Jakub Gubar nr albumu 29205

14.01.2025r.

Sprawozdanie

Internet rzeczy: PREZENTACJA DANYCH W OPARCIU O INSTRUKCJE WARUNKOWE I PĘTLE

1. Cel ćwiczenia - Celem ćwiczenia było zapoznanie z zasadą działania wbudowanej matrycy LED, instrukcjami warunkowymi i pętlami.

2. Przebieg ćwiczenia.

2.1. Implementacja licznika – prezentacja w postaci cyklicznego załączania w wyłączania odpowiednich diod macierzy led.

Postępując zgodnie z instrukcją, wykorzystaliśmy kod w niej zawarty i stworzyliśmy program, którego zadaniem było odliczanie od 0 do 5, stan licznika zmieniał się co sekundę. Następnie dokonaliśmy jego modyfikacji w celu dodania funkcji odpowiadających za prezentację liczb od 6 do 9. Poniżej rezultat tychże modyfikacji.

```
#include "Arduino_LED_Matrix.h"
ArduinoLEDMatrix matrix;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  matrix.begin();

}

uint8_t frame[8][12] = {
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

void diode_off()
{
  for (int index_1 = 0; index_1 <= 7; index_1++)
  {
      for (int index_2 = 0; index_2 <= 7; index_2++)
      {
        frame[index_1][index_2] = 0;
      }

  }
}
```

```
void zero_number(){
  diode_off();
  frame[0][0] = 1;
  frame[0][1] = 1;
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
  frame[7][1] = 1;
  frame[7][0] = 1;
  frame[6][0] = 1;
  frame[5][0] = 1;
  frame[4][0] = 1;
  frame[3][0] = 1;
  frame[2][0] = 1;
  frame[1][0] = 1;
}

void one_number(){
  diode_off();
  frame[2][0] = 1;
  frame[1][1] = 1;
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
}
```

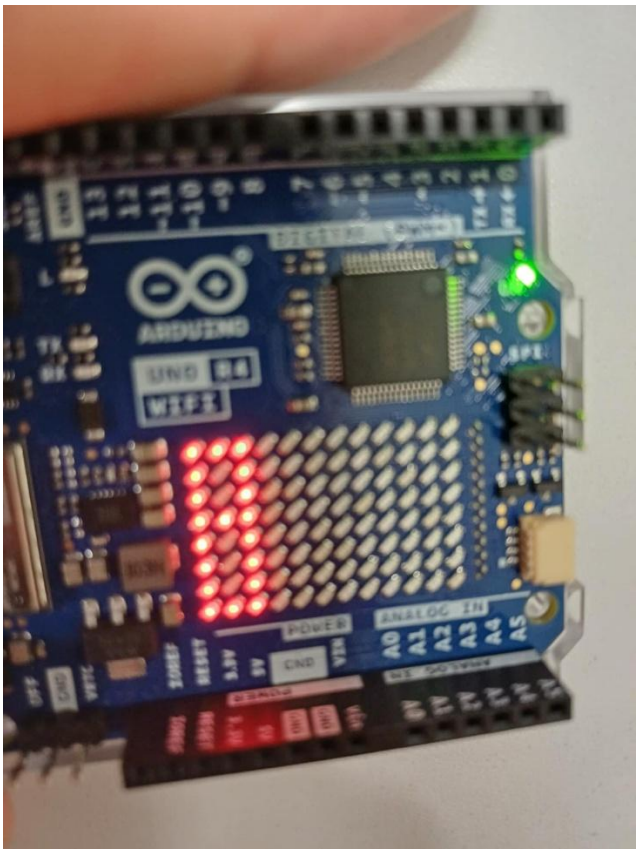
```
void two_number(){
  diode_off();
  frame[0][0] = 1;
  frame[0][1] = 1;
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[3][1] = 1;
  frame[3][0] = 1;
  frame[4][0] = 1;
  frame[5][0] = 1;
  frame[6][0] = 1;
  frame[7][0] = 1;
  frame[7][1] = 1;
  frame[7][2] = 1;
}

void three_number(){
  diode_off();
  frame[0][0] = 1;
  frame[0][1] = 1;
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[3][1] = 1;
  frame[3][0] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
  frame[7][1] = 1;
  frame[7][0] = 1;
}
```

```
void four_number(){
  diode_off();
  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
  frame[3][0] = 1;
  frame[3][1] = 1;
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
}

void five_number(){
  diode_off();
  frame[0][2] = 1;
  frame[0][1] = 1;
  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
  frame[3][0] = 1;
  frame[3][1] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
  frame[7][1] = 1;
  frame[7][0] = 1;
}
```

```
void six_number(){
  diode_off();
  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
  frame[3][0] = 1;
  frame[4][0] = 1;
  frame[5][0] = 1;
  frame[6][0] = 1;
  frame[7][0] = 1;

  frame[0][1] = 1;
  frame[4][1] = 1;
  frame[7][1] = 1;

  frame[0][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
}

void seven_number(){
  diode_off();
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;

  frame[0][1] = 1;

  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
}
```

```
void eight_number(){
  diode_off();
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;

  frame[0][1] = 1;
  frame[3][1] = 1;
  frame[7][1] = 1;

  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
  frame[3][0] = 1;
  frame[4][0] = 1;
  frame[5][0] = 1;
  frame[6][0] = 1;
  frame[7][0] = 1;
}

void nine_number(){
  diode_off();
  frame[0][2] = 1;
  frame[1][2] = 1;
  frame[2][2] = 1;
  frame[3][2] = 1;
  frame[4][2] = 1;
  frame[5][2] = 1;
  frame[6][2] = 1;
  frame[7][2] = 1;
  frame[0][1] = 1;
  frame[3][1] = 1;
  frame[7][1] = 1;
  frame[0][0] = 1;
  frame[1][0] = 1;
  frame[2][0] = 1;
  frame[3][0] = 1;
  frame[6][0] = 1;
  frame[7][0] = 1;
}
```

```
int cnt = 0;

void loop() {
  // put your main code here, to run repeatedly:
if (cnt==0)
{
  zero_number();
}
else if (cnt == 1)
{
  one_number();
}
else if (cnt == 2)
{
  two_number();
}
else if (cnt == 3)
{
  three_number();
}
else if (cnt == 4)
{
  four_number();
}
else if (cnt == 5)
{
  five_number();
}
else if (cnt == 6)
{
  six_number();
}
else if (cnt == 7)
{
  seven_number();
}
else if (cnt == 8)
{
  eight_number();
}
else
{
  nine_number();
}

if(cnt == 9)
{
  cnt = 0;
}
else
{
  cnt = cnt + 1;
}

matrix.renderBitmap(frame, 8, 12);
delay(1000);
}
```



