



Politechnika Wrocławskiego

Technika cyfrowa i mikroprocesorowa
Projekt - Domowa stacja pogodowa

Hanna Grzebieluch, 264209
Termin zajęć: Środa, TP, 15¹⁰ – 18¹⁵
Kod kursu: K00-41h

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI
INFORMATYCZNE SYSTEMY AUTOMATYKI

31 maja 2023

Spis treści

1	Lista funkcjonalności projektu	3
2	Lista użytych czujników/modułów	3
3	Schemat	4
4	Układ	5
5	Spełnienie założeń projektowych	6
5.1	Wyświetlacz - podłączenie	6
5.2	Zegar czasu rzeczywistego	6
5.3	Czujnik temperatury i wilgotności	7
5.4	Ustawienie wariantów wyświetlacza	7
5.4.1	Przycisk tact switch	7
5.4.2	Zmienne pomocnicze	8
5.4.3	Wariant 1	9
5.4.4	Wariant 2	10
5.4.5	Wariant 3	11
5.5	Dioda RGB	13
5.6	Dioda LED	14
5.7	Bateria	15

1 Lista funkcjonalności projektu

W ramach projektu zaliczeniowego z kursu *Technika cyfrowa i mikroprocesorowa* wykonuję domową stację pogodową, która ma za zadanie powiadomić użytkownika o wybranych, zmierzonych parametrach. Lista funkcjonalności projektu:

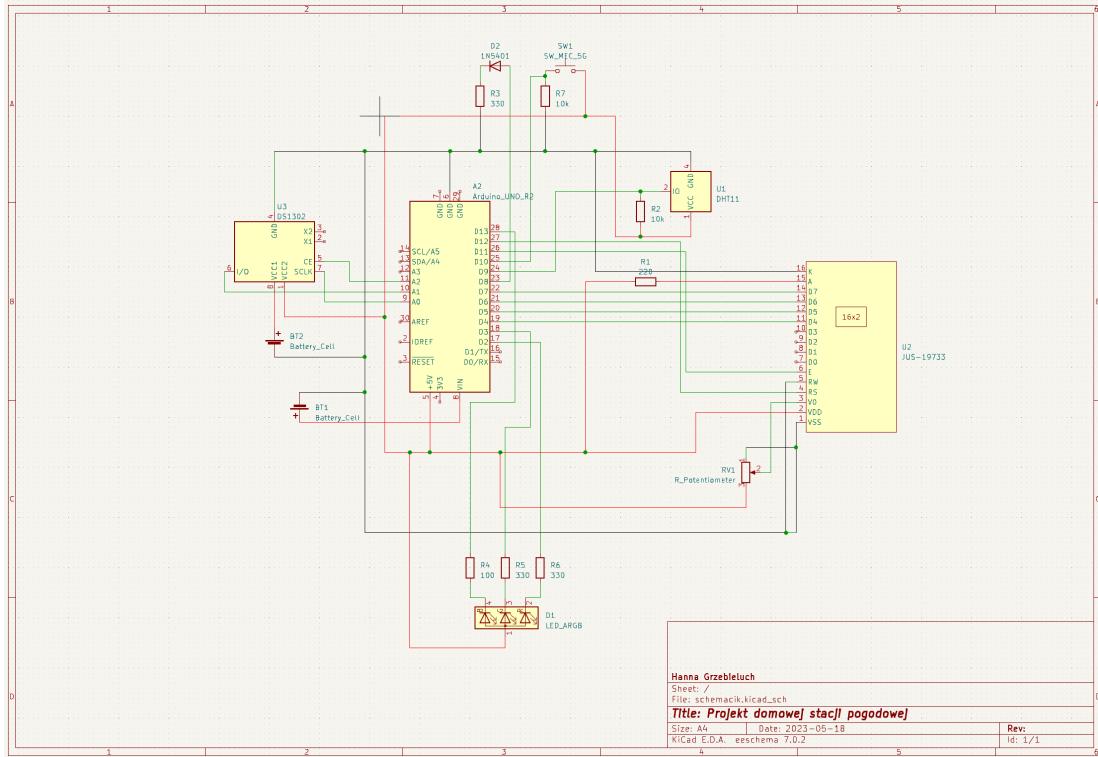
1. Wyświetlanie - w zależności od wybranego wariantu - daty, godziny, dnia tygodnia, temperatury i wilgotności,
2. Zmiana wariantu wyświetlacza poprzez naciśnięcie przycisku,
3. Sygnalizowanie wilgotności powietrza poprzez zieloną diodę LED,
4. Sygnalizowanie temperatury w sposób kolorystyczny, z użyciem programowalnej diody RGB.

