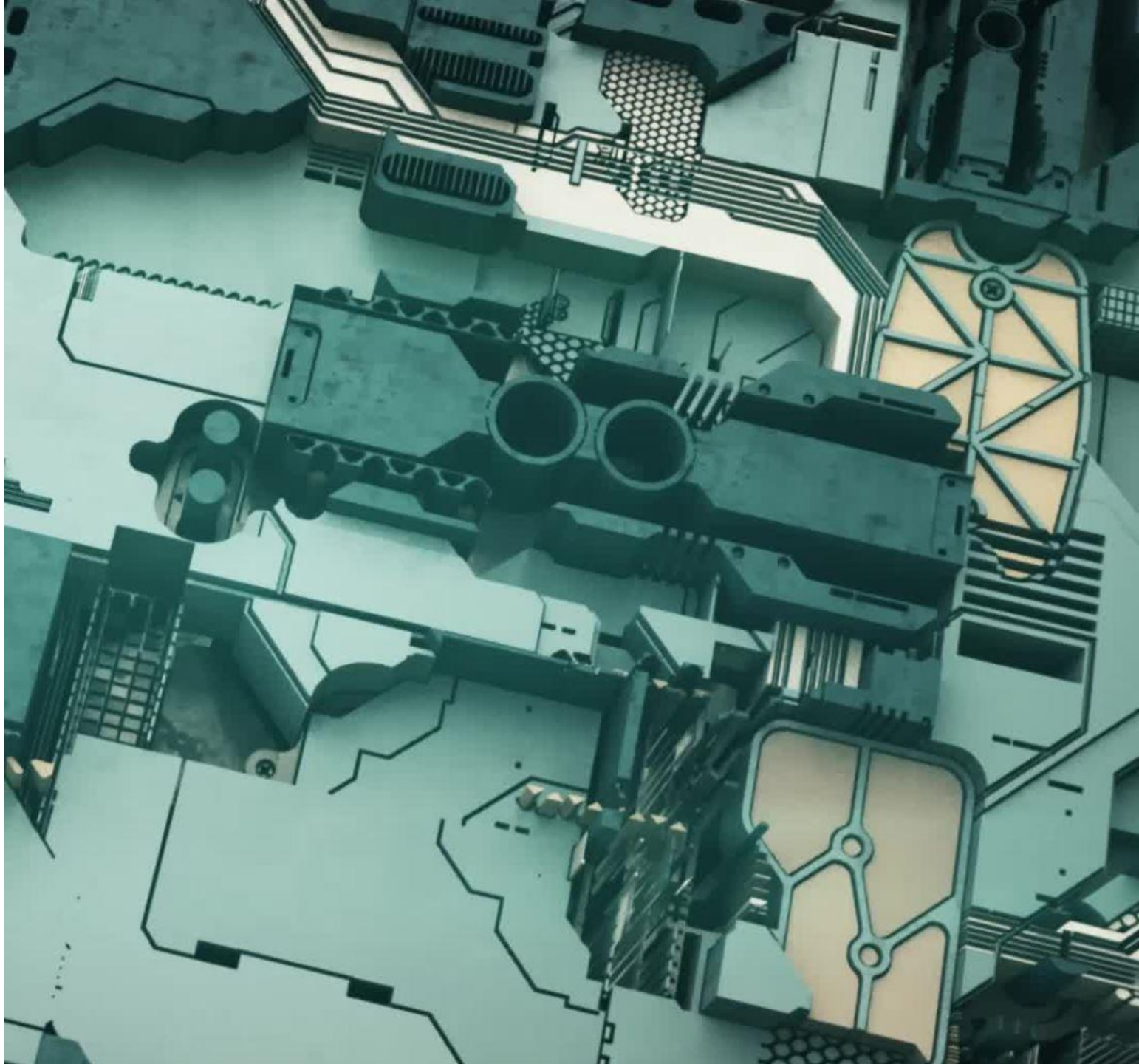
MINIATUROWA STACJA POGODOWA

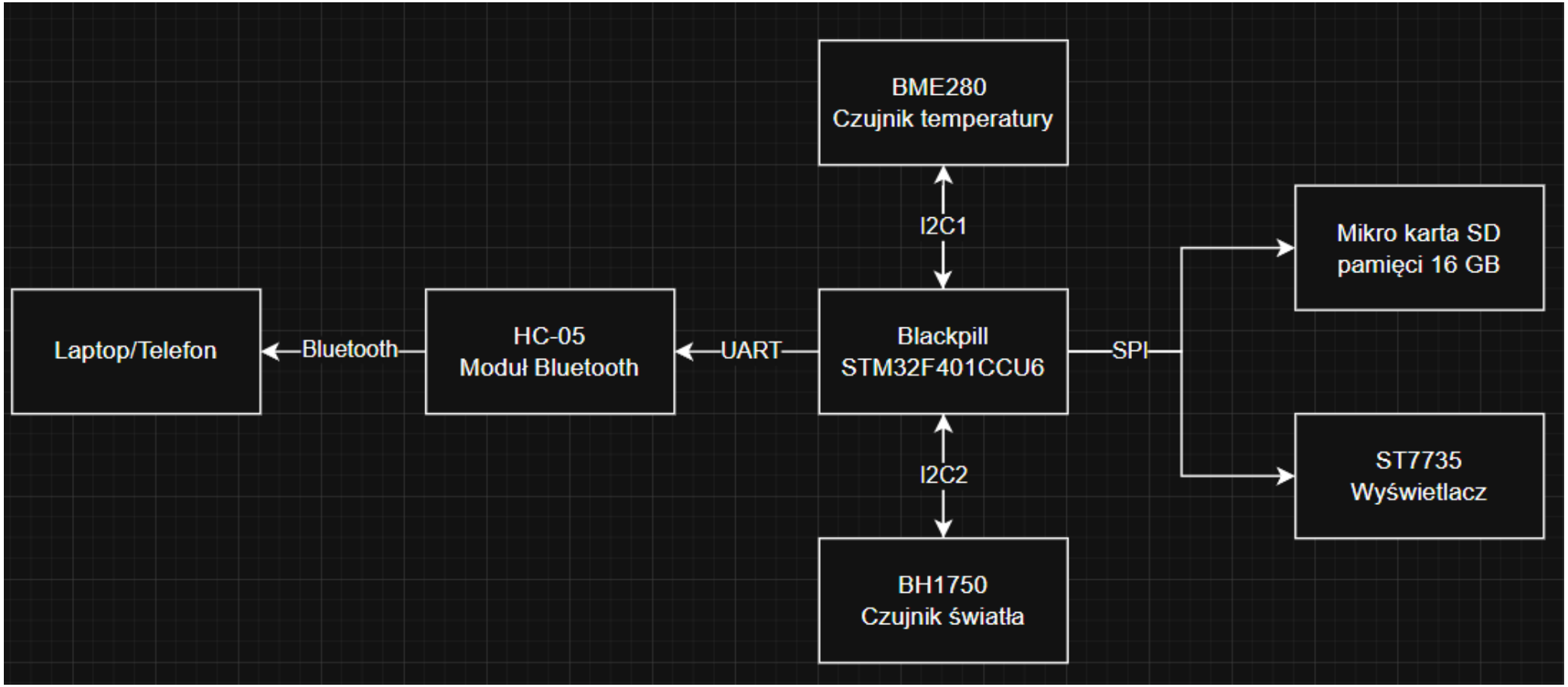
PMIK

Artur Skrzypczak

Illia Kovalenko



SCHEMAT BLOKOWY



HARMONOGRAM PRAC

Etap	Zadanie	Odpowiedzialna osoba	Termin zakończenia zadania i czas realizacji
4	Implementacja odczytu temperatury, wilgotności, ciśnienia i przetwarzania danych	Artur Skrzypczak	6.04 (1 tydzień)
6	Implementacja pomiaru natężenia światła oraz konfiguracja RTC	Illia Kovalenko	20.04 (1 tydzień)
7	Integracja karty microSD oraz zapis przetwarzanych danych	Illia Kovalenko	27.04 (1 tydzień)
8	Implementacja komunikacji bezprzewodowej (Bluetooth)	Artur Skrzypczak	27.04 (2 tygodnie)
10	Optymalizacja kodu i testy urządzenia, ewentualne korekty	Illia Kovalenko	11.05 (1 tydzień)
11	Dokumentacja i prezentacja całego projektu	Zespół	25.05 (2 tygodnie)

