

EMILIA STARCZYK
249005

**PROJEKT STACJI POGODOWEJ OPARTEJ
NA ATMEGA32**

Podstawy Techniki Mikroprocesorowej 2
DR INŻ. KRZYSZTOF HALAWA
26 stycznia 2021

Spis treści

1 Założenia projektowe	2
2 Przebieg pracy nad projektem	2
3 Elementy projektu	2
3.1 Pisanie kodu w języku C oraz testowanie urządzenia	2
3.2 Wykonanie schematu połączeń w programie Autodesk Eagle	5
3.3 Wykonanie obudowy stacji pogodowej	7
4 Możliwości rozwojowe projektu	8
5 Efekt końcowy pracy	8
6 Zdjęcia projektu prezentujące ostateczny wygląd	11

1. Założenia projektowe

Stacja pogodowa składa się z czujników temperatury, wilgotności, czujnika barometrycznego - z możliwością wyznaczenia wysokości, klawiatury matrycowej pozwalającej na wybór punktów z menu, wyświetlacza LCD ze sterowaniem HD44780 oraz zegara RTC potrzebnego do wyświetlania daty. W projekcie należało wykorzystać komunikację I^2C oraz SPI, oprogramować obsługę klawiatury oraz napisać bibliotekę obsługującą wyświetlacz LCD.

2. Przebieg pracy nad projektem

Prace nad projektem przebiegały w kilku etapach:

- Wykonanie zestawienia elementów potrzebnych do realizacji
- Pisanie kodu w języku C oraz testowanie urządzenia
- Wykonanie schematu ogólnego w programie Autodesk Eagle
- Wykonanie wewnętrznego urządzenia - lutowanie, wykonanie połączeń
- Wykonanie obudowy stacji pogodowej

Każdy z etapów wymagał odpowiedniego nakładu czasu. Musiałam skupić się na szczegółach, tak by urządzenie działało w pełni poprawnie, ale również było w pełni bezpieczne.

3. Elementy projektu

Poniżej znajduje się spis wszystkich elementów użytych do zbudowania stacji pogodowej. Jako czujnika temperatury oraz wilgotności użyłam czujnika DHT11, czujnik barometryczny zastosowany w projekcie to czujnik firmy POLULU LPS331AP. Zegar RTC to standardowy zegar DS1307 z dodatkowym modułem pamięci EEPROM.

Tabela 1: Zestawienie elementów projektowych wraz z oznaczeniami schematowymi projektu

L.p.	Oznaczenie na schemacie	Opis elementu	Kod urządzenia	Parametry pomocnicze
1	IC1	mikrokontroler	U-35460K	ATMEGA32
2	U\$1	zegar RTC	DS1307	+ 32kB pamięci EEPROM
3	U\$2	czujnik temperatury i wilgotności	DHT11	AOSONG DHT11
4	U\$3	czujnik barometryczny	LPS331AP	POLULU
5	DIS1	wyświetlacz LCD	HD44780	HITACHI, 2x16
6	KLA_MATRY	klawiatura matrycowa	-	-
7	P1	potencjometr	RM-065	rezystancja 10kΩ
8	P2	potencjometr	RM-065	rezystancja 10kΩ
9	Q1	rezonator kwarcowy	HC49-S	THT, 16MHz
10	C1	Kondensator ceramiczny	-	22pF, 100V
11	C2	Kondensator ceramiczny	-	22pF, 100V
12	R1	rezystor	-	4.7kΩ, 1/4W
13	R2	rezystor	-	10kΩ, 1/4W
14	R3	rezystor	-	4.7kΩ, 1/4W
15	R4	rezystor	-	4.7kΩ, 1/4W
16	S1	tact switch	-	0.05A/12VDC, 4pin

3.1. Pisanie kodu w języku C oraz testowanie urządzenia

W początkowej fazie reałizacji projektu złożyłam układ testowy oparty na Arduino UNO (Atmega328P). Wykorzystałam płytę stykową i wykonałam połączenia początkowo tylko wyświetlacza LCD 2x16, potencjometrów oraz klawiatury matrycowej. Rozpoczęłam pisanie własnej biblioteki obsługującej wyświetlacz o sterowniku HD44780. Założyłam wysył danych za pomocą 4 linii z danymi, aby zaoszczędzić miejsce. Bibliotekę napisałam tak, aby móc

wykorzystać ją wielokrotnie. Założyłem również, że nie pobieramy z wyświetlacza żadnych danych, zatem pin R/W podłączylem na stałe do uziemienia. Dodatkowo dla ułatwienia zakładam, że użytkownik podpina wszystkie piny od wysyłu danych po kolej w jednym rzędzie. Poniżej znajduje się fragment kodu z pliku *LCD.h* załączonego do projektu, który trzeba odpowiednio ustawić, aby skorzystać z biblioteki.

```
/*Data to set*/
/*RW to GND - writing mode and all data pins are in one row, one by one*/
#define LCD_E_DDR DDRC
#define LCD_E_PORT PORTC
#define LCD_E_PIN 6
#define LCD_RS_DDR DDRC
#define LCD_RS_PORT PORTC
#define LCD_RS_PIN 7
#define LCD_DATA_DDR DDRC
#define LCD_DATA_PORT PORTC
#define LCD_DATA_FIRST_PIN 2
```