2 Lista użytych czujników/modułów

1. wyświetlacz LCD 2x16 (sterownik zgodny z HD44780) oraz użyta do tego biblioteka Arduino LiquidCrystal,
2. płytka Arduino UNO z mikrokontrolerem Atmega328,
3. czujnik temperatury i wilgotności DHT11,
4. moduł czasu rzeczywistego DS1302,
5. zielona dioda LED,
6. programowalna dioda RGB,
7. przełącznik typu tact-switch,
8. bateria 9V,
9. potencjometr 10k Ω ,
10. rezystory: 100, 220, 330, 10k Ω ,
11. płytka stykowa,
12. przewody męsko-męskie

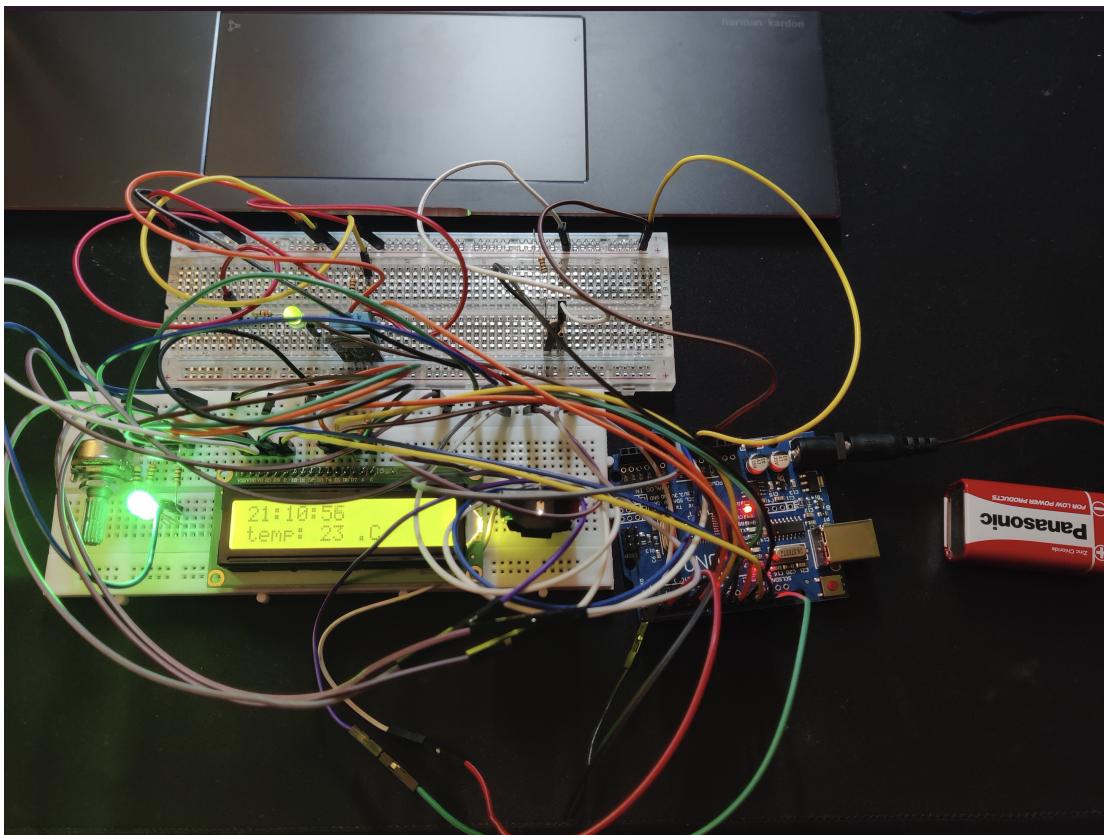
Projekt wstępnie wykonany został na płytce stykowej, dając możliwość późniejszego lutowania wszystkich części.

