Guess Pattern

MicroController project

Ashkan Hafezi , Amirhossein Zendevani

Table of Contents

01

Rules

Here, you can play this game and learn its rules.

Game Rules

6 round

The game has 6 round.
Each round, one pattern

6 lives

Max 6 again lives. And
having 6 wrong choice.

Win or Loss?

After completion of 6
round, you Win ☺

Pointing Rules

Correct Choice

- Increase 3 point.

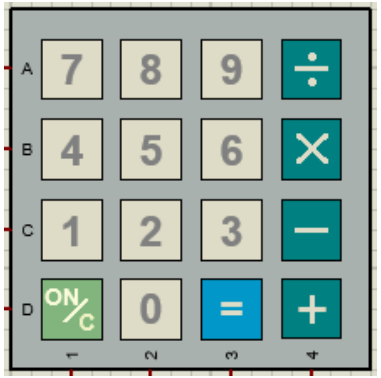
Wrong Choice

- Decrease 1 point.

Repeated Choice

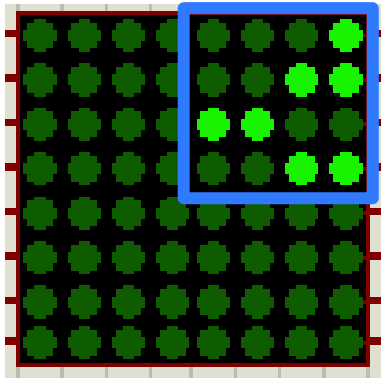
- No effect.

How to play



Input

we have 16 location and here have a correspond key for each location.



Board of Game

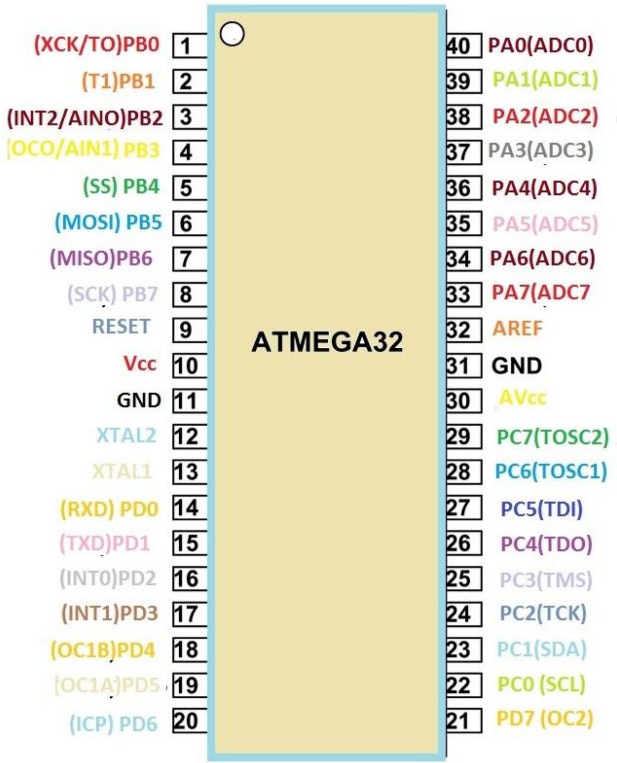
The pattern hide after 6 sec and then you should guess it to win.

02

Applience

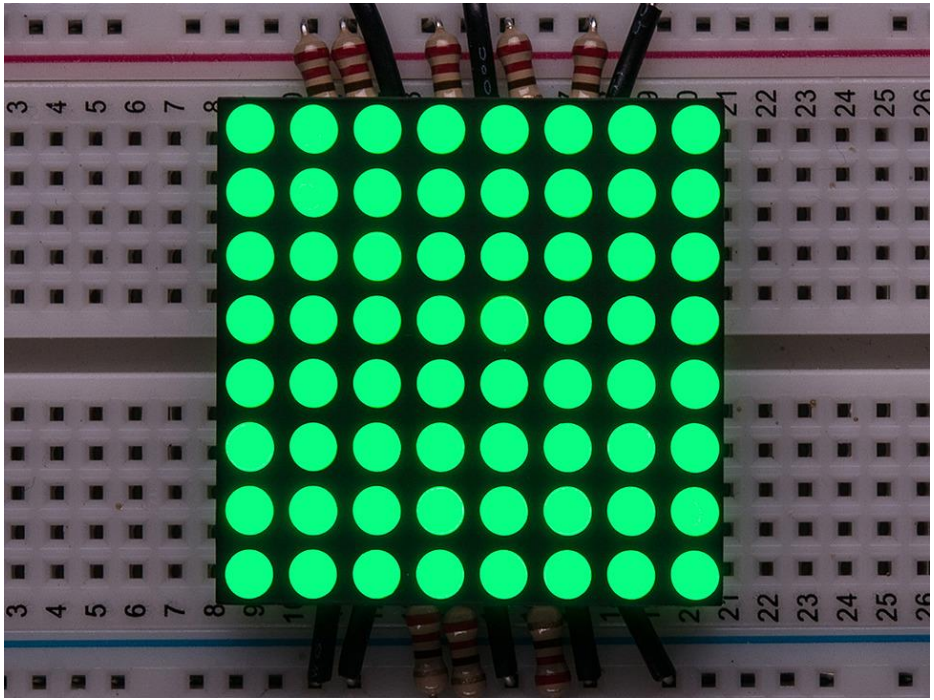
Here we describe the appliences used in this project