W kolejnych liniach pliku *LCD.h* znajdują się definicje stałych potrzebnych do ustawiania trybu przesyłu danych oraz innych potrzebnych ustawień. Definicje zostały wykonane zgodnie z dokumentacją wyświetlacza [1]. Po definicjach stałych znajdują się definicje podstawowych funkcji potrzebnych użytkownikowi do korzystania z wyświetlacza.

```
/* Functions*/
/*Initialization of LCD*/
void LCD_Init();
/*Clear screen*/
void LCD_Clear();
/*Function to send instruction to LCD*/
void LCD_SendInstruction(int8_t mask);
/*Function to write one char to LCD, use twice to send char = 4bit/4bit*/
void LCD_Write(int8_t symbol);
/*Function to send text to LCD*/
void LCD_WriteText(char *text);
/*Function to send number - int - to LCD*/
void LCD_WriteInt(int number);
/*Move cursor to (row, column)
The top left corner is a point: (1,1)*/
void LCD_GoTo(int row, int column);
/*Move cursor to home (1,1)*/
void LCD_GoHome();
/*Function to add special character to memory*/
void LCD_AddSpecialChar(int8_t symbol[8], int where);
```

Po przetestowaniu działania biblioteki uzupełniłem plik główny o funkcję obsługującą klawiaturę *scan_keyboard()*.

```
uint8_t scan_keyboard() {
    uint8_t r, c;
    for (r = 0; r < 4; r++) {
        PORTA &= ~(1 << r); // set low
        _delay_ms(1); // Sprawdzenie wartości w każdym wierszu
        for (c = 0; c < 4; c++) { // w każdej kolumnie
            if (!(PIN0 & (1 << (c + 4)))) { // check if on PORTC is sth
                PORTA |= (1 << r);
                _delay_ms(100);
                return r * 4 + (c + 1);
            }
        }
        PORTA |= (1 << r); // set high
    }
    _delay_ms(100);
    return 99; // nothing pressed
}
```

Dodałem w funkcji głównej funkcję wyświetlającą numer wciśniętego przycisku i przetestowałem działanie klawiatury. Po ukończeniu tych podstawowych prac podłączylem czujnik DHT11 i dodałem bibliotekę do obsługi połączenia czujnika z mikrokontrolerem. Skorzystałem z biblioteki udostępnionej przez Davide Gironi na jego stronie internetowej. Ponownie przetestowałem działanie, wyświetlając temperaturę i wilgotność porównując wyniki z innymi czujnikami zewnętrznymi. Kolejnym etapem prac nad projektem była komunikacja z czujnikiem barometrycznym poprzez protokół transmisji SPI. Skorzystałem tutaj z dokumentacji ATmega32 [2], gdzie bardzo dobrze opisany jest ten interfejs komunikacyjny. Wszystkie napisane przeze mnie funkcje potrzebne do poprawnego uzyskania danych z

czujnika znajdują się w plikach *spi.h*, *spi.c*. Informacje potrzebne do odpowiedniego ustawienia komunikacji - odpowiednie rejestyry oraz dane potrzebne do wpisania - znalazłam w dokumentacji czujnika firmy POLULU [3]. Poniżej znajdują się definicje zaimplementowanych funkcji.

```
/*Initialization of communication with the sensor*/
void spi_init();
/*Read from address*/
char spi_read(char address);
/*Write data to address*/
void spi_write(char address, char data);
/*Download pressure data*/
int32_t spi_download_press();
/*Download temperature data*/
uint8_t spi_download_temp();
```