3 Schemat

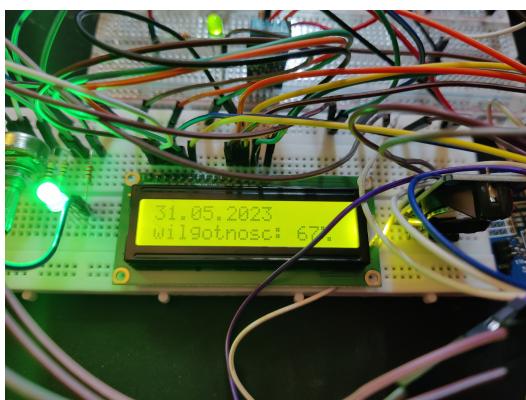


Rysunek 1: Schemat

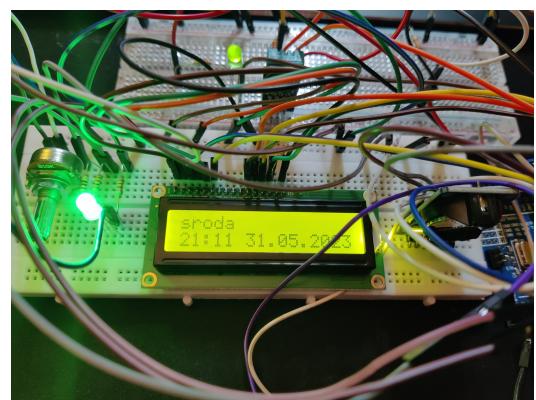
4 Układ



Rysunek 2: Zdjęcie układu - wariant pierwszy wyświetlacza



Drugi wariant wyświetlacza



Trzeci wariant wyświetlacza

5 Spełnienie założeń projektowych

5.1 Wyświetlacz - podłączenie

Aby prawidłowo podłączyć wyświetlacz LCD do płytki Arduino, wykorzystana została zainstalowana wcześniej biblioteka LiquidCrystal. Zaczęłam od przypisania wyświetlaczowi następujących pinów i utworzyłam obiekt odpowiadający wyświetlaczowi:

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Zadeklarowane nazwy pinów to odpowiednio: **rs** - wybór rejestrów (wybór rejestru instrukcji wyświetlacza (stan niski)/wybór rejestru danych (stan wysoki)), **en** - (*enable*) zapis do rejestrów, **d4-d7** - dane. Jak widać na schemacie, reszta pinów wyświetlacza podłączona jest następująco:

- VSS (masa), RW (wybór opcji odczytu (stan niski) i zapisu (stan wysoki)), K (masa podświetlenia) - masa,
- VDD (zasilanie), A (zasilanie podświetlenia) - napięcie,
- V0 (regulacja kontrastu) - potencjometr,
- piny D0-D3 nie zostały wykorzystane, ponieważ wyświetlacz pracuje w trybie 4-bitowym

Zgodnie z dokumentacją używanej biblioteki, inicjujemy wyświetlacz w następujący sposób:

```
lcd.begin(16, 2);
```

5.2 Zegar czasu rzeczywistego

Aby móc odpowiednio posługiwać się wybranym modułem RTC, korzystamy z biblioteki virtuabotixRTC [1].

```
#include <virtuabotixRTC.h>
virtuabotixRTC myRTC(14, 15, 16);
```