1. AVR Atmega32



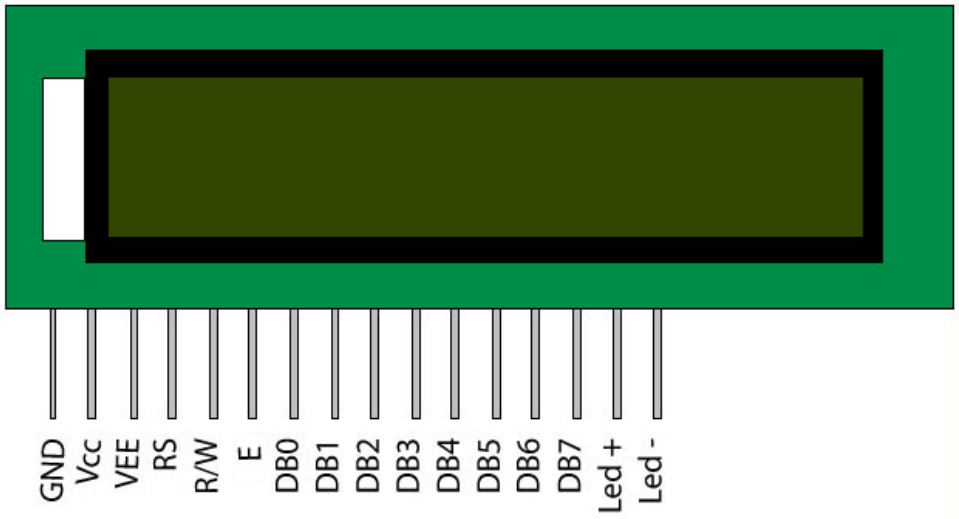
2. KEYPAD-SMALLCALC

3. MATRIX LED-8×8-GREEN



4. LED-GREEN, LED-RED

5. LCD 20-LM032L



