System do monitorowania poziomu glukozy we krwi

inż. Mateusz Woźniak

Grudzień 2025

1 Wprowadzenie

Celem niniejszego projektu jest opracowanie urządzenia IoT do monitorowania poziomu glukozy we krwi. Założenia systemu:

- Centralnym komponentem urządzenia jest Arduino UNO
- Czujnikiem, który generuje sygnał o poziomie glukozy we krwi jest potencjometr (w rzeczywistości jest to prawdziwy czujnik)
- Urządzenie posiada wyświetlacz LCD do pokazywania bieżących i historycznych odczytów
- Pomiary są archiwizowane na karcie SD w pliku CSV.
- Buzer informuje o krytycznym poziomie glukozy we krwi.
- Przycisk powoduje przełączanie się między trybami wyświetlania w LCD (bieżące/archiwalne).

2 Realizacja projektu

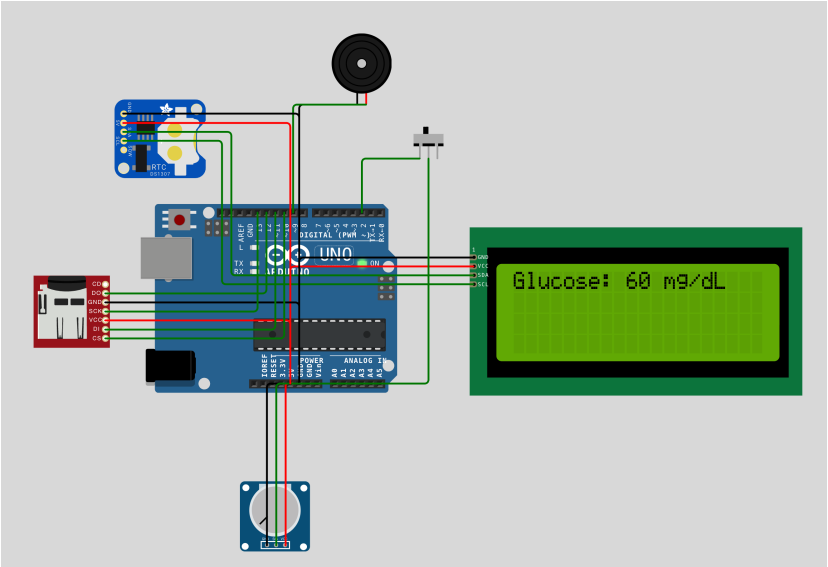
2.1 Schemat połączenia obwodu

Używając symulatora Wokwi utworzono schemat połączeń komponentów takich jak Arduino UNO, Buzzer, czytnik kard SD, wyświetlacz, potencjometr i zegar RTC.

Schemat przedstawiono na rys. 2.1.

Poniżej przedstawiono użyte protokoły komunikacji:

- Zegar RTC jest podłączony z użyciem protokołu I2C.
- Wyświetlacz LCD jest podłączony z użyciem protokołu I2C.
- Czytnik kart SD jest podłączony z użyciem protokołu SPI.



Rysunek 1: Schemat połączeń układu

- Potencjometr jest podłączony za pomocą wejścia anaologowego.
- Buzzer jest podłączony za pomocą wyjścia PWM.
- Przełącznik bistabilny jest podłączony za pomocą wejścia cyfrowego.

2.2 Implementacja oprogramowania

Oprogramowanie zostało napisane w środowisku C/Arduino.