Załączamy wybraną bibliotekę i tworzymy nowy obiekt o nazwie myRTC, a następnie ustawiamy opowiedni czas w funkcji **setup**.

```
myRTC.setDS1302Time(00, 56, 13, 3, 10, 5, 2023);
\\seconds, minutes, hours, day of the week, day, month, year
```

Po pierwszym przesłaniu programu komentujemy powyższą linijkę, aby uniknąć ciągłej zmiany czasu przy kolejnych przesyłaniach. W funkcji **loop** natomiast wywołujemy następującą linijkę:

```
myRTC.updateTime();
```

co umożliwia modułowi zapamiętać zapisane wartości.

5.3 Czujnik temperatury i wilgotności

Do odpowiedniego posługiwania się modułem DHT11 wykorzystujemy zadeklarowaną na początku programu bibliotekę DHT11 i definiujemy numer pinu łączący się z czujnikiem. Następnie utworzymy nowy obiekt i nazywamy go *dht*.

```
#include "DHT.h"  
#define DHT11_PIN 9  
DHT dht;
```

W funkcji **setup** natomiast inicjujemy nasz obiekt:

```
dht.setup(DHT11_PIN);
```

W funkcji **loop** tworzymy 2 zmienne odpowiedzialne za przechowywanie wartości temperatury i wilgotności, które będą potem używane do wyświetlania wartości na wyświetlaczu oraz do odpowiedniego programowania diody RGB.

```
int hum = dht.getHumidity();  
int tem = dht.getTemperature();
```

5.4 Ustawienie wariantów wyświetlacza

Mając już odpowiednio podłączony zarówno wyświetlacz, jak i moduł czasu rzeczywistego, możemy przejść do dalszego programowania wyświetlacza.

5.4.1 Przycisk tact switch

Przycisk tact-switch został podłączony do płytki poprzez pin 10. Przed funkcją **setup** zadeklarowany został stały, przypisany do switcha numer pinu komendą

```
const int switchPIN = 10;
```

, a następnie w funkcji **setup** przycisk został ustawiony na tryb INPUT_PULLUP.

```
pinMode(switchPIN, INPUT_PULLUP);
```

Jak widać na schemacie, nasz przełącznik podłączony jest poprzez rezystor $10k\Omega$. Został on zastosowany, aby wejście naszego mikrokontrolera nie zbierało szumów (stan na nim byłby przypadkowy), gdy przełącznik jest rozwarty. [2] Rezystor sprawia, że gdy switch jest rozwarty występuje na wejściu stan wysoki, a gdy zostaje zwarty, stan przechodzi w niski (połączenie pull-up).

5.4.2 Zmienne pomocnicze

Aby ułatwić pracę z wariantami wyświetlacza, przed funkcją **setup** deklarujemy 3 zmienne, które pomogą Nam napisać każdą z opcji wyświetlania osobno.

```
int buttonCounter = 0; // licznik naciśnięcia przycisku
bool buttonState = LOW; // zakładamy, że przycisk jest w stanie LOW
bool lastButtonState = LOW; // analogicznie
```

W funkcji **setup** przepisujemy zmiennej Buttonstate faktyczny stan przycisku:

```
buttonState = digitalRead(switchPIN);
```

Na podstawie tej zmiennej możemy bez problemu napisać instrukcję switch-case w funkcji **loop**. Na sam koniec sprawdzamy, czy stan naszego przycisku się zmienił i odpowiednio modyfikujemy zmienną buttonCounter (jednak gdy tylko wartość zmiennej buttonCounter przekracza liczbę 2, to wracamy do zera, ponieważ mamy jedynie 3 warianty). Ponadto przy każdym naciśnięciu przycisku czyścimy wyświetlacz, aby nie pojawiały się niepożądane błędy.

```
if (buttonState != lastButtonState)
{
    delay(1000);
    if (buttonState == HIGH)
    {
        buttonCounter++;
        lcd.noDisplay();
        lcd.display();
        if (buttonCounter > 2)
        {
            buttonCounter = 0;
        }
    }
    lastButtonState = buttonState;
}
```

5.4.3 Wariant 1

Pierwszy wariant wyświetlacza z założenia powinien wyświetlać:

1. w linijce pierwszej dzień tygodnia i pełną godzinę,
2. w linijce drugiej pełną datę.