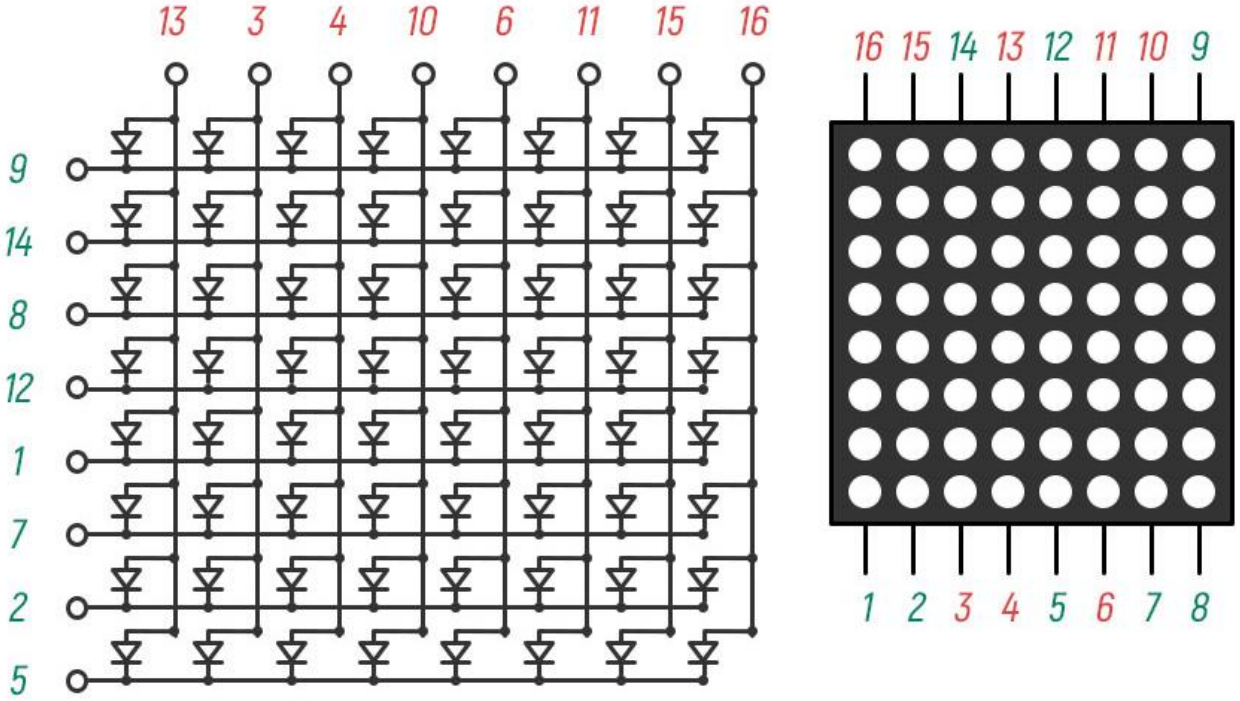
6. POT

03

Implementation

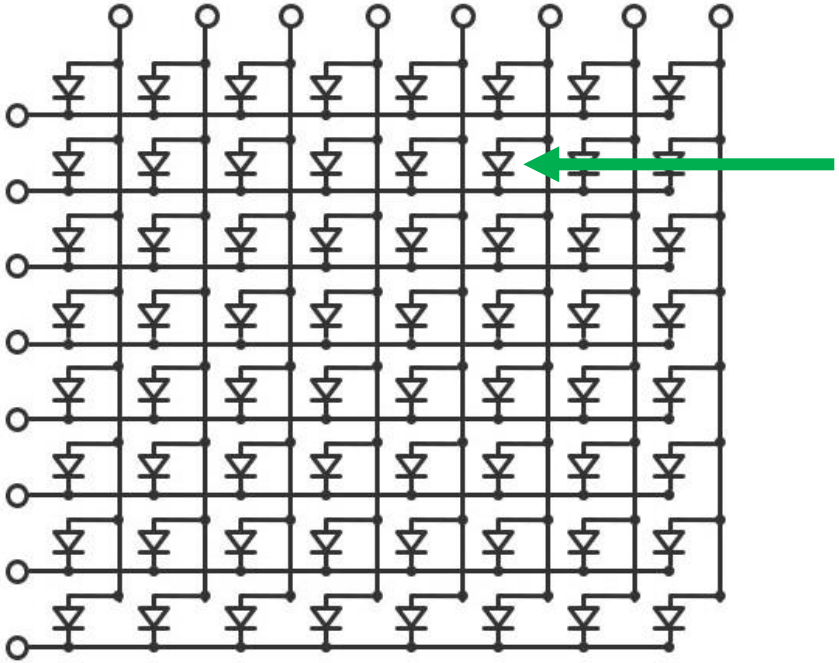
Here you see usage of each appliance and its related code

