

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. ТАРАСА ШЕВЧЕНКА
ФІЗИЧНИЙ ФАКУЛЬТЕТ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №4 (CRAZY SNAKE AND ULTIMATE
TETRIS)

Виконали:

студент 2 курсу 5-А групи спеціальності
104 «Фізика та астрономія»
Свінтозельський Володимир
Ярославович

студентка 2 курсу 5-А групи
спеціальності
104 «Фізика та астрономія»
Бучинська Марія Євгенівна

студентка 2 курсу 5-А групи
спеціальності
104 «Фізика та астрономія»
Топчій Поліна

Науковий керівник:

викладач
Єрмоленко Руслан Вікторович

РОЛІ АВТОРІВ

У цій лабораторній роботі брали участь такі студенти:

- Свінтозельський Володимир Ярославович
 - *концепт*
 - *сборка електричної схеми*
 - *написання коду програми*
 - *відеофіксація легендарного досягнення*
 - *проектування джойстика*
 - *виготворення джойстика*
 - *оформлення звіту*
- Бучинська Марія Євгенівна
 - *відлагодження електричної схеми*
 - *дизайн*
 - *концепт*
 - *виготворення джойстика*
 - *написання коду програми*
- Топчій Поліна
 - *відлагодження електричної схеми*
 - *дизайн*
 - *концепт*
 - *політична робота*

ЗМІСТ

Вступ	4
Розділ 1 Ближче до справи	6
1.1 Підключення матриці	6
1.2 Змійка	6
1.3 Тетріс	12
Висновки	22

ВСТУП

У даній роботі нами було створено гри епохи 00х: змійка та тетріс! Створено це чудо програмної та інженерної думки на базі arduino та світлодіодної матриці 8x8.

Для гри у змійку було виготовлено джойстик 1 із кнопок, проводів, макетної плати та рук, що нагло претендують на звання "прямі". По суті, він являв собою кнопки, які при натисканні притягували відповідні виходи до землі (ардуїнка була у режимі *INPUT_PULLUP*). Нажаль, до того часу, як почалась розробка тетрісу, даний джойстик не дожив :(



Рис. 1: Джойстик, що використовувався у змійці.

РОЗДІЛ 1

БЛИЖЧЕ ДО СПРАВИ

Оскільки тут описувати по суті нічого, адже автори щиро вірять у казочку, що усі грали ці ігри, перейдемо відразу до технічних деталей.

1.1 Підключення матриці

Розпіновка світлодіодної матриці показана на 1.1

VCC	GND	CIN	CS	CLK
+5V	GND	11	9	13

Рис. 1.1: Розпіновка світлодіодної матриці.

1.2 Змійка

Відео гри одного з авторів у отриману змійку знаходиться у нашій папці на гітхабі.

Програмний код:

```

1 #include <SPI.h>
2 #include <Adafruit_GFX.h>
3 #include <Max72xxPanel.h>
4
5 int pinCS = 9;
6 int numberOfHorizontalDisplays = 1;
7 int numberOfVerticalDisplays = 1;
8

```



```

54     }
55     matrix.drawPixel(1,1+4*k, HIGH);
56     matrix.drawPixel(5,1+4*k, HIGH);
57     break;
58
59     case 1:
60     for(int i=1; i<6; i++)
61     matrix.drawPixel(i,2+4*k, HIGH);
62     break;
63
64     case 2:
65     for(int i=1; i<6; i++)
66     {
67         if(i!=4)
68         { matrix.drawPixel(i,4*k, HIGH);
69             if(i!=2) matrix.drawPixel(i,1+4*k, HIGH);
70         }
71         if(i!=2)
72         matrix.drawPixel(i,2+4*k, HIGH);
73     }
74     break;
75
76     case 3:
77     for(int i=1; i<6; i++)
78     {
79     matrix.drawPixel(i,2+4*k, HIGH);
80     if(i%2==1)
81     {matrix.drawPixel(i,4*k, HIGH);
82     matrix.drawPixel(i,1+4*k, HIGH);}
83     }
84     break;
85
86     case 4:
87     for(int i=1; i<6; i++)
88     {
89     matrix.drawPixel(i,2+4*k, HIGH);
90     if(i>2) matrix.drawPixel(i,4*k, HIGH);
91     }
92     matrix.drawPixel(3,1+4*k, HIGH);
93     break;
94
95     case 5:
96
97     for(int i=1; i<6; i++)
98     {
99         if(i!=2)

