Realizacja praktyczna układu z zegarem czasu rzeczywistego RTC, pomiarem temperatury, wilgotności powietrza oraz ciśnienia atmosferycznego, z wyświetlaniem informacji na ekranie TFT LCD. Realizacja łączności z wykorzystaniem interfejsów I2C oraz SPI.

|  |  |
| --- | --- |
| Autor: Marcel Kozerski | Data wykonania: 28.05.2025r.  Wydział Inżynierii Elektrycznej i Komputerowej  Rok akademicki: 2025/2026 |

1. Podstawy teoretyczne

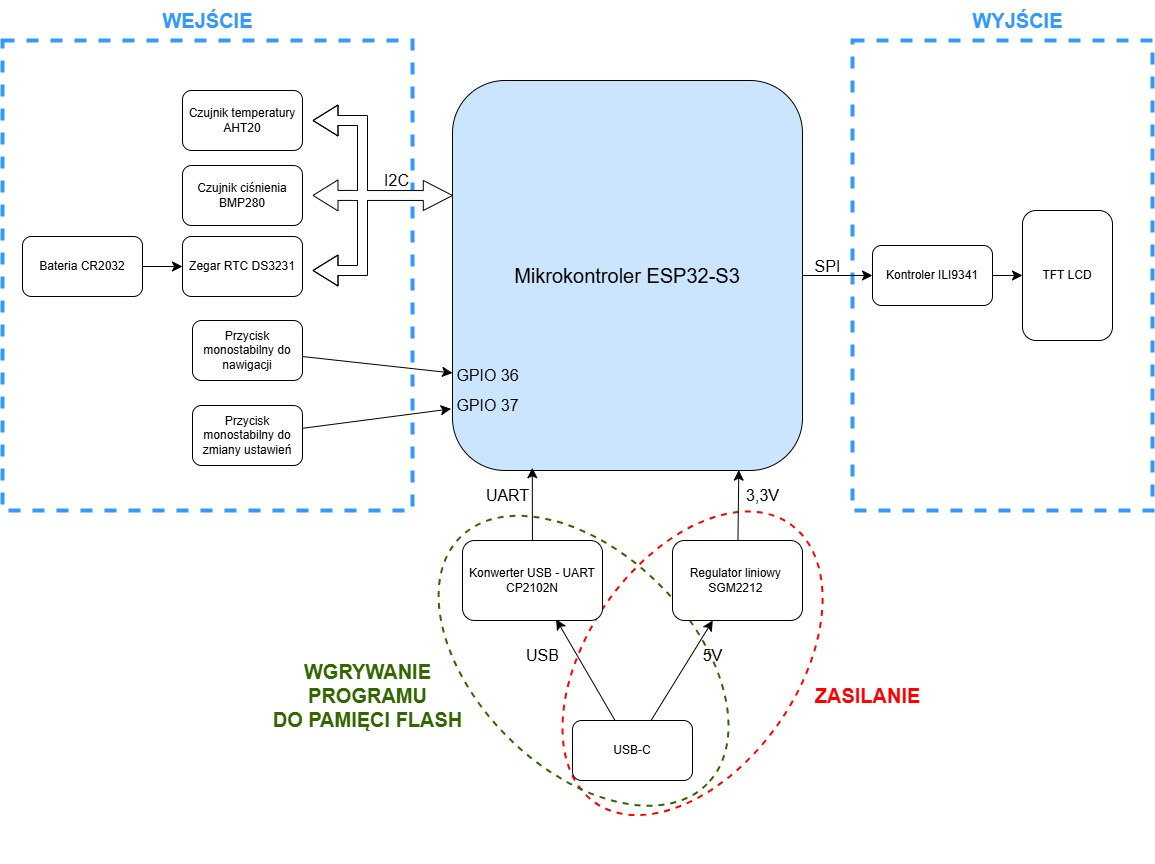
Układ elektroniczny realizuje funkcje pomiarowe i wyświetlające, integrując mikrokontroler ESP32-S3 z czujnikami AHT20 (temperatura i wilgotność), BMP280 (ciśnienie atmosferyczne) oraz zegarem RTC DS3231. Komunikacja z czujnikami AHT20 i BMP280 oraz zegarem DS3231 odbywa się przez interfejs I2C, natomiast kolorowy wyświetlacz TFT LCD wyposażony w kontroler ILI9341 wykorzystuje interfejs SPI. Dwa przyciski umożliwiają interakcję użytkownika: wybór parametru do edycji (sekunda, minuta, godzina lub data) oraz zwiększanie jej wartości. Zegar RTC jest podtrzymywany baterią CR2032, zapewniając ciągłość czasu przy braku zasilania. Układ jest zasilany przez port USB-C na pytce deweloperskiej ESP32-S3.