MATRIX LED-8×8-GREEN

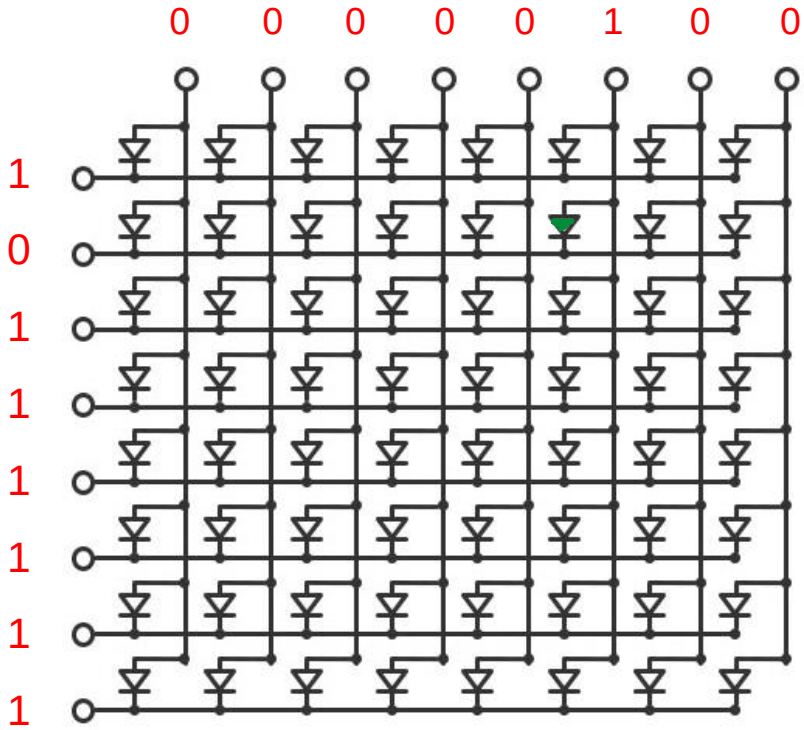


Question

Here how I can turn on a LED?

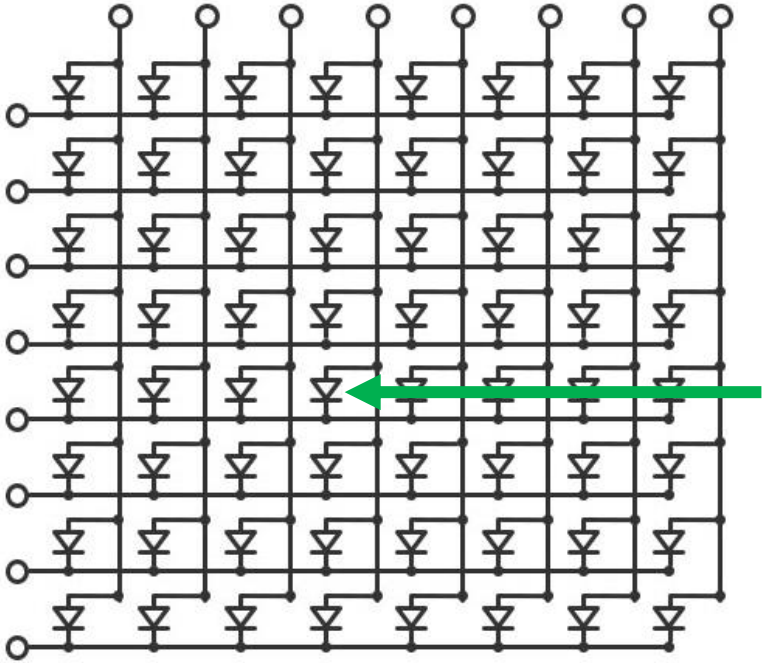


Answer

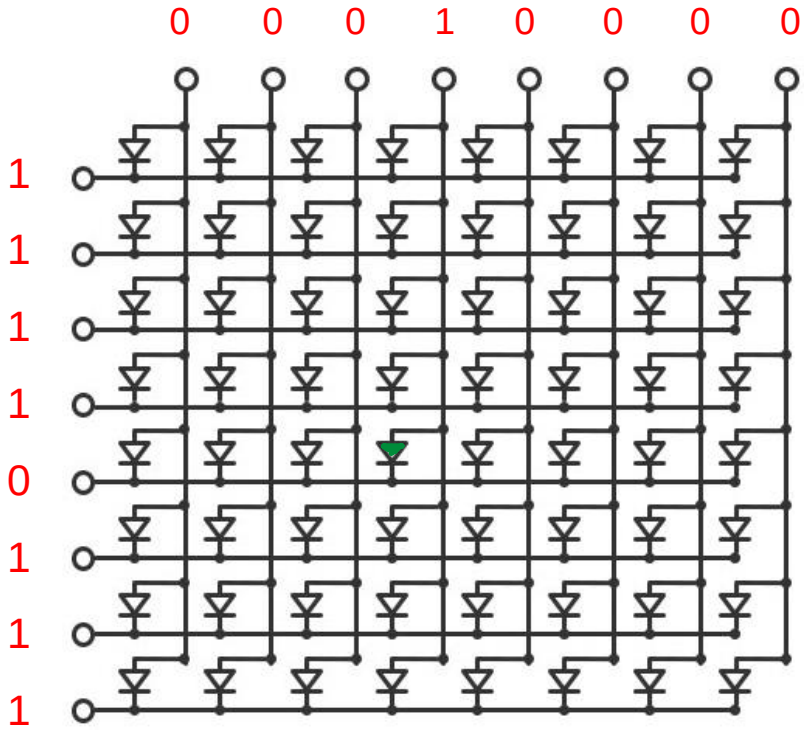


Question

Here how I can turn on another LED?