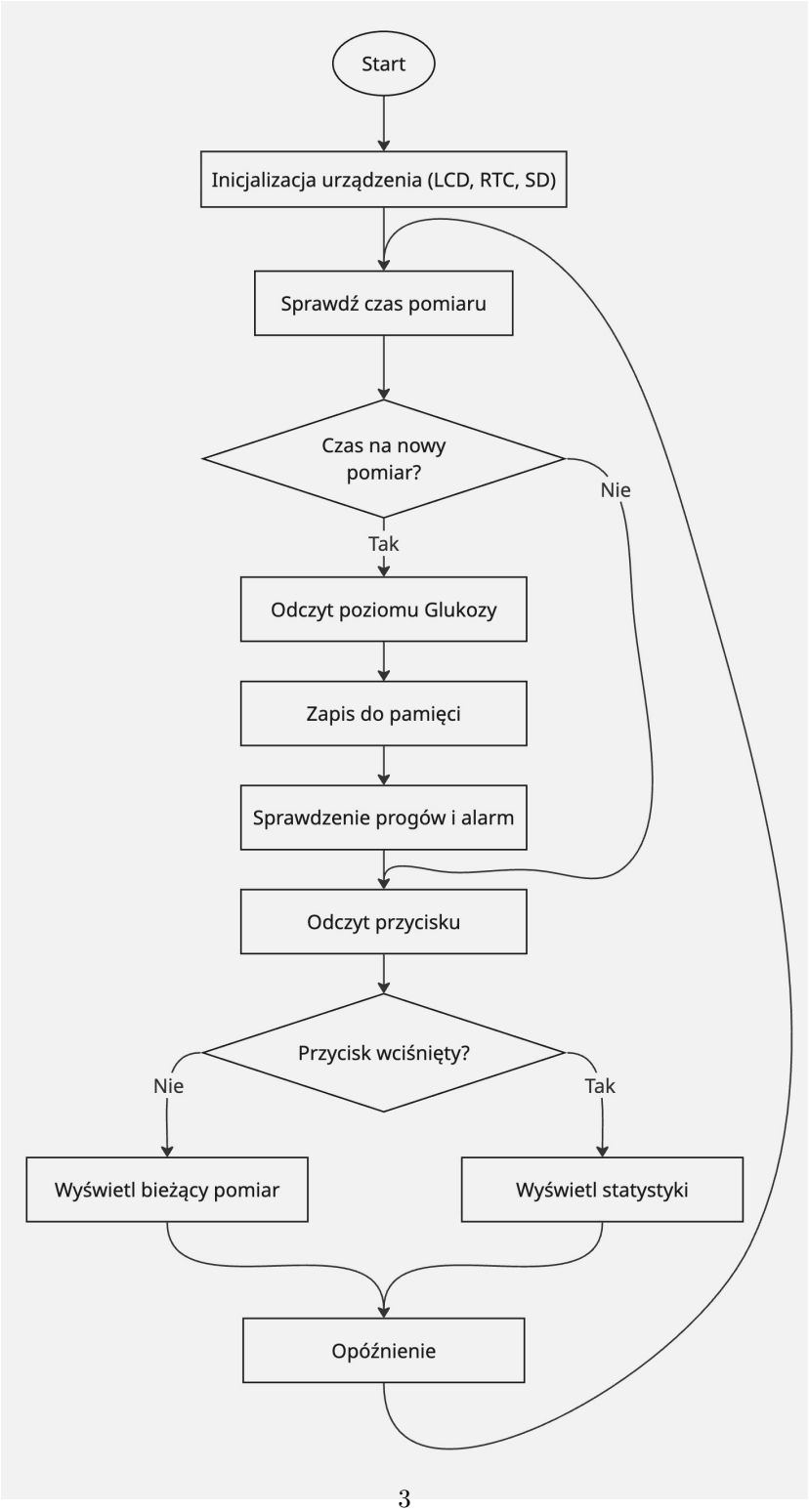
W programie istnieje pętla główna, która jest wykonywana z częstotliwością 100 Hz. Ma ona za zadanie reagować na sygnały wejścia oraz aktualizować wyświetlacz LCD. Pętla ta również sprawdza, czy w danej chwili system powinien pomiar. Interwał pomiaru jest ustawiony na 1 sekundę. Jeżeli warunek czasowy wykonania pomiaru zostanie spełniony, urządzenie wykona odczyt z wejścia analogowego, a następnie wykona przetworzenie odczytu. W ramach przetwarzania urządzenie zapisze odczyt do pliku CSV wraz z aktualną pieczęcią czasu Unix, sprawdzi, czy pomiar powinien spowodować uruchomienie alarmu (buzzer) oraz wyświetli wynik na ekranie LCD.

