Wydział Informatyki, Architektura Komputerów, Laboratoria 2023	Data: 31.01.2024
Ćwiczenie nr 8 Temat: Projekt Grupowy Grupa ćw. 5 1. Bartłomiej Karanowski 2. Jakub Jakacki 3. Rafał Jaroma	Prowadzący: dr inż. Mirosław Omieljanowicz Ocena:

Stacja pogodowa

1. Stacja pogodowa powinna wyświetlać aktualną temperaturę, ciśnienie oraz wilgotność na ekranie LCD. Należy zadbać o odpowiedni opis, jednostki (m. in. °C) oraz estetykę danych prezentowanych na ekranie LCD.
2. Pomiary powinny być prezentowane na 3 osobnych ekranach (widokach), które cyklicznie automatycznie przełączają się między sobą na jednym wyświetlaczu LCD.
3. W pierwszej linijce należy wyświetlać aktualny pomiar, natomiast w drugiej linijce należy wyświetlić pasek, który będzie graficznie odwzorowywać wartość pomiaru.
4. Na ekranie z temperaturą należy również wyświetlić symbol słońca, gdy temperatura przekracza 30°C, symbol zachmurzonego słońca, gdy temperatura jest powyżej 24°C, ale nie przekracza 30°C oraz symbol chmur, gdy temperatura jest poniżej 24°C.
5. Ekrany można przełączać również ręcznie przyciskami LEFT i RIGHT. Cykliczne przełączanie się ekranów powinno zostać zablokowane na pewien czas po ręcznym przełączeniu ekranu.
6. Temperatura powinna być wyświetlana w jednostkach °C lub °F. Przycisk UP powinien powodować przełączanie się między jednostkami.

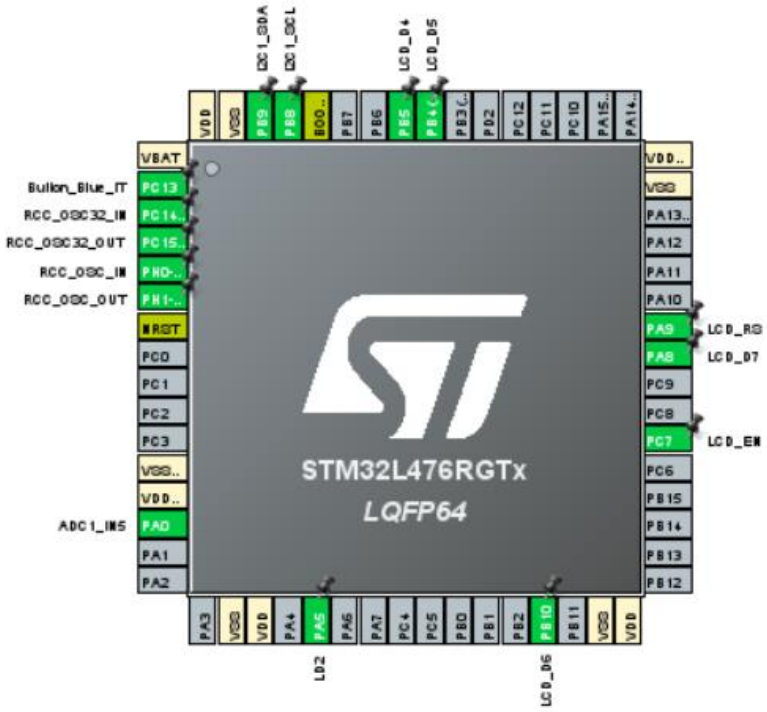
Przygotowanie programu do pracy:

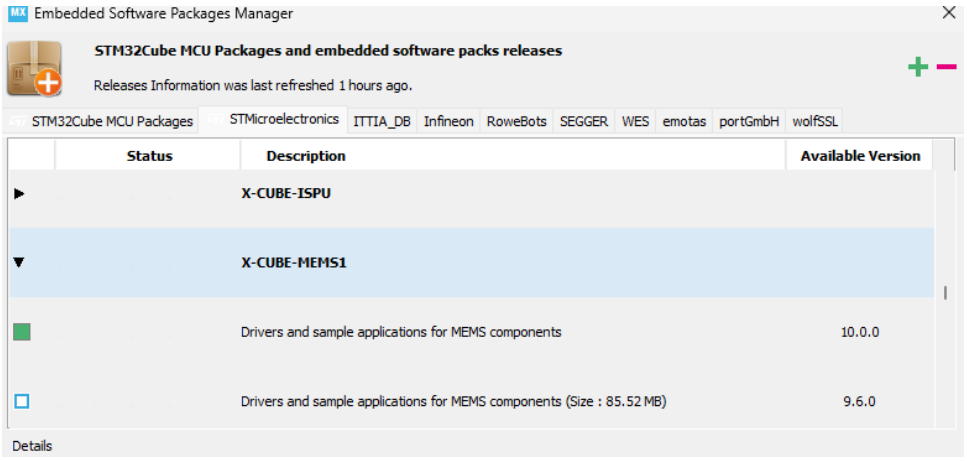
Na początku naszego zadania przygotowaliśmy odpowiednie piny zgodnie z poniższym schematem:

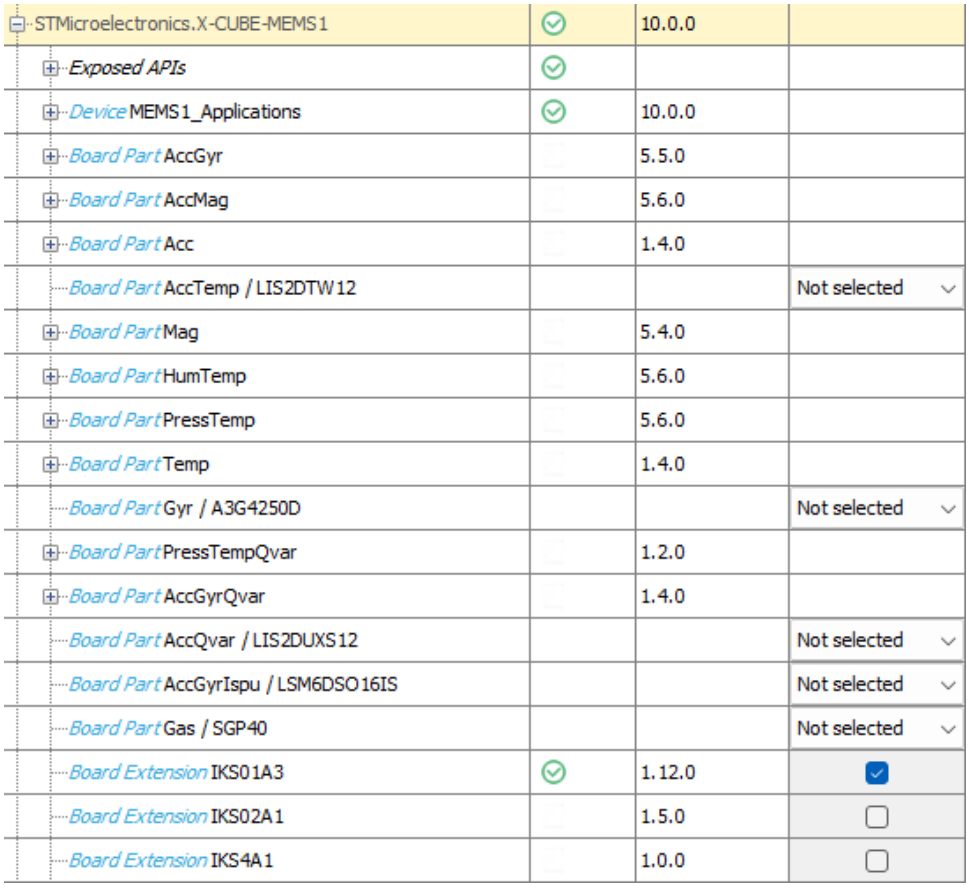
Pin	Nazwa
PC13	Button_Blue_IT
PC14	RCC_OSC32_IN
PC15	RCC_OSC32_OUT
PH0	RCC_OSC_IN
PH1	RCC_OSC_OUT
PA0	ADC1_IN5
PA5	LD2
PB10	LCD_D6

PC7	LCD_EN
PA8	LCD_D7
PA9	LCD_RS
PB4	LCD_D5
PB5	LCD_D4
PB8	I2C1_SCL
PB9	I2C1_SDA

Następujące piny posłużyły do wyświetlacza lcd ,obsługi diody ld2 oraz do obsługi MEMS. MEMS posłużył nam do realizacji tego zadania, ponieważ odpowiada za zaczytywanie temperatury, wilgoci itd. Żeby go aktywować trzeba było wejść w paczki i zainstalować MEMS a następnie zainstalować dodatkowe rozszerzenia. Po realizacji przeszliśmy do clocka i kliknęliśmy resolve clock i go poprawiliśmy. Następnie przystąpiliśmy do realizacji danego zadania.