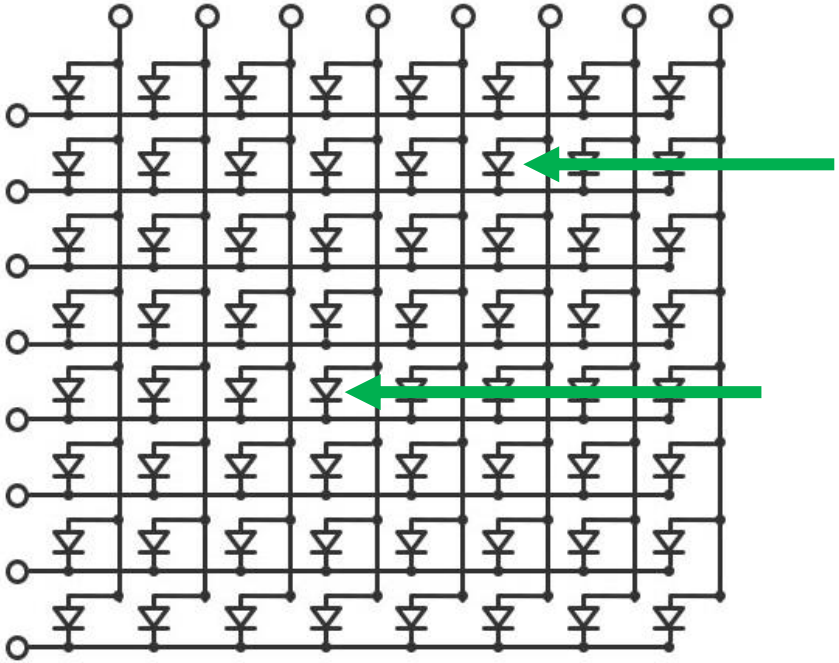


Answer



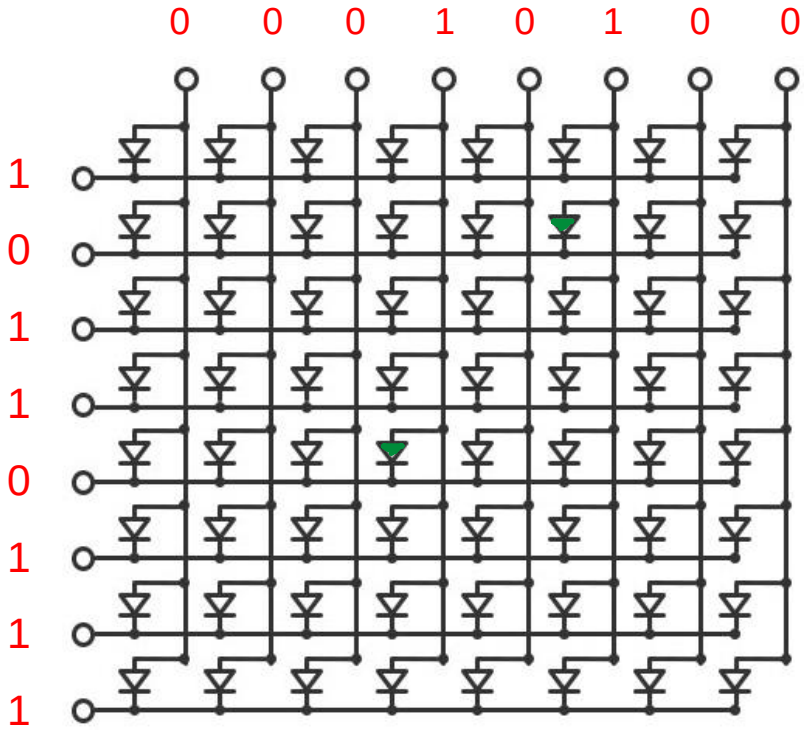
Question

Here how I can turn on these two LED simultaneously?