NAGŁÓWEK INFORMACYJNY.
ZMIENNE I STRUKTURY - POCZĄTEK

```
/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
I2C_HandleTypeDef hi2c2;

RTC_HandleTypeDef hrtc;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
uint8_t getdata;

RTC_TimeTypeDef sTime; // 2 struktury do ustawienia daty i godziny
RTC_DateTypeDef sDate;
char timeBuf[16]; //godzina
char dateBuf[16]; //data

struct bme280_dev bme;
struct bme280_data sensor_data;
uint8_t bme280_addr = BME280_I2C_ADDR_PRIM;  // adres 0x76

// Tablica pomocnicza do tworzenia napisów wyświetlanych na ekranie
char debugBuf[32];
/* USER CODE END PV */
```

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Program obslugujacy stacje pogodowa dokonujaca pomiarow
  *     temperatury, wilgotnosci, cisnienia, naslonecznienia oraz wyswietlajacej te dane
  *     na wyswietlaczu. Obsluguje takze zegar RTC, z ktorego czas tez jest wyswietlany.
  *     Program powstaje w ramach przedmiotu PMIK na Politechnice Warszawskiej
  *
  *     JEST TO WERSJA ROBOCZA, NIE OBSLUGUJE JESZCZE WSZYSTKICH FUNKCJONALNOSCI
  *         TE BEDA STOPNIOWO DODAWANE
  * @authors        : Artur Skrzypczak, Illia kovalenko
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "ST7735.h"
#include "GFX_FUNCTIONS.h"
#include <stdio.h> //dla wykorzystania sprintf
#include "bme280.h"
#include "bme280_stm32_hal.h"
#include "bh1750.h"
#include <string.h>
```

FUNKCJA MAIN

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_SPI1_Init();
  MX_RTC_Init();
  MX_I2C1_Init();
  MX_I2C2_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  // Uruchomienie przerwań od timera TIM1
  //(używane do okresowego odświeżania danych)
  HAL_TIM_Base_Start_IT(&htim1);

  // Konfiguracja bieżącego czasu zegara RTC
  //(np. po uruchomieniu mikrokontrolera)
  // Ustawiamy czas: 00:00:00
  sTime.Hours = 0;
  sTime.Minutes = 0;
  sTime.Seconds = 0;
  HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BIN);

  // Ustawiamy date: Poniedziałek, 1 stycznia 2025
  sDate.WeekDay = RTC_WEEKDAY_MONDAY;
  sDate.Month = RTC_MONTH_JANUARY;
  sDate.Date = 1;
  sDate.Year = 25;
  HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BIN);

  /*Dezaktywacja WakeUp Timer RTC
  // musi być wykonana przed jego ponowną konfiguracją
  // RTC nie pozwala nadpisać aktywnego
  timera bez wcześniejszego wyłaczenia*/
  HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);

  // Inicjalizacja wyświetlacza ST7735 i czyszczenie ekranu
  ST7735_Init(0);
  fillScreen(BLACK);
```

```
  // Konfiguracja struktury czujnika BME280 (wilgotność, temperatura, ciśnienie)
  bme.intf = BME280_I2C_INTF;
  bme.intf_ptr = &bme280_addr;
  bme.read = user_i2c_read;
  bme.write = user_i2c_write;
  bme.delay_us = user_delay_us;

  // Ustawienie parametrów pomiarowych BME280
  struct bme280_settings settings;
  settings.osr_h = BME280_OVERSAMPLING_1X;
  settings.osr_p = BME280_OVERSAMPLING_1X;
  settings.osr_t = BME280_OVERSAMPLING_1X;
  settings.filter = BME280_FILTER_COEFF_2;
  settings.standby_time = BME280_STANDBY_TIME_1000_MS;

  uint8_t settings_sel = BME280_SEL_OSR_PRESS | BME280_SEL_OSR_TEMP |
                         BME280_SEL_OSR_HUM | BME280_SEL_FILTER;

  if (bme280_init(&bme) != BME280_OK) {
      Error_Handler();
  }
  bme280_set_sensor_settings(settings_sel, &settings, &bme);

  // Ponowna dezaktywacja WakeUp Timer (na wypadek jeśli był jeszcze aktywny)
  HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);

  // Aktywacja WakeUp Timer RTC z przerwaniem – co określony czas wywoła callback
  HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 4095, RTC_WAKEUPCLOCK_RTCCLK_DIV16);

  // Inicjalizacja czujnika światła BH1750 w trybie ciągłego
  // pomiaru o wysokiej rozdzielczości
  BH1750_Init(&hi2c2, BH1750_CONT_HIGH_RES_MODE);

  //wysylanie 1 komunikatu przez Bluetooth
  char msg[] = "Hello from STM32\n";
  HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
  HAL_Delay(1000);
```