Jednym z ostatnich etapów było dodanie zegara RTC oraz dodanie kodu zapewniającego transmisję danych do urządzenia i z niego. Wykorzystany protokół transmisji to I^2C . Tutaj również wykorzystałam dokumentację mikrokontrolera, ponieważ zawiera ona dokładny opis tego interfejsu [2]. W przypadku zegara RTC potrzebny jest zapis liczb do formatu BCD - zgodnie z dokumentacją [4]. Zatem uzupełniłam kod również o funkcje przekształcające liczby dziesiętne do tego formatu i na odwrót. Wszystkie zaimplementowane funkcje oraz definicje stałych związanych ze sprawdzaniem poprawności przesyłu oraz odbioru danych znajdują się w pliku *i2c.h* dołączonym do projektu.

```
/*initialize i2c */
void i2c_init();
/*Send start to slave_address*/
int i2c_send_start(char slave_address);
/*Send stop (slave address not needed) */
void i2c_send_stop();
/*Send (read/write) address of memory*/
int i2c_rw_address(char rw_address);
/*Send (read/write) address of memory without checking ACK*/
int i2c_rwspecial_address(char rw_address);
/*Write data to slave */
int i2c_write(char data);
/*Send repeated start to slave_address*/
int i2c_send_repeated_start(char slave_address);
/*Read from slave
end_of_reading is to set whether to send ack, or nack bit
1 - means end reading, send nack bit
0 - means that there is sth to read there*/
int i2c_read(int end_of_reading);
```

Ostatnim elementem było uzupełnienie projektu o menu i obsługę wyświetlania wybranego punktu przez kilka sekund. Tutaj posłużyły się timerami i obsługą przerwania, tak aby po odpowiednim czasie wyświetlania np. temperatury urządzenie powróciło do wyświetlania menu. Wykorzystałam timer nr 1 z prescalerem CS12=1 oraz CS10=1 (1024). Po odpowiednich obliczeniach otrzymujemy czas około 4.2s wyświetlania wybranego elementu z menu. Przerwanie zatrzymuje timer i ustawia jego początkową wartość na 0. Timer znalazł również zastosowanie przy zmienianiu stron menu głównego. Wykorzystuję tutaj porównywanie wartości bieżącej timera z wartością zadaną. Uzyskałam w ten sposób odliczanie do około dwóch sekund i zmianę na następną część menu.

Dodatkowo wykorzystałam pamięć EEPROM umieszczoną na płytce wraz z zegarem RTC. Zaimplementowałam dwie funkcje: zapisującą do pamięci oraz odczytującą z pamięci zewnętrznej. Obie funkcje opierają się o komunikację I^2C . Zgodnie z dokumentacją układu AT24C32 [5], aby skomunikować się z układem należy wysłać odpowiedni adres urządzenia. W moim przypadku jest to dokładnie 1010 000 R/W. Ostatni bit odpowiada za wysłanie do urządzenia informacji, o tym czy chcemy z czytać z pamięci, czy też zapisywać. Aby korzystać z pamięci, musiałam wymyślić odpowiedni schemat zapisu danych. W związku z tym, że danymi do zapisu są: data(dd-mm-yy), temperatura, wilgotność oraz ciśnienie, większość zapisu nie sprawiała większego problemu. Jedynie ciśnienie wymagało dodatkowych przeliczeń. Tabela numer 2 przedstawia sposób zapisu do EEPROMU.

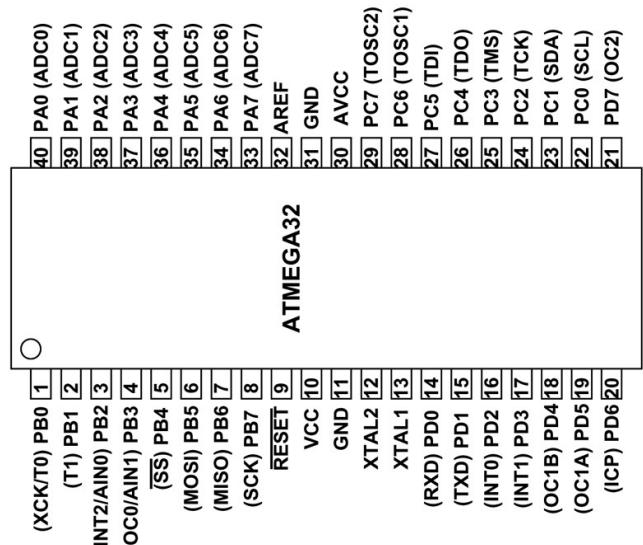
Tabela 2: Zestawienie elementów projektowych wraz z oznaczeniami schematowymi projektu

Adres pamięci	Zapis bitowy
i*8	dzień
i*8+1	miesiąc
i*8+2	rok
i*8+3	temperatura
i*8+4	wilgotność
i*8+5	(ciśnienie - 900)

