Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji
Kierunek: Informatyczne Systemy Automatyki
Techniki cyfrowa i mikroprocesorowa
Laboratorium

Projekt zegara z budzikiem

Autor: Zuzanna Gorczyca
Nr indeksu: 264182
Semestr: 4
Grupa: Środa 10^{10} - 13^{00} TP, K00-41f
Data zajęć: 07 Czerwca 2023
Prowadzący: Mgr inż. Wojciech Tarnawski

Czerwiec 2023

Spis treści

1 Wstęp

Projekt zakładał zbudowanie zegara z funkcją budzika używając mikro-kontrolera ATMEGA328P. Układ zbudowano wykorzystując płytkę Arduino UNO oraz wyświetlacz LCD do wyświetlania zegara, daty oraz ustawień budzika i minutnika.

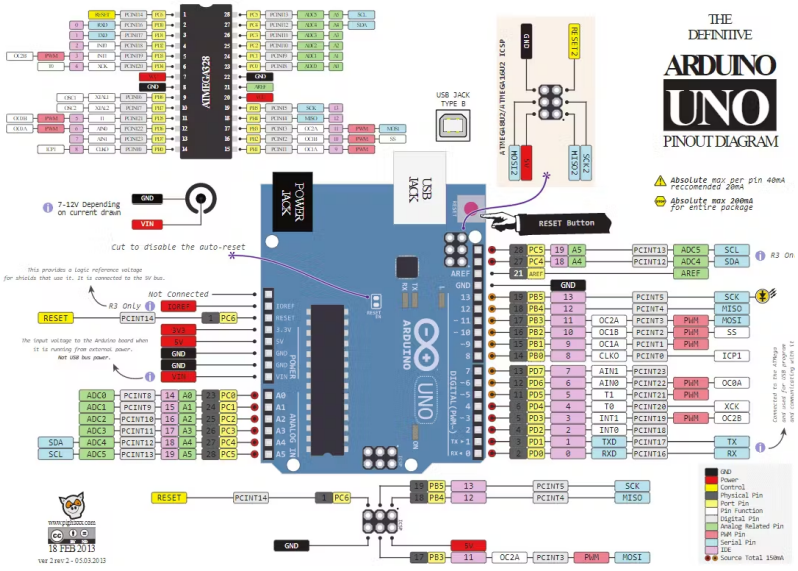
2 Komponenty

Do zbudowania projektu użyto poniższych komponentów:

- Płytka Arduino UNO z mikroprocesorem ATMEGA328P
- Wyświetlacz LCD z modułem I2C LCM1602 ułatwiającym komunikację między wyświetlaczem, a mikrokontrolerem
- Przyciski do ustawiania godziny daty oraz alarmu
- Moduł z buzzerem Iduino ST1143 - służący jako dzwonek budzika
- Kable
- Płytka stykowa

2.1 Ardunio UNO

Jest to popularna płytka z mikrokontrolerem ATMEGA328P, ułatwiająca programowanie mikrokontrolerów.

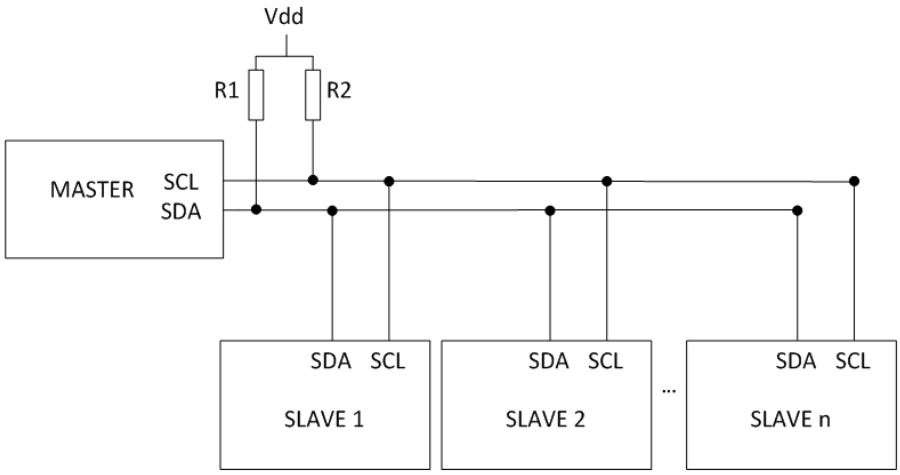


Rysunek 1: Opis pinów na płytce Arduino UNO

2.2 Interface I2C

I2C to nazwa dwuprzewodowego interfejsu, służącego do przesyłania danych pomiędzy dwoma lub większą liczbą układów cyfrowych. Interface umożliwia komunikację pomiędzy wieloma urządzeniami, które nie wymagają szybkiej transmisji danych. Do komunikacji użyte zostały dwa przewody sygnałowe, wspólna masa oraz zasilanie. Przewody sygnałowe:

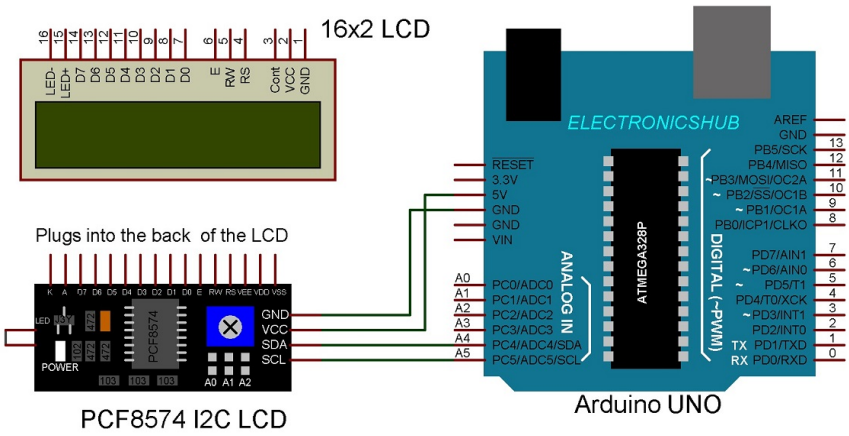
- **SDA** - linia danych służąca do przesyłania danych do innych urządzeń
- **SCL** - linia zegarowa służąca do synchronizacji czasu odbierania oraz nadawania informacji przez inne urządzenia



Rysunek 2: Przykładowe połączenie urządzeń interfacem I2C

2.3 Konwerter I2C

Do wyświetlacza LCD został dołączony moduł I2C LCM1602 z konwerterem opartym na układzie PCF8574 i potencjometrem do regulacji kontrastu. Moduł ten znacznie ułatwia używanie wyświetlacza i pozwala go podłączyć do modułu Arduino UNO tylko 4 przewodami, masa (GND), zasilanie (VCC), magistrala danych I2C (SDA), magistrala zegara I2C (SCL).
SDA podłącza się do wejścia analogowego A4, pinu PC4, natomiast SCL do wejścia analogowego A5, pinu PC5. Pozostałe piny konwertera wlutowane zostały do pinów wyświetlacza LCD.



Rysunek 3: Układ PCF8574, wyświetlacz LCD i moduł Arduino UNO

Poniżej przedstawiono fragment dokumentacji PCF8574 opisujący piny układu.

SYMBOL	PIN		DESCRIPTION
	DIP16; SO16	SSOP20	
A0	1	6	address input 0
A1	2	7	address input 1
A2	3	9	address input 2
P0	4	10	quasi-bidirectional I/O 0
P1	5	11	quasi-bidirectional I/O 1
P2	6	12	quasi-bidirectional I/O 2
P3	7	14	quasi-bidirectional I/O 3
V_{SS}	8	15	supply ground
P4	9	16	quasi-bidirectional I/O 4
P5	10	17	quasi-bidirectional I/O 5
P6	11	19	quasi-bidirectional I/O 6
P7	12	20	quasi-bidirectional I/O 7
$\overline{INT}$	13	1	interrupt output (active LOW)
SCL	14	2	serial clock line
SDA	15	4	serial data line
V_{DD}	16	5	supply voltage
n.c.	–	3	not connected
n.c.	–	8	not connected
n.c.	–	13	not connected
n.c.	–	18	not connected

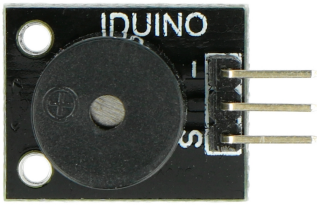
Rysunek 4: Fragment dokumentacji PCF8574 przedstawiający piny

2.4 Moduł Iduino ST1143, czyli buzzer

Moduł buzzera z generatorem, umożliwia tworzenie sygnałów dźwiękowych przy pomocy stałego sygnału napięciowego. Specyfikacja:

- Napięcie zasilania: 5 V
- Pobór prądu: ok. 30 mA
- Sygnał generowany po podaniu stanu wysokiego