2.2. Implementacja licznika – prezentacja w postaci animacji „rysowania” liczb.

Korzystając z kodu użytego w kolejnym zadaniu, dokonaliśmy jego modyfikacji w celu dodania funkcji odpowiadających za rysowanie liczb od 6 do 9.

Poniżej przykład funkcji określający sposób rysowania danej liczby.

```
34  void zero_number(int cnt_led_array[19]){
35    diode_off();
36    //int cnt_led_array_buff[18] =cnt_led_array;
37    frame[0][0] = cnt_led_array[0];
38    frame[0][1] = cnt_led_array[1];
39    frame[0][2] = cnt_led_array[2];
40    frame[1][2] = cnt_led_array[3];
41    frame[2][2] = cnt_led_array[4];
42    frame[3][2] = cnt_led_array[5];
43    frame[4][2] = cnt_led_array[6];
44    frame[5][2] = cnt_led_array[7];
45    frame[6][2] = cnt_led_array[8];
46    frame[7][2] = cnt_led_array[9];
47    frame[7][1] = cnt_led_array[10];
48    frame[7][0] = cnt_led_array[11];
49    frame[6][0] = cnt_led_array[12];
50    frame[5][0] = cnt_led_array[13];
51    frame[4][0] = cnt_led_array[14];
52    frame[3][0] = cnt_led_array[15];
53    frame[2][0] = cnt_led_array[16];
54    frame[1][0] = cnt_led_array[17];
55  }
```