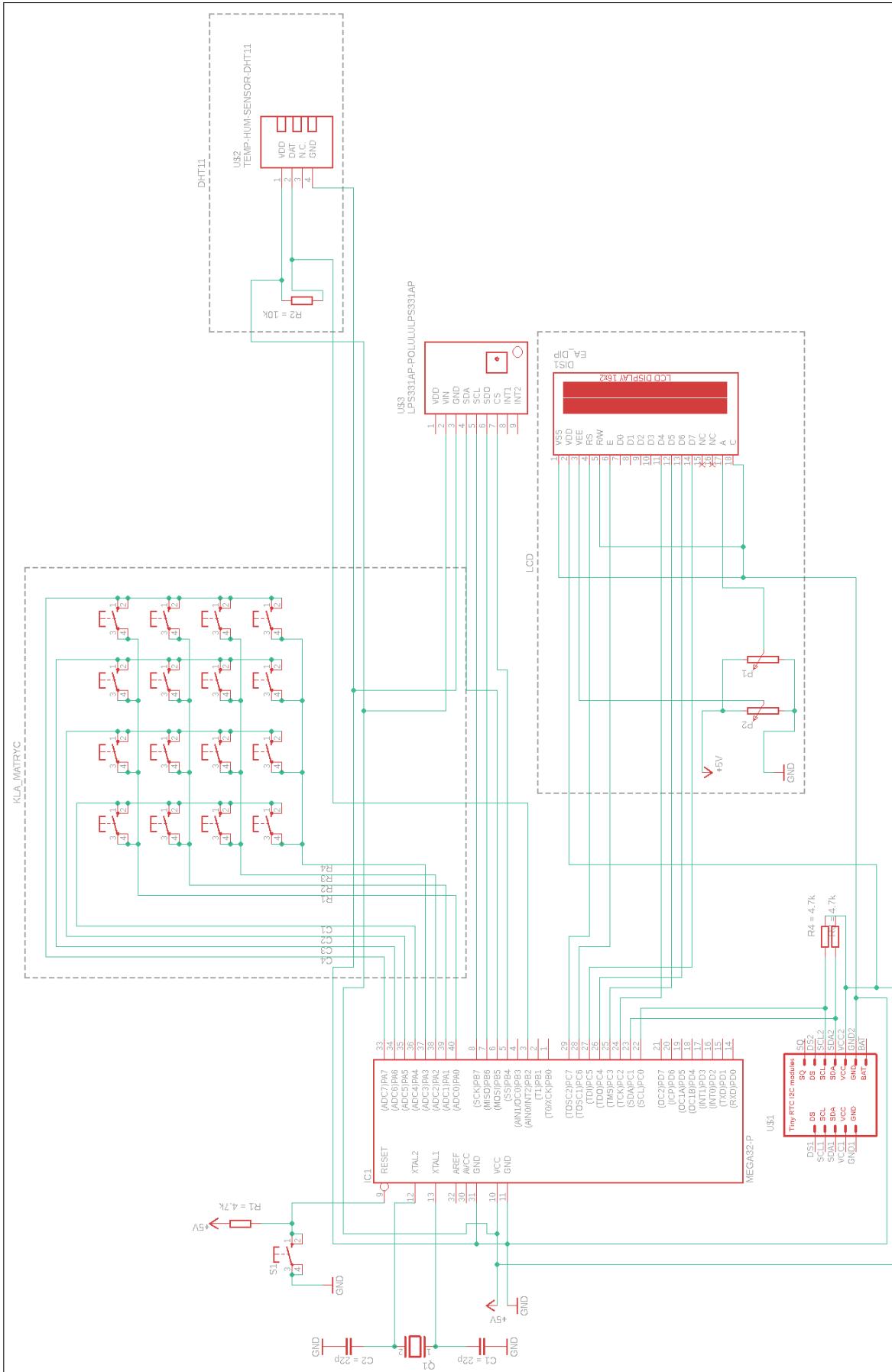
Przykładowo: Uznałam, że adresem pierwszych danych jest adres 0x0000. Zatem dla pierwszego zestawy danych adresy prezentują się następująco: 0x0000, 0x0001, 0x0002, 0x0003, 0x0004, 0x0005. Odczytując zapisane dane otrzymujemy bezpośrednio wartości do wyświetlania. W przypadku ciśnienia do wartości odczytanej z pamięci należy dodać wartość 900. W ten sposób zakres zapisanego ciśnienia jest od 900 do 1155. Jes to wystarczający zakres patrząc na ciśnienie otoczenia. Gdyby zaistniała potrzeba posiadania pełnego zakresu czujnika, należałoby rozbić zapis na dwa rejestr MSB, LSB.

3.2. Wykonanie schematu połączeń w programie Autodesk Eagle

Schemat wykonałem na podstawie dokumentacji wszystkich urządzeń zawartych w projekcie. Zgodnie z wyprowadzeniami Atmegi32 - przedstawionymi na poniższym rysunku - kryształ 16MHz umieszczałem miedzy pinami 12 oraz 13. Do każdej z nóżek kryształu podłączylem kondensator ceramiczny 22pF. Do podłączenia klawiatury wykorzystałem cały port A. Zgodnie ze schematem czujnik barometryczny POLULU podłączylem do pinów: 5,6,7,8, potrzebnych do komunikacji za pomocą SPI. Czujnik DHT11 wymaga tylko jednego pinu, zatem podłączylem go do pinu 2 portu B. Zegar RTC podłączylem do pinów 22 oraz 23 portu C, dodatkowo zostały one podłączone przez rezystory do 5V, zgodnie z dokumentacją Atmegi32 [2]. Wyświetlacz LCD podłączylem do pozostałych 6 pinów portu C, dwa piny zostały wykorzystane na RS oraz E, pozostałe wykorzystane są do wysyłu danych. Dodatkowo do wyświetlacza podpięte zostały dwa potencjometry, aby odpowiednio dobrać podświetlenie oraz kontrast na wyświetlaczu.