Plik app_mems.c

Biblioteki użyte do realizacji zadania:

```
#include "app_mems.h"
#include <stdio.h>
#include <lcd.h>
#include "iks01a3_motion_sensors.h"
#include "iks01a3_env_sensors.h"
#include "math.h"
```

```
static void floatToInt float in, displayFloatToInt_t *out_value, int32_t
dec_prec
{
if(in >= 0.0f)
{
out_value->sign = 0;
 else
{
out_value->sign = 1;
in = -in;
}

in = in + (0.5f / pow(10, dec_prec));
out_value->out_int =  int32_t in;
in = in -  float (out_value->out_int);
out_value->out_dec =  int32_t trunc(in * pow 10, dec_prec));
}
```

W skrócie, funkcja ta służy do konwersji liczby zmiennoprzecinkowej na format przyjazny do wyświetlenia. Ta funkcja konwertuje liczbę na parę liczb całkowitych reprezentująca jej część całkowita i ułamkowa formę.

```
void Temp_Sensor_Handler(uint32_t Instance, uint8_t jednostka_temperatury)
{
float temperature;
displayFloatToInt_t out_value;
if (IKS01A3_ENV_SENSOR_GetValue(Instance, ENV_TEMPERATURE, &temperature))
{
snprintf(dataOut, MAX_BUF_SIZE, "T: Error");
}
else
{
floatToInt(temperature, &out_value, 2);
switch(jednostka_temperatury)
{
case 0:
snprintf(dataOut, MAX_BUF_SIZE, "T:%c%02d.%02dC",(out_value.sign) ? '-' :
'+'), (int)out_value.out_int,
 int)out_value.out_dec);
 break;
case 1:
int fahrenheit_decimal=(int)out_value.out_int*1.8+32;
uint8_t fahrenheit_float=(int)out_value.out_dec*1.8;
fahrenheit_decimal+=fahrenheit_float/100;
```

```
fahrenheit_float%=100;
snprintf(dataOut, MAX_BUF_SIZE, "T:%c%02d.%02dF",((out_value.sign) ? '-' :
'+'), fahrenheit_decimal,fahrenheit_float);
 break;
}
}
lcd_clear();
if ((int out_value.out_int>=30)
{
lcd_print(2,3,"Slonecznie");
lcd_char(2,1,3);
}
else if ((int)out_value.out_int<30 && ((int out_value.out_int>=24))
{
lcd_print(2,3,"Za chmurami");
lcd_char(2,1,4);
}
else
{
lcd_print(2,3,"Zachmurzenie");
lcd_char(2,1,5);
}

lcd_print(1,1, dataOut);
lcd_char(1,16,2);
HAL_Delay(1000);
lcd_clear();
}
```

Ta funkcja służy do czytania wartości temperatury. Jeżeli coś nie zadziała poprawnie zwracany jest błąd. Za pomocą funkcji floattoint zamieniamy wartości, żeby potem łatwiej było je wypisać. Następnym krokiem jest funkcja switch case, która posłużyła do sprawdzania warunków temperatur. W case 0 została podana funkcja do wypisy w Celsjuszach a w case 1 została podana w Fahrenheitach. Następnie sprawdzamy nasza wartość temperatur i wypisujemy Słonecznie, gdy temperatura wyniesie >=30 Stopni za chmurami, gdy <30 i >=24 lub w pozostałych przypadkach zachmurzenie i wypisujemy dla nich poszczególny znak w zależności od temperatury. Pod koniec wypisujemy temperaturę oraz znak termometru po czym czyścimy ekran.

```
void Press_Sensor_Handler(uint32_t Instance)
{
float pressure;
displayFloatToInt_t out_value;
char buf[16]="";

if (IKS01A3_ENV_SENSOR_GetValue(Instance, ENV_PRESSURE, &pressure))
{
snprintf(dataOut, MAX_BUF_SIZE, "P: Error");
}
else
```

```
{
floatToInt(pressure, &out_value, 2);
snprintf(dataOut, MAX_BUF_SIZE, "P:%dhPa", int(out_value.out_int));
if( int out_value.out_int==1013)
{
sprintf(buf,"W normie");
}
if( int out_value.out_int>1013)
{
sprintf(buf,"Za wysokie");
}
else
{
sprintf(buf,"Za niskie");
}

lcd_clear();
lcd_print(1,1, dataOut);
lcd_char(1,16,1);
lcd_print(2,1,buf);
HAL_Delay(1000);
lcd_clear();
}
```