Używając funkcji dostępnych poprzez używaną bibliotekę, jesteśmy w stanie wyświetlić osobno godzinę, minuty i sekundy używając odpowiednio funkcji myRTC.hours, myRTC.minutes i myRTC.seconds, gdzie myRTC to przypisana nazwa naszego czujnika.

W pętlach zawarte zostały dodatkowe warunki, aby zamiast samych jednoliczbowych wartości minut i sekund wyświetlić przed nimi jeszcze 0 - przykładowo zamiast 12 : 5 : 9 dostajemy wartość 12 : 05 : 09.

```
case 0:  
    lcd.clear();  
    lcd.setCursor(0, 0);  
    lcd.print(myRTC.hours);  
    lcd.print(":");  
    for(int i=0;i<10;i++)  
    {  
        if(myRTC.minutes==i)  
        {  
            lcd.print("0");  
            lcd.print(myRTC.minutes);  
        }  
    }  
    if(myRTC.minutes>9)  
    {  
        lcd.print(myRTC.minutes);  
    }  
    lcd.print(":");  
    for(int i=0;i<10;i++)  
    {  
        if(myRTC.seconds==i)  
        {  
            lcd.print("0");  
            lcd.print(myRTC.seconds);  
        }  
    }  
    if(myRTC.seconds>9)
```

```

{
    lcd.print(myRTC.seconds);
}
lcd.setCursor(0, 1);
lcd.print("temp: ");
lcd.print(tem);
lcd.print(" .C");
delay(1000);
break;

```

5.4.4 Wariant 2

Drugi wariant wyświetlacza z założenia powinien wyświetlać:

1. w linijce pierwszej dzień, miesiąc i rok,
2. w linijce drugiej wilgotność.

Pełną datę uzyskujemy wykorzystując funkcje myRTC.year, myRTC.month oraz myRTC.dayofmonth.

```

case 1:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(myRTC.dayofmonth);
    lcd.print(".");
    for(int i=0;i<10;i++)
    {
        if(myRTC.month==i)
        {
            lcd.print("0");
            lcd.print(myRTC.month);
        }
    }
    if(myRTC.month>9)
    {
        lcd.print(myRTC.month);
    }
    lcd.print(".");
    lcd.print(myRTC.year);
    lcd.setCursor(0, 1);
    lcd.print("wilgotnosc: ");

```

```
lcd.print(hum);
lcd.print("%");
delay(1000);
break;
```

5.4.5 Wariant 3

Trzeci wariant wyświetlacza z założenia powinien wyświetlać:

1. w linijce pierwszej dzień tygodnia,
2. w linijce drugiej godzinę i minuty oraz pełną datę.

W przypadku case 2 został zawarty dodatkowy case, który w zależności od zmiennej myRTC.dayofweek (gdzie 1 - poniedziałek, 7- niedziela) wypisuje odpowiedni dzień tygodnia jako tekst. Pełna godzina wypisywana jest analogicznie jak w wariantie 1, natomiast pełną datę uzyskujemy wykorzystując funkcje myRTC.year, myRTC.month oraz myRTC.dayofmonth (analogicznie jak w wariantie 2).

```
case 2:
lcd.clear();
lcd.setCursor(0, 0);
switch (myRTC.dayofweek)
{
    case 1:
        lcd.print("poniedzialek");
        break;

    case 2:
        lcd.print("wtorek");
        break;

    case 3:
        lcd.print("sroda");
        break;

    case 4:
        lcd.print("czwartek");
        break;

    case 5:
        lcd.print("piatek");
```

```

break;

case 6:
lcd.print("sobota");
break;

case 7:
lcd.print("niedziela");
break;
}

lcd.setCursor(0, 1);
lcd.print(myRTC.hours);
lcd.print(":");
for(int i=0;i<10;i++)
{
  if(myRTC.minutes==i)
  {
    lcd.print("0");
    lcd.print(myRTC.minutes);
  }
}
if(myRTC.minutes>9)
{
  lcd.print(myRTC.minutes);
}
lcd.print(" ");
lcd.print(myRTC.dayofmonth);
lcd.print(".");
for(int i=1;i<10;i++)
{
  if(myRTC.month==i)
  {
    lcd.print("0");
    lcd.print(myRTC.month);
  }
}
if(myRTC.month>9)
{
  lcd.print(myRTC.month);
}