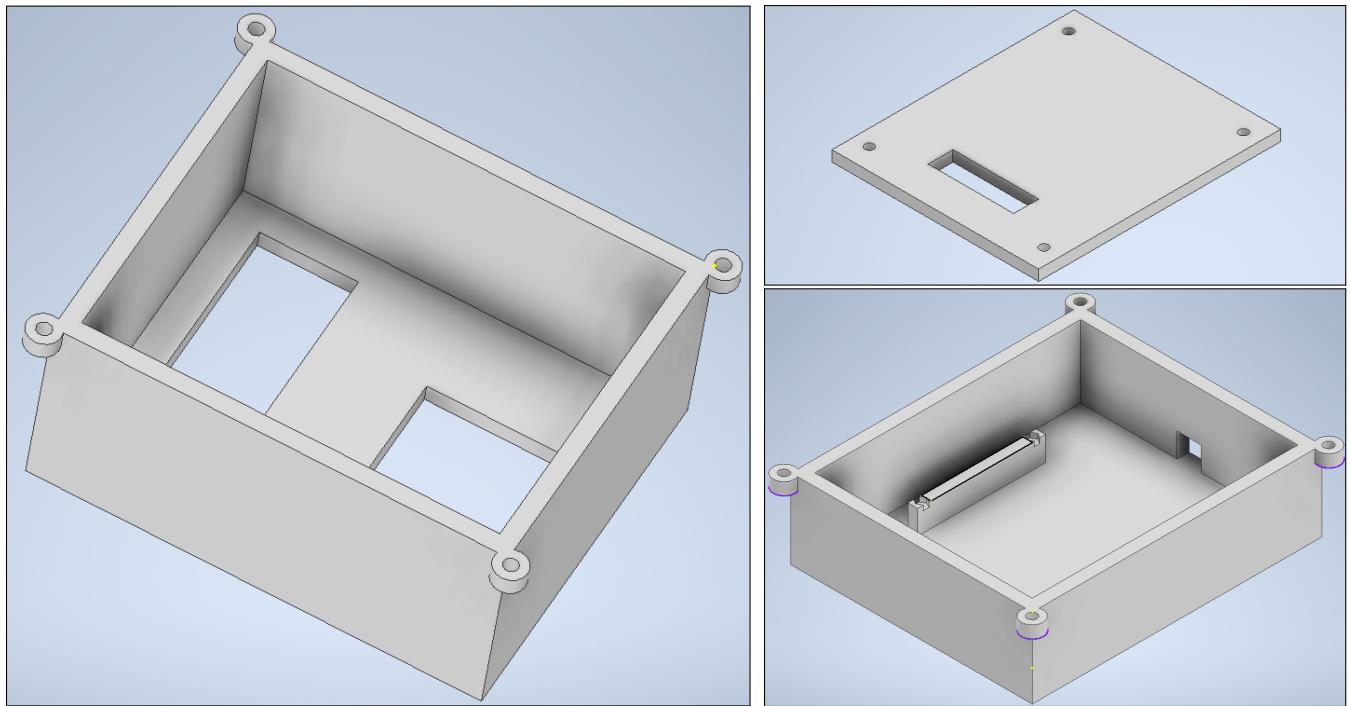
Rysunek 1: Schemat wejść/wyjść Atmegi 32



Rysunek 2: Schemat połączeń elementów stacji pogodowej

3.3. Wykonanie obudowy stacji pogodowej

Obudowę wykonałem w kilku częściach. Do zaprojektowania obudowy wykorzystałem program Autodesk Inventor Professional 2021. Osobno wykonałem góre, spód oraz specjalną podkładkę pod klawiaturę. Poniżej znajdują się screeny elementów z programu.



