

# 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.

```
1  #include "Arduino_LED_Matrix.h"
2  ArduinoLEDMatrix matrix;
3
4  void setup() {
5      // put your setup code here, to run once:
6      Serial.begin(115200);
7      matrix.begin();
8
9  }
10
11  uint8_t frame[8][12] = {
12      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
13      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
14      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
15      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
16      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
17      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
18      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
19      { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
20  };
21
22  void diode_off()
23  {
24      for (int index_1 = 0; index_1 <= 7; index_1++)
25      {
26          for (int index_2 = 0; index_2 <= 7; index_2++)
27          {
28              frame[index_1][index_2] = 0;
29          }
30      }
31  }
32
33
34  void zero_number(){
35      diode_off();
36      frame[0][0] = 1;
37      frame[0][1] = 1;
38      frame[0][2] = 1;
39      frame[1][2] = 1;
40      frame[2][2] = 1;
41      frame[3][2] = 1;
42      frame[4][2] = 1;
43      frame[5][2] = 1;
44      frame[6][2] = 1;
45      frame[7][2] = 1;
46      frame[7][1] = 1;
47      frame[7][0] = 1;
48      frame[6][0] = 1;
49      frame[5][0] = 1;
50      frame[4][0] = 1;
51      frame[3][0] = 1;
52      frame[2][0] = 1;
53      frame[1][0] = 1;
54  }
55
56  void one_number(){
57      diode_off();
58      frame[2][0] = 1;
59      frame[1][1] = 1;
60      frame[0][2] = 1;
61      frame[1][2] = 1;
62      frame[2][2] = 1;
63      frame[3][2] = 1;
64      frame[4][2] = 1;
65      frame[5][2] = 1;
66      frame[6][2] = 1;
67      frame[7][2] = 1;
68  }
```

```

70 void two_number(){
71     diode_off();
72     frame[0][0] = 1;
73     frame[0][1] = 1;
74     frame[0][2] = 1;
75     frame[1][2] = 1;
76     frame[2][2] = 1;
77     frame[3][2] = 1;
78     frame[3][1] = 1;
79     frame[3][0] = 1;
80     frame[4][0] = 1;
81     frame[5][0] = 1;
82     frame[6][0] = 1;
83     frame[7][0] = 1;
84     frame[7][1] = 1;
85     frame[7][2] = 1;
86 }
87
88 void three_number(){
89     diode_off();
90     frame[0][0] = 1;
91     frame[0][1] = 1;
92     frame[0][2] = 1;
93     frame[1][2] = 1;
94     frame[2][2] = 1;
95     frame[3][2] = 1;
96     frame[3][1] = 1;
97     frame[3][0] = 1;
98     frame[4][2] = 1;
99     frame[5][2] = 1;
100    frame[6][2] = 1;
101    frame[7][2] = 1;
102    frame[7][1] = 1;
103    frame[7][0] = 1;
104 }

106 void four_number(){