Answer

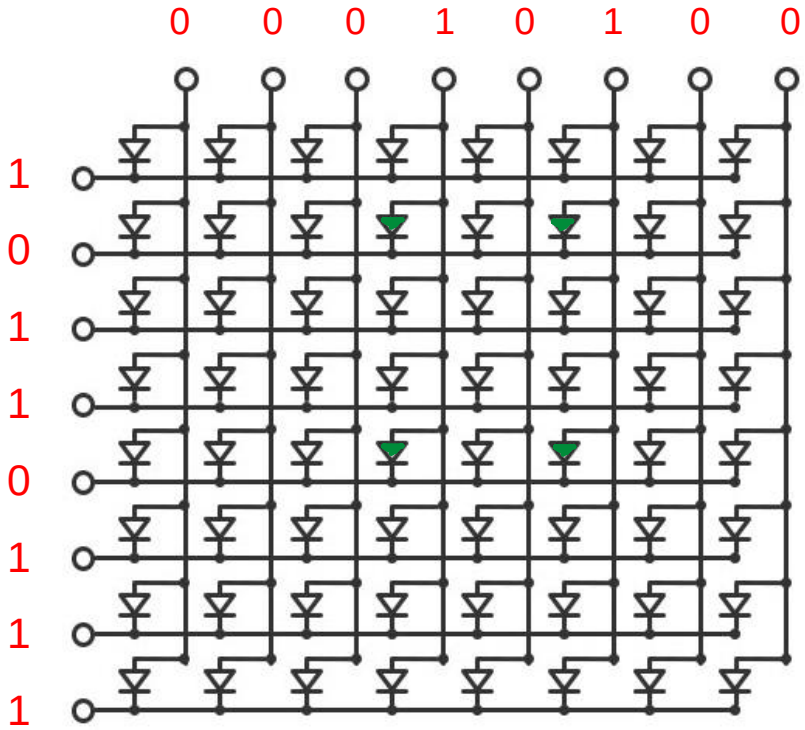
This is true?



Answer

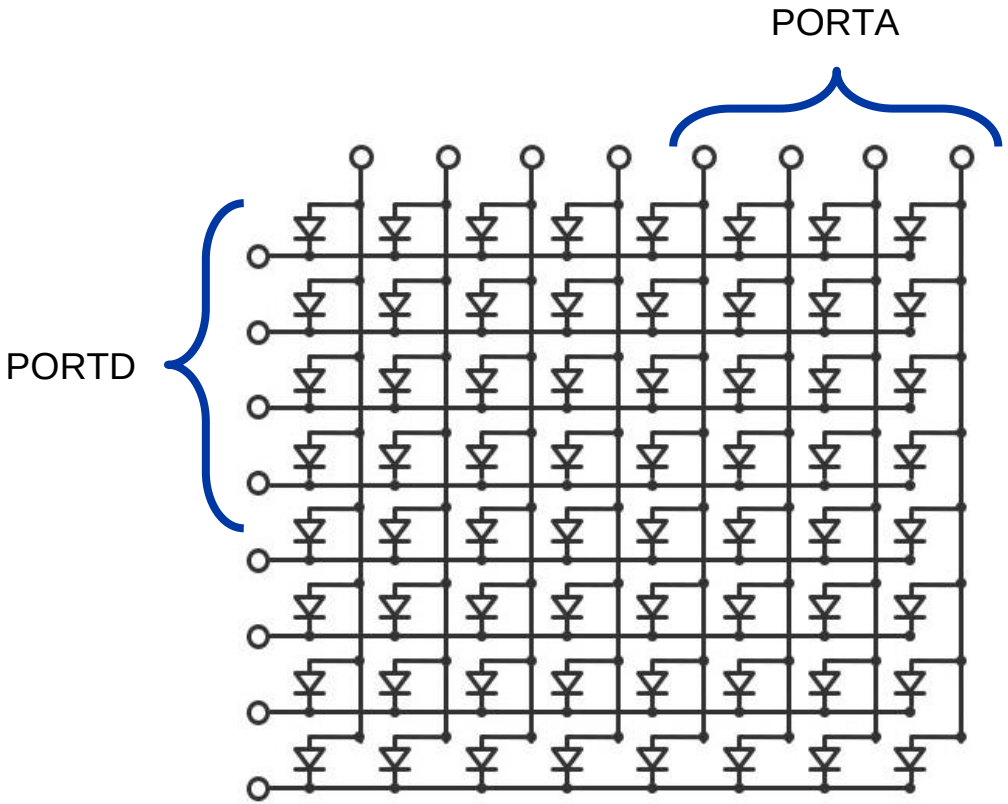
This is wrong.

