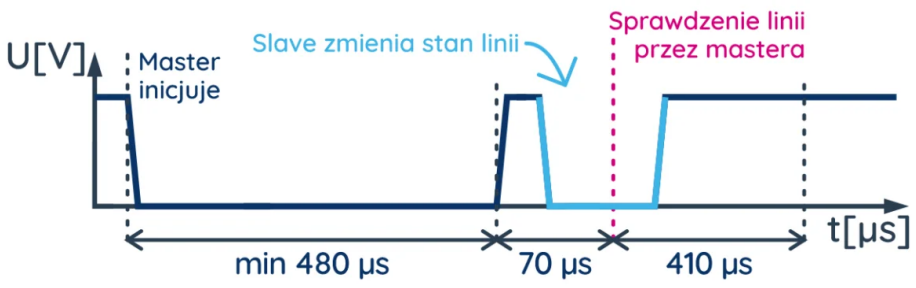
Protokół 1-wire

W tym protokole jedna linia służy zarówno do wysyłania jak i odbierania danych. Podobnie działa linia SDA w protokole I2C. Pin ustawiany jest w tryb otwartego drenu. Dlatego na nakładce dodatkowo podpięty jest sygnał zasilania przez rezystor ograniczający prąd.

Domyślnym stanem linii jest stan wysoki, a układy podłączone mogą wymuszać stan niski. Ważne jest to aby zachowywać tutaj czasy trwania sekwencji, które są podawane na linię danych.

Sekwencja reset w 1-wire

Komunikacja rozpoczyna się od wygenerowania przez układ odpowiedniej sekwencji. Sygnał ten informuje układy podrzędne o rozpoczęciu transmisji.



Rys. 6. *Sekwencja reset, informująca o początku transmisji 1-wire*

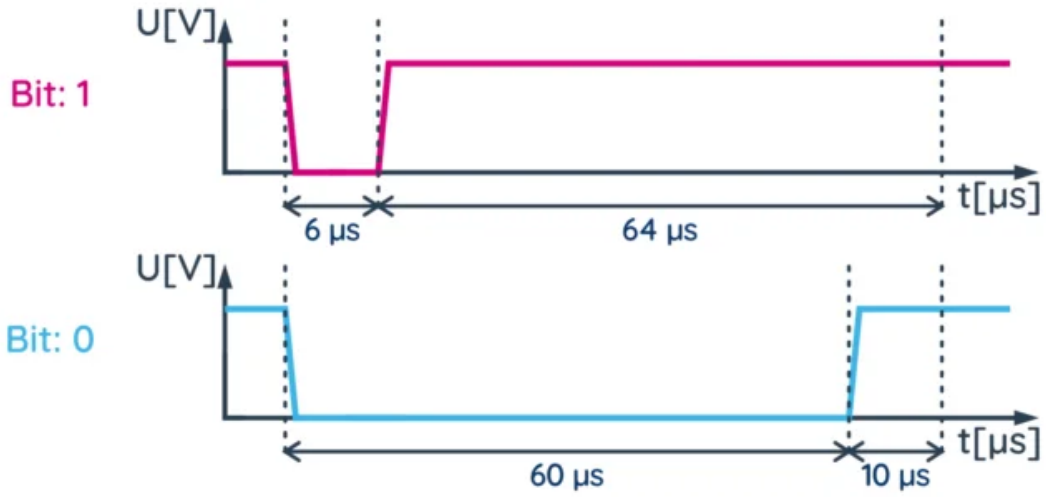
Odczyt danych 1-wire

Odczyt danych jest również inicjowany przez układ nadrzędny. W tym przypadku jest to wymuszenie stanu niskiego przez 6 µs, potem master czeka 9 µs, po czym następuje odczyt stanu linii. Stan niski wskazuje na odbiór bitu oznaczającego 0, a wysoki – 1. Po odczycie układ nadrzędny musi też odczekać 55 µs przed kolejną transmisją.

Rys. 7. Sekwencja rozpoczynająca odczyt danych przez 1-wire

Wysyłanie danych (kodowanie bitów) przez 1-wire

W przypadku 1-wire wysyłanie bitu reprezentującego logiczną jedynkę to 6 µs stanu niskiego oraz 64 µs stanu wysokiego. Z kolei zero realizowane jest przez wymuszenie przez 60 µs stanu stanu niskiego, a następnie stanu wysokiego przez kolejne 10 µs.