107     diode_off();
108     frame[0][0] = 1;
109     frame[1][0] = 1;
110     frame[2][0] = 1;
111     frame[3][0] = 1;
112     frame[3][1] = 1;
113     frame[0][2] = 1;
114     frame[1][2] = 1;
115     frame[2][2] = 1;
116     frame[3][2] = 1;
117     frame[4][2] = 1;
118     frame[5][2] = 1;
119     frame[6][2] = 1;
120     frame[7][2] = 1;
121 }
122
123 void five_number(){
124     diode_off();
125     frame[0][2] = 1;
126     frame[0][1] = 1;
127     frame[0][0] = 1;
128     frame[1][0] = 1;
129     frame[2][0] = 1;
130     frame[3][0] = 1;
131     frame[3][1] = 1;
132     frame[3][2] = 1;
133     frame[4][2] = 1;
134     frame[5][2] = 1;
135     frame[6][2] = 1;
136     frame[7][2] = 1;
137     frame[7][1] = 1;
138     frame[7][0] = 1;
139 }

141 void six_number(){
142     diode_off();
143     frame[0][0] = 1;
144     frame[1][0] = 1;
145     frame[2][0] = 1;
146     frame[3][0] = 1;
147     frame[4][0] = 1;
148     frame[5][0] = 1;
149     frame[6][0] = 1;
150     frame[7][0] = 1;
151
152     frame[0][1] = 1;
153     frame[4][1] = 1;
154     frame[7][1] = 1;
155
156     frame[0][2] = 1;
157     frame[4][2] = 1;
158     frame[5][2] = 1;
159     frame[6][2] = 1;
160     frame[7][2] = 1;
161 }
162
163 void seven_number(){
164     diode_off();
165     frame[0][2] = 1;
166     frame[1][2] = 1;
167     frame[2][2] = 1;
168     frame[3][2] = 1;
169     frame[4][2] = 1;
170     frame[5][2] = 1;
171     frame[6][2] = 1;
172     frame[7][2] = 1;
173
174     frame[0][1] = 1;
175
176     frame[0][0] = 1;
177     frame[1][0] = 1;
178     frame[2][0] = 1;
179 }

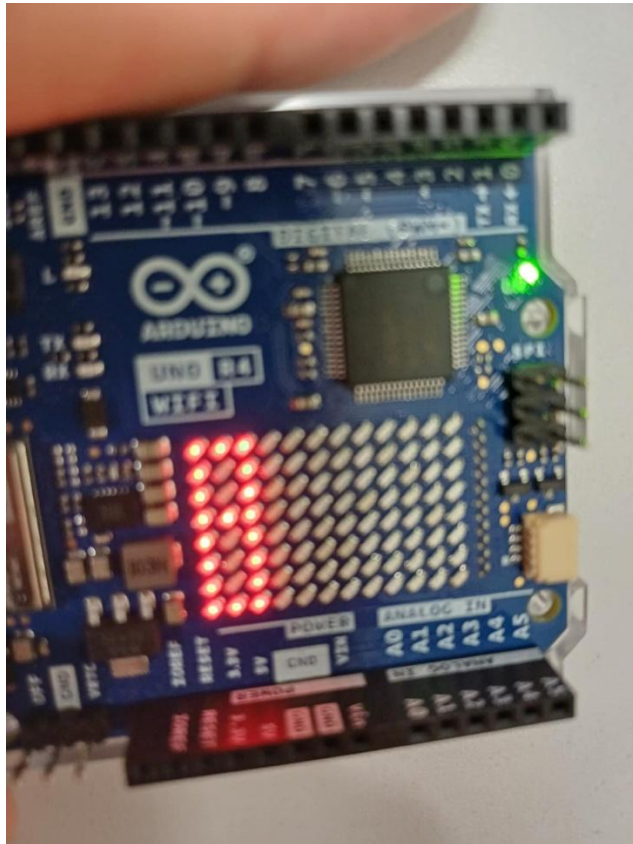
181 void eight_number(){
182     diode_off();
183     frame[0][2] = 1;
184     frame[1][2] = 1;
185     frame[2][2] = 1;
186     frame[3][2] = 1;
187     frame[4][2] = 1;
188     frame[5][2] = 1;
189     frame[6][2] = 1;
190     frame[7][2] = 1;
191
192     frame[0][1] = 1;
193     frame[3][1] = 1;
194     frame[7][1] = 1;
195
196     frame[0][0] = 1;
197     frame[1][0] = 1;
198     frame[2][0] = 1;
199     frame[3][0] = 1;
200     frame[4][0] = 1;
201     frame[5][0] = 1;
202     frame[6][0] = 1;
203     frame[7][0] = 1;
204 }
205
206 void nine_number(){
207     diode_off();
208     frame[0][2] = 1;
209     frame[1][2] = 1;
210     frame[2][2] = 1;
211     frame[3][2] = 1;
212     frame[4][2] = 1;
213     frame[5][2] = 1;
214     frame[6][2] = 1;
215     frame[7][2] = 1;
216     frame[0][1] = 1;
217     frame[3][1] = 1;
218     frame[7][1] = 1;
219     frame[0][0] = 1;
220     frame[1][0] = 1;
221     frame[2][0] = 1;
222     frame[3][0] = 1;
223     frame[6][0] = 1;
224     frame[7][0] = 1;
225 }