W przypadku, gdy użytkownik przełączy przycisk na tryb archiwalny, pętla główna będzie wyświetlać zagregowane pomiary na ekranie LCD (średnie czasowe).

Logikę działania systemu przedstawiono na rys. 2.2.

W listingu 1 przedstawiono kod źródłowy.

Listing 1: Kod źródłowy



Rysunek 2: Diagram blokowy działania systemu

```
1  // Glucose Meter
2  // Author: Mateusz Wozniak
3  #include <Wire.h>
4  #include <LiquidCrystal_I2C.h>
5  #include <RTClib.h>
6  #include <SD.h>
7
8  #define POT_PIN A0
9  #define BUTTON_PIN 2
10 #define BUZZER_PIN 9
11 #define SD_CS 10
12
13 const bool DEBUG = true;
14
15 const unsigned long LOOP_INTERVAL = 10;
16 const unsigned long MEAS_INTERVAL = 1000;
17
18 const unsigned long GLUCOSE_MIN = 60;
19 const unsigned long GLUCOSE_MAX = 250;
20
21 const unsigned long GLUCOSE_WARNING_THRESHOLD = 180;
22 const unsigned long GLUCOSE_CRITICAL_THRESHOLD = 200;
23
24 LiquidCrystal_I2C lcd(0x27, 16, 2);
25 RTC_DS1307 rtc;
26
27 unsigned long lastMeasurementTime = 0;
28 unsigned long lastMeasurementValue = 0;
29
30 int lastDisplayedGlucose = -1;
31
32 int readGlucose() {
33   int raw = analogRead(POT_PIN);
34   int glucose = map(raw, 0, 1023, GLUCOSE_MIN,
35       GLUCOSE_MAX); // mg/dL
36   return glucose;
37 }
38
39 void printDataFile() {
40   Serial.println("---");
41   File dataFile = SD.open("glucose.csv");
42   while (dataFile.available()) {
43     Serial.write(dataFile.read());
44   }
45   dataFile.close();
46 }
```

```
47
48 void archive(int glucose) {
49   DateTime now = rtc.now();
50
51   File dataFile = SD.open("glucose.csv", FILE_WRITE);
52   dataFile.print(now.unixtime());
53   dataFile.print(",");
54   dataFile.println(glucose);
55   dataFile.close();
56 }
57
58 void printCurrentMeasurement(int glucose) {
59   if (glucose != lastDisplayedGlucose) {
60     lcd.clear();
61     lcd.setCursor(0, 0);
62     lcd.print("Glucose: ");
63     lcd.print(glucose);
64     lcd.print(" mg/dL");
65     lastDisplayedGlucose = glucose;
66
67     if (glucose > GLUCOSE_CRITICAL_THRESHOLD) {
68       lcd.setCursor(0, 1);
69       lcd.print("Critical: ");
70       lcd.setCursor(0, 2);
71       lcd.print("Glucose > ");
72       lcd.print(GLUCOSE_CRITICAL_THRESHOLD);
73       lcd.print(" mg/dL");
74     } else if (glucose > GLUCOSE_WARNING_THRESHOLD) {
75       lcd.setCursor(0, 1);
76       lcd.print("Warning: ");
77       lcd.setCursor(0, 2);
78       lcd.print("Glucose > ");
79       lcd.print(GLUCOSE_WARNING_THRESHOLD);
80       lcd.print(" mg/dL");
81     } else {
82       lcd.setCursor(0, 1);
83       int blocks = map(glucose, GLUCOSE_MIN,
84         GLUCOSE_WARNING_THRESHOLD, 0, 7);
85       for(int i = 0; i < blocks; i++) {
86         lcd.write(byte(255));
87       }
88     }
89   }
90 }
91
92 void printStatistics() {
```

```
93  lastDisplayedGlucose = -1;
94  File dataFile = SD.open("glucose.csv");
95  if (!dataFile) {