Rys. 8. Sposób reprezentacji bitów w 1-wire

```
case 1:
    // pomiar temperatury
    ds18b20_start_measure(NULL);
    HAL_Delay(500);
    float temp = ds18b20_get_temp(NULL);
    displayed_value = temp;
    printf("T=%.2f\n", temp);
    fflush(stdout);
    break;
  }
```

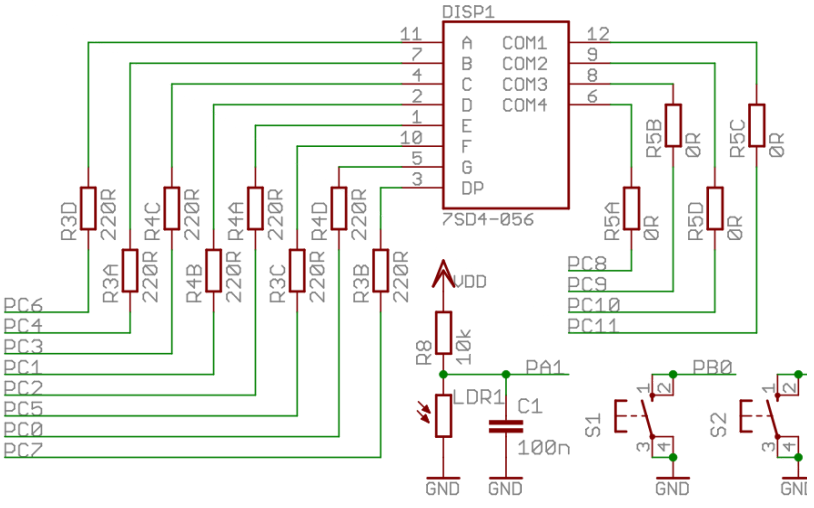
c) Wyświetlanie wartości na wyświetlaczu 7 - segmentowym

Wyświetlacz siedmiosegmentowy – wyświetlacz składający się z siedmiu segmentów, przeznaczony do wyświetlania cyfr dziesiętnych; wiele modeli ma też ósmy segment będący kropką. Na wyświetlaczu możliwe jest też wyświetlanie niektórych liter, a także umownych znaków, co jest używane do prezentacji różnych informacji. Wyświetlacze te mają wyprowadzenia sterujące każdym segmentem oddzielnie.

Schemat połączenia na KA-Nucleo-Multisensor:

Kodowanie szesnastkowe do wyświetlania znaków od 0 do F

Cyfra	Wyświetlacz	gfedcba	abcdefg	a	b	c	d	e	f	g
0	0	0x3F	0x7E	wł	wł	wł	wł	wł	wł	wył
1	1	0x06	0x30	wył	wł	wł	wył	wył	wył	wył
2	2	0x5B	0x6D	wł	wł	wył	wł	wł	wył	wł
3	3	0x4F	0x79	wł	wł	wł	wł	wył	wył	wł
4	4	0x66	0x33	wył	wł	wł	wył	wył	wł	wł
5	5	0x6D	0x5B	wł	wył	wł	wł	wył	wł	wł
6	6	0x7D	0x5F	wł	wył	wł	wł	wł	wł	wł
7	7	0x07	0x70	wł	wł	wł	wył	wył	wył	wył
8	8	0x7F	0x7F	wł	wł	wł	wł	wł	wł	wł
9	9	0x6F	0x7B	wł	wł	wł	wł	wył	wł	wł
A	A	0x77	0x77	wł	wł	wł	wył	wł	wł	wł
b	b	0x7C	0x1F	wył	wył	wł	wł	wł	wł	wł
C	C	0x39	0x4E	wł	wył	wył	wł	wł	wł	wył
d	d	0x5E	0x3D	wył	wł	wł	wł	wł	wył	wł
E	E	0x79	0x4F	wł	wył	wył	wł	wł	wł	wł
F	F	0x71	0x47	wł	wył	wył	wył	wł	wł	wł



Z układu można wywnioskować że podłączone segmentu są do wspólnej katody. To znaczy że musimy zapewnić multipleksowanie sygnału, aby zapalać po kolei każdy segment z odpowiednią wartością jaką chcemy wyświetlać.