```

```

228 int cnt = 0;
229
230 void loop() {
231     // put your main code here, to run repeatedly:
232     if (cnt==0)
233     {
234         zero_number();
235     }
236     else if (cnt == 1)
237     {
238         one_number();
239     }
240     else if (cnt == 2)
241     {
242         two_number();
243     }
244     else if (cnt == 3)
245     {
246         three_number();
247     }
248     else if (cnt == 4)
249     {
250         four_number();
251     }
252     else if (cnt == 5)
253     {
254         five_number();
255     }
256     else if (cnt == 6)
257     {
258         six_number();
259     }
260     else if (cnt == 7)
261     {
262         seven_number();
263     }
264     else if (cnt == 8)
265     {
266         eight_number();
267     }
268     else
269     {
270         nine_number();
271     }
272
273     if(cnt == 9)
274     {
275         cnt = 0;
276     }
277     else
278     {
279         cnt = cnt + 1;
280     }
281
282     matrix.renderBitmap(frame, 8, 12);
283     delay(1000);
284 }

```



## 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 }
```

```

182 void eight_number(int cnt_led_array[19]){
183     diode_off();
184
185     frame[0][0] = cnt_led_array[0];
186     frame[0][1] = cnt_led_array[1];
187     frame[0][2] = cnt_led_array[2];
188     frame[1][2] = cnt_led_array[3];
189     frame[2][2] = cnt_led_array[4];
190     frame[3][2] = cnt_led_array[5];
191     frame[4][2] = cnt_led_array[6];
192     frame[5][2] = cnt_led_array[7];
193     frame[6][2] = cnt_led_array[8];
194     frame[7][2] = cnt_led_array[9];
195     frame[7][1] = cnt_led_array[10];
196     frame[7][0] = cnt_led_array[11];
197     frame[6][0] = cnt_led_array[12];
198     frame[5][0] = cnt_led_array[13];
199     frame[4][0] = cnt_led_array[14];
200     frame[3][0] = cnt_led_array[15];
201     frame[3][1] = cnt_led_array[16];
202     frame[2][0] = cnt_led_array[17];
203     frame[1][0] = cnt_led_array[18];
208 void nine_number(int cnt_led_array[19]){
209     diode_off();
210     frame[0][2] = cnt_led_array[10];
211     frame[1][2] = cnt_led_array[9];
212     frame[2][2] = cnt_led_array[8];
213     frame[3][2] = cnt_led_array[7];
214     frame[4][2] = cnt_led_array[6];
215     frame[5][2] = cnt_led_array[5];
216     frame[6][2] = cnt_led_array[4];
217     frame[7][2] = cnt_led_array[3];
218
219     frame[0][1] = cnt_led_array[11];
220     frame[3][1] = cnt_led_array[16];
221     frame[7][1] = cnt_led_array[2];
222
223     frame[0][0] = cnt_led_array[12];
224     frame[1][0] = cnt_led_array[13];
225     frame[2][0] = cnt_led_array[14];
226     frame[3][0] = cnt_led_array[15];
227
228     frame[6][0] = cnt_led_array[0];
229     frame[7][0] = cnt_led_array[1];
230 }

```

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