96    lcd.clear();
97    lcd.print("No data found");
98    return;
99  }
100
101 unsigned long nowUnix = rtc.now().unixtime();
102
103 unsigned long lastMinuteSum = 0, lastMinuteCount = 0;
104 unsigned long lastHourSum = 0, lastHourCount = 0;
105
106 while (dataFile.available()) {
107   String line = dataFile.readStringUntil('\n');
108   int commaIndex = line.indexOf(',');
109   if (commaIndex == -1) continue;
110
111   unsigned long timestamp = line.substring(0,
112       commaIndex).toInt();
113   int glucose = line.substring(commaIndex + 1).toInt();
114
115   unsigned long age = nowUnix - timestamp;
116
117   // Last minute: 60 seconds
118   if (age <= 60UL) {
119     lastMinuteSum += glucose;
120     lastMinuteCount++;
121   }
122   // Last hour: 3600 seconds
123   if (age <= 3600UL) {
124     lastHourSum += glucose;
125     lastHourCount++;
126   }
127 }
128 dataFile.close();
129
130 lcd.clear();
131 lcd.setCursor(0, 0);
132 if(lastMinuteCount) {
133   lcd.print("1min avg:");
134   lcd.print(lastMinuteSum / lastMinuteCount);
135 } else {
136   lcd.print("1min avg:N/A");
137 }
138
```

```
139   lcd.setCursor(0, 1);
140   if(lastHourCount) {
141     lcd.print("1h avg:");
142     lcd.print(lastHourSum / lastHourCount);
143   } else {
144     lcd.print("1h avg:N/A");
145   }
146 }
147
148
149 void draw(int glucose) {
150   int buttonState = digitalRead(BUTTON_PIN);
151   if(buttonState == LOW) {
152     printCurrentMeasurement(glucose);
153   } else {
154     printStatistics();
155   }
156 }
157
158 void setup() {
159   Serial.begin(115200);
160   pinMode(BUTTON_PIN, INPUT);
161
162   lcd.init();
163   lcd.backlight();
164   lcd.print("Glucose Meter");
165   lcd.setCursor(0, 1);
166   lcd.print("Mateusz Wozniak");
167
168   if (!rtc.begin()) {
169     Serial.println("RTC setup failed!");
170     lcd.setCursor(0, 1);
171     lcd.print("RTC error!");
172   }
173   if (!SD.begin(SD_CS)) {
174     Serial.println("SD Card setup failed!");
175     lcd.setCursor(0, 1);
176     lcd.print("SD error!");
177   }
178   Serial.println("Initialized");
179
180   delay(100);
181   lcd.clear();
182 }
183
184 void checkAlert(int glucose) {
```

```
185   if (glucose > GLUCOSE_CRITICAL_THRESHOLD) {
186     tone(BUZZER_PIN, 1000);
187   } else if (glucose > GLUCOSE_WARNING_THRESHOLD) {
188     tone(BUZZER_PIN, 600, 100);
189   } else {
190     noTone(BUZZER_PIN);
191   }
192 }
193
194 void loop() {
195   unsigned long now = millis();
196
197   if (now - lastMeasurementTime >= MEAS_INTERVAL) {
198     // Fetch
199     int glucose = readGlucose();
200     lastMeasurementTime = now;
201     lastMeasurementValue = glucose;
202     // Process
203     archive(glucose);
204     checkAlert(glucose);
205     if(DEBUG) {
206       printDataFile();
207     }
208   }
209   // Display
210   draw(lastMeasurementValue);
211   delay(LOOP_INTERVAL);
212 }
```