```



```

100     { matrix.drawPixel(i,4*k, HIGH);
101         if(i!=4) matrix.drawPixel(i,1+4*k, HIGH);
102     }
103     if(i!=4)
104     matrix.drawPixel(i,2+4*k, HIGH);
105 }
106 break;
107
108 case 6:
109 for(int i=1; i<6; i++)
110 {
111     matrix.drawPixel(i,4*k, HIGH);
112     matrix.drawPixel(i,2+4*k, HIGH);
113     if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
114 }
115     matrix.drawPixel(4,2+4*k, LOW);
116     break;
117
118 case 7:
119 for(int i=1; i<6; i++)
120 {
121     matrix.drawPixel(i,2+4*k, HIGH);
122 }
123     matrix.drawPixel(5,4*k, HIGH);
124     matrix.drawPixel(5,1+4*k, HIGH);
125     break;
126
127 case 8:
128 for(int i=1; i<6; i++)
129 {
130     matrix.drawPixel(i,4*k, HIGH);
131     matrix.drawPixel(i,2+4*k, HIGH);
132     if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
133 }
134     break;
135
136 case 9:
137 for(int i=1; i<6; i++)
138 {
139     matrix.drawPixel(i,4*k, HIGH);
140     matrix.drawPixel(i,2+4*k, HIGH);
141     if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
142 }
143     matrix.drawPixel(2,4*k, LOW);
144     break;
145 }

```

```

146     a=d%10;
147 }
148 matrix.write();
149 }
150
151
152 void loop() {
153
154
155 snake[0].x=(snake[0].x+8)%8;
156 snake[0].y=(snake[0].y+8)%8;
157 for(int i=0; i<len; i++)
158 {
159     matrix.drawPixel((snake[i].x+8)%8, (snake[i].y+8)%8, LOW);
160 }
161 matrix.write();
162
163 for(int j=len-1; j>0; j--)
164 {
165     snake[j].x=(snake[j].x+8)%8;
166     snake[j].y=(snake[j].y+8)%8;
167     if(snake[j].x != snake[j-1].x || snake[j].y != snake[j-1].y)
168     {
169         snake[j].x=snake[j-1].x;
170         snake[j].y=snake[j-1].y;
171     }
172 }
173
174
175 snake[0].x+=dirX;
176 snake[0].y+=dirY;
177
178 snake[0].x = (snake[0].x + 8) % 8;
179 snake[0].y = (snake[0].y + 8) % 8;
180
181 for(int i=0; i<len; i++)
182 {
183     matrix.drawPixel((snake[i].x+8)%8, (snake[i].y+8)%8, HIGH);
184 }
185 matrix.drawPixel(eat.x, eat.y, HIGH);
186 matrix.write();
187 for(int k=0; k<len; k++)
188 {
189     for( int j=k+1; j<len; j++)
190     {
191         if(snake[k].x==snake[j].x&&snake[k].y==snake[j].y)

```

```

192         {
193             printdigit(len);
194             while(1==1)
195             {
196
197             }
198         }
199     }
200 }
201
202 if(eat.x==snake[0].x&&eat.y==snake[0].y)
203 {
204     snake[len].x=snake[len-1].x;
205     snake[len].y=snake[len-1].y;
206     len++;
207     matrix.drawPixel(eat.x, eat.y, LOW);
208     eat={random(8), random(8)};
209     matrix.drawPixel(eat.x, eat.y, HIGH);
210     matrix.write();
211 }
212 for(int k=0;k<500;k++)
213 {
214     if(k%100==0)
215     {
216         matrix.drawPixel(snake[0].x, snake[0].y, LOW);
217         matrix.drawPixel(eat.x, eat.y, LOW);
218         matrix.write();
219     }
220     if(k%100==50)
221     {
222         matrix.drawPixel(snake[0].x, snake[0].y, HIGH);
223         matrix.drawPixel(eat.x, eat.y, HIGH);
224         matrix.write();
225     }
226
227
228 if(digitalRead(2)==LOW&&dirX!=-1)
229 {   dirX=1;
230     dirY=0;}
231 if(digitalRead(3)==LOW&&dirY!=-1)
232 {   dirX=0;
233     dirY=1;}
234 if(digitalRead(4)==LOW&&dirX!=1)
235 {   dirX=-1;
236     dirY=0;}
237 if(digitalRead(5)==LOW&&dirY!=1)

```

```

238 {   dirX=0;
239     dirY=-1;}
240     delay(1);
241 }
242
243
244
245
246 }

```

1.3 Тетріс

Відео гри одного з авторів у отриманий тетріс знаходиться у нашій папці на гітхабі.