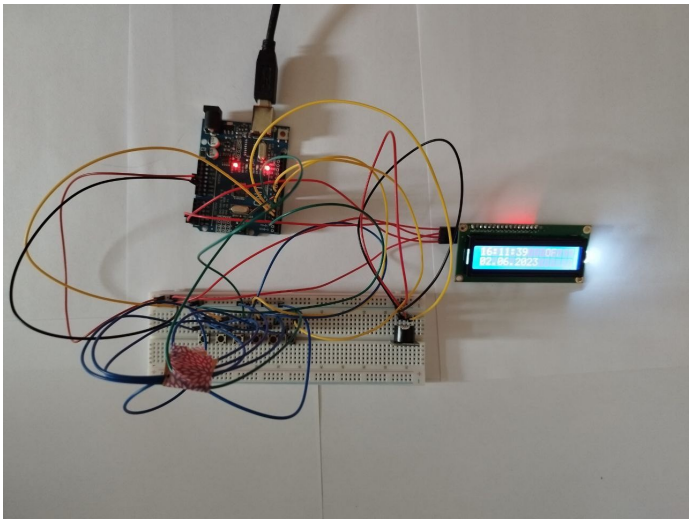
Moduł posiada trzy piny. Jeden podłącza się do masy (GND), drugi do zasilania 5V (VCC), a ostatni do pinu na module Arduino UNO. Buzzer włącza się kiedy na trzecim pinie jest podany stan wysoki.



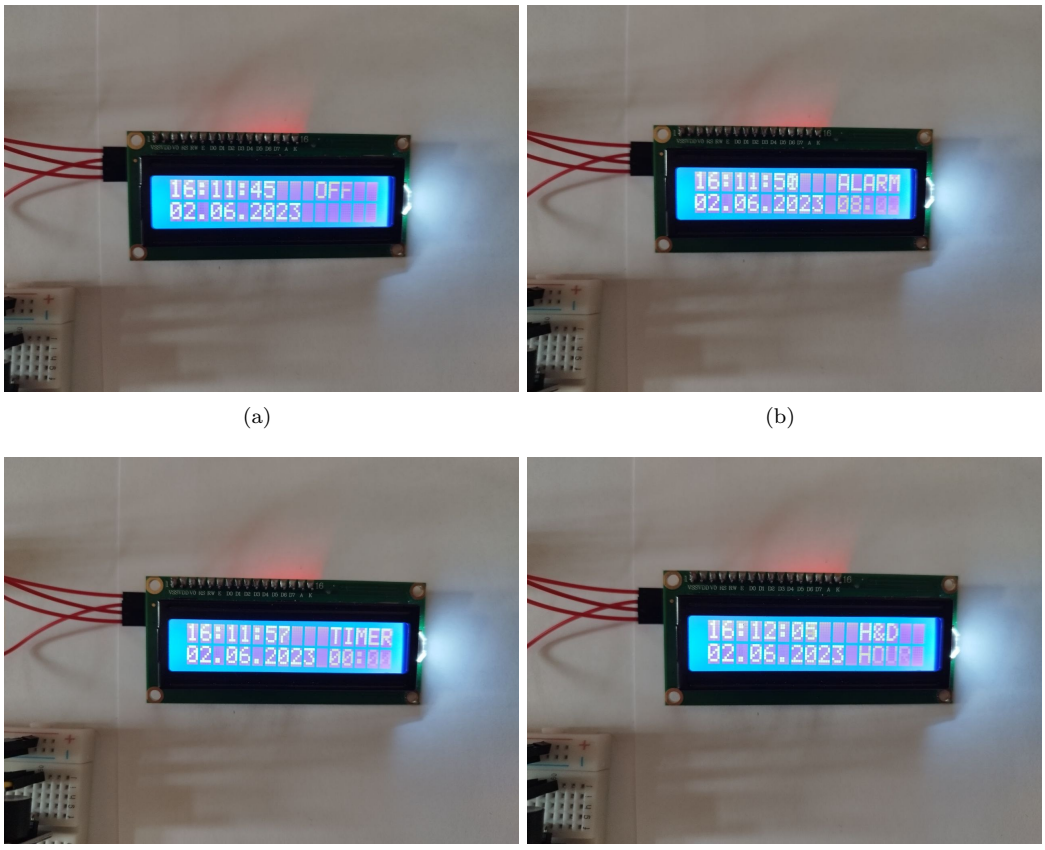
Rysunek 5: Zdjęcie modułu z buzzerem

3 Układ z mikro-kontrolerem

Układ zbudowano używając płytki stykowej. Poniżej zamieszczone są zdjęcia gotowego układu.