PĘTLA WHILE

```
while (1)

{
    if (getdata) {
        getdata = 0;//zerowanie flagi od timera tim1

        // Odczyt danych z czujnika swiatla BME280
        bme280_set_sensor_mode(BME280_POWERMODE_FORCED, &bme);

       /*Tryb FORCED oznacza, ze czujnik sie budzi, aby wykonac jeden pomiar, po czym
       * przechodzi w stan uspienia SLEEP. W tej wersji wybudzamy go na poczatku petli,
       * potem idzie spac, co zmniejsza pobor pradu.
       * Inna mozliwoscia jest zastosowanie trybu NORMAL, w ktorym bme wykonuje pomiary ciagle,
       * bez przechodzenia w SLEEP. Wowczas: BME280_POWERMODE_NORMAL
       */

        bme.delay_us(2000 * 1000, bme.intf_ptr);//odczekanie, az sie obudzi i zmierzy
        if (bme280_get_sensor_data(BME280_ALL, &sensor_data, &bme) == BME280_OK) {
            char buf[32]; //tablica pomocnicza do wyswietlania danych

            //wyswietlanie temperatury
            sprintf(buf, "%.1f C", sensor_data.temperature);
            ST7735_WriteString(10, 10, buf, Font_16x26, WHITE, BLACK);

            //wyswietlanie wilgotnosci
            sprintf(buf, "HUM: %.0f %%", sensor_data.humidity);
            ST7735_WriteString(35, 100, buf, Font_7x10, WHITE, BLACK);

            //wyswietlanie cisnienia
            sprintf(buf, "PRESS: %.1f HPA", sensor_data.pressure / 100.0);
            ST7735_WriteString(3, 120, buf, Font_7x10, WHITE, BLACK);

            /*ODCZYT DANYCH Z BH1750 - czujnik nasłonecznienia
            *Tryb ciągły (CONTINUOUS HIGH RES MODE) pozwala odczytywać dane
            *bez każdorazowego inicjalizowania pomiaru.
            *Jednak ze względu na prostotę i dokładność, w tym przypadku
            *wykonujemy pomiar recznie w każdej iteracji.
            * Uruchomienie nowego pomiaru w trybie ustawionym przy inicjalizacji
            */

            BH1750_StartMeasurement();
            HAL_Delay(180); //Odczekanie 180 ms - tyle potrzebuje czujnik na wykonanie pomiaru


            uint8_t data[2]; // Odbiór 2 bajtów danych - surowa wartość nasłonecznienia
            uint16_t lux;

            // Odczyt danych z czujnika (adres BH1750 to 0x23 przesuniete w lewo o 1 bit = 0x46)
            HAL_I2C_Master_Receive(&hi2c2, 0x46, data, 2, 100);

            // Przetwarzanie danych - łaczenie dwóch bajtów w jedna wartość 16-bitowa (w luxach)
            lux = (data[0] << 8) | data[1];

            // Wyświetlenie zaktualizowana wartość natężenia światła
            sprintf(debugBuf, "Lux: %4u lx", lux);
            ST7735_WriteString(10, 150, debugBuf, Font_7x10, WHITE, BLACK);

            // Wysłanie danych pomiarowych przez Bluetooth
            // (temperatura, wilgotność, ciśnienie, nasłonecznienie)
            char btBuf[64];
            sprintf(btBuf, "Temp: %.1f C, Hum: %.0f%%, Press: %.1f hPa, Lux: %u\r\n",
                    sensor_data.temperature,
                    sensor_data.humidity,
                    sensor_data.pressure / 100.0,  //Ciśnienie jest przeliczane z Pa na hPa
                    lux);
            BT_SendString(btBuf);  // Przesłanie przez UART2 do modułu HC-05 (Bluetooth)
            }
}
```