```

```

lcd.print(".");
lcd.print(myRTC.year);
delay(1000);
break;

```

5.5 Dioda RGB

Dioda RGB w układzie służy do przekazywania wartości temperatury poprzez przypisany do niej kolor. W projekcie użyta została dioda RGB ze wspólną anodą, dlatego poza 3 pinami odpowiadającymi trzem podstawowym kolorom definiujemy również i ja.

```

#define red 2
#define green 3
#define blue 13
#define COMMON_ANODE

```

Zmiana koloru diody została zaprogramowana poprzez proste instrukcje warunkowe (przykład poniżej) oraz zaimplementowanie funkcji setColor(). W funkcji **setup** natomiast deklarujemy wszystkie piny jako wyjście:

```

pinMode(red, OUTPUT);
pinMode(green, OUTPUT);
pinMode(blue, OUTPUT);

```

W funkcji **loop**:

```

if(temp >= 15 && temp < 20)
{
    setColor(0,153,153);
}
else if(temp >= 20 && temp < 25)
{
    setColor(0,204,0);
}
else if...

```

Kolory diody RGB w zależności od temperatury:

1. dla temperatury mniejszej niż 15 stopni - fioletowy (255,0,255),
2. dla temperatury większej bądź równej 15 i jednocześnie mniejszej niż 20 - niebieski (0,153,153),

3. dla temperatury większej bądź równej 20 i jednocześnie mniejszej niż 25 - zielony (0,204,0),
4. dla temperatury większej lub równej 25 - pomarańczowy () .

Funkcja **setColor()**:

```
void setColor(int r, int g, int b)
{
    #ifdef COMMON_ANODE
        r = 255 - r;
        g = 255 - g;
        b = 255 - b;
    #endif
    analogWrite(red, r);
    analogWrite(green, g);
    analogWrite(blue, b);
}
```

Przy implementowaniu funkcji zmiany koloru, szczególną uwagę należało poświęcić wspólnej anodzie. Przez fakt, że używamy diody ze wspólną anodą, musimy odpowiednio zmienić wartości ustalone analogowo, aby każdy kolor był poprawnie odjęty od wartości 255. [3]

5.6 Dioda LED

Dioda LED zamontowana według schematu informuje o tym, czy wilgotność powietrza przekracza 60%. Jeśli warunek ten zostanie spełniony, dioda zaczyna świecić na zielono.

```
const int greenLED = 8;
```

W funkcji **setup**:

```
pinMode(greenLED, OUTPUT);
```

W funkcji **loop**:

```
if(hum>=60)
{
    digitalWrite(greenLED, HIGH);
}
```

5.7 Bateria

Układ zasilany jest zewnętrze poprzez baterię o mocy 9V, co pozwala na bezprzewodowe działanie stacji pogodowej.

Literatura

- [1] “Arduinortlibrary.” <https://github.com/chrisfryer78/ArduinoRTCLibrary>. data dostępu: 17.05.2023.
- [2] “Microswitch jako proste czujniki przeszkód.” <https://forbot.pl/blog/microswitch-proste-czujniki-przeszkod-id1870>. data dostępu: 17.05.2023.
- [3] “Arduino lesson 3. rgb leds.” <https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/arduino-sketch>. data dostępu: 18.05.2023.