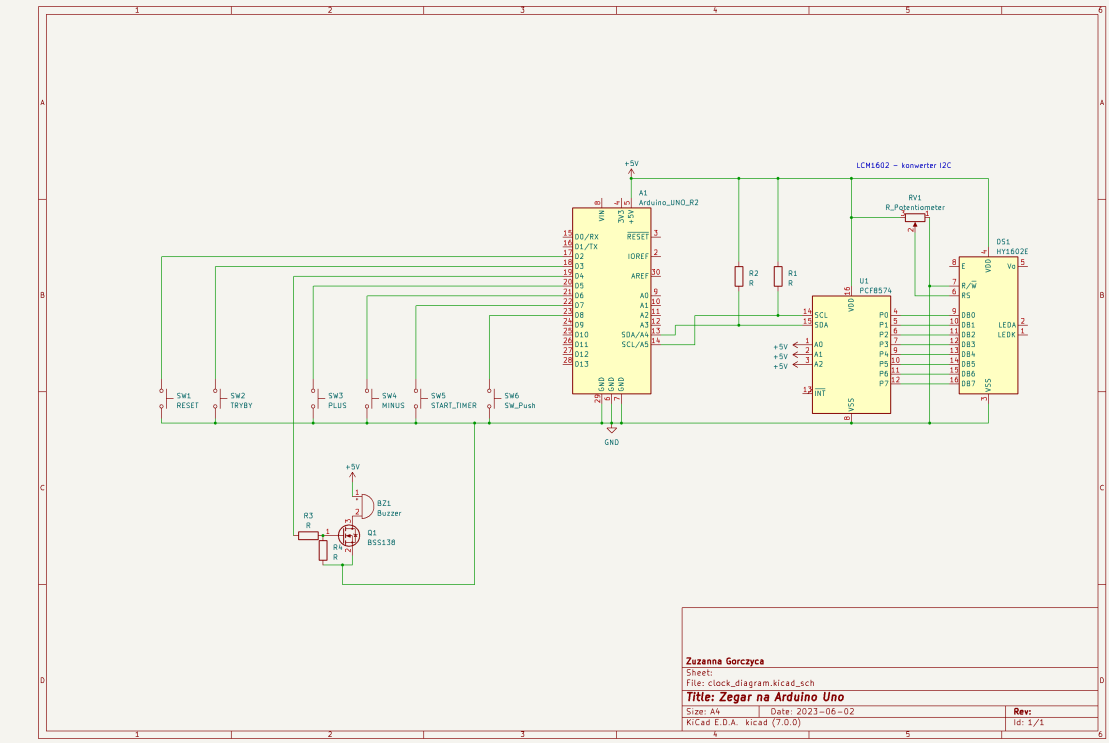
Rysunek 6: Zdjęcie gotowego układu



(a) (b)

(c) (d)

Rysunek 7: Tryby zegara: (a) wszystko wyłączone zwykły zegar, (b) włączony alarm, (c) włączony minutnik, (d) włączona tryb do zmiany godziny i daty.



Rysunek 8: Schemat układu utworzony w programie KiCad

4 Implementacja kodu w języku AVR C

Program do projektu zaimplementowano w języku AVR C. Do obsługi wyświetlacza LCD użyto biblioteki `LcdI2c`. Do ustawiania czasu użyto wewnętrznego przerwania mikrokontrolera. Kod na mikro-kontroler pisany w języku C musi mieć określoną strukturę. Na początku należy zaimplementować biblioteki potrzebne do pisania na mikro-kontrolerze. Potem jest główna funkcje programu, czyli `main()`, w której jest wywoływana pętla `while(1)`, czyli nieskończona pętla, w której powinny się znajdować wszystkie czynności, które mają być wykonywane przez mikro-kontroler, poza tymi które ustawia się tylko na samym początku.

4.1 Deklaracja zmiennych

Zadeklarowano sześć wyjść na przyciski, jedno wyjście na buzzer oraz jedno na ekran LCD.

```
1       //WEJSCIA
2       cbi(DDRD,PD2);  // przycisk przerwanie INT0 RESET
3       cbi(DDRD,PD3);  // przycisk przerwanie INT1 TRYBY
4       cbi(DDRD,PD5);  // przycisk ++
5       cbi(DDRD,PD6);  // przycisk --
6       cbi(DDRD,PD7);  // przycisk startuje minutnik
7       cbi(DDRB,PB0);  // ustawianie godziny i daty
8       // WEJSCIA PULL-UP
9       sbi(PORTD,PD3);
10      sbi(PORTD,PD2);
11      sbi(PORTD,PD5);
12      sbi(PORTD,PD6);
13      sbi(PORTD,PD7);
14      sbi(PORTB,PB0);
15      //WYJSCIA
16      sbi(DDRD, PD4); // buzzer
17      cbi(PORTD,PD4); // ustawia stan niski
18      LCD_I2C lcd = LCD_I2C(0x27); // wyswietlacz
```