Interfejs I2C pozwala na dwukierunkową komunikację z urządzeniami przy użyciu dwóch linii: SDA (dane) i SCL (zegar). SPI wykorzystuje cztery linie: MOSI, MISO, SCK i CS, zapewniając szybką transmisję danych do wyświetlacza. Do realizacji projektu zostało wykorzystane środowisko Arduino.

1. Podstawowe parametry układu
   1. Parametry wejściowe

* Czujnik AHT20: Pomiar temperatury.
* Czujnik BMP280: Pomiar ciśnienia.
* Zegar RTC DS3231: Czas (godziny, minuty, sekundy), data (dzień, miesiąc, rok), dzień tygodnia.
* Przyciski: Dwa przyciski monostabilne (pull-down, stan wysoki po naciśnięciu).
  1. Parametry wyjściowe
* Wyświetlacz TFT LCD z kontrolerem ILI9341 wyświetlanie: temperatura, wilgotność, ciśnienie, czas, data, dzień tygodnia.
  1. Zasilanie
* Układ główny zasilany jest napięciem stałym 5V przez złącze USB-C umieszczone na płytce deweloperskiej ESP32-S3. Do zasilenia mikrokontrolera oraz pozostałych komponentów układu napięciem 3,3V wykorzystano liniowy regulator napięcia SGM2212.
* Do podtrzymania zasilania układu RTC wykorzystano baterię CR2032 o nominalnym napięciu 3V.

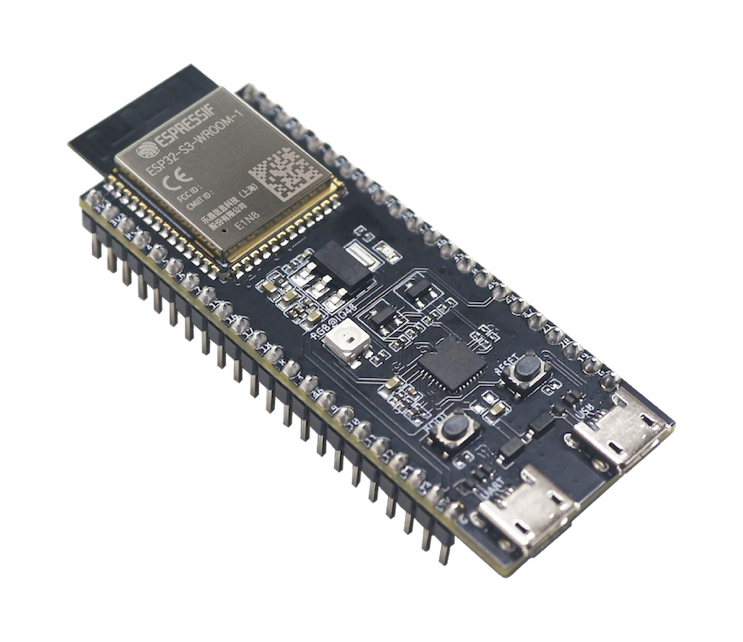
1. Schemat blokowy i opis działania



Rysunek 1 Schemat blokowy układu.