Програмний код:

```

1  #include <SPI.h>
2  #include <Adafruit_GFX.h>
3  #include <Max72xxPanel.h>
4
5  int pinCS = 9;
6  int numberOfHorizontalDisplays = 1;
7  int numberOfVerticalDisplays = 1;
8
9  Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays,
    numberOfVerticalDisplays);
10
11 // 4 5 6
12 // 2 3
13 int** matrixMask = new int*[8];
14 short score = 0;
15
16 void setup() {
17     randomSeed(analogRead(A0));
18     pinMode(4, INPUT_PULLUP);
19     pinMode(5, INPUT_PULLUP);
20     pinMode(6, INPUT_PULLUP);
21     pinMode(2, INPUT_PULLUP);
22     pinMode(3, INPUT_PULLUP);
23     matrix.setIntensity(0);
24     for(int i =0;i<8;i++){
25         matrixMask[i] = new int[8];
26         for(int j = 0;j<8;j++)
27             matrixMask[i][j]=0;
28     }
29     matrix.fillScreen(LOW);
30     matrix.write();

```

```

31 Serial.begin(9600);
32 }
33
34 void printdigit(int d)
35 {
36     matrix.setRotation( 0, 1 );
37     for(int i=0; i<8; i++)
38     {
39         for(int j=0; j<8; j++)
40             matrix.drawPixel(i,j, LOW);
41         matrix.write();
42     }
43     int a=d/10;
44     for(int k=0; k<2; k++)
45     {
46         switch(a)
47         {
48             case 0:
49                 for(int i=1; i<6; i++)
50                 {
51                     matrix.drawPixel(i,4*k, HIGH);
52                     matrix.drawPixel(i,2+4*k, HIGH);
53                 }
54                 matrix.drawPixel(1,1+4*k, HIGH);
55                 matrix.drawPixel(5,1+4*k, HIGH);
56                 break;
57
58             case 1:
59                 for(int i=1; i<6; i++)
60                     matrix.drawPixel(i,2+4*k, HIGH);
61                 break;
62
63             case 2:
64                 for(int i=1; i<6; i++)
65                 {
66                     if(i!=4)
67                         { matrix.drawPixel(i,4*k, HIGH);
68                         if(i!=2) matrix.drawPixel(i,1+4*k, HIGH);
69                     }
70                     if(i!=2)
71                         matrix.drawPixel(i,2+4*k, HIGH);
72                 }
73                 break;
74
75             case 3:
76                 for(int i=1; i<6; i++)

```

```

77     {
78         matrix.drawPixel(i,2+4*k, HIGH);
79         if(i%2==1)
80         {matrix.drawPixel(i,4*k, HIGH);
81         matrix.drawPixel(i,1+4*k, HIGH);}
82     }
83     break;
84
85     case 4:
86     for(int i=1; i<6; i++)
87     {
88         matrix.drawPixel(i,2+4*k, HIGH);
89         if(i>2) matrix.drawPixel(i,4*k, HIGH);
90     }
91     matrix.drawPixel(3,1+4*k, HIGH);
92     break;
93
94     case 5:
95
96     for(int i=1; i<6; i++)
97     {
98         if(i!=2)
99         { matrix.drawPixel(i,4*k, HIGH);
100             if(i!=4) matrix.drawPixel(i,1+4*k, HIGH);
101         }
102         if(i!=4)
103         matrix.drawPixel(i,2+4*k, HIGH);
104     }
105     break;
106
107     case 6:
108     for(int i=1; i<6; i++)
109     {
110         matrix.drawPixel(i,4*k, HIGH);
111         matrix.drawPixel(i,2+4*k, HIGH);
112         if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
113     }
114     matrix.drawPixel(4,2+4*k, LOW);
115     break;
116
117     case 7:
118     for(int i=1; i<6; i++)
119     {
120         matrix.drawPixel(i,2+4*k, HIGH);
121     }
122     matrix.drawPixel(5,4*k, HIGH);