Na początku inicjalizujemy zmienne pressure oraz buf oraz out_value. Następnie w zależności czy pojawi się błąd czy nie zmieniamy za pomocą funkcji sprintf wartość liczbową na wypisywanie na ekran. W następnej części kodu analizujemy podane wartości liczbowe i w zależności czy są one większe od 1013, mniejsze od 1013 lub równe 1013 wypisujemy odpowiednio za wysokie, za niskie, w normie. Pod koniec czyścimy ekran i wypisujemy wartości ciśnienia i wypisujemy ja na ekran. Dodatkowo wypisujemy specjalny znak, który symbolizuje wypisanie ciśnienia.

```
void Hum_Sensor_Handler(uint32_t Instance)
{
float humidity;
displayFloatToInt_t out_value;

if( IKS01A3_ENV_SENSOR_GetValue(Instance, ENV_HUMIDITY, &humidity))
{
snprintf(dataOut, MAX_BUF_SIZE, "H: Error");
}
else
{
floatToInt(humidity, &out_value, 2);
snprintf(dataOut, MAX_BUF_SIZE, "H:%d.%02d%%", (int out_value.out_int,
 int out_value.out_dec);
}
```

```
lcd_clear();
lcd_print(1,1, dataOut);
lcd_char(1,16,0);
lcd_char(2,3,'[');
int wartosc=(int)out_value.out_int/10;
for(int i=1;i<wartosc+1;i++)
{
 lcd_char(2,i+3,0);
}
lcd_char(2,14,']');
HAL_Delay(1000);
lcd_clear();
}
```

Na początku zainicjalizowaliśmy odpowiednie zmienne humidity oraz out_value a następnie jak w poprzednich fragmentach zamieniliśmy wartość liczbowa na taka która będzie możliwa do wypisania na ekran. Nasza funkcja działa w taki sposób ze na ekran wypisuje wartość wilgoci w procentach a następnie w prawym górnym rogu wypisuje specjalny znak kropelki, co symbolizuje wilgoć. Dodatkowo w zależności od stopnia wilgoci dodaliśmy specjalny pasek, który wypisuje daną ilość kropelek w zależności od procentów, czyli dla wartości 40 wypisze 4 krople, dla 80 wypisze 8 itd. Dalsza cześć projektu została zrealizowana w części main.

Struktura pliku main.c

Nagłówki i Inicjalizacja

#include "main.h": Dołączenie głównego pliku nagłówkowego.

Dołączenie plików nagłówkowych dla peryferiów ADC, RTC, GPIO oraz MEMS.

#include "adc.h", #include "rtc.h",

#include "gpio.h", #include "app_mems.h":

#include "lcd.h": Dołączenie pliku nagłówkowego dla obsługi LCD.

#include "stdbool.h": Dołączenie standardowej biblioteki dla typu bool.

Główny Kod Użytkownika

Zmienne Globalne:

bool przerwanie=false;: Zmienna dla obsługi przerwań.

uint8_t jednostka_temperatury=0;: Wybór jednostki temperatury.

uint8_t mnoznik=10;: Mnożnik czasu.

Funkcja adc_konwersja(bool wyswietl):

Obsługa ADC; wyswietl kontroluje, czy wynik ma być wyświetlony.

```
uint32_t adc_konwersja(bool wyswietl)
{
    char buf[16];
    uint32_t raw;
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,HAL_MAX_DELAY
    raw=HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
    if(wyswietl){
        sprintf(buf,"Odczytano: %lu",raw);
        lcd_print(1,1,buf);
        HAL_Delay(1000);
        lcd_clear();
    }
    return raw;
}
```

Funkcja zmiana_temp(struct przyciski p):

Zmiana jednostki temperatury przy użyciu przycisków.

```
uint8_t zmiana_temp(struct przyciski p)
{
    HAL_Delay(150);
    bool petla=true;
    uint8_t j=0;
    while(petla)
    {
        lcd_clear();
        lcd_print(1,1,"Jednostka");
        switch(j)
        {
        case 0:
            lcd_print(2,1,"Celsjusz");
            break;
        case 1:
            lcd_print(2,1,"Fahrenheit");
            break;
        }
        HAL_Delay(100);
        uint16_t raw=adc_konwersja(false);
        if (raw>= p.up-150 && raw<=p.up+150)
        {
            if(j==1)
            {
                j=0;
            }
            else
                j++;
        }
        if(raw>= p.down-150 && raw<=p.down+150)
        {
            if(j==0)
            {
                j=1;
            }
            else
                j--;
        }
        if (raw>= p.select-150 && raw<=p.select+150)
        {
            petla=false;
        }
        lcd_clear();
        }
        return j;
```

Funkcja wyswietl_date():