Mikrokontroler cyklicznie odczytuje dane z czujników AHT20 i BMP280 oraz zegara DS3231 przy pomocy interfejsu I2C. Dane są przetwarzane i wyświetlane na ekranie TFT za pośrednictwem kontrolera ILI9341 skomunikowanego z mikrokontrolerem ESP32-S3 przez interfejs SPI. Pierwszy przycisk przełącza między parametrami zegara do edycji (sekundy, minuta, godzina, dzień, miesiąc, rok). Drugi przycisk zwiększa wybrany parametr o 1. Zegar czasu rzeczywistego RTC jest podtrzymywany dodatkowym zasilaniem z baterii CR2032.

1. Opis i parametry urządzeń składowych
   1. Mikrokontroler ESP32-S3



Rysunek 2 Mikrokontroler ESP32-S3 [1]

|  |  |
| --- | --- |
| **Procesor** | Dwurdzeniowy Xtensa LX7, 32-bit |
| **Obsługiwany protokół Wi-Fi i Bluetooth** | 802.11 b/g/n, 2.4 GHz; Bluetooth 5.0 |
| **Pamięć Flash** | 8 MB |
| **SRAM** | 512 KB |
| **Obsługiwane interfejsy komunikacyjne** | I2C (2 kanały), SPI (4 kanały),  UART (3 kanały) |
| **Napięcie zasilania** | 3V – 3,6V |
| **Ilość pinów GPIO** | 45 |

Tabela 1 Parametry mikrokontrolera ESP32-S3. [2]

* 1. Czujnik AHT20

A purple circuit board with black and silver wires

Description automatically generated

Rysunek 3 Moduł z czujnikiem AHT20 oraz BMP280 [3]

|  |  |
| --- | --- |
| **Typ czujnika** | Temperatura i wilgotność względna |
| **Interfejs komunikacyjny** | I2C |
| **Zakres temperatury** | -40°C do +85°C |
| **Dokładność temperatury** | ± 0,3 °C |
| **Zakres wilgotności** | 0 – 100% RH |
| **Dokładność wilgotności** | 2% RH |
| **Napicie zasilania** | 2.2V - 5.5 V (w projekcie: 3.3 V) |
| **Pobór prądu** | 40 µA (pomiar), 0.5 µA (uśpienie) |
| **Czas odpowiedzi** | 8 sekund (wilgotność), 6 sekund (temperatura) |

Tabela 2 Parametry czujnika AHT20. [4]

* 1. Czujnik BMP280

|  |  |
| --- | --- |
| Typ czujnika | Ciśnienie atmosferyczne, temperatura |
| Interfejs komunikacyjny | I2C (do 3,4 MHz); SPI (do 10 MHz) |
| Zakres ciśnienia | 300 - 1100 hPa |
| Dokładność ciśnienia | ±1 hPa |
| Zakres temperatury | -40°C do +85°C |
| Napicie zasilania | 1.71V - 3.6 V (w projekcie: 3.3 V) |
| Pobór prądu | 2.8 µA (typowo przy pomiarze), 0,3 µA (uśpienie) |

Tabela 3 Parametry czujnika BMP280. [5]

* 1. Zegar RTC DS3231

A close-up of a battery

Description automatically generated

Rysunek 4 Zegar RTC DS3231 [6]

|  |  |
| --- | --- |
| Typ urządzenia | Zegar czasu rzeczywistego (RTC) |
| Interfejs komunikacyjny | I2C |
| Dokładność | 2 ppm (przy temperaturze od 0°C do +40°C) |
| Napicie zasilania | 2.3V – 5.5 V (w projekcie: 3.3 V) |
| Pobór prądu | 200 µA (maksymalna wartość podczas przesyłania danych) |
| Funkcje dodatkowe | Generator fali prostokątnej, alarmy |

Tabela 4 Parametry zegara RTC DS3231. [7]