Posługując się danym przykładem stworzyliśmy odpowiednio funkcje **void** określające liczby od 6 do 9.

```
142 void six_number(int cnt_led_array[19]){
143   diode_off();
144   frame[0][0] = cnt_led_array[13];
145   frame[1][0] = cnt_led_array[12];
146   frame[2][0] = cnt_led_array[11];
147   frame[3][0] = cnt_led_array[10];
148   frame[4][0] = cnt_led_array[9];
149   frame[5][0] = cnt_led_array[8];
150   frame[6][0] = cnt_led_array[7];
151   frame[7][0] = cnt_led_array[6];
152
153   frame[0][1] = cnt_led_array[14];
154   frame[4][1] = cnt_led_array[0];
155   frame[7][1] = cnt_led_array[5];
156
157   frame[0][2] = cnt_led_array[14];
158   frame[4][2] = cnt_led_array[1];
159   frame[5][2] = cnt_led_array[2];
160   frame[6][2] = cnt_led_array[3];
161   frame[7][2] = cnt_led_array[4];
162 }
```

```
164 void seven_number(int cnt_led_array[19]){
165   diode_off();
166   frame[0][2] = cnt_led_array[4];
167   frame[1][2] = cnt_led_array[5];
168   frame[2][2] = cnt_led_array[6];
169   frame[3][2] = cnt_led_array[7];
170   frame[4][2] = cnt_led_array[8];
171   frame[5][2] = cnt_led_array[9];
172   frame[6][2] = cnt_led_array[10];
173   frame[7][2] = cnt_led_array[11];
174
175   frame[0][1] = cnt_led_array[3];
176
177   frame[0][0] = cnt_led_array[2];
178   frame[1][0] = cnt_led_array[1];
179   frame[2][0] = cnt_led_array[0];
180 }
```

```
void eight_number(int cnt_led_array[19]){
  diode_off();

  frame[0][0] = cnt_led_array[0];
  frame[0][1] = cnt_led_array[1];
  frame[0][2] = cnt_led_array[2];
  frame[1][2] = cnt_led_array[3];
  frame[2][2] = cnt_led_array[4];
  frame[3][2] = cnt_led_array[5];
  frame[4][2] = cnt_led_array[6];
  frame[5][2] = cnt_led_array[7];
  frame[6][2] = cnt_led_array[8];
  frame[7][2] = cnt_led_array[9];
  frame[7][1] = cnt_led_array[10];
  frame[7][0] = cnt_led_array[11];
  frame[6][0] = cnt_led_array[12];
  frame[5][0] = cnt_led_array[13];
  frame[4][0] = cnt_led_array[14];
  frame[3][0] = cnt_led_array[15];
  frame[3][1] = cnt_led_array[16];
  frame[2][0] = cnt_led_array[17];
  frame[1][0] = cnt_led_array[18];
```

```
void nine_number(int cnt_led_array[19]){
  diode_off();
  frame[0][2] = cnt_led_array[10];
  frame[1][2] = cnt_led_array[9];
  frame[2][2] = cnt_led_array[8];
  frame[3][2] = cnt_led_array[7];
  frame[4][2] = cnt_led_array[6];
  frame[5][2] = cnt_led_array[5];
  frame[6][2] = cnt_led_array[4];
  frame[7][2] = cnt_led_array[3];

  frame[0][1] = cnt_led_array[11];
  frame[3][1] = cnt_led_array[16];
  frame[7][1] = cnt_led_array[2];

  frame[0][0] = cnt_led_array[12];
  frame[1][0] = cnt_led_array[13];
  frame[2][0] = cnt_led_array[14];
  frame[3][0] = cnt_led_array[15];

  frame[6][0] = cnt_led_array[0];
  frame[7][0] = cnt_led_array[1];
}
```