```

472 else if (cnt == 8)
473 {
474     for (int index=0; index<=18; index++)
475     {
476         cnt_led_array[index] = 1;
477         eight_number(cnt_led_array);
478         matrix.renderBitmap(frame, 8, 12);
479         delay(50);
480     }
481     delay(1000);
482     cnt_led_array[0] = 0;
483     cnt_led_array[1] = 0;
484     cnt_led_array[2] = 0;
485     cnt_led_array[3] = 0;
486     cnt_led_array[4] = 0;
487     cnt_led_array[5] = 0;
488     cnt_led_array[6] = 0;
489     cnt_led_array[7] = 0;
490     cnt_led_array[8] = 0;
491     cnt_led_array[9] = 0;
492     cnt_led_array[10] = 0;
493     cnt_led_array[11] = 0;
494     cnt_led_array[12] = 0;
495     cnt_led_array[13] = 0;
496     cnt_led_array[14] = 0;
497     cnt_led_array[15] = 0;
498     cnt_led_array[16] = 0;
499     cnt_led_array[17] = 0;
500     cnt_led_array[18] = 0;
501     diode_off();
502     matrix.renderBitmap(frame, 8, 12);
503
504 }

```

## 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.

```
34 void number(int cnt_led_array[45]){
35     diode_off();
36
37     frame[7][0] = cnt_led_array[0];
38     frame[6][0] = cnt_led_array[1];
39     frame[5][0] = cnt_led_array[2];
40     frame[4][0] = cnt_led_array[3];
41     frame[3][0] = cnt_led_array[4];
42
43     frame[7][1] = cnt_led_array[5];
44     frame[6][1] = cnt_led_array[6];
45     frame[5][1] = cnt_led_array[7];
46     frame[4][1] = cnt_led_array[8];
47     frame[3][1] = cnt_led_array[9];
48
49     frame[7][2] = cnt_led_array[10];
50     frame[6][2] = cnt_led_array[11];
51     frame[5][2] = cnt_led_array[12];
52     frame[4][2] = cnt_led_array[13];
53     frame[3][2] = cnt_led_array[14];
54
55     frame[7][3] = cnt_led_array[15];
56     frame[6][3] = cnt_led_array[16];
57     frame[5][3] = cnt_led_array[17];
58     frame[4][3] = cnt_led_array[18];
59     frame[3][3] = cnt_led_array[19];
60
61     frame[7][4] = cnt_led_array[20];
62     frame[6][4] = cnt_led_array[21];
63     frame[5][4] = cnt_led_array[22];
64     frame[4][4] = cnt_led_array[23];
65     frame[3][4] = cnt_led_array[24];
66
67     frame[7][5] = cnt_led_array[25];
68     frame[6][5] = cnt_led_array[26];
69     frame[5][5] = cnt_led_array[27];
70     frame[4][5] = cnt_led_array[28];
71     frame[3][5] = cnt_led_array[29];
72
73     frame[7][6] = cnt_led_array[30];
74     frame[6][6] = cnt_led_array[31];
75     frame[5][6] = cnt_led_array[32];
76     frame[4][6] = cnt_led_array[33];
77     frame[3][6] = cnt_led_array[34];
78
79     frame[7][7] = cnt_led_array[35];
80     frame[6][7] = cnt_led_array[36];
81     frame[5][7] = cnt_led_array[37];
82     frame[4][7] = cnt_led_array[38];
83     frame[3][7] = cnt_led_array[39];
84
85     frame[7][8] = cnt_led_array[40];
86     frame[6][8] = cnt_led_array[41];
87     frame[5][8] = cnt_led_array[42];
88     frame[4][8] = cnt_led_array[43];
89     frame[3][8] = cnt_led_array[44];
90 }
```

Na podstawie poniższych tablic opracowaliśmy odpowiednie warianty dla brakujących liczb.