What is solution?



Solution

```
char dotControl[4] = {1, 2, 4, 8};

void fill2()
{
    for (int i = 0; i < 4; i++)
    {
        PORTA = 0x00;
        PORTD = pattern[i];
        PORTA = dotControl[i];
        _delay_ms(2);
    }
}
```



Random patterns

How to generate random pattern?

we need **random numbers** to generate random patterns for each time of new game playing.

Problem:

Apparently, There is no random number generation like computers in AVR.

Our Solution:

Storing patterns in EEPROM, and after each execution, the elements of the pattern undergo a circular shift and are then stored again in EEPROM.

For each play, patterns are read from EEPROM. Due to the shifts applied in the previous game, these patterns are not the same as those in the previous game.

Read patterns from EEPROM

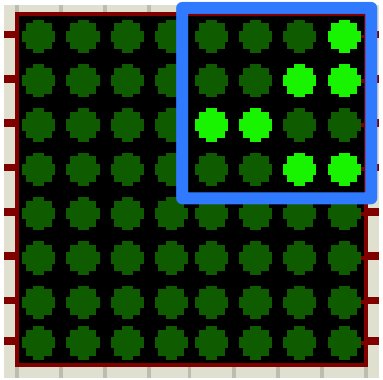
```
uint8_t numbers[24] = {
                            0x1, 0x3, 0xc, 0x3, 0x6, 0xf, 0x8, 0x6,
                            0x7, 0x0, 0x5, 0x2, 0x4, 0xe, 0xd, 0xb,
                            0x8, 0x2, 0xa, 0x9, 0xb, 0x4, 0x1, 0x0
                       };
void getPatterns()
{
    if(eeprom_read_byte((uint8_t*)24) != 255)
    {
        wins = eeprom_read_byte((uint8_t*)24);
        losts = eeprom_read_byte((uint8_t*)25);
        point = eeprom_read_byte((uint8_t*)26);

        for (int i = 0; i < 6; i++)
            for (int j = 0; j < 4; j++)
                patterns[i][j] = eeprom_read_byte((uint8_t*)(i*4 + j));
    }
    else
    {
        for (int i = 0; i < 6; i++)
            for (int j = 0; j < 4; j++)
            {
                patterns[i][j] = numbers[i*4 + j];
                eeprom_write_byte((uint8_t*)i, numbers[i]);
            }
    }
}
```

Circular Shift of elements

```
void cyclically_shift() {
    int temp;
    int shift = 13;

    for (int i = 0; i < 24; i++)
        numbers[i] = eeprom_read_byte((uint8_t*)i);

    while (shift > 0) {
        temp = numbers[23];

        for (int i = 23; i > 0; i--) {
            numbers[i] = numbers[i - 1];
        }

        numbers[0] = temp;
        shift--;
    }

    for(int i = 0; i < 24; i++)
        eeprom_write_byte((uint8_t*)i, numbers[i]);
}
```

Verify input

pattern[0] = 14	1	1	1	0
pattern[1] = 12	1	1	0	0
pattern[2] = 3	0	0	1	1
pattern[3] = 12	1	1	0	0



pattern[i]

1	1	1	0
1	1	1	0
0	1	1	1
1	1	0	1

Next choice

Pattern up to now

tmp3 = 4

patterns[r][i]

1	1	1	0
1	1	0	0
0	0	1	1
1	1	0	0

main pattern

tmp4 = 0

```
// wrong coordination
if( tmp3 && tmp4 )
```

pattern[i]

Next choice

1	1	1	0
1	1	1	0
0	1	1	1
1	1	0	1

Pattern up to now

patterns[r][i]

1	1	1	0
1	1	0	0
0	0	1	1
1	1	0	0

main pattern

Wrong chioce

```
// repeated coordination
if( tmp3 == 0 && tmp4 == 0)
```

pattern[i]

1	1	1	0
1	1	1	0
0	1	1	1
1	1	0	1

Pattern up to now

Next choice

patterns[r][i]

1	1	1	0
1	1	0	0
0	0	1	1
1	1	0	0

main pattern

repeated chioce

Not of wrong choice

pattern[i]

1	1	1	0
1	1	1	0
0	1	1	1
1	1	0	1

Next choice

Pattern up to now

patterns[r][i]

1	1	1	0
1	1	0	0
0	0	1	1
1	1	0	0

main pattern

Correct chioce

Thanks

Do you have any questions?