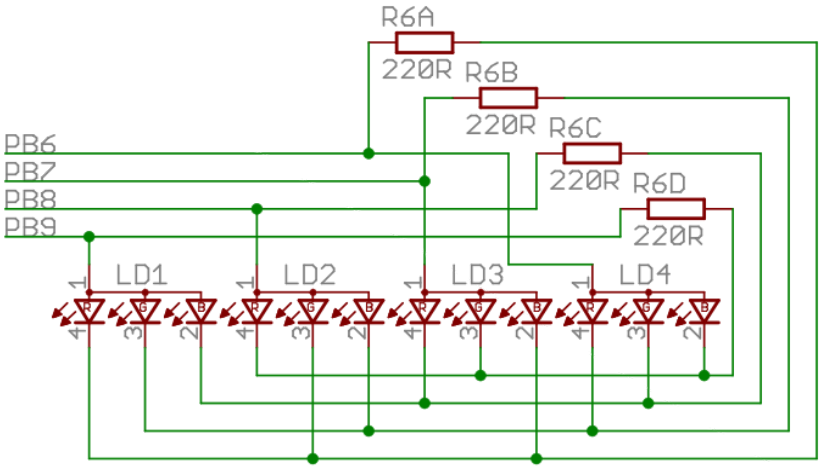
Na wyświetlaczu zostało zrealizowane wyświetlanie wartości mierzonych. Wstępnie pojawia się litera odpowiadająca i wskazująca na wyświetlaną wartość, a następnie następuje zobrazowanie wyników.

L - Wartość natężenia światła
t - Wartość temperatury

```
// obsługa wyświetlacza 7 segmentowego
void interrupt_display(float num) {
    // wyświetlanie litery odpowiadającej rodzajowi pomiaru
    if (display_letter) {
        switch (menu) {
        case 0:
            display_digit(17, 1);
            break;
        case 1:
            display_digit(16, 1);
            break;
        }
        display_counter++;
        if (display_counter > 1000) {
            display_letter = false;
            display_counter = 0;
        }
    }
    // wyświetlanie wartości pomiaru
    else {
        int calkowita = (int) num;
        int ulamkowa = (int) ((num - calkowita) * 100);

        if (menu == 0) {
            int num1 = (int) num % 10;
            int num10 = (int) num / 10 % 10;
            int num100 = (int) num / 100 % 10;
            int num1000 = (int) num / 1000 % 10;

            switch (display) {
            case 4:
                display_digit(num1, 4);
                display = 0;
                break;
            case 3:
                if ((num1000 != 0) | (num100 != 0) | (num10 != 0)) display_digit(num10, 3);
                break;
            case 2:
                if ((num1000 != 0) | (num100 != 0)) display_digit(num100, 2);
                break;
            case 1:
                if (num1000 != 0) display_digit(num1000, 1);
                break;
            }
        }
        else {
```

```
        else {
            int calkowita1 = calkowita % 10;
            int calkowita10 = calkowita / 10 % 10;
            int ulamkowa1 = ulamkowa % 10;
            int ulamkowa10 = ulamkowa / 10 % 10;
            switch (display) {
            case 4:
                display_digit(ulamkowa1, 4);
                display = 0;
                break;
            case 3:
                display_digit(ulamkowa10, 3);
                break;
            case 2:
                display_digit(calkowita1, 2);
                HAL_GPIO_WritePin(GPIOC, S7DP_Pin, 1);
                break;
            case 1:
                if (calkowita10 != 0) display_digit(calkowita10, 1);
                break;
            }
        }

        display++;
    }
}
```

d) Sterowanie diodami LED (MENU)



Rys. 9. Schemat podłączenia diod LED RGB

Na diodach zostało zrealizowane wyświetlanie aktualnego położenia w menu, a tym samym wskazanie jaką wartość wyświetlamy.