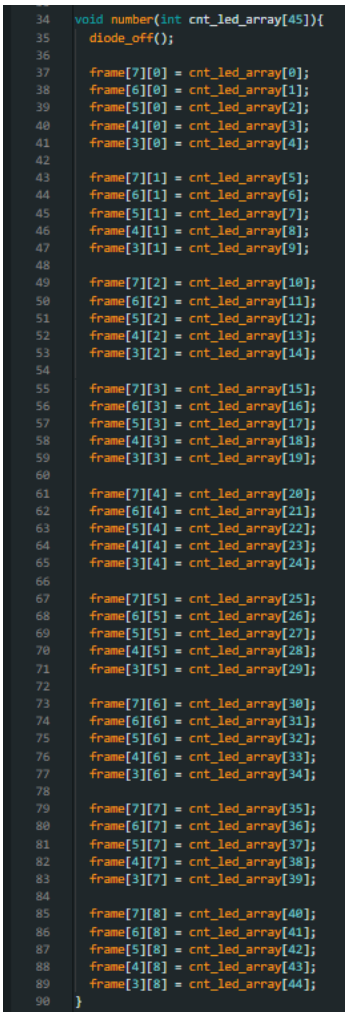
W celu wywołania funkcji wykorzystaliśmy warunek **else if** i pętlę **for** tak jak w przykładzie poniżej.

```
else if (cnt == 8)
{
  for (int index=0; index<=18; index++)
  {
    cnt_led_array[index] = 1;
    eight_number(cnt_led_array);
    matrix.renderBitmap(frame, 8, 12);
    delay(50);
  }
  delay(1000);
  cnt_led_array[0] = 0;
  cnt_led_array[1] = 0;
  cnt_led_array[2] = 0;
  cnt_led_array[3] = 0;
  cnt_led_array[4] = 0;
  cnt_led_array[5] = 0;
  cnt_led_array[6] = 0;
  cnt_led_array[7] = 0;
  cnt_led_array[8] = 0;
  cnt_led_array[9] = 0;
  cnt_led_array[10] = 0;
  cnt_led_array[11] = 0;
  cnt_led_array[12] = 0;
  cnt_led_array[13] = 0;
  cnt_led_array[14] = 0;
  cnt_led_array[15] = 0;
  cnt_led_array[16] = 0;
  cnt_led_array[17] = 0;
  cnt_led_array[18] = 0;
  diode_off();
  matrix.renderBitmap(frame, 8, 12);

}
```

2.3. Implementacja licznika – prezentacja w postaci animacji zwiększania rozmiaru liczb.

Następnie korzystając z kodu umieszczonego w zadaniu 3, do jego funkcji odliczającej liczby od 0 do 5 w postaci animacji dodaliśmy animacje dla liczb od 3 do 5.

W tej części wykorzystaliśmy zadeklarowaną funkcję **void number**, która pobierała dane z 45 elementowej tablicy wejściowej.



```
void number(int cnt_led_array[45]){
  diode_off();

  frame[7][0] = cnt_led_array[0];
  frame[6][0] = cnt_led_array[1];
  frame[5][0] = cnt_led_array[2];
  frame[4][0] = cnt_led_array[3];
  frame[3][0] = cnt_led_array[4];

  frame[7][1] = cnt_led_array[5];
  frame[6][1] = cnt_led_array[6];
  frame[5][1] = cnt_led_array[7];
  frame[4][1] = cnt_led_array[8];
  frame[3][1] = cnt_led_array[9];

  frame[7][2] = cnt_led_array[10];
  frame[6][2] = cnt_led_array[11];
  frame[5][2] = cnt_led_array[12];
  frame[4][2] = cnt_led_array[13];
  frame[3][2] = cnt_led_array[14];

  frame[7][3] = cnt_led_array[15];
  frame[6][3] = cnt_led_array[16];
  frame[5][3] = cnt_led_array[17];
  frame[4][3] = cnt_led_array[18];
  frame[3][3] = cnt_led_array[19];

  frame[7][4] = cnt_led_array[20];
  frame[6][4] = cnt_led_array[21];
  frame[5][4] = cnt_led_array[22];
  frame[4][4] = cnt_led_array[23];
  frame[3][4] = cnt_led_array[24];

  frame[7][5] = cnt_led_array[25];
  frame[6][5] = cnt_led_array[26];
  frame[5][5] = cnt_led_array[27];
  frame[4][5] = cnt_led_array[28];
  frame[3][5] = cnt_led_array[29];

  frame[7][6] = cnt_led_array[30];
  frame[6][6] = cnt_led_array[31];
  frame[5][6] = cnt_led_array[32];
  frame[4][6] = cnt_led_array[33];
  frame[3][6] = cnt_led_array[34];

  frame[7][7] = cnt_led_array[35];
  frame[6][7] = cnt_led_array[36];
  frame[5][7] = cnt_led_array[37];
  frame[4][7] = cnt_led_array[38];
  frame[3][7] = cnt_led_array[39];

  frame[7][8] = cnt_led_array[40];
  frame[6][8] = cnt_led_array[41];
  frame[5][8] = cnt_led_array[42];
  frame[4][8] = cnt_led_array[43];
  frame[3][8] = cnt_led_array[44];
}
```