PROCEDURY OBSŁUGI PRZERWAŃ

```
615 /* USER CODE BEGIN 4 */
616 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
617 {
618     if (htim == &htim1) {
619         getdata = 1;//flaga, po ktorej nastepuje pobranie wynikow i ich wyswietlanie
620     }
621 }
622
623 void HAL_RTCEx_WakeUpTimerEventCallback(RTC_HandleTypeDef *hrtc)
624 {
625     HAL_RTC_GetTime(hrtc, &sTime, RTC_FORMAT_BIN);
626     HAL_RTC_GetDate(hrtc, &sDate, RTC_FORMAT_BIN);
627     sprintf(timeBuf, "%02d:%02d", sTime.Hours, sTime.Minutes);
628     sprintf(dateBuf, "%02d:%02d:%02d", sDate.Date, sDate.Month, sDate.Year);
629     ST7735_WriteString(45, 40, timeBuf, Font_7x10, WHITE, BLACK);
630     ST7735_WriteString(35, 60, dateBuf, Font_7x10, WHITE, BLACK);
631
632 }
633
634 void BT_SendString(char *str) {
635     HAL_UART_Transmit(&huart2, (uint8_t*)str, strlen(str), HAL_MAX_DELAY);
636 }
```

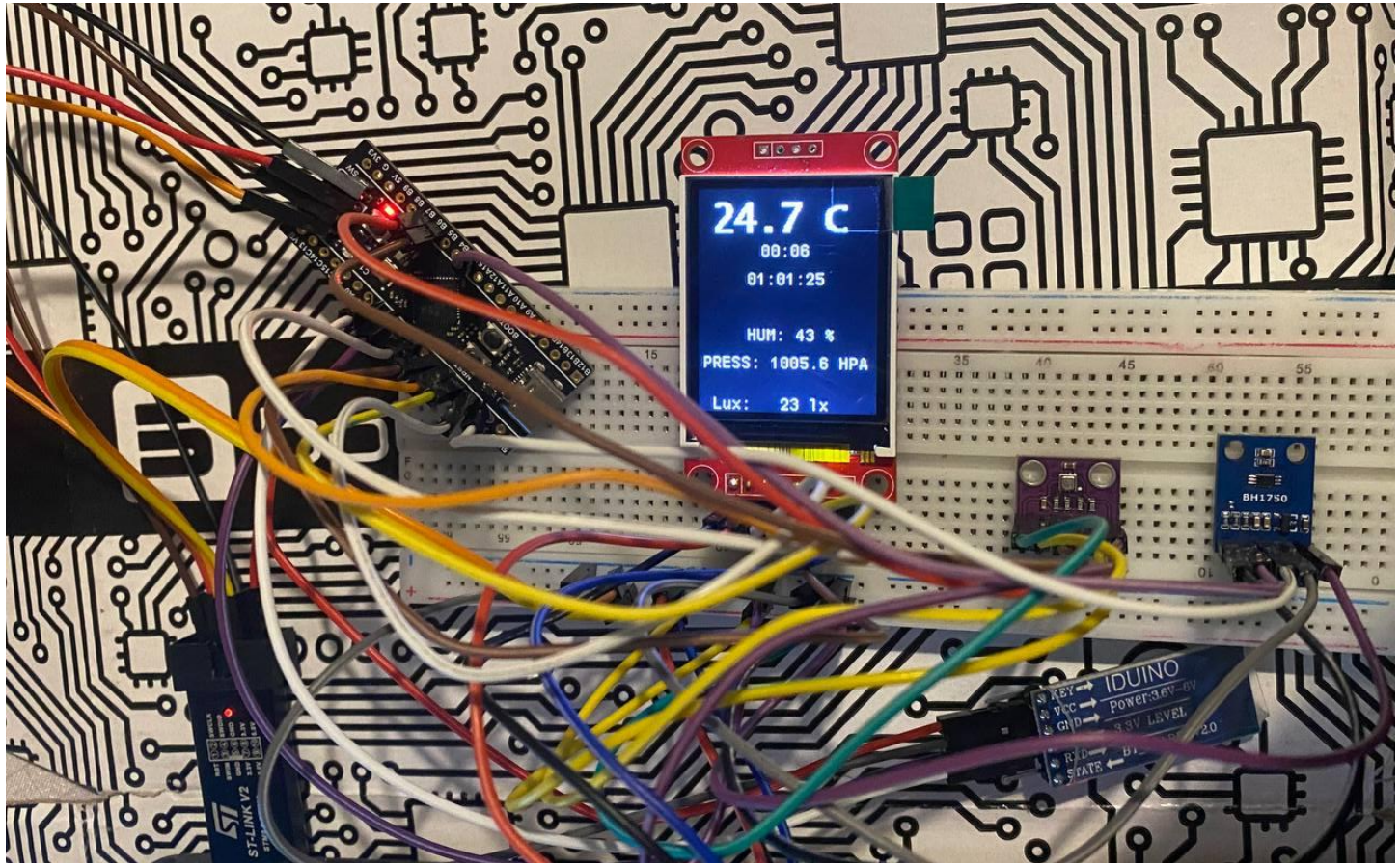
CZUJNIK NATĘŻENIA ŚWIATŁA

```c
#include "bh1750.h"
// Wewnetrzne wskaźniki do konfiguracji czujnika
static I2C_HandleTypeDef *bh1750_i2c;     // Wskaźnik na uchwyt I2C
static BH1750_Mode bh1750_mode;           // Tryb pracy wybrany przy inicjalizacji
static uint8_t bh1750_address = BH1750_ADDR_LOW;  // Domyślny adres I2C (ADDR = GND)
/**
 * @brief Inicjalizacja czujnika BH1750.
 * @param hi2c - uchwyt do magistrali I2C
 * @param mode - tryb pracy (ciągły / jednorazowy / rozdzielczość)
 */
void BH1750_Init(I2C_HandleTypeDef *hi2c, BH1750_Mode mode) {