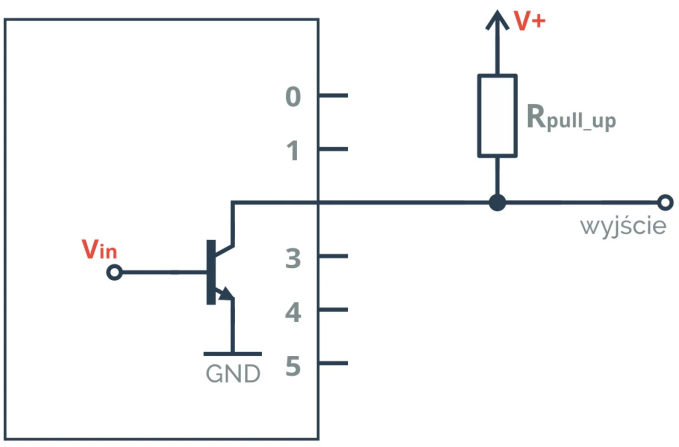
```
else if (htim == &htim4) {
    switch (menu){
      case 0:
        HAL_GPIO_WritePin(PB9_GPIO_Port, PB9_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PB8_GPIO_Port, PB8_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PB7_GPIO_Port, PB7_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(PB6_GPIO_Port, PB6_Pin, GPIO_PIN_SET);
        break;
      case 1:
          HAL_GPIO_WritePin(PB9_GPIO_Port, PB9_Pin, GPIO_PIN_RESET);
          HAL_GPIO_WritePin(PB8_GPIO_Port, PB8_Pin, GPIO_PIN_RESET);
          HAL_GPIO_WritePin(PB7_GPIO_Port, PB7_Pin, GPIO_PIN_SET);
          HAL_GPIO_WritePin(PB6_GPIO_Port, PB6_Pin, GPIO_PIN_RESET);
          break;
  }

}
// obsługa przycisków
```

Diody są zapalane poprzez ustawienie stanu wysokiego na jednej linii oraz stanu niskiego na pozostałych. W poszczególnej diodzie są zapalane wszystkie kolory RGB dzięki czemu dioda emituje białe światło.

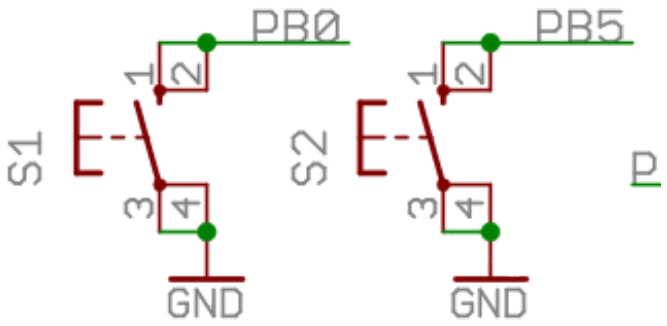
e) Przyciski do przełączania wartości

Aby zapewnić odpowiednie działanie przycisków ważne jest pociągnięcie ich do rezystora zasilającego (pull-up).

Rezystor podciągający to zwykły opornik który podłączony jest między wejście układu, a dodatnią szynę zasilania. Jeśli wyprowadzenie układu cyfrowego zostało skonfigurowane jako wejście cyfrowe to stan na tym pinie nie będzie zbyt stabilny. Wszystkie zakłócenia elektrostatyczne i elektromagnetyczne będą odbierane przez układ jako chaotyczne zmiany stanu logicznego na wyjściu. Rezystor podciągający pozwala spolaryzować wejście układu stabilnym napięciem pochodzącym z dodatniej szyny zasilania układu.



Rys. 10. Schemat podłączenia przycisku (pull-up)



Rys. 10. Schemat podłączenia na nakładce KA-Nucleo-Multisensor

f) Wysyłanie danych przez UART

W mikrokontrolerach stosuje się najczęściej uproszczoną wersję tego interfejsu, która działa np. na napięciach 0 V i 3,3 V. Moduł odpowiedzialny za obsługę takiej komunikacji nazywany jest UART (ang. *universal asynchronous receiver and transmitter*) lub też jako USART (ang. *universal synchronous asynchronous receiver-transmitter*). Druga wersja oznacza po prostu, że ten sam moduł może działać jako interfejs synchroniczny lub asynchroniczny (z tej części korzysta się najczęściej).