* 1. Wyświetlacz TFT LCD ILI9341

A close up of a screen

Description automatically generated

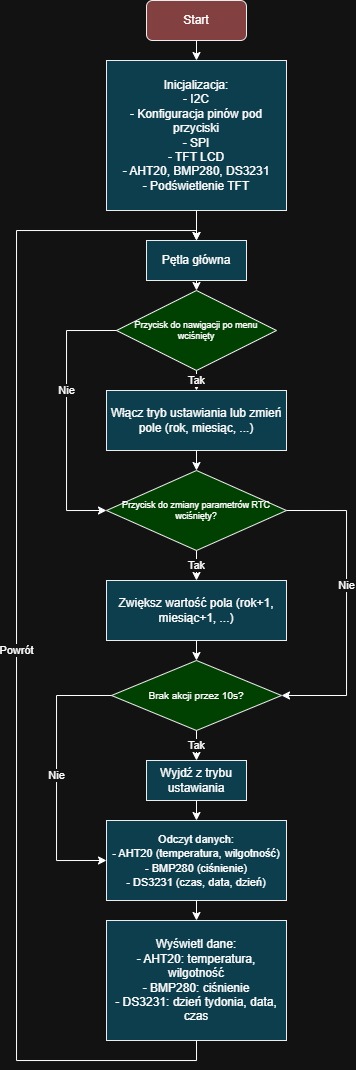
Rysunek 5 Wyświetlacz TFT LCD [8]

|  |  |
| --- | --- |
| Typ wyświetlacza | TFT LCD, kolorowy |
| Sterownik | ILI9341 |
| Interfejs komunikacyjny | SPI (4-przewodowy: MOSI, SCK, CS, DC) |
| Rozdzielczość | 240x320 pikseli |
| Głębia kolorów | 16-bit (65 536 kolorów) |
| Napicie zasilania | 2.5V – 3.3V (w projekcie 3.3V) |
| Rozmiar ekranu | 2.4 cala |

Tabela 5 Parametry wyświetlacza TFT LCD ILI9341. [9]

1. Algorytm programu
   1. Inicjalizacja:

* Inicjalizacja magistrali I2C z pinami (SDA: GPIO 45, SCL: GPIO 48).
* Konfiguracja przycisków nawigacji (GPIO 36) i ustawiania (GPIO 37) jako wejścia.
* Inicjalizacja wyświetlacza ILI9341 TFT LCD przez SPI (piny: CS=10, DC=2, MOSI=11, SCK=12, RST= pin reset mikrokontrolera, LED=8), ustawienie orientacji ekranu, ustawienie koloru tła na bordowy, konfiguracja tekstu (biały kolor, rozmiar 2).
* Inicjalizacja czujnika AHT20 (I2C, adres 0x38); zatrzymanie programu w przypadku niepowodzenia.
* Inicjalizacja czujnika BMP280 (I2C, adres 0x77); zatrzymanie programu w przypadku niepowodzenia.
* Inicjalizacja zegara RTC DS3231 (I2C, adres 0x68); ustawienie czasu na czas kompilacji w przypadku utraty zasilania; zatrzymanie programu w przypadku niepowodzenia.
* Włączenie podświetlenia TFT przez ustawienie pinu LED na stan wysoki.
  1. Pętla główna:
* **Obsługa przycisków**:
  + Monitorowanie przycisku do nawigacji. Przy stanie wysokim (z eliminacją drgań 100 ms), włączenie trybu ustawiania lub przełączenie między parametrami RTC (rok, miesiąc, dzień, godzina, minuta).
  + Monitorowanie przycisku zmiany parametrów RTC. Przy stanie wysokim (z eliminacją drgań 100 ms), w trybie ustawiania zwiększenie wartości wybranego pola (np. rok do 2099, miesiąc do 12, dzień do 31, godzina do 23, minuta do 59).
  + Wyjście z trybu ustawiania po 10 sekundach braku aktywności przycisków.
* **Odczyt danych z czujników**:
  + Odczyt temperatury i wilgotności z AHT20.
  + Odczyt ciśnienia z BMP280 (przeliczenie z Pa na hPa).
  + Odczyt bieżącego czasu i daty z DS3231.
* **Aktualizacja wyświetlacza**:
  + Dla każdego typu danych (temperatura i wilgotność AHT20, ciśnienie BMP280, dzień, data, czas RTC, status trybu ustawiania) formatowanie danych do ciągu znaków.
  + Czyszczenie linii wyświetlacza, jeśli nowy tekst jest krótszy od poprzedniego, aby uniknąć dodatkowych znaków za wyrazem.
  + Wyświetlanie danych z czujnika AHT20,
  + Wyświetlanie danych z czujnika BMP280.
  + Wyświetlanie danych z DS3231.