Rysunek 3: Elementy obudowy stacji pogodowej

Elementy obudowy zostały wydrukowane za pomocą drukarki 3D. Pozostałe elementy potrzebne do zmontowania obudowy:

- M2x5mm x4 do płytka uniwersalnej z atmegą
- M3x16mm x4 do wyświetlacza LCD
- M3x16mm x4 do obudowy
- M3x16mm x2 do modułu TINY RTC
- M1x5mm x1 do czujnika barometrycznego
- M2x25mm x4 do podkładki pod klawiaturę
- nakrętka M3 x10
- nakrętka M2 x8
- nakrętka M1 x1
- podkładka M3 x10
- podkładka M2 x8
- podkładka M1 x1

4. Możliwości rozwojowe projektu

Projekt można rozwijać chociażby o wyznaczanie wysokości, na której się znajdujemy. Zaletą jest to, że korzystając z czujnika LPS331AP, nie ma potrzeby dokładania nowych czujników, ponieważ na postawie uzyskiwanych z niego danych możemy wyznaczyć wysokość. Dużym usprawnieniem układu byłby również moduł WiFi. Można by wtedy pokusić się o stronę internetową, na której byłyby wyświetlane dane z czujników oraz historia. Warto też zadbać o mniejsze gabaryty układu - warto przemyśleć zaprojektowanie własnej płytki PCB ze wszystkimi czujnikami umieszczonymi na płytce. Zmniejszyłoby to ryzyko przerwania kabla, a zarazem zdecydowanie pomniejszyłby rozmiary obecnego układu.

5. Efekt końcowy pracy

Układ posiada możliwość:

- wyświetlenia obecnej temperatury,
- wyświetlenia obecnej wilgotności,
- wyświetlenia obecnego ciśnienia,
- wyświetlenia daty z dokładnością do sekund,
- zapisu zestawu danych do zewnętrznej pamięci,
- odczytu zapisanego zestawu danych.

Układ wykorzystuje do obsługi menu klawiaturę matrycową, z której wiele przycisków można jeszcze wykorzystać. Wszystkie dane wyświetlane są na wyświetlaczu LCD o sterowniku HD44780. Poniżej znajdują się zdjęcia poszczególnych stron menu wraz z opisem ich funkcjonalności. Menu główne składa się z 3 stron wyświetlanych na zmianę. Każda z 3 stron wyświetla po dwie opcje do wybrania. Poniżej znajdują się zdjęcia poszczególnych stron.



Po naciśnięciu przycisku nr 1 na klawiaturze matrycowej zgodnie z wyświetlonym menu uzyskujemy dostęp do informacji o obecnej temperaturze. Po naciśnięciu przycisku nr 2 dostajemy informację o obecnej wilgotności w pomieszczeniu.

Rysunek 4: Pierwsza strona menu głównego



Rysunek 5: Podstrony wyświetlanego menu: temperatura oraz wilgotność



Po naciśnięciu przycisku nr 3 na klawiaturze matrycowej zgodnie z wyświetlonym menu uzyskujemy dostęp do informacji o obecnym ciśnieniu. Po naciśnięciu przycisku nr 4 wyświetlana zostaje pełna data z chwili naciśnięcia przycisku i aktualizuje się ona co sekundę.

Rysunek 6: Druga strona menu głównego



Rysunek 7: Podstrony wyświetlonego menu: ciśnienie oraz data



Po naciśnięciu przycisku nr 5 , jak i przycisku nr 6, w obu przypadkach przechodzimy do podokna wyboru miejsca zapisu lub odczytu z pamięci. Należy wybrać, korzystając z klawiatury, liczbę od 1 - 5 w celu wyboru miejsca pamięci zewnętrznej. Możliwe jest jeszcze naciśnięcie przycisku nr 16. Spowoduje to anulowanie obecnego ekranu i powrót do menu głównego.

Rysunek 8: Trzecia strona menu głównego



Rysunek 9: Podstrony zapisu oraz odczytu z pamięci

Oznaczenie "p" przy liczbach oznacza, że w takim miejscu podczas obecnej pracy urządzenia nie zostały jeszcze zapisane żadne dane. W momencie zapisu na dane miejsce pamięci "p" zmienia się na "u", co jest zaprezentowane na powyższych zdjęciach



Rysunek 10: Podstrony po zapisie oraz odczytanie danych

W przypadku zapisu po wybraniu odpowiedniego miejsca w pamięci, wyświetlana jest informacja o poprawnym zapisie. W przypadku odczytu pojawiają się dane z czujników oraz data zapisu owych danych.