```
95 int zero_size_1[45] = {1,1,1,0,0,
96                       1,0,1,0,0,
97                       1,0,1,0,0,
98                       1,0,1,0,0,
99                       1,1,1,0,0,
100                      0,0,0,0,0,
101                      0,0,0,0,0,
102                      0,0,0,0,0,
103                      0,0,0,0,0};
104 int zero_size_2[45] = {1,1,1,1,0,
105                       1,0,0,1,0,
106                       1,0,0,1,0,
107                       1,0,0,1,0,
108                       1,0,0,1,0,
109                       1,0,0,1,0,
110                       1,1,1,1,0,
111                       0,0,0,0,0,
112                       0,0,0,0,0};
113 int zero_size_3[45] = {1,1,1,1,1,
114                       1,0,0,0,1,
115                       1,0,0,0,1,
116                       1,0,0,0,1,
117                       1,0,0,0,1,
118                       1,0,0,0,1,
119                       1,0,0,0,1,
120                       1,0,0,0,1,
121                       1,1,1,1,1};
```

Poniżej przedstawienie brakującego kodu.

```
178 int three_size_1[45] = {
179     1,1,1,0,0, // wiensz 0
180     0,0,1,0,0, // wiensz 1
181     1,1,1,0,0, // wiensz 2
182     0,0,1,0,0, // wiensz 3
183     1,1,1,0,0, // wiensz 4
184     0,0,0,0,0, // wiensz 5
185     0,0,0,0,0, // wiensz 6
186     0,0,0,0,0, // wiensz 7
187     0,0,0,0,0 // wiensz 8
188 };
189
190 int three_size_2[45] = {
191     1,1,1,1,1, // wiensz 0
192     0,0,0,0,1, // wiensz 1
193     0,0,0,0,1, // wiensz 2
194     1,1,1,1,1, // wiensz 3
195     0,0,0,0,1, // wiensz 4
196     0,0,0,0,1, // wiensz 5
197     1,1,1,1,1, // wiensz 6
198     0,0,0,0,0, // wiensz 7
199     0,0,0,0,0 // wiensz 8
200 };
201
202 int three_size_3[45] = {
203     1,1,1,1,1, // wiensz 0
204     0,0,0,0,1, // wiensz 1
205     0,0,0,0,1, // wiensz 2
206     0,0,0,0,1, // wiensz 3
207     1,1,1,1,1, // wiensz 4
208     0,0,0,0,1, // wiensz 5
209     0,0,0,0,1, // wiensz 6
210     0,0,0,0,1, // wiensz 7
211     1,1,1,1,1 // wiensz 8
212 };
225 int four_size_2[45] = {
226     1,0,0,1,0,
227     1,0,0,1,0,
228     1,0,0,1,0,
229     1,1,1,1,1,
230     0,0,0,1,0,
231     0,0,0,1,0,
232     0,0,0,1,0,
233     0,0,0,0,0,
234     0,0,0,0,0
235 };
236 int four_size_3[45] = {
237     1,0,0,0,1,
238     1,0,0,0,1,
239     1,0,0,0,1,
240     1,0,0,0,1,
241     1,1,1,1,1,
242     0,0,0,0,1,
243     0,0,0,0,1,
244     0,0,0,0,1,
245     0,0,0,0,1
246 };
249 int five_size_1[45] = {
250     1,1,1,0,0,
251     1,0,0,0,0,
252     1,1,1,0,0,
253     0,0,1,0,0,
254     1,1,1,0,0,
255     0,0,0,0,0,
256     0,0,0,0,0,
257     0,0,0,0,0,
258     0,0,0,0,0
259 };
260 int five_size_2[45] = {
261     1,1,1,1,0,
262     1,0,0,0,0,
263     1,1,1,1,0,
264     0,0,0,1,0,
265     0,0,0,1,0,
266     0,0,0,1,0,
267     1,1,1,1,0,
268     0,0,0,0,0,
269     0,0,0,0,0
270 };
271 int five_size_3[45] = {
272     1,1,1,1,1,
273     1,0,0,0,0,
274     1,1,1,1,1,
275     0,0,0,0,1,
276     0,0,0,0,1,
277     0,0,0,0,1,
278     1,1,1,1,1,
279     0,0,0,0,0,
280     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.