```

```

123     matrix.drawPixel(5,1+4*k, HIGH);
124     break;
125
126     case 8:
127     for(int i=1; i<6; i++)
128     {
129     matrix.drawPixel(i,4*k, HIGH);
130     matrix.drawPixel(i,2+4*k, HIGH);
131     if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
132     }
133     break;
134
135     case 9:
136     for(int i=1; i<6; i++)
137     {
138     matrix.drawPixel(i,4*k, HIGH);
139     matrix.drawPixel(i,2+4*k, HIGH);
140     if(i%2==1) matrix.drawPixel(i,1+4*k, HIGH);
141     }
142     matrix.drawPixel(2,4*k, LOW);
143     break;
144 }
145 a=d%10;
146 }
147 matrix.write();
148 }
149
150
151 struct Figure {
152     int nx;
153     int ny;
154     int** mask;
155
156     ~Figure(){
157         for(int i = 0;i<ny;i++)
158             delete [] mask[i];
159         delete [] mask;
160     }
161     Figure(int nx, int ny, int** mask): nx(nx),ny(ny),mask(mask){}
162     Figure(int type){
163         mask = new int*[2];
164         for(int i =0;i<2;i++){
165             mask[i] = new int[3];
166             for(int j =0;j<3;j++)
167                 mask[i][j]=0;
168         }

```

```
169     if(type == 1){
170         nx = 1;
171         ny =1;
172         mask[0][0] = 1;
173     }
174     if(type == 2){
175         nx = 3;
176         ny =2;
177         mask[0][0] = 1;
178         mask[0][1] = 1;
179         mask[1][1] = 1;
180         mask[1][2] = 1;
181     }
182     if(type == 3){
183         nx = 3;
184         ny =2;
185         mask[1][0] = 1;
186         mask[0][1] = 1;
187         mask[1][1] = 1;
188         mask[0][2] = 1;
189     }
190     if(type == 4){
191         nx = 3;
192         ny =2;
193         mask[0][1] = 1;
194         mask[1][0] = 1;
195         mask[1][1] = 1;
196         mask[1][2] = 1;
197     }
198     if(type == 5){
199         nx = 3;
200         ny =2;
201         mask[0][0] = 1;
202         mask[1][0] = 1;
203         mask[1][1] = 1;
204         mask[1][2] = 1;
205     }
206     if(type == 6){
207         nx = 3;
208         ny =2;
209         mask[0][0] = 1;
210         mask[1][0] = 1;
211         mask[0][1] = 1;
212         mask[0][2] = 1;
213     }
214     if(type == 7){
```



```

215     nx = 3;
216     ny =1;
217     mask[0][0] = 1;
218     mask[0][1] = 1;
219     mask[0][2] = 1;
220 }
221 if(type == 8){
222     nx = 2;
223     ny =2;
224     mask[0][0] = 1;
225     mask[0][1] = 1;
226     mask[1][0] = 1;
227 }
228 if(type == 9){
229     nx = 2;
230     ny =2;
231     mask[0][0] = 1;
232     mask[0][1] = 1;
233     mask[1][0] = 1;
234     mask[1][1] = 1;
235 }
236 }
237 };
238
239 bool hasActiveFig = false;
240 Figure* activef;
241 int x=0;
242 int y=0;
243
244 bool isSomethingUnder(){
245     if(y == activef->ny-1) {Serial.println("GROUND");return true;}
246     for(int j =0;j<activef->nx;j++){
247         for(int i = activef->ny-1;i>=0;i--){
248             if(activef->mask[i][j]){
249                 if(matrixMask[y-i-1][x+j])
250                     return true;
251                 break;
252             }
253         }
254     }
255     return false;
256 }
257
258 bool isSomethingRight(){
259     if(x == 8-activef->nx) {Serial.println("Right border");return true;}
260     for(int j =0;j<activef->ny;j++){