4.2 Pętla while(1)

Najważniejsza pętla w kodzie. Znajdują się w niej wszystkie funkcje programu, które ma wykonywać mikrokontroler. Jeżeli funkcja w pętli spełnia odpowiednie warunki takie jak wciśnięty przycisk, odpowiednia wartość zmiennej itp. to jest wywoływana. W innym wypadku funkcja jest pomijana i czeka na odpowiednie warunki.

```
1       while(1){
2           lcd.clear();     //wyczysc ekran
3           set();           // zmienia tryby
4           displayCLOCKandDATE(lcd);    // wyswietla zegar
5           displayALARMandTIMER(lcd);  // wyswietla alarm i timer
6           startTIMER(); //jesli wybrane wystartuj minutnik
7           setBuzzer();  // jesli spelnione warunki wlacz buzzer
8       }
```

4.3 Przerwania

Przerwanie jest zdarzeniem, które przerywa wykonywanie programu głównego i uruchamia specjalną funkcję obsługi przerwania. Gdy funkcja obsłuży przerwanie, następuje powrót do przerwanego programu i wznowienie jego wykonywania od miejsca, w którym został przerwany. Zaletą tego rozwiązania jest to, ze mikrokontroler nie musi programowo sprawdzać cały czas, czy zaszło zdarzenie związane z przerwaniem. Dlatego też doskonale nadają się do zaprogramowania zegara. Przerwanie wykonywane co sekundę zapewnia, że zegar nie będzie mieć opóźnień, a czas policzy się nawet,

jeśli program główny będzie wtedy wykonywać inną czynność. Aby uruchomić przerwanie w funkcji `main()` należy zainicjować funkcje `sei()`, która powoduje uruchomienie przerwań na mikrokontrolerze.
Istnieją dwa rodzaje przerwań: zewnętrzne oraz wewnętrzne. Przerwania zewnętrzne są generowane przez urządzenie zewnętrzne, którym steruje mikrokontroler. Natomiast przerwania wewnętrzne pochodzą od wewnętrznych składników mikrokontrolera.
W programie użyto trzech przerwań. Jednego wewnętrznego wywołującego się co sekundę do ustawiania czasu i dwóch zewnętrznych do obsługi funkcji budzika i minutnika.

4.3.1 ISR(TIMER1_COMPA_vect)

Przerwanie wewnętrzne TIMER1 COMPA, wywołuje się co sekundę i ustawia zegar oraz jeśli funkcja minutnik jest włączona odpowiada też za liczenie czasu minutnika.

4.3.2 ISR(INT0_vect)

Przerwanie zewnętrzne INT0 na pinie PD2. Wywołuje się po naciśnięciu przycisku podłączonego do pinu PD2. W programie służy do wyłączania buzzera oraz do resetowania funkcji budzika oraz minutnika.

```
1       ISR(INT0_vect){
2           cbi(PORTD,PD4);
3           ALARMSTSTUS = 0;
4           TIMERSTSTUS = 0;
5           TIMERSTART = 0;
6           choice = 0;
7           next = 0;
8           ALARMHOU2 = 8;
9           ALARMHOU1 = 0;
10          _delay_ms(200);
11      }
```

4.3.3 ISR(INT1_vect)

Przerwanie zewnętrzne INT1 na pinie PD3. Wywołuje się po naciśnięciu przycisku podłączonego do pinu PD3. W programie służy do przełączania się pomiędzy kolejnymi trybami budzika i minutnika.

```
1       ISR(INT1_vect){
2           _delay_ms(30);
3           if(bit_is_clear(PIND,PD3)){
4               _delay_ms(20);
5               choice++;
6           }
7           if (choice >5){
8               choice = 0;
9           }
10      }
```

4.4 Funkcje

W programie zaimplementowano funkcję do obsługi wyświetlacza, ustawiania godziny i daty oraz do obsługi trybów budzika i minutnika. Dzięki temu można ograniczyć ilość kodu w nieskończonej pętli `while(1)`, co powoduje, że kod jet bardziej czytelny.

4.4.1 Funkcja wyświetlania

Zaimplementowano dwie funkcje wyświetlania: `void displayCLOCKandDATE(LCD_I2C lcd)` oraz `void displayALARMandTIMER(LCD_I2C lcd)`.