Na podstawie poniższych tablic opracowaliśmy odpowiednie warianty dla brakujących liczb.

```
int zero_size_1[45] = {1,1,1,0,0,
                       1,0,1,0,0,
                       1,0,1,0,0,
                       1,0,1,0,0,
                       1,1,1,0,0,
                       0,0,0,0,0,
                       0,0,0,0,0,
                       0,0,0,0,0,
                       0,0,0,0,0};
int zero_size_2[45] = {1,1,1,1,0,
                       1,0,0,1,0,
                       1,0,0,1,0,
                       1,0,0,1,0,
                       1,0,0,1,0,
                       1,0,0,1,0,
                       1,1,1,1,0,
                       0,0,0,0,0,
                       0,0,0,0,0};
int zero_size_3[45] = {1,1,1,1,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,0,0,0,1,
                       1,1,1,1,1};
```

Poniżej przedstawienie brakującego kodu.

```
int three_size_1[45] = {
  1,1,1,0,0,  // wiersz 0
  0,0,1,0,0,  // wiersz 1
  1,1,1,0,0,  // wiersz 2
  0,0,1,0,0,  // wiersz 3
  1,1,1,0,0,  // wiersz 4
  0,0,0,0,0,  // wiersz 5
  0,0,0,0,0,  // wiersz 6
  0,0,0,0,0,  // wiersz 7
  0,0,0,0,0   // wiersz 8
};

int three_size_2[45] = {
  1,1,1,1,1,  // wiersz 0
  0,0,0,0,1,  // wiersz 1
  0,0,0,0,1,  // wiersz 2
  1,1,1,1,1,  // wiersz 3
  0,0,0,0,1,  // wiersz 4
  0,0,0,0,1,  // wiersz 5
  1,1,1,1,1,  // wiersz 6
  0,0,0,0,0,  // wiersz 7
  0,0,0,0,0   // wiersz 8
};

int three_size_3[45] = {
  1,1,1,1,1,  // wiersz 0
  0,0,0,0,1,  // wiersz 1
  0,0,0,0,1,  // wiersz 2
  0,0,0,0,1,  // wiersz 3
  1,1,1,1,1,  // wiersz 4
  0,0,0,0,1,  // wiersz 5
  0,0,0,0,1,  // wiersz 6
  0,0,0,0,1,  // wiersz 7
  1,1,1,1,1   // wiersz 8
};
```

```
int four_size_2[45] = {
  1,0,0,1,0,
  1,0,0,1,0,
  1,0,0,1,0,
  1,1,1,1,1,
  0,0,0,1,0,
  0,0,0,1,0,
  0,0,0,1,0,
  0,0,0,0,0,
  0,0,0,0,0
};
int four_size_3[45] = {
  1,0,0,0,1,
  1,0,0,0,1,
  1,0,0,0,1,
  1,0,0,0,1,
  1,1,1,1,1,
  0,0,0,0,1,
  0,0,0,0,1,
  0,0,0,0,1,
  0,0,0,0,1
};
```

```
int five_size_1[45] = {
  1,1,1,0,0,
  1,0,0,0,0,
  1,1,1,0,0,
  0,0,1,0,0,
  1,1,1,0,0,
  0,0,0,0,0,
  0,0,0,0,0,
  0,0,0,0,0,
  0,0,0,0,0
};
int five_size_2[45] = {
  1,1,1,1,0,
  1,0,0,0,0,
  1,1,1,1,0,
  0,0,0,1,0,
  0,0,0,1,0,
  0,0,0,1,0,
  1,1,1,1,0,
  0,0,0,0,0,
  0,0,0,0,0
};
int five_size_3[45] = {
  1,1,1,1,1,
  1,0,0,0,0,
  1,1,1,1,1,
  0,0,0,0,1,
  0,0,0,0,1,
  0,0,0,0,1,
  1,1,1,1,1,
  0,0,0,0,0,
  0,0,0,0,0
};
```

3. Wnioski.

- Testowanie programu pokazało, że instrukcje warunkowe są kluczowe dla poprawnego przełączania między różnymi stanami.

- Brak instrukcji **break** w strukturze **switch case** może prowadzić do niepożądanych wyników, co zostało zweryfikowane podczas testów.

- Zastosowanie animacji oraz różnorodnych rozmiarów cyfr wymagało odpowiedniego zarządzania tablicami oraz opóźnieniami (funkcja delay), co pozwoliło na płynne przejścia.