Wyświetla bieżącą datę i czas.

```
void wyswietl_date()
{
    RTC_TimeTypeDef time;
    RTC_DateTypeDef date;
    char buf[17];
    HAL_RTC_GetTime(&hrtc,&time,RTC_FORMAT_BIN);
    HAL_RTC_GetDate(&hrtc,&date, RTC_FORMAT_BIN);
    for(int i=0;i<2;i++)
    {
        lcd_char(1,16,6);
        if(i%2==0)
        {
            sprintf(buf,"%02d:%02d %02d.%02d.20%02d",time.Hours,time.Minutes,date.Date,date.Month,date.Year);
        }
        else
        {
            sprintf(buf,"%02d %02d %02d.%02d.20%02d",time.Hours,time.Minutes,date.Date,date.Month,date.Year);
        }
        lcd_print(1,1,"Godzina i data");
        lcd_print(2,1,buf);
        HAL_Delay(1000);
        lcd_clear();
    }

}
```

Funkcja wywolaj_funkcje(uint8_t licznik):

Wywołanie funkcji sensorycznych na podstawie licznik.

```
void wywolaj_funkcje(uint8_t licznik)
{
    switch(licznik)
    {
        case 0:
            Temp_Sensor_Handler(0,jednostka_temperatury);
            break;
        case 1:
            Hum_Sensor_Handler(0);
            break;
        case 2:
            Press_Sensor_Handler(1);
            break;
        case 3:
            wyswietl_date();
            break;

    }
}
```

Funkcja menu_acykliczne(uint8_t *licz, struct przyciski p):

Menu umożliwiające wybór wyświetlanego ekranu przez użytkownika.

```
void menu_acykliczne(uint8_t *licz,struct przyciski p){
    uint8_t licznik=*licz;
    uint32_t raw;
    bool petla=true;
    while(petla)
    {
        wywolaj_funkcje(licznik);
        raw=adc_konwersja(false);
        if(raw>=p.right && raw<=p.right+150)
        {
              if(licznik==3)
              {
                  licznik=0;
              }
              else
                  licznik++;
        }
        if(raw>=p.left-150 && raw<=p.left+150)
        {
              if(licznik==0)
              {
                  licznik=3;
              }
              else
              {
                  licznik--;
              }
        }
        if (raw>=p.select-150 && raw<=p.select+150)
        {
            petla=false;

        }
```

Funkcja zmiana_mnoznika(struct przyciski p):

Zmiana mnożnika czasu.

```
void zmiana_mnoznika(struct przyciski p)
{
    uint16_t raw;
    uint8_t j=mnoznik;
    bool edytowanie_nieukonczone=true;
    char buf[17];
    HAL_Delay(150);
    do
    {
        raw=adc_konwersja(false);
        if (raw>= p.up-150 && raw<=p.up+150)
        {
            if(j==60)
            {
                j=1;
            }
            else
                j++;
              }
        if (raw>= p.down-150 && raw<=p.down+150)
        {
            if(j==1)
            {
                j=60;
            }
            else
                j--;
        }
        if (raw>= p.select-150 && raw<=p.select+150)
        {
            edytowanie_nieukonczone=false;
            mnoznik=j;
        }
        sprintf(buf,"%d sekund",j);
        lcd_print(1,1,"Czas wysw");
        lcd_print(2,1,buf);
        HAL_Delay(150);
        lcd_clear();

    }
    while(edytowanie_nieukonczone);
```

Tworzenie Znaków na LCD (lcd_create_custom_char):

Definicje niestandardowych znaków do wyświetlania na LCD.

```
void lcd_create_custom_char(uint8_t char_addr, char* char_data) {
 char_addr &= 0x07;
 lcd_cmd(0x40 + (char_addr * 8));
 for (int i = 0; i < 8; ++i) {
 lcd_char_cp(char_data[i]);
 }}
```

Funkcje Edycji Czasu i Daty:

Funkcje do edycji poszczególnych elementów czasu i daty.

```c
void edytowanie_godziny(RTC_TimeTypeDef *time,uint8_t *migniecie,struct przyciski p)
{
    uint16_t raw;
    char buf[16];
    lcd_print(1,1,"HH:MM");
            if(*migniecie<=6)
            {
                sprintf(buf,"%02d:%02d",time->Hours,time->Minutes);
                *migniecie+=1;
            }
            else if(*migniecie<=9)
            {
                sprintf(buf,"  :%02d",time->Minutes);
                *migniecie+=1;
            }else
            {
                sprintf(buf,"  :%02d",time->Minutes);
                *migniecie=0;
            }
                lcd_print(2,1,buf);
                HAL_Delay(100);

            raw=adc_konwersja(false);
            if(raw>=p.up-150 && raw<=p.up+150)
            {
                  if(time->Hours==23)
                  {
                      time->Hours=0;
                  }
                  else
                      time->Hours++;
            }

             if(raw>=p.down-150 && raw<=p.down+150)
             {
                   if(time->Hours==0)
                   {
                       time->Hours=23;
                   }
                   else
                   {
                       time->Hours--;
                   }
             }

 }
```

```c
void edytowanie_minuty(RTC_TimeTypeDef *time,uint8_t *migniecie,struct przyciski p)
{
    uint16_t raw;
    char buf[16];
    lcd_print(1,1,"HH:MM");
            if(*migniecie<=6)
            {
                sprintf(buf,"%02d:%02d",time->Hours,time->Minutes);
                *migniecie+=1;
            }
            else if(*migniecie<=9)
            {
                sprintf(buf,"%02d:  ",time->Hours);
                *migniecie+=1;
            }else
            {
                sprintf(buf,"%02d:  ",time->Hours);
                *migniecie=0;
            }
                lcd_print(2,1,buf);
                HAL_Delay(100);

            raw=adc_konwersja(false);
            if(raw>=p.up-150 && raw<=p.up+150)
            {
                  if(time->Minutes==59)
                  {
                      time->Minutes=0;
                  }
                  else
                      time->Minutes++;
            }
```

```c
if(raw>=p.down-150 && raw<=p.down+150)
{
      if(time->Minutes==0)
      {
          time->Minutes=59;
      }
      else
      {
          time->Minutes--;
      }
}
```

```
void edytowanie_dnia(RTC_DateTypeDef *date,uint8_t *migniecie,struct przyciski p)
{
    uint16_t raw;
    char buf[16];
    lcd_print(1,1,"DD.MM.YYYY");
            if(*migniecie<=6)
            {
                sprintf(buf,"%02d.%02d.20%02d",date->Date,date->Month,date->Year);
                *migniecie+=1;
            }
            else if(*migniecie<=9)
            {
                sprintf(buf,"  .%02d.20%02d",date->Month,date->Year);
                *migniecie+=1;
            }else
            {
                sprintf(buf,"  .%02d.20%02d",date->Month,date->Year);
                *migniecie=0;
            }
                lcd_print(2,1,buf);
                HAL_Delay(100);

            raw=adc_konwersja(false);

             if(raw>=p.up-150 && raw<=p.up+150)
             {
                   if(date->Date==31)
                   {
                       date->Date=1;
                   }
                   else
                       date->Date++;
             }
             if(raw>=p.down-150 && raw<=p.down+150)
             {
                   if(date->Date==1)
                   {
                       date->Date=31;
                   }
                   else
                   {
                       date->Date--;
                   }
             }
}

void edytowanie_miesiaca(RTC_DateTypeDef *date,uint8_t *migniecie,struct przyciski p)
{
    uint16_t raw;
    char buf[16];
    lcd_print(1,1,"DD.MM.YYYY");
            if(*migniecie<=6)
            {
                sprintf(buf,"%02d.%02d.20%02d",date->Date,date->Month,date->Year);
                *migniecie+=1;
            }
            else if(*migniecie<=9)
            {
                sprintf(buf,"%02d.  .20%02d",date->Date,date->Year);
                *migniecie+=1;
            }else
            {
                sprintf(buf,"%02d.  .20%02d",date->Date,date->Year);
                *migniecie=0;
            }
                lcd_print(2,1,buf);
                HAL_Delay(100);

            raw=adc_konwersja(false);
```

```
            if(raw>=p.up-150 && raw<=p.up+150)
            {
                  if(date->Month==12)
                  {
                      date->Month=1;
                  }
                  else
                      date->Month++;
            }
            if(raw>=p.down-150 && raw<=p.down+150)
            {
                  if(date->Month==1)
                  {
                      date->Month=12;
                  }
                  else
                  {
                      date->Month--;
                  }
            }
}
```

```
void edytowanie_roku(RTC_DateTypeDef *date,uint8_t *migniecie,struct przyciski p)
{
    uint16_t raw;
    char buf[16];
    lcd_print(1,1,"DD.MM.YYYY");
            if(*migniecie<=6)
            {
                sprintf(buf,"%02d.%02d.20%02d",date->Date,date->Month,date->Year);
                *migniecie+=1;
            }
            else if(*migniecie<=9)
            {
                sprintf(buf,"%02d.%02d.    ",date->Date,date->Month);
                *migniecie+=1;
            }else
            {
                sprintf(buf,"%02d.%02d.    ",date->Date,date->Month);
                *migniecie=0;
            }
                lcd_print(2,1,buf);
                HAL_Delay(100);

            raw=adc_konwersja(false);
```

```
if(raw>=p.up-150 && raw<=p.up+150)
{
      if(date->Year==99)
      {
          date->Year=0;
      }
      else
          date->Year++;
}
if(raw>=p.down-150 && raw<=p.down+150)
{
      if(date->Year==0)
      {
          date->Year=99;
      }
      else
      {
          date->Year--;
      }
}
```