Wyświetlanie aktualnej godziny oraz daty

```
1       void displayCLOCKandDATE(LCD_I2C lcd){
2           char _clock[20];
3           char _date[20];
4           lcd.goTo(0,0);
5           sprintf(_clock,"%02d:%02d:%02d", HOU, MIN, SEC);
6           lcd.writeText(_clock);
7           lcd.goTo(0,1);
8           sprintf(_date,"%02d.%02d.%04d", DAY, MONTH, YEAR);
9           lcd.writeText(_date);
10          if(choice == 5){
11          sprintf(_status, "H&D");
12          switch(next){
13            /*wyswitla nazwe pola do zmiany*/
14          }
15          lcd.goTo(11,0);
16          lcd.writeText(_status);
17          lcd.goTo(11,1);
18          lcd.writeText(_position);
19        }
20      }
```

Wyświetlanie aktualnych danych budzika lub minutnika.

```
1       void displayALARMandTIMER(LCD_I2C lcd){
2           char _alarm[20];
3           char _ststus[20];
4           if(choice != 5){
5               if(ALARMSTSTUS == 0 && TIMERSTSTUS == 0 ){
6                   sprintf(_ststus, "OFF");
7                   sprintf(_alarm, "-----");
8               }
9               else if(ALARMSTSTUS == 1){
10                  sprintf(_ststus, "ALARM");
11                  sprintf(_alarm, "%02d:%02d", ALARMHOU, ALARMMIN);
12              }
13              else if(TIMERSTART == 1 || TIMERSTSTUS == 1){
14                  sprintf(_ststus, "TIMER");
15                  sprintf(_alarm, "%02d:%02d", TIMERMIN, TIMERSEC);
16              }
17              lcd.goTo(11,0);
18              lcd.writeText(_ststus);
19              lcd.goTo(11,1);
20              lcd.writeText(_alarm);
21          }
22      }
```

4.4.2 Włączanie buzzera

Funkcja do włączania buzzera `void setBuzzer()`. Buzzer działa w częstotliwości 2Hz z wypełnieniem 50%.

```
1       void setBuzzer(){
2           if(ALARMSTSTUS && HOU == ALARMHOU && MIN == ALARMMIN){
3             tbi(PORTD,PD4);
4             _delay_ms(1000);
5             tbi(PORTD,PD4);
6             _delay_ms(1000);
7           }
8           if(TIMERSTART && TIMERMIN == 0 && TIMERSEC == 0 ){
9             tbi(PORTD,PD4);
10            _delay_ms(1000);
11            tbi(PORTD,PD4);
12            _delay_ms(1000);
13          }
14      }
```

4.4.3 Start minutnika

Funkcja `void startTIMER()` uruchamia odliczanie czasu minutnika. Jeśli przycisk na pinie PD7 zostanie wciśnięty to zmienna `TIMERSTART` zmienia wartość na 1 co powoduje start minutnika, którego odliczanie odbywa się w przerwaniu `ISR(TIMER1 COMPA vect)`.

```
1       void startTIMER(){
2           if(bit_is_clear(PIND,PD7)){
3               TIMERSTART = 1;
4               _delay_ms(200);
5           }
6       }
```

4.4.4 Ustawianie daty i godziny

Ta funkcja załącza się tylko jeśli jesteśmy w trybie zmiany godziny, czyli zmienna `choice = 5`. Dla kolejnych wartości zmiennej `next` możemy ustawiać wartości godziny, minut, dnia, miesiąca i roku.

```
1       void setCLOCKandDATE(){
2           if(bit_is_clear(PINB,PB0)){
3           _delay_ms(30);
4               if(bit_is_clear(PINB,PD0)){
5               _delay_ms(20);
6               next = (next+1)%6;
7               }
8           }
9           /*ustawianie kolejnych wartosci*/
10      }
```