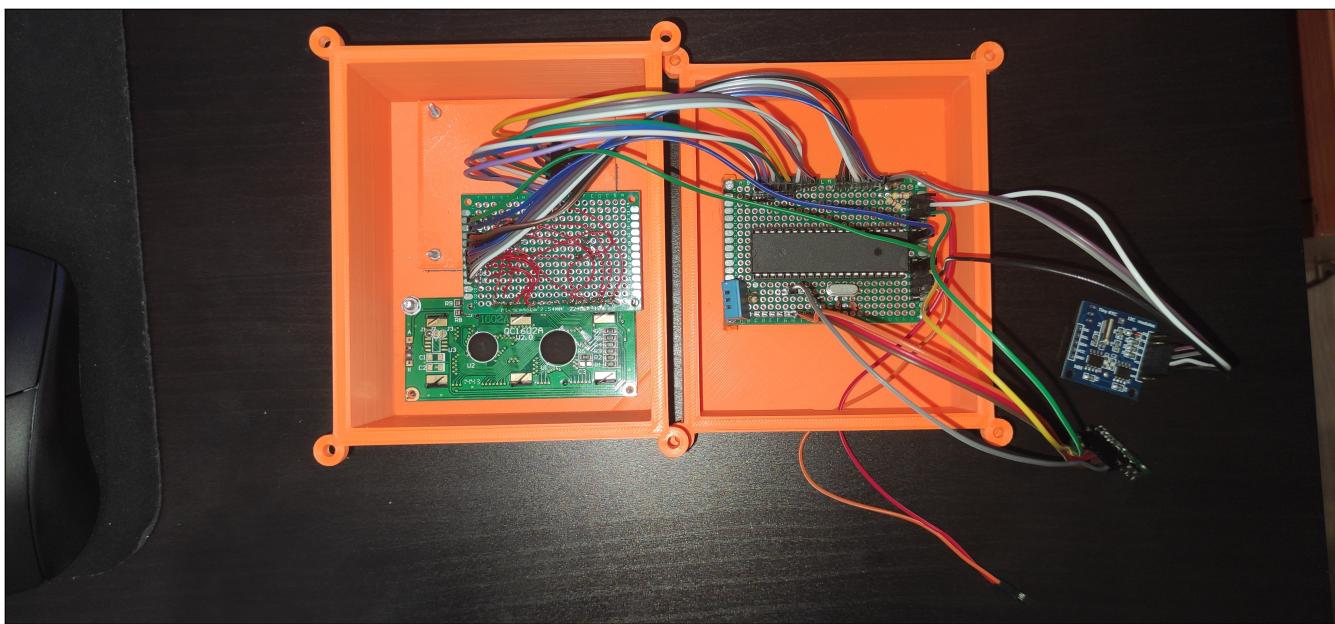
W przypadku zapisu oraz odczytu możliwe jest wycofanie się z tych ekranów. Naciskając przycisk nr 16, cofujemy zadaną wcześniej operację. Po naciśnięciu przycisku na ekranie wyświetlana jest informacja o anulacji polecenia.



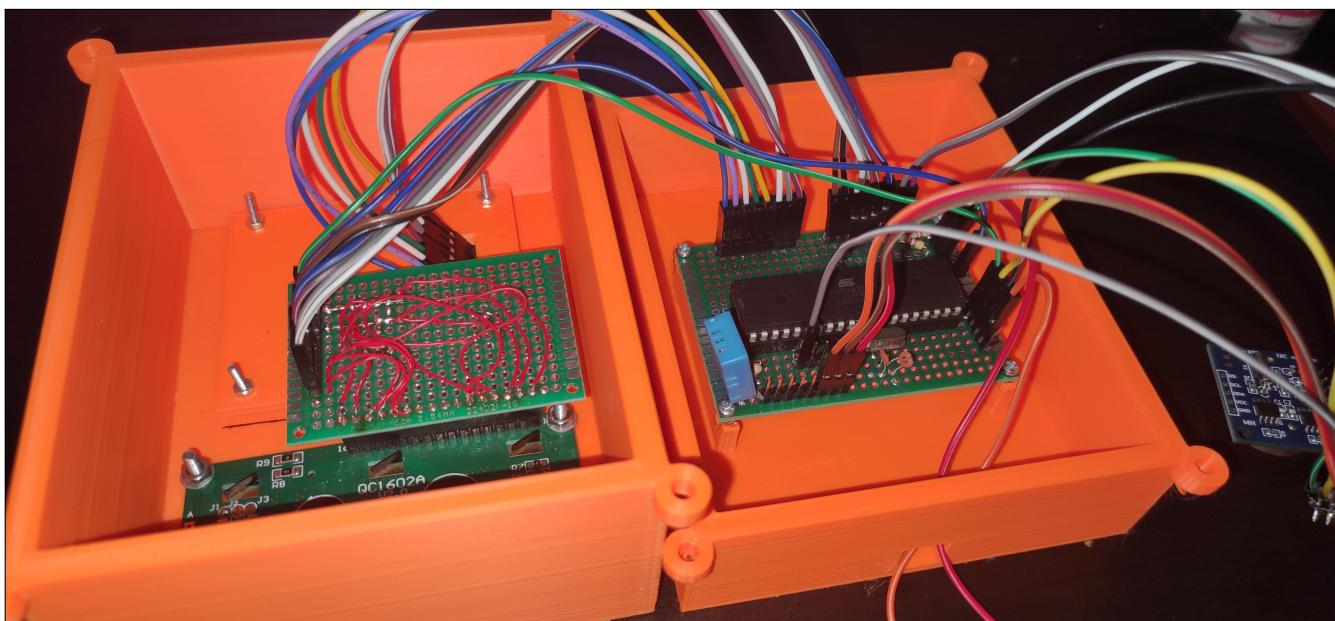
Rysunek 11: Anulowanie zapisu lub odczytu z pamięci

Kod źródłowy projektu wraz z modelami obudowy znajduje się na stronie github.com na moim prywatnym koncie. Link prowadzący do kodu źródłowego: <https://github.com/Emilysta/WeatherStation>, [link].