Funkcja konfiguracja_daty

Umożliwia użytkownikowi konfigurację daty i czasu.

```
void konfiguracja_daty(RTC_TimeTypeDef *time,RTC_DateTypeDef *date,struct przyciski p)
{
    lcd_clear();
    lcd_print(1,1,"Konfiguracja");
    lcd_print(2,1,"daty i godziny");
    HAL_Delay(2000);
    lcd_clear();
    uint8_t edycja=0;
    bool edytowanie_nieukonczone=true;
    uint8_t migniecie=0;
    uint16_t raw;
    do
    {
        switch(edycja)
        {
            case 0:
                edytowanie_dnia(date, &migniecie, p);
                break;
            case 1:
                edytowanie_miesiaca(date, &migniecie, p);
                break;
            case 2:
                edytowanie_roku(date, &migniecie, p);
                break;
        }
        raw=adc_konwersja(false);
        if(raw>=p.left-150 && raw<=p.left+150)
        {
            if(edycja!=0)
            {
                edycja--;
            }
        }

        if(raw>=p.right && raw<=p.right+150)
        {
            if(edycja!=2)
            {
                edycja++;
            }
        }
        if(raw>=p.select-150 && raw<=p.select+150)
        {
            edytowanie_nieukonczone=false;
        }
    }

while(edytowanie_nieukonczone);
edytowanie_nieukonczone=true;
HAL_Delay(300);
lcd_clear();
edycja=0;
do
{
    switch(edycja)
    {
        case 0:
            edytowanie_godziny(time,&migniecie,p);
            break;
        case 1:
            edytowanie_minuty(time, &migniecie,p);
            break;
    }
    raw=adc_konwersja(false);
    if(raw>=p.left-150 && raw<=p.left+150)
    {
        edycja=0;
    }

    if(raw>=p.right && raw<=p.right+150)
    {
        edycja=1;
    }
    if(raw>=p.select-150 && raw<=p.select+150)
    {
        edytowanie_nieukonczone=false;
    }
}
while(edytowanie_nieukonczone);
HAL_RTC_SetTime(&hrtc,time,RTC_FORMAT_BIN);
HAL_RTC_SetDate(&hrtc,date,RTC_FORMAT_BIN);
HAL_Delay(150);
lcd_clear();
```

Funkcja wywolaj_menu:

Obsługa menu i wybór funkcji przez użytkownika.

```
void wywolaj_menu(uint8_t *licznik,struct przyciski p)
{
    uint8_t j=0;
    while(przerwanie==true)
    {
        uint32_t raw=adc_konwersja(false);
        lcd_print(1,1,"wybierz opcje:");
        if (raw>= p.up-150 && raw<=p.up+150)
        {
            if(j==4)
            {
                j=0;
            }
            else
                j++;
              }
        if (raw>= p.down-150 && raw<=p.down+150)
        {
            if(j==0)
            {
                j=4;
            }
            else
                j--;
        }
        if (raw>= p.select-150 && raw<=p.select+150)
        {
            switch(j)
            {
                case 0:
                    przerwanie=false;
                    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin,GPIO_PIN_RESET);
                    break;
                case 1:
                    menu_acykliczne(licznik,p);
                    HAL_Delay(150);
                    break;

                case 2:
                    jednostka_temperatury=zmiana_temp(p);
                    break;
                case 3:
                    RTC_TimeTypeDef time;
                    RTC_DateTypeDef date;
                    HAL_RTC_GetTime(&hrtc,&time,RTC_FORMAT_BIN);
                    HAL_RTC_GetDate(&hrtc,&date, RTC_FORMAT_BIN);
                    konfiguracja_daty(&time, &date, p);
                    break;
                case 4:
                    zmiana_mnoznika(p);
                    break;
            }
        }
        switch(j)
        {
            case 0:
                lcd_print(2,1,"wysw cyklicznie");
                break;
            case 1:
                lcd_print(2,1,"wysw acyklicznie");
                break;
            case 2:
                lcd_print(2,1,"jednostka temp");
                break;
            case 3:
                lcd_print(2,1,"zmiana daty");
                break;
            case 4:
                lcd_print(2,1,"czas wysw");
                break;
        }
        HAL_Delay(150);
        lcd_clear();
    }
}
```

Funkcja konfiguracja_przyciskow:

Konfiguracja przycisków.

```
void konfiguracja_przyciskow(struct przyciski *p)
{
    lcd_print(1,1,"Konfiguracja");
    lcd_print(2,1,"przyciskow");
    HAL_Delay(2000);
    lcd_clear();
    lcd_print(1,1,"wcisnij SELECT");
    HAL_Delay(1000);
    p->select=wcisniety_przycisk();
    lcd_clear();
    lcd_print(1,1,"SELECT zapisany");
    HAL_Delay(1000);
    lcd_clear();
    lcd_print(1,1,"wcisnij LEFT");
    HAL_Delay(1000);
    p->left=wcisniety_przycisk();
    lcd_clear();
    lcd_print(1,1,"LEFT zapisany");
    HAL_Delay(1000);
    lcd_clear();
    lcd_print(1,1,"wcisnij RIGHT");
    HAL_Delay(1000);
    p->right=wcisniety_przycisk();
    lcd_clear();
    lcd_print(1,1,"RIGHT zapisany");
    HAL_Delay(1000);
    lcd_clear();
    lcd_print(1,1,"wcisnij UP");
    HAL_Delay(1000);
    p->up=wcisniety_przycisk();
    lcd_clear();
    lcd_print(1,1,"UP zapisany");
    HAL_Delay(1000);
    lcd_clear();
    lcd_print(1,1,"wcisnij DOWN");
    HAL_Delay(1000);
    p->down=wcisniety_przycisk();
    lcd_clear();
    lcd_print(1,1,"DOWN zapisany");
    HAL_Delay(1000);
    lcd_clear();
    lcd_print(1,1,"Skonfigurowano");
    lcd_print(2,1,"przyciski");
    HAL_Delay(2000);
    lcd_clear();
}
```

Główna Funkcja main

Inicjalizacja Znaków LCD:

Definicje graficzne dla niestandardowych znaków LCD.

```
char kropelka[] = {
  0B00010,
  0B00110,
  0B01110,
  0B01110,
  0B11110,
  0B11111,
  0B11111,
  0B01110
};
char termometr[] = {
  0B00100,
  0B01010,
  0B01010,
  0B01010,
  0B01010,
  0B10001,
  0B10001,
  0B01110
};
```

```
char cisnienie[] = {
  0B00100,
  0B00100,
  0B00100,
  0B10101,
  0B01110,
  0B00100,
  0B00000,
  0B11111
};
char chmura[] = {
  0B00000,
  0B00000,
  0B00000,
  0B00000,
  0B01110,
  0B11111,
  0B11111,
  0B01110
};
```

```
char sloncechmura[] = {
  0B00000,
  0B00100,
  0B10101,
  0B01110,
  0B01010,
  0B11111,
  0B11111,
  0B01110
};
char slonce[] = {
  0B00000,
  0B00100,
  0B10101,
  0B01110,
  0B11011,
  0B01110,
  0B10101,
  0B00100
};
```

```
char zegar[] = {
  0B00000,
  0B00100,
  0B01110,
  0B10101,
  0B10111,
  0B10001,
  0B01110,
  0B00000
};
```

Konfiguracja MCU:

Inicjalizacja peryferiów i ustawienie początkowych wartości zegara systemowego.

```
lcd_init(_LCD_4BIT,_LCD_FONT_5x8, _LCD_2LINE);
lcd_create_custom_char(0, kropelka);
lcd_create_custom_char(1, cisnienie);
lcd_create_custom_char(2, termometr);
lcd_create_custom_char(3, slonce);
lcd_create_custom_char(4, sloncechmura);
lcd_create_custom_char(5, chmura);
lcd_create_custom_char(6, zegar);
struct przyciski p;
uint8_t licznik;
//p.select=3694;
//p.left=2405;
//p.right=0;
//p.up=558;
//p.down=1510;
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
time.Hours=0;
time.Minutes=0;
date.Date=1;
date.Month=1;
date.Year=0;
```

Wywołania funkcji "Konfiguracja przycisków" oraz "daty"

```
konfiguracja_przyciskow(&p);
konfiguracja_daty(&time,&date,p);
```

Pętla Główna:

while (1): Główna pętla programu obsługująca logikę działania urządzenia.

```
while (1)
{
  /* USER CODE END WHILE */
      for(licznik=0;licznik<3;licznik++){
          for(int i=0;i<mnoznik;i++)
          {
          wywolaj_menu(&licznik,p);
          wywolaj_funkcje(licznik);
          }
      }
      licznik=3;
      for(int i=-1;i<mnoznik/2;i++)
      {

          wywolaj_menu(&licznik,p);
          wywolaj_funkcje(licznik);
      }
```