    bh1750_i2c = hi2c;
    bh1750_mode = mode;

    uint8_t power_on = 0x01;  // Komenda "Power ON"
    uint8_t reset = 0x07;     // Komenda "Reset"

    HAL_I2C_Master_Transmit(bh1750_i2c, bh1750_address, &power_on, 1, 100);
    HAL_Delay(10); // Czekaj po włączeniu zasilania
    HAL_I2C_Master_Transmit(bh1750_i2c, bh1750_address, &reset, 1, 100);
}
/**
 * @brief Rozpoczęcie pomiaru w trybie jednorazowym (ONE_TIME_*)
 */
void BH1750_StartMeasurement(void) {
    uint8_t cmd = bh1750_mode;
    HAL_I2C_Master_Transmit(bh1750_i2c, bh1750_address, &cmd, 1, 100);
}
/**
 * @brief Odczyt danych świetlnych z czujnika.
 * @return Wartość oświetlenia w luxach (16-bit)
 */
uint16_t BH1750_ReadLux(void) {
    uint8_t data[2]; // Bufor na dane: MSB, LSB
    HAL_Delay(180);  // Maksymalny czas konwersji dla High-Res mode
    HAL_I2C_Master_Receive(bh1750_i2c, bh1750_address, data, 2, 100);
    return (data[0] << 8) | data[1]; // Połączenie bajtów w wartość 16-bitową
}
```

UKŁAD


24.7 C
00:06
01:01:25
HUM: 43 %
PRESS: 1005.6 HPA
Lux: 23 lx
BH1750
IDUINO
ST-LINK V2

CIEKAWE ROZWIĄZANIA I DALSZE POMYSŁY

```
Hello from STM32
Temp: 24.8 C, Hum: 32%, Press: 1001.8 hPa, Lux: 244

Temp: 24.8 C, Hum: 32%, Press: 1001.8 hPa, Lux: 245

Temp: 24.8 C, Hum: 32%, Press: 1001.7 hPa, Lux: 272

Temp: 24.8 C, Hum: 32%, Press: 1001.8 hPa, Lux: 273

Temp: 24.8 C, Hum: 32%, Press: 1001.8 hPa, Lux: 281

Temp: 24.8 C, Hum: 32%, Press: 1001.8 hPa, Lux: 276
```


DZIĘKUJEMY
ZA UWAGĘ