W przypadku UART-u transmisja rozpoczyna się od bitu startu (na rysunku jako BS); zawsze jest to bit będący logicznym zerem (0 V). Potem, zależnie od konfiguracji, następuje po sobie 7, 8 lub 9 bitów danych (na rysunku jest ich 8, od B0-B7), które są wysyłaną informacją (najczęściej jeden bajt). Z kolei bit stopu (tutaj jako BK) to bit będący logiczną jedynką – mówi o końcu transmisji. Format ramki oraz sposób transmisji UART-u jest właściwie niezmieniony względem RS-232.

Rys. 11. Przykładowy przebieg UART-u

Płytka Nucleo posiada wbudowany konwerter z UART na USB. Nie mamy więc po drodze nigdzie „prawdziwego” RS-232. Po podłączeniu do PC przejściówka z płytki Nucleo będzie widziana jako port COM, czyli nasz port szeregowy.

W projekcie został użyty USART2 (w kategorii Connectivity). W ustawieniach wybieramy tryb asynchroniczny (Mode na Asynchronous). Wszystkie pozostałe ustawienia zostały ustawione na domyślne:

- **Baud Rate** – to prędkość komunikacji przez nasz port szeregowy; domyślna wartość to standardowe 115 200 bitów na sekundę (tyle nam teraz wystarczy).
- **Word Length** – domyślnie przesyłamy dane 8-bitowe, co odpowiada wielkości typowego bajta. Można jednak wysyłać też dane w formacie 7- lub 9-bitowym.
- **Parity** – bit parzystości; nie używamy go, więc zostawiamy opcję None.
- **Stop Bits** – jeden bit stopu jest bardzo popularnym trybem, ale warto wiedzieć, że można używać też innych wartości.

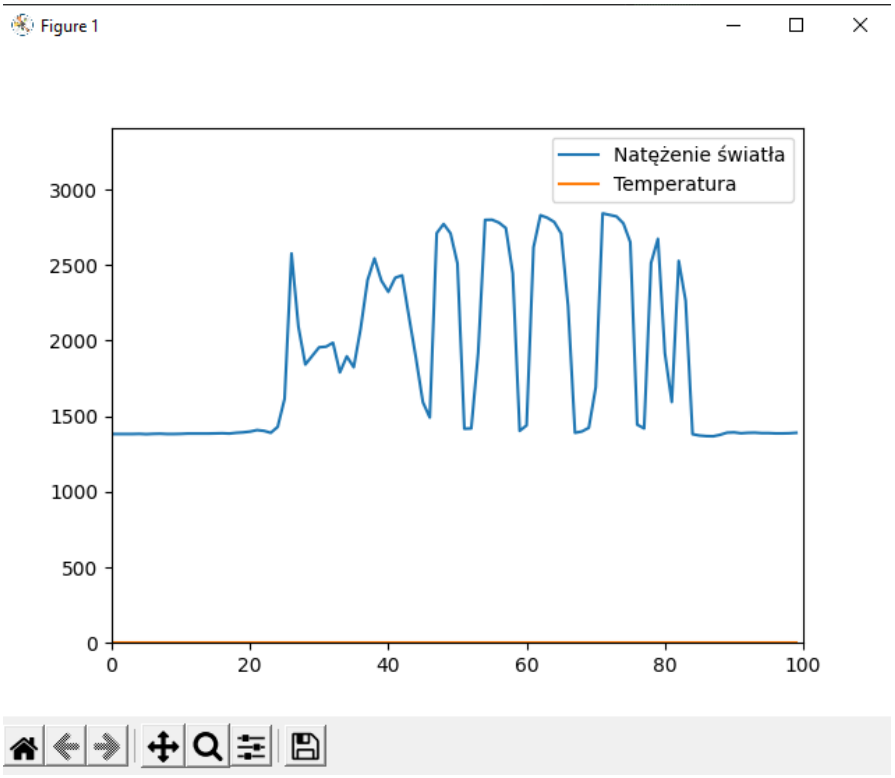
Dla ułatwienia wysyłania danych przez UART została nadpisana funkcja ***__io_putchar()***, która wykorzystana jest w funkcji ze standardowej biblioteki *stdio.h, **printf()***

```
// wysyłanie znaku na UART
int __io_putchar(int ch)
{
    if (ch == '\n') {
        uint8_t ch2 = '\r';
        HAL_UART_Transmit(&huart2, &ch2, 1, HAL_MAX_DELAY);
    }

    HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
    return 1;
}
```