```

```

261     for(int i = activef->nx-1;i>=0;i--){
262         if(activef->mask[j][i]){
263             if(matrixMask[y-j][x+i+1])
264                 return true;
265             break;
266         }
267     }
268 }
269 return false;
270 }
271
272 bool isSomethingLeft(){
273     if(x == 0) {Serial.println("Left border");return true;}
274     for(int j =0;j<activef->ny;j++){
275         for(int i = 0;i<activef->nx;i++){
276             if(activef->mask[j][i]){
277                 if(matrixMask[y-j][x+i-1])
278                     return true;
279                 break;
280             }
281         }
282     }
283     return false;
284 }
285
286 void removeLine(int line){
287     for(int j = line;j<6;j++){
288         for(int i =0;i<8;i++){
289             matrixMask[j][i] = matrixMask[j+1][i];
290         }
291     }
292
293 void checkIsFullLineExist(){
294     bool isfull;
295     for(int j =0;j<6;j++){
296         isfull = true;
297         for(int i =0;i<8;i++){
298             if(!matrixMask[j][i]){
299                 isfull = false;
300                 break;
301             }
302         }
303         if(isfull)
304             removeLine(j);
305     }
306 }

```

```

307 void endGame(){
308     matrix.fillScreen(LOW);
309     printdigit(score);
310     matrix.write();
311     while(true){}
312 }
313
314 void moveActive(){
315     bool check = isSomethingUnder();
316     Serial.println(check);
317     if(check){
318         if(y >= 6) endGame();
319         for(int i = y; i >= activef->ny; i--)
320             for(int j = x; j < activef->nx; j++)
321                 matrixMask[i][j] = (activef->mask[y-i][j-x] || matrixMask[i][j]);
322         hasActiveFig=false;
323         checkIsFullLineExist();
324         return;
325     }
326     y--;
327 }
328
329 void falldown(){
330     while(hasActiveFig)
331         moveActive();
332 }
333
334 void graph(){
335     matrix.fillScreen(LOW);
336     for(int i = 0; i < 8; i++)
337         for(int j = 0; j < 8; j++)
338             matrix.drawPixel(i, j, matrixMask[j][i]);
339     for(int i = 0; i < activef->nx; i++)
340         for(int j = 0; j < activef->ny; j++)
341             if(activef->mask[j][i])
342                 matrix.drawPixel(x+i, y-j, activef->mask[j][i]);
343     matrix.write();
344 }
345
346 void rotateRight(){
347     int xsize = activef->nx;
348     int ysize = activef->ny;
349     Figure* rotatedF = new Figure(ysize, xsize, 0);
350     rotatedF->mask = new int*[xsize];
351     for(int i = 0; i < xsize; i++){
352         rotatedF->mask[i] = new int[ysize];

```

```

353     for(int j = 0; j < ysize; j++)
354         rotatedF->mask[i][j] = 0;
355 }
356 for(int i = 0; i < xsize; i++)
357     for(int j = 0; j < ysize; j++)
358         rotatedF->mask[i][j] = activef->mask[ysize-j-1][i];
359
360 activef = rotatedF;
361 }
362
363 void wait(){
364     for(int t = 0; t < 4; t++){
365         bool turnRight = !digitalRead(4);
366         bool turnDown = !digitalRead(3);
367         bool turnLeft = !digitalRead(2);
368         bool rotRight = !digitalRead(5);
369         bool rotLeft = !digitalRead(6);
370         if(turnRight && !turnLeft)
371             if(!isSomethingRight()){
372                 x++;
373                 graph();
374             }
375         if(!turnRight && turnLeft)
376             if(!isSomethingLeft()){
377                 x--;
378                 graph();
379             }
380         if(turnDown){
381             falldown();
382             graph();
383         }
384         if(rotRight && !rotLeft){
385             rotateRight();
386             graph();
387         }
388         delay(150);
389     }
390 }
391
392 void loop() {
393     if(!hasActiveFig){
394         score ++;
395         Serial.print("creating new figure with type: ");
396         hasActiveFig = true;
397         int type = random(1, 10);
398         Serial.print(type);

```

```
399     Serial.print("    ");
400     activef = new Figure(type);
401     x = 4;
402     y = 7;
403     Serial.println(type);
404     graph();
405     wait();
406 }
407
408 moveActive();
409 Serial.print("moved ");
410 Serial.print(x);
411 Serial.print(" ");
412 Serial.println(y);
413
414 graph();
415 wait();
416 }
```

ВИСНОВКИ

В процесі виконання даної роботи автори відточили навички з написання не завжди корисних програм на дитячій платформі arduino. Ну і звісно порозважались! :)

Автори надзвичайно вдячні викладачу Єрмоленку Р. В. за надану ардуїнку, матрицю, та кучу баракла до неї! Ну хоч одне речення без сарказму..