3 Działanie i testy systemu

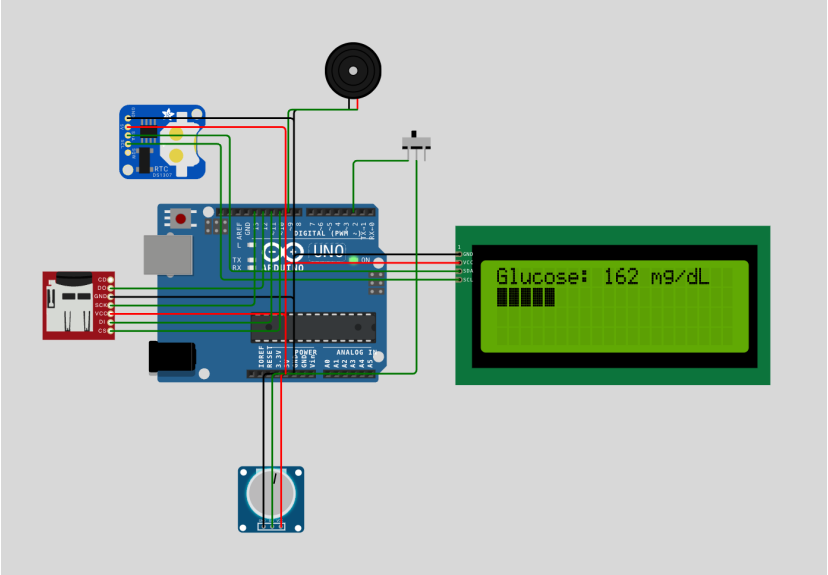
Urządzenie było przede wszystkim testowane manualnie wraz z rozwojem funkcjonalności projektu.

Stwierdzono, że urządzenie działa w pełni zgodnie z wyznaczonymi założeniami, a każda funkcjonalność działa bez błędów.

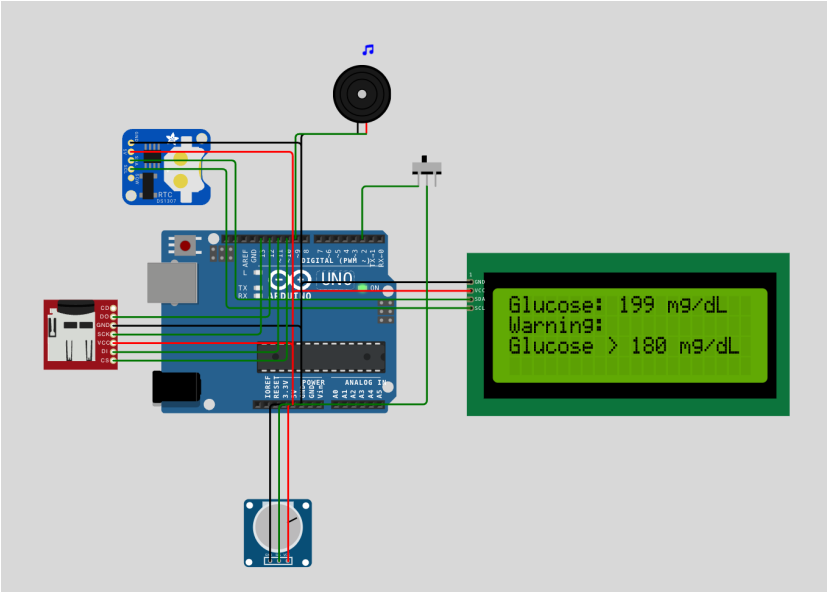
Na załączonych zrzutach ekranu przedstawiono stan testowania funkcji.