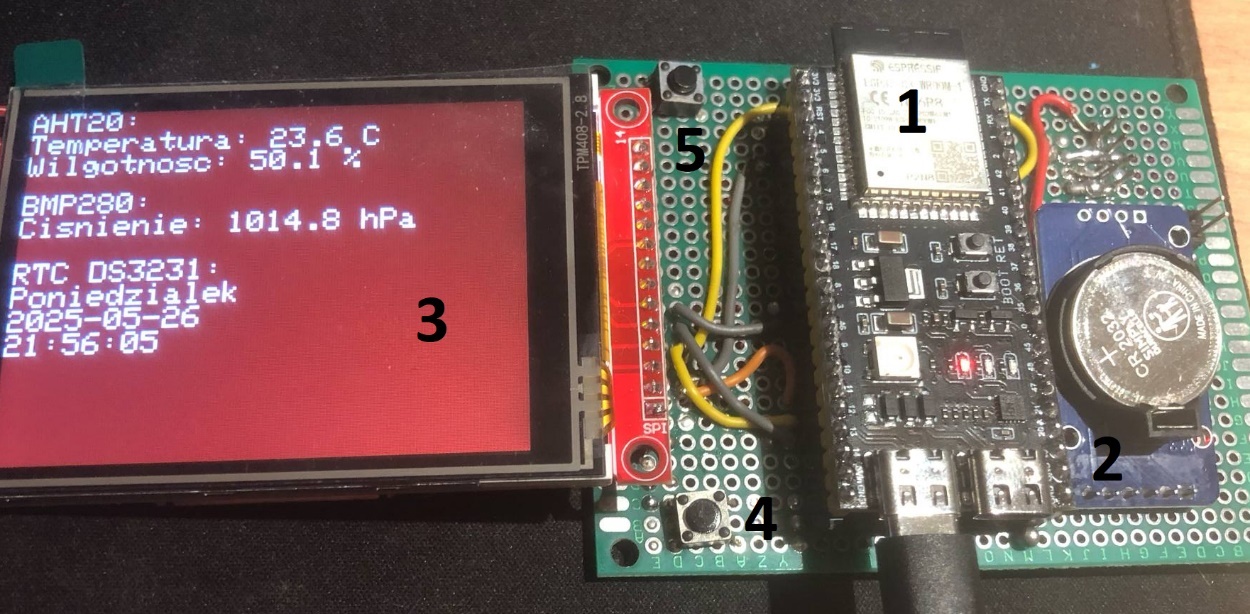
Rysunek 6 Algorytm programu przedstawiony przy pomocy schematu blokowego.