4.4.5 Ustawianie budzika i minutnika

Do ustawiania budzika oraz minutnika używana jest funkcja `void setALARM()`. Funkcja wykorzystuje przerwanie INT1, do ustawiania wartości zmiennej `choice`. Jeśli zmiana jest równa 0 to budzik oraz minutnik są wyłączone. Dla wartości 1 oraz 2 włączamy tryb alarmu, dla wartości 1 można ustawić godzinę, o której ma zadzwonić budzik, a przy wartości 2 ustawiamy minuty. Dla wartości 3 włącza się tryb minutnika i wyłącza się tryb budzika. Można wtedy ustawić na minutniku liczbę minut. Dla wartości 4 możliwe jest wystartowanie minutnika.

```
1       void setALARM(){
2          //======USTAWIA MINUTY BUDZIKA======
3          if(choice == 1){
4              if(bit_is_clear(PIND,PD5)){
5                  ALARMMIN++;
6                  _delay_ms(1000);
7                  if(ALARMMIN > 59){
8                      ALARMMIN = 0;} }
9              ALARMSTSTUS = 1;
10             TIMERSTSTUS = 0; }
11         //======USTAWIA GODZINY BUDZIKA======
12         else if(choice == 2){
13             if(bit_is_clear(PIND,PD5)){
14                 ALARMHOU++;
15                 _delay_ms(500);
16                 if(ALARMHOU >23){
17                     ALARMHOU = 0; } }
18             ALARMSTSTUS = 1;
19             TIMERSTSTUS = 0; }
20         //======USTAWIA MINUTNIK======
21         else if(choice == 3){
22             if(bit_is_clear(PIND,PD5)){
23                 TIMERMIN++;
24                 _delay_ms(1000);
25                 if(TIMERMIN > 99){
26                     TIMERMIN = 0;} }
27             ALARMSTSTUS = 0;
28             TIMERSTSTUS = 1;
29             TIMERSTART = 0; }
30         //======MOZLIWY START MINUTNIKA======
31         else if(choice == 4){
32             ALARMSTSTUS = 0;
33             TIMERSTSTUS = 1;}
34         else{
35             ALARMSTSTUS = 0;
36             TIMERSTSTUS = 0;
37             TIMERSTART = 0;} }
```

4.4.6 Inicjacja przerwań

Funkcja `void interupt()` inicjuje przerwania INT0 oraz INT1.

```
1       void interupt() {
2           sbi(EICRA, ISC01);
3           sbi(EICRA, ISC11);
4           sbi(EIMSK,INT0);
5           sbi(EIMSK, INT1);
6       }
```

4.4.7 Inicjacja wewnętrznego zegara

Funkcja `void clockInit()` inicjuje zegar wewnętrzny.

```
1       void clockInit() {
2           sbi(TCCR1B, WGM12);
3           sbi(TCCR1B, CS12);
4           OCR1A = 62500;
5           sbi(TIMSK1, OCIE1A);
6       }
```

4.5 Drgania styków

Jednym z trudniejszych problemów, napotkanych w trakcje tworzenia projektu, były drgania styków, które występują po naciśnięciu przycisku. Kiedy przycisk zostaje naciśnięty powoduje to odkształcenie się styków, co prowadzi do kilku drgań zanim styki się ze sobą zewrą. Ten proces zazwyczaj trwa około 20ms, w tym czasie może się zdarzyć, że mikrokontroler zamiast jednego naciśnięcia przycisków odczyta kilka naciśnięć co może powodować błędy w działaniu urządzenia. Poniższy wykres obrazuje zachowanie się sygnału po wciśnięciu przycisku.

Rysunek 9: Wykres obrazujący drgania styków

Jest kilka możliwych rozwiązań tego problemu, zarówno sprzętowych jak i przy pomocy kodu. W tym projekcie wykorzystano rozwiązanie przy pomocy kodu. Rozwiązanie można zobaczyć na przykład w funkcji przerwania INT1.

```
1       ISR(INT1_vect){
2           _delay_ms(30);
3           if(bit_is_clear(PIND,PD3)){
4               _delay_ms(20);
5               choice++;
6           }
7           if (choice >5){
8               choice = 0;
9           }
10      }
```

Na początku program czeka 30ms i jeśli po tych 30ms przycisk faktycznie będzie wciśnięty (warunek `if`) to program czeka jeszcze 20ms i zmienia wartość zmiennej `choice`.

5 Instrukcja obsługi

Układ działa tylko podłączony do zasilania, nie ma modułu zasilającego więc nie jest przenośny. Do obsługi zegara służy sześć przycisków. Każdy z nich ma inną funkcję.

1. Przycisk 1 (PD2) - wywołuje przerwanie INT0, odpowiada za wyjście z trybu budzika, minutnika oraz ustawiania czasu i wyzerowania jego wartości. Wyłącza również buzzer, jeśli ten jest załączony.

2. Przycisk 2 (PD3) - wywołuje przerwanie INT1, służy do zmiany trybów.
   Jedno naciśnięcie uruchamia tryb budzika i pozwala ustawić minuty.
   Drugie naciśnięcie pozwala edytować godzinę alarmu.
   Trzecie naciśnięcie uruchamia tryb minutnika. Pozwala ustawić minuty, które mają być odliczone.
   Czwarte naciśnięcie pozwala na uruchomienie minutnika przez naciśnięcie przycisku 5.
   Piąte naciśnięcie uruchamia tryb edycji godziny i daty.

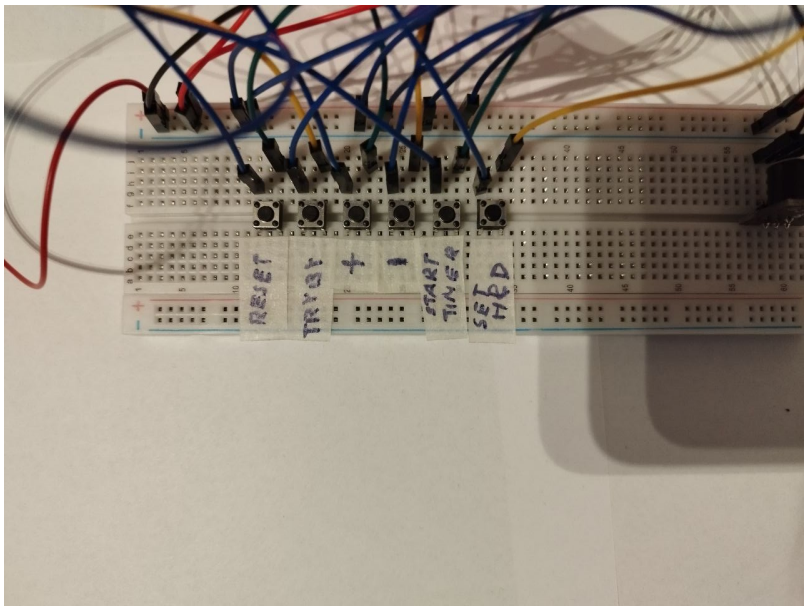
3. Przycisk 3 (PD5) - pozwala zwiększyć o jeden wartość, która jest aktualnie edytowana (wybrana wcześniej przyciskiem 2 lub 6).

4. Przycisk 4 (PD6) - pozwala zmniejszenie o jeden wartość, która jest aktualnie edytowana (wybrana wcześniej przyciskiem 2 lub 6).

5. Przycisk 5 (PD7) - jego wciśnięcie uruchamia minutnik, ale tylko jeśli wcześniej przycisk 2 został wciśnięty 4 razy tzn. był w trybie startu minutnika.

6. Przycisk 6 (PB0) - jeśli przycisk 2 został wciśnięty 5 razy i zegar jest w trybie edycji daty i godziny przycisk 6 służy do przełączania się miedzy kolejnymi pozycjami.
   Jedno wciśnięcie pozwala edytować godzinę (sygnalizowane pojawieniem się w prawym dolnym rogu napisu ”HOUR”.

Drugie wciśnięcie pozwala edytować minuty (sygnalizowane pojawieniem się w prawym dolnym rogu napisu "MIN".
Trzecie wciśnięcie pozwala edytować dni (sygnalizowane pojawieniem się w prawym dolnym rogu napisu "DAY".
Czwarte wciśnięcie pozwala edytować miesiąc (sygnalizowane pojawieniem się w prawym dolnym rogu napisu "MONTH".
Piąte wciśnięcie pozwala edytować rok (sygnalizowane pojawieniem się w prawym dolnym rogu napisu "YEAR".



Rysunek 10: Przyciski do obsługi zegara, przyciski są liczone od lewej

6 Podsumowanie

Projekt napisany został w języku AVR C. Zostały zrealizowane wszystkie założenia projektu. Projekt posiada funkcje budzika oraz minutnika oraz możliwość ustawiania czasu przez użytkownika. Jedyna zmiana w stosunku do oryginalnych założeń to generowanie czasu zegara przy pomocy przerwania TIMER1, zamiast używając modułu DS1302 RTC. Zmiany dokonano gdyż uznano, że opcja z przerwaniem będzie ciekawsza i lepiej zaprezentuje możliwości mikrokontrolera.

7 Bibliografia

1. https://eduinf.waw.pl/inf/prg/009_kurs_avr_old/0027.php
2. http://w-tarnawski.pl/materialy-do-zajec/podstawy-techniki-mikroprocesorowej-i/podstawy-techniki-mikroprocesorowej-semestr-i/laboratorium-1-1/
3. https://gradhelp96.blogspot.com/2018/11/arduino-avr-alarm-clock-using-atmel.html
4. https://serwis.avt.pl/files/kurs_c/22_KursAVR_cz01.pdf
5. https://www.youtube.com/watch?v=W6iGhPm2eUE&ab_channel=ArduinoPL
6. Forbot

Spis rysunków