4 Wnioski

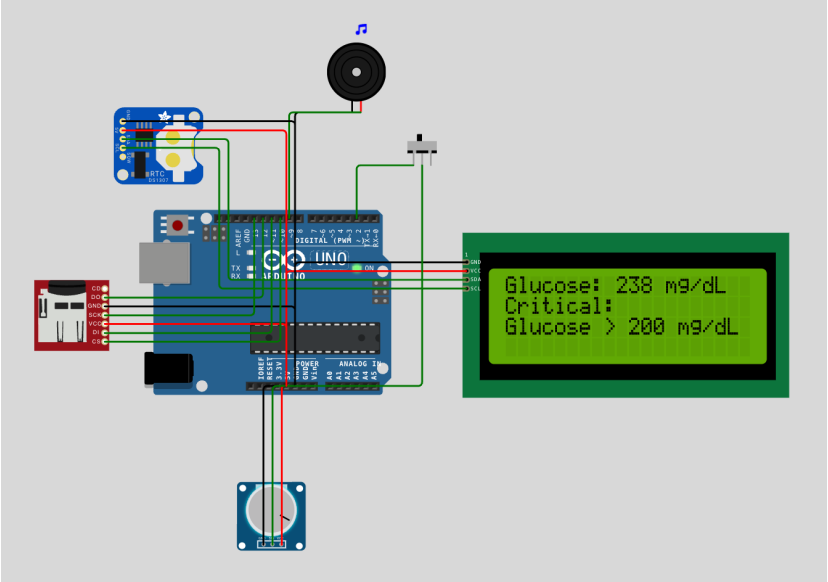
Opracowanie systemu do monitorowania poziomu glukozy we krwi stanowiło dobrą okazję do nauczenia się budowania kompletnych urządzeń wbudowanych. W szczególności trudne było dobre zaprojektowanie pętli, tak by odczyt był wykonywany z częstotliwości 1Hz, a jednocześnie wyświetlacz LCD był odświeżany z częstotliwością 100Hz - ten kluczowy element oprogramowania powinien zostać



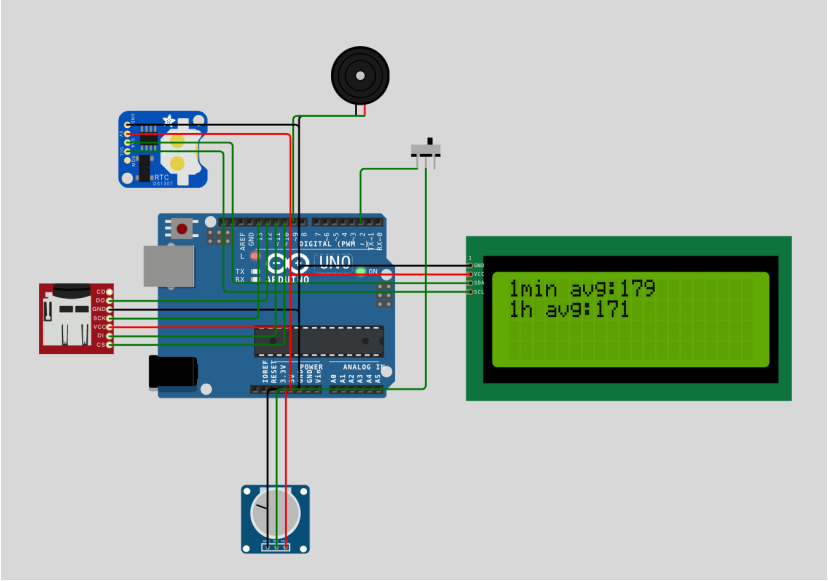
Rysunek 3: Podstawowy tryb wyświetlania wraz z paskiem informacyjnym



Rysunek 4: Urządzenie w stanie ostrzegawczym - buzzer "pika"



Rysunek 5: Urządzenie w stanie krytycznym - buzzer wydaje dźwięk ciągły



Rysunek 6: Urządzenie w trybie archiwalnym - LCD pokazuje średnie czasowe

zaplanowany na samym początku implementacji kodu, a nie w trakcie. Pomimo małych przeszkód, projekt został wykonany w pełni, zgodnie z założeniami początkowymi.

5 Możliwe ścieżki rozwoju projektu

W trakcie budowy projektu pojawiły się pomysły potencjalnego rozszerzenia projektu o:

- Moduł Bluetooth HC-06 - umożliwiałby on strumieniowanie odczytów do innych komponentów np. smartfon
- Moduł GSM - umożliwiałby on strumieniowanie odczytów do sieci internetu komórkowego

Opisane powyżej ścieżki rozwoju mogłyby pozwolić na sprawne przekazywanie informacji lekarzowi w czasie rzeczywistym, co pozwoliłoby na jeszcze lepsze efekty leczenia chorób.