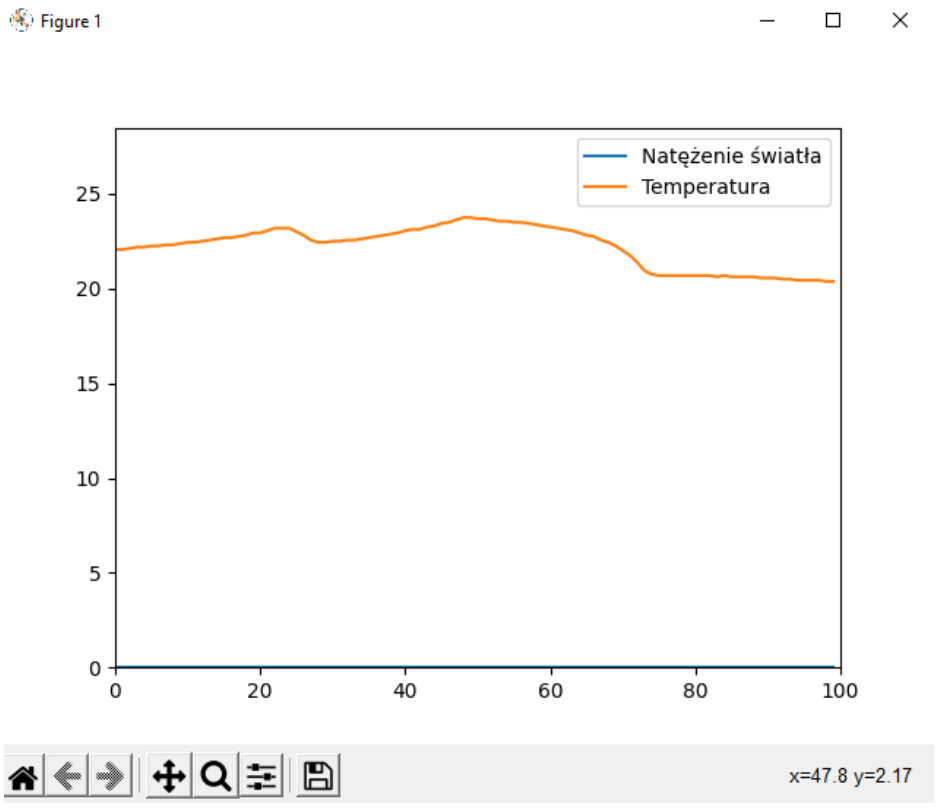
4. Program do wizualizacji danych na wykresach czasu rzeczywistego

Program do wizualizacji danych został napisany w języku *PYTHON*, przy pomocy bibliotek matplotlib.
Do uruchomienia programu należy wywołać w konsoli:

```
python3 main.py --port COM3
```



Rys. 12. Przebieg odczytywanego natężenia światła [mV]



Rys. 13. Przebieg odczytywanej [ºC]

5. Wnioski.

Powyższy projekt jest świetnym przykładem możliwości jakie daje nam płytka Nucleo STM32F411RE wraz z nakładką KA-Nucleo-Multisensor. Podczas wykonywania tego projektu zmierzyliśmy się z pewnymi problemami, które wymagały od nas znajomości zasad działania elementów elektronicznych oraz umiejętności obsługi mikroprocesora tego typu. Oprócz powyższych funkcjonalności próbowaliśmy jeszcze wykonać pomiar ciśnienia z czujnika LPS25HB. Podczas prób odczytu danych zauważyliśmy nie pełną kompatybilność ekspandera z płytką Nucleo. Wyjścia SCL i SDA z ekspandera nie pokrywają się z wyjściami SCL i SDA, które znajdują się na naszej płytce Nucleo. Dużą zaletą naszego projektu jest odczyt danych z magistrali UART oraz tworzenie wykresu w czasie rzeczywistym za pomocą pakietu matplotlib. Wczytanie danych do języka Python daje nam w zasadzie nieograniczone możliwości ich wykorzystania. Kolejną funkcjonalnością do zaimplementowania w przyszłości mogłoby być stworzenie bazy danych pogodowych za pomocą pakietu Pandas.