1. Implementacja programowa
2. #include <Adafruit\_GFX.h>      // Biblioteka graficzna
3. #include <Adafruit\_ILI9341.h>  // Biblioteka dla ILI9341
4. #include <Wire.h>              // Biblioteka I2C
5. #include "Adafruit\_AHTX0.h"    // Biblioteka dla AHT20
6. #include "Adafruit\_BMP280.h"   // Biblioteka dla BMP280
7. #include "RTClib.h"            // Biblioteka dla RTC DS3231
8. #include <string.h>
9. // Definicja pinów dla TFT
10. #define TFT\_CS   10
11. #define TFT\_DC   2
12. #define TFT\_MOSI 11
13. #define TFT\_SCK  12
14. #define TFT\_RST  -1  // RST podłączone do RESET ESP32
15. #define TFT\_LED  8   // Pin podświetlenia
16. // Piny I2C
17. #define I2C\_SDA 45
18. #define I2C\_SCL 48
19. // Piny przycisków
20. #define BUTTON\_NAV 36  // Przycisk nawigacji
21. #define BUTTON\_SET 37  // Przycisk ustawiania
22. // Inicjalizacja obiektów
23. Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC, TFT\_MOSI, TFT\_SCK, TFT\_RST);
24. Adafruit\_AHTX0 aht;
25. Adafruit\_BMP280 bmp;
26. RTC\_DS3231 rtc;
27. // Zmienne dla obsługi ustawiania czasu
28. bool settingMode = false;
29. int settingField = 0; // 0: rok, 1: miesiąc, 2: dzień, 3: godzina, 4: minuta
30. DateTime now;
31. // Poprzednie wartości tekstowe do porównania długości
32. char prevAhtTemp[48] = "";
33. char prevAhtHum[48] = "";
34. char prevBmpTemp[48] = "";
35. char prevBmpPress[48] = "";
36. char prevDayOfWeek[48] = "";
37. char prevDate[48] = "";
38. char prevTime[48] = "";
39. char prevSetMode[48] = "";
40. // Funkcja ustawiająca podświetlenie
41. void setBacklight(bool state) {
42. pinMode(TFT\_LED, OUTPUT);
43. digitalWrite(TFT\_LED, state ? HIGH : LOW);
44. }
45. // Funkcja czyszcząca linię, jeśli nowy tekst jest krótszy
46. void clearIfShorter(int x, int y, const char\* newText, char\* prevText) {
47. if (strlen(newText) < strlen(prevText)) {
48. // Czyszczenie obszaru linii (zakładamy szerokość 320 pikseli, wysokość 16 pikseli dla textSize=2)
49. tft.fillRect(x, y, 320, 16, ILI9341\_MAROON);
50. }
51. strcpy(prevText, newText); // Aktualizacja poprzedniego tekstu
52. }
53. // Funkcja wyświetlania daty i godziny
54. void displayDateTime() {
55. now = rtc.now();
56. char buffer[32];
58. // Dzień tygodnia
59. const char\* daysOfWeek[] = {"Niedziela", "Poniedzialek", "Wtorek", "Sroda",
60. "Czwartek", "Piatek", "Sobota"};
61. tft.setCursor(0, 108);
62. tft.println("RTC DS3231:");
64. strcpy(buffer, daysOfWeek[now.dayOfTheWeek()]);
65. clearIfShorter(0, 124, buffer, prevDayOfWeek);
66. tft.setCursor(0, 124);
67. tft.print(buffer);
68. tft.println("  ");
70. // Data
71. sprintf(buffer, "%04d-%02d-%02d", now.year(), now.month(), now.day());
72. clearIfShorter(0, 140, buffer, prevDate);
73. tft.setCursor(0, 140);
74. tft.print(buffer);
75. tft.println("  ");
77. // Godzina
78. sprintf(buffer, "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
79. clearIfShorter(0, 156, buffer, prevTime);
80. tft.setCursor(0, 156);
81. tft.print(buffer);
83. // Wskazanie trybu ustawiania
84. if (settingMode) {
85. const char\* fieldNames[] = {"Rok", "Miesiac", "Dzien", "Godzina", "Minuta"};
86. sprintf(buffer, " SET:%s", fieldNames[settingField]);
87. } else {
88. strcpy(buffer, "");
89. }
90. clearIfShorter(156, 156, buffer, prevSetMode);
91. tft.setCursor(156, 156);
92. tft.print(buffer);
93. }
94. void setup() {
95. // Inicjalizacja I2C
96. Wire.begin(I2C\_SDA, I2C\_SCL);
97. // Inicjalizacja przycisków
98. pinMode(BUTTON\_NAV, INPUT);
99. pinMode(BUTTON\_SET, INPUT);
100. // Inicjalizacja TFT
101. tft.begin();
102. setBacklight(true); // Włączenie podświetlenia
103. tft.setRotation(1); // Orientacja ekranu (0-3)