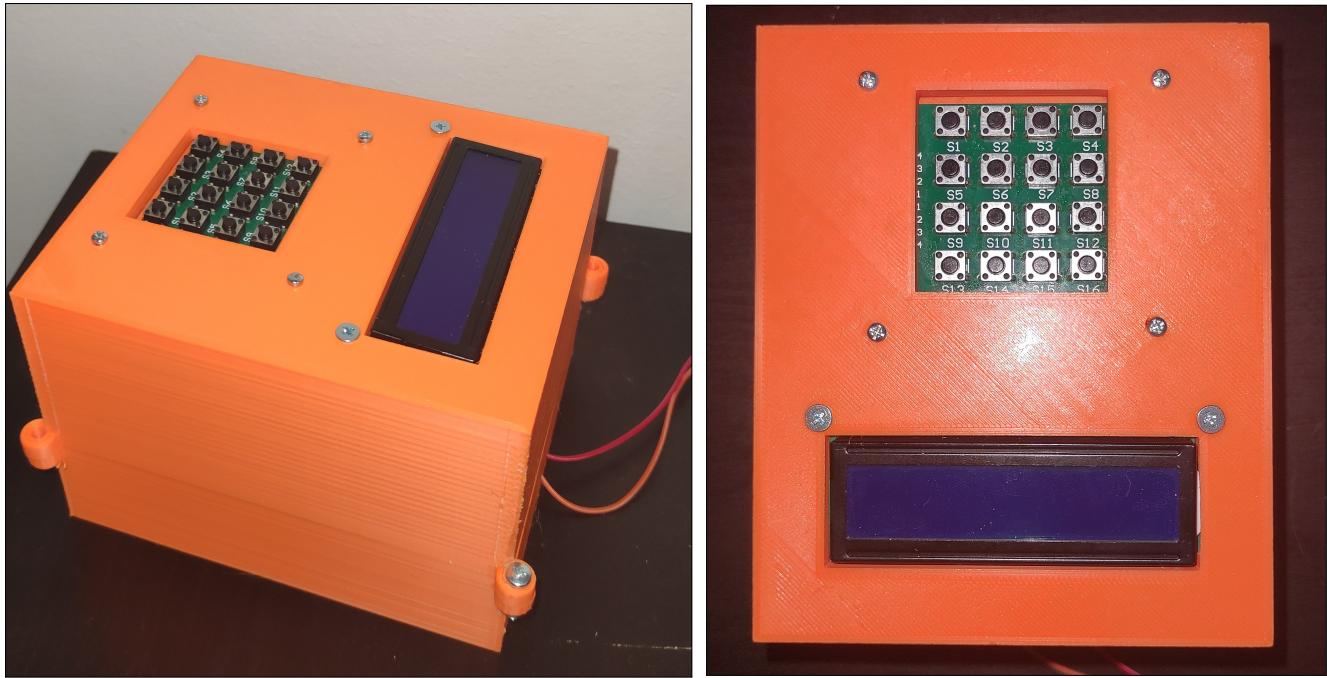
6. Zdjęcia projektu prezentujące ostateczny wygląd



Rysunek 12: Zdjęcie wnętrza - płytka uniwersalna z ATmega32, wyświetlacz, dodatkowa płytka do wyświetlacza



Rysunek 13: Zdjęcie wnętrza z bliska



Rysunek 14: Zdjęcia projektu z zewnątrz

Film prezentujący działanie układu umieszczony został pod linkiem:
<https://drive.google.com/file/d/1BawenwPFYtUKA13hmQIXgNxF-ID4po51/view?usp=sharing>, [link].

Literatura

- [1] HITACHI. *HD44780 Documentacja*.
- [2] ATMEL. *8-bit AVR Microcontroller with 32KBytes In-System Programmable Flash*. [link].
- [3] POLULU. Mems pressure sensor: 260-1260 mbar absolute digital output barometer. [link].
- [4] OEM. *DS1307 Documentacja*.
- [5] ATMEL. *2-wire Serial EEPROM, AT24C32, AT24C64*. [link].
- [6] Davide Gironi. Reading temperature and humidity on avr atmega using dht11 library 01. [link].