104. tft.fillScreen(ILI9341\_MAROON); // Początkowe czyszczenie ekranu
105. tft.setTextColor(ILI9341\_WHITE, ILI9341\_MAROON); // Kolor tekstu i tła
106. tft.setTextSize(2);
107. // Inicjalizacja AHT20
108. if (!aht.begin()) {
109. tft.setCursor(0, 0);
110. tft.println("AHT20 problem!");
111. while (1);
112. }
113. // Inicjalizacja BMP280
114. if (!bmp.begin(0x77)) {
115. tft.setCursor(0, 20);
116. tft.println("BMP280 problem!");
117. while (1);
118. }
119. // Inicjalizacja RTC DS3231
120. if (!rtc.begin()) {
121. tft.setCursor(0, 40);
122. tft.println("RTC problem!");
123. while (1);
124. }
125. if(rtc.lostPower()) {
126. rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_))); // Ustawienie RTC na czas kompilacji
127. }
128. // Ustawienia BMP280
129. bmp.setSampling(Adafruit\_BMP280::MODE\_NORMAL,     // Tryb pracy
130. Adafruit\_BMP280::SAMPLING\_X2,     // Oversampling temperatury
131. Adafruit\_BMP280::SAMPLING\_X16,    // Oversampling ciśnienia
132. Adafruit\_BMP280::FILTER\_X16,      // Filtr
133. Adafruit\_BMP280::STANDBY\_MS\_500); // Czas uśpienia
134. }
135. void loop() {
136. static unsigned long lastUpdate = 0;
137. static unsigned long lastNavPress = 0;
138. static unsigned long lastSetPress = 0;
139. unsigned long currentMillis = millis();
140. // Obsługa przycisku nawigacji (przełączanie pól lub trybu)
141. if (digitalRead(BUTTON\_NAV) == HIGH && currentMillis - lastNavPress > 100) {
142. lastNavPress = currentMillis;
143. if (!settingMode) {
144. settingMode = true;
145. settingField = 0;
146. } else {
147. settingField = (settingField + 1) % 5; // Przełącz między polami
148. }
149. }
150. // Obsługa przycisku ustawiania (zmiana wartości)
151. if (digitalRead(BUTTON\_SET) == HIGH && currentMillis - lastSetPress > 100) {
152. lastSetPress = currentMillis;
153. if (settingMode) {
154. now = rtc.now();
155. int year = now.year();
156. int month = now.month();
157. int day = now.day();
158. int hour = now.hour();
159. int minute = now.minute();
160. switch (settingField) {
161. case 0: year++; if (year > 2099) year = 2025; break;
162. case 1: month++; if (month > 12) month = 1; break;
163. case 2: day++; if (day > 31) day = 1; break;
164. case 3: hour++; if (hour > 23) hour = 0; break;
165. case 4: minute++; if (minute > 59) minute = 0; break;
166. }
167. rtc.adjust(DateTime(year, month, day, hour, minute, 0));
168. } else {
169. settingMode = true;
170. settingField = 0;
171. }
172. }
173. // Wyjście z trybu ustawiania po 10 sekundach bezczynności
174. if (settingMode && currentMillis - lastNavPress > 10000 && currentMillis - lastSetPress > 10000) {
175. settingMode = false;
176. }
178. char buffer[32];
179. // Odczyt z AHT20
180. sensors\_event\_t humidity, temp;
181. aht.getEvent(&humidity, &temp);
182. // Wyświetlanie danych AHT20
183. tft.setCursor(0, 0);
184. tft.println("AHT20:");
186. sprintf(buffer, "Temperatura: %.1f C", temp.temperature);
187. clearIfShorter(0, 16, buffer, prevAhtTemp);
188. tft.setCursor(0, 16);
189. tft.println(buffer);
191. sprintf(buffer, "Wilgotnosc: %.1f %%", humidity.relative\_humidity);
192. clearIfShorter(0, 32, buffer, prevAhtHum);
193. tft.setCursor(0, 32);
194. tft.println(buffer);
195. // Odczyt z BMP280
196. float temperature = bmp.readTemperature();
197. float pressure = bmp.readPressure() / 100.0F; // Konwersja Pa na hPa
198. // Wyświetlanie danych BMP280
199. tft.setCursor(0, 60);
200. tft.println("BMP280:");
202. sprintf(buffer, "Cisnienie: %.1f hPa", pressure);
203. clearIfShorter(0, 76, buffer, prevBmpPress);
204. tft.setCursor(0, 76);
205. tft.println(buffer);
206. // Wyświetlanie danych RTC
207. displayDateTime();
209. }

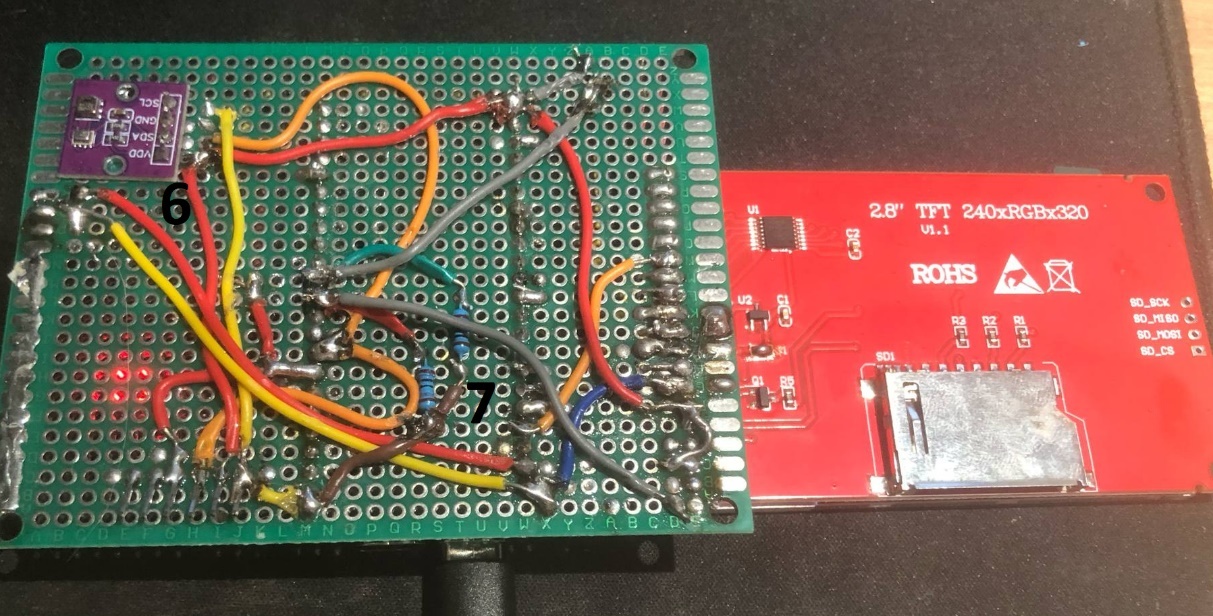
7. Realizacja praktyczna



Rysunek 7 Wyświetlanie danych z czujników oraz aktualnej daty.



Rysunek 8 1 - ESP32-S3, 2 - Moduł DS3231, 3 - Wyświetlacz TFT LCD, 4 - Przycisk do nawigacji po menu, 5 - Przycisk do zmiany parametrów RTC.



Rysunek 9 6 - Moduł AHT20 i BMP280, 7 - Rezystory pull-down 10kΩ do przycisków



Rysunek 10 Zmiana roku z 2025 na 2026 przy pomocy przycisków oraz automatyczna zmiana dnia tygodnia.



Rysunek 11 Zmiana miesiąca z maja na czerwiec przy pomocy przycisków oraz automatyczna zmiana dnia tygodnia.

Źródła:

[1] https://docs.espressif.com/projects/esp-idf/zh\_CN/v4.4/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html

[2] <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>

[3] https://www.aliexpress.com/item/1005004329890958.html

[4] https://asairsensors.com/wp-content/uploads/2021/09/Data-Sheet-AHT20-Humidity-and-Temperature-Sensor-ASAIR-V1.0.03.pdf

[5] <https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf>

[6] https://www.majju.pk/product/ds3231-rtc-module-ds3231-real-time-clock-module/

[7] https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf

[8] https://www.bidorbuy.co.za/item/562791530/2\_8\_Inch\_ILI9341\_240x320\_SPI\_TFT\_LCD\_Display\_Touch\_Panel\_SPI\_Serial\_Port\_Module.html

[9] https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf