სენსორული პანელის მართვა მიკროკონტროლერის საშუალებით

ოთარ ქართველიშვილი, სიმონ ხოშტარია
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

განიხილება სენსორული ეკრანის მართვის შესაძლებლობები მიკროკონტროლერის საშუალებით. პროექტში გამოყენებულია სენსორული პანელი, გრაფიკული დისპლეი და AVR ოჯახის მიკროკონტროლერი Atmega 128. ნაჩვენებია მართვის ალგორითმი და C ენაზე პროგრამის ერთ-ერთი ვარიანტი, რომლის საშუალებით შესაძლებელია ეკრანზე სხვადასხვა ხასიათის ინფორმაციის გამოყვანა - ტექსტური ან გრაფიკული, ეკრანზე შეხების ადგილის შესაბამისად. სისტემის მუშაობა შემოწმებულია ლაბორატორიულ სტენდზე.

საკვანძო სიტყვები: სენსორული პანელი. გრაფიკული დისპლეი. მიკრო-კონტროლერი. ალგორითმი, პროგრამა.

1. შესავალი

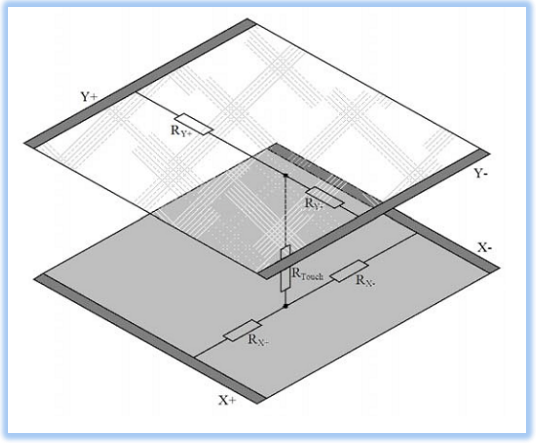
თანამედროვე გამოთვლით სისტემებში დიდი პოპულარობით სარგებს სენსორული ეკრანები, რომელთა დანიშნულებაა ეკრანის სხვადასხვა არეზე დაჭერით გამოიწვიოს რომელიმე პროგრამის გააქტიურება და შედეგის გამოტანაა ეკრანზე. არსებობს სენსორული ეკრანის დიდი სახესხვაობა, რომლებიც განსხვავდება თავისი მახასიათებლებით და მუშაობის პრინციპით. ერთ-ერთი მათგანი, სენსორული პანელი TS12864CRNA, ჩვენ მიერ არის გამოყენებული. მისი დანიშნულებაა დაჭერის ადგილის კოორდინატების განსაზღვრა, რომელთა გაანალიზების შედეგად შემდგომში პროგრამის მიერ ეკრანზე გამოიყვანება შესაბამისი გრაფიკული ან ტექსტური გამოსახულება.

2. სენსორული პანელის სტრუქტურა და მუშაობის პრინციპი

ჩვენ განვიხილავთ რეზისტორული ტიპის ოთხ გამოსასვლელიან სენსორულ პანელს TS12864CRNA. პანელი, რომელიც იყენებს რეზისტორულ ტექნოლოგიებს, შედგება ორი ნაწილისაგან - მოქნილი ზედა და ხისტი ქვედა ფენებისაგან. მოქნილი ფენისათვის გამოიყენება სხვადასხვა პლასტიკური ან პოლიეფირული აფსკები, ხოლო მეორე მზადდება მინისაგან. ორივე ფენის შიგა ზედაპირი დაფარულია დენის გამტარი თხელი ფენით, რომელსაც გააჩნია გარკვეული წინააღმდეგობა . მათ შორის სივრცე შევსებულია მიკროსკოპიული იზოლატორით, რომელიც თანაბრად არის განაწილებული მთელ ზედაპირზე, ყოფს ორ ფენას და არ აძლევს მათ შეხების საშუალებას.

მოქნილ ფენაზე დაჭერის შემთხვევაში, გამტარი ფენები ეხება ერთმანეთს და ქმნის კონტაქტს შეხების წერტილში, ამასთან შეხების წერტილი ქმნის მარტივი ძაბვის გამყოფს. ამის გამო დაჭერის წერტილის კოორდინატები მარტივად შეიძლება გამოითვალოს აცგ-ს საშუალებით ორი გაზომვის შედეგად (ჯერ ერთი კოორდინატის,

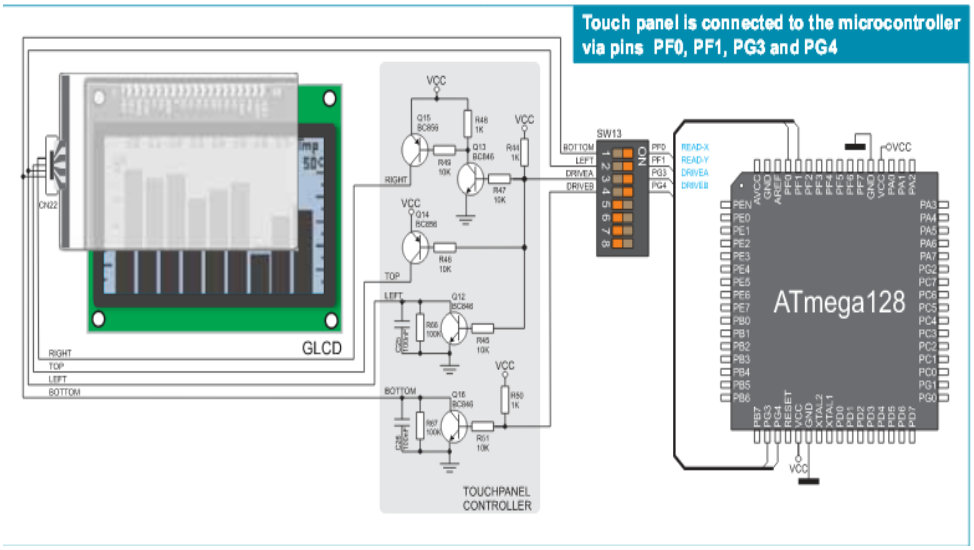
შემდეგ მეორე კოორდინატის). X კოორდინატის გამოთვლისათვის აფსკის განაპირა სალტეზე უნდა მივაწოდოთ მუდმივი ძაბვა, მაგალითად +5ვ (მარცხენა X+ სალტეზე ეწოდება 5ვ, მარჯვენა სალტეზე X- - მიწა). ამის შედეგად, ქვედა ფენის ყოველ ჰორიზონტალურ უბანზე დენი იწვევს ძაბვის ვარდნას, რომლის მნიშვნელობა უბნის სიგრძის პროპორციულია. ეს მნიშვნელობა ჩვენ მიერ უნდა იყოს წაკითხული. ამისათვის ვიყენებთ ზედა ფენას, რომლის სალტეებზე ამ შემთხვევაში ძაბვა არ არის მიწოდებული. მის ერთ-ერთ გამოსასვლელზე, რომელიც აცგ-სთან არის მიერთებული გამოდის ქვედა ფენასთან შეხების წერტილში ძაბვის მნიშვნელობა. ეს იქნება X კოორდინატა. ანალოგიურად წაიკითხება Y კოორდინატა. ამ შემთხვევაში კვება მიეწოდება ზედა ფენის განაპირა სალტეებს (მარცხენა Y+ სალტეს -5ვ, მარჯვენა Y- სალტეს - მიწა). ზედა ფენის დაჭერილ წერტილში ძაბვის მნიშვნელობა მიეწოდება აცგ-ს ქვედა ფენის ერთ-ერთ სალტიდან. სურ.1-ზე ნაჩვენებია სენსორული პანელის ორგანიზაცია.



ნახ.1

3. სისტემის პრინციპიალური სქემა

სენსორული პანელის მართვის სისტემა შედგენილია თვით TS12864CRNA სენსორული პანელისა, WG12864A 128x54 გრაფიკული დისპლეისა და AVR ოჯახის Atmega 128 მიკროკონტროლერისაგან. სქემის რეალიზაციისათვის გამოყენებულია ფირმა MicroElektronics-ის მიერ დამუშავებულ ლაბორატორიული სტენდი BigAVR6. სურ.2-ზე ნაჩვენებია სენსორული პანელის დაკავშირება მიკროკონტროლერის გამომყვანებთან.



ნახ.2

მოყვანილი სქემის მიხედვით მიკროკონტროლერის PG3, PG4 გამოსასვლელები მიერთებულია სენსორული პანელის სამართავ შესასვლელებთან DRIVEA, DRIVEB. ამ გამოსასვლელებით მიკროლონტროლერი გასცემს ორთანრიგა კოდს, რის სფუძველზე პანელის ცალკეული ფენის სალტეებზე ფორმირდებიან X და Y კოორდინატას წაკითხვის ძაბვის მნიშვნელობები. მიკროკონტროლერის PF0, PF1 გამომყვანები უკავშირდებიან სენსორული პანელის გამოსასვლელებს, საიდანაც ამოიკითხებიან დაჭერის ადგილის X და Y კორდინატას მნიშვნელობა. მიკროკონტროლერის გრაფიკულ დისპლეისთან კავშირი ხორციელდება შემდეგი სახით: დისპლეის მონაცემთა გამომყვანები მიერთაბულია პორტ A-სთან, სამართავი შესასვლელები - პორტ E-თან: CS1-PE2-თან; CS2-PE3; RS-PE4; RW-PE5; EN-PE6; RST-PE7.

4. სისტემის ფუნქციონირების ალგორითმი

ა) საწყისი გაწყობის ოპერაციები

- სენსორული პანელის და მიკროკონტროლერის გამომყვანების კავშირის განსაზღვრა;
- პანელზე დაჭერის აღმოჩენის ფუნქციის განსაზღვრა;
- X კოორდინატის განსაზღვრის ფუნქციის წარმოდგენა;
- Y კოორდინატის განსაზღვრის ფუნქციის წარმოდგენა;
- ეკრანის დაკალიბრების ფუნქციის წარმოდგენა.

ბ) ოპერაციები, რომლებიც ქმნის პროგრამის ტანს

- სისტემის ინიციალიზაცია;
- დაჭერის აღმოჩენა;
- ეკრანის დაკალიბრება;
- დაჭერის ადგილის კოორდინატების განსაზღვრა;
- დაჭერის ადგილის შესაბამისი ინფორმაციის გამოყვანა გრაფიკულ დისპლეიზე.

5. სისტემის მუშაობის პროგრამა

ჩვენ მიერ დამუშავებულია მიკროკონტროლერის პროგრამის რამდენიმე ვარიანტი C ენაზე MicroC PRO for AVR პროგრამულ არეში. გამოყენებულია აღნიშნული კომპილატორის ბიბლიოთეკის ბრძანებები. ქვევით მოყვანილია პროგრამის ერთ-ერთი ვარიანტი.

// GLCD *დისპლეის დაკავშირება მიკროკონტროლერთან*

```
1.  char GLCD_DataPort at PORTA;
2.  char GLCD_DataPort_Direction at DDRA;
3.  sbit GLCD_CS1 at PORTE2_bit;
4.  sbit GLCD_CS2 at PORTE3_bit;
5.  sbit GLCD_RS  at PORTE4_bit;
6.  sbit GLCD_RW  at PORTE5_bit;
7.  sbit GLCD_EN  at PORTE6_bit;
8.  sbit GLCD_RST at PORTE7_bit;
9.  sbit GLCD_CS1_Direction at DDE2_bit;
10. sbit GLCD_CS2_Direction at DDE3_bit;
```

```
11. sbit GLCD_RS_Direction  at DDE4_bit;
12. sbit GLCD_RW_Direction  at DDE5_bit;
13. sbit GLCD_EN_Direction  at DDE6_bit;
14. sbit GLCD_RST_Direction at DDE7_bit;
// Touch Panel მოდულის დაკავშირება მიკროკონტროლერთან
15. sbit DRIVE_A at PORTG3_bit;
16. sbit DRIVE_B at PORTG4_bit;
17. sbit DRIVE_A_Direction at DDG3_bit;
18. sbit DRIVE_B_Direction at DDG4_bit;
19. const char READ_X_CHANNEL = 0;  // READ X ხაზი დაკავშირებულია აცგ-ს 0 ანალოგურ არხთან
20. const char READ_Y_CHANNEL = 1;  // READ Y ხაზი დაკავშირებულია აცგ-ს 1 ანალოგურ არხთან
21. unsigned int x_coord, y_coord;
22.  int        cal_x_min, cal_y_min, cal_x_max, cal_y_max; // მაკალიბრებელი კონსტანტები
23.  char write_msg[] = "WRITE";              // GLCD შეტყობინების მენიუ
24.   char clear_msg[] = "CLEAR";
25.    char erase_msg[] = "ERASE";
26.    const unsigned int ADC_THRESHOLD = 900;  // ზღვრის მნიშვნელობა დაჭერის აღმოჩენისათვის
 27. char PressDetect() {    // დაჭერის აღმოჩენის ფუნქცია
 28. unsigned adc_rd;
 29. char result;
 // დაჭერის აღმოჩენა
 30. DRIVE_A = 0;        // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
 31. DRIVE_B = 0;        // DRIVEB = 0 (BOTTOM drive off)
 32. Delay_ms(5);
 33. adc_rd = ADC_Read(READ_Y_CHANNEL);     // Y-წაკითხვა
 34. result = (adc_rd > ADC_THRESHOLD);
 // დაჭერის გამეორება 2 წმ შემდეგ
 35.  Delay_ms(2);
 36.  adc_rd = ADC_Read(READ_Y_CHANNEL);     // Y-წაკითხვა
 37.  result = result & (adc_rd > ADC_THRESHOLD);
 38.  return result;
}
 39. unsigned  int  GetX() // X კორდინატას წაკითხვის ფუნქცია
{
 40. unsigned int result;
 41. DRIVE_A = 1;       // DRIVEA = 1 (LEFT drive on, RIGHT drive on, TOP drive off)
 42. DRIVE_B = 0;            // DRIVEB = 0 (BOTTOM drive off)
 43. Delay_ms(5);
 44.   result = ADC_Read(READ_X_CHANNEL);     // X-ის წაკითხვა (BOTTOM)
 45.   return result;
 }
```

```
46.  unsigned int GetY()  // Y-წაკითხვის ფუნქცია
     {
47.  unsigned int result;
48.  DRIVE_A = 0;      // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
49.  DRIVE_B = 1;     // DRIVEB = 1 (BOTTOM drive on)
50.  Delay_ms(100);
51.  result = ADC_Read(READ_Y_CHANNEL); // Y-ის წაკითხვა  (LEFT)
52.  return result;
}
53. void Calibrate() {              //მაკალიბრებელი ფუნქცია
54. Glcd_Dot(0,63,1);
55. Glcd_Write_Text("TOUCH BOTTOM LEFT",12,3,1);
56.  while (!PressDetect());
57.  cal_x_min = GetX() - 10;
58.  cal_y_min = GetY() - 10;
59.  Delay_ms(1000);
60. Glcd_Fill(0);
61. Glcd_Dot(127,0,1);
62. Glcd_Write_Text("TOUCH UPPER RIGHT",12,4,1);
63. while (!PressDetect()) ;
// მაკალიბრებელი კონსტანტის მიღება
64. cal_x_max = GetX() + 5;
65. cal_y_max = GetY() + 5;
66.Delay_ms(1000);
}
67. void Initialize() {  ინიციალიზაციის ფუნქცია
68. DRIVE_A_Direction = 1;   // DRIVE_ A კონტაქტის დაყენება როგორც გამოსასვლელი
69. DRIVE_B_Direction = 1;  // DRIVE_ B Bკონტაქტის დაყენება როგორც გამოსასვლელი
70. Glcd_Init();                    // GLCD-ს ინიციალიზაცია
}

71. void main() {
72.  Initialize();
73.  Glcd_Fill(0x00);                  // GLCD-ს გასუფთავება
74.  Glcd_Set_Font(font5x7, 5, 7, 32);      // შრიფტის არჩევა
75. Glcd_Write_Text("CALIBRATION", 30, 2, 1);
76. Delay_ms(1500);
77. Glcd_Fill(0);
78. Calibrate();
79. Glcd_Fill(0);
80. Glcd_Write_Text("WRITE ON SCREEN", 20, 5, 1) ;
81. Delay_ms(1000);
82. Glcd_Fill(0);
83. Glcd_Fill(0);
84. Glcd_V_Line(0,7,0,1);
```

```
85.  Glcd_Write_Text(clear_msg,1,0,1);
86.  Glcd_V_Line(0,7,97,1);
87.  Glcd_Write_Text(erase_msg,98,0,1);
88.  Glcd_write_text("Y", 50, 6, 1);
89.  Glcd_write_text("X", 50, 4, 1);
90 .   while (1) {
91.   if(PressDetect())
  {
92.   x_coord = GetX() - cal_x_min;
93.  y_coord = GetY()- cal_y_min;
94.   x_coord = x_coord / 7;
95.    y_coord = (y_coord / 4) - 130;
96.    if (x_coord > 88 && y_coord > 60){
97.      Glcd_Fill(0);
98.      Glcd_write_text("gtu", 2, 0, 1);
        }
99.   else {
100.   Glcd_Fill(0);
101.   Glcd_Circle(60,30,10,1);
         }
102.  Delay_ms(1500);
         }
       }
   }
```

პროგრამის მოყვანილი ვარიანტი ითვალისწინებს სხვადასხვა გამოსახულების გამოყვანას გრაფიკულ დისპლეიზე სენსორულ პანელზე დაჭერილი ადგილის შესაბამისად.

პროგრამის დასაწყისში სრულდება მიკროკონტროლერის და გრაფიკული დისპლეის გამომყვანებს შორის კავშირის (სტრიქონი 1-8) და გადაცემის მიმართულების (სტრიქონი 9-14) გამოცხადება.

15,16 სტრიქონებში ცხადდება კავშირი მიკროკონტროლერის **PG3,PG4** გამომყვანებსა და სენსორული პანელის მართვის **DDRIVEA, DRIVEB** შესასვლელებს შორის, შესაბამისად.17,18 სტრიქონებში განისაზღვრება გადაცემის მიმართულება აღნიშნული გამომყვანებისათვის. 19,20 სტრიქონებში ცხადდება კონსტანტები **READ_X_CHANNEL = 0; READ_Y_CHANNEL =1**, რომლებიც პროგრამაში გამოიყენება შესასვლელი არხის მითითებისათვის აცგ-სთან მიმართვის შემთხვევაში- პირველი აკავშირებს ზედა ფენის BOTTOM სალტეს მიკროკონტროლერის PF0 გამომყვანთან (X კოორდინატის წაკითხვა), მეორე - ქვედა ფენის LEFT სალტეს მიკროკონტროლერის PF1 გამომყვანთან (Y კოორდინატის წაკითხვა). 21 სტრიქონში ცხადდება ცვლადები **x_coord, y_coord,** რომლებშიც პროგრამის შესრულების დროს ჩაიწერება პანელზე დაჭერის კოორდინატები. 22 სტრიქონში გამოცხადებულია მაკალიბრებელი კონსტანტები, რომლითაც ხდება მიღებული კოორდინატების მნიშვნელობების

კორექცია. 23-25 სტრიქონებში ცხადდება ტექსტური მასივი, რომლის ელემენტებში იწერება სხვა და სხვა შეტყობინების ტექსტი. 26 სტრიქონზე განსაზღვრულია კონსტანტა **ADC_THRESHOLD,** რომელშიც ჩაიწერება აცგ-ს გარდასახვის ზღვარი. ეს არის გარდასახული კოდის მნიშვნელობა, როდესაც აცგ-ს შესასვლელზე არ არის მიწოდებული ძაბვა (პანელი არ არის დაჭერილი). აღნიშნული კოდი განისაზღვრება აცგ-ს საყრდენი ძაბვის მნიშვნელობით და წარმოადგენს გარდასახვის ათვლის საწყის მნიშვნელობას (პროგრამაში მიღებულია 900). ამრიგად, აცგ-ს შესასვლელებზე ძაბვის მიწოდების შემთხვევაში (პანელზე დაჭერის დროს) მისი მნიშვნელობა აითვლება გარდასახვის ზღვრიდან.

27-34 სტრიქონზე განსაზღვრულია დაჭერის აღმოჩენის ფუნქცია **PressDetect().** ფუნქცია სრულდება შემდეგი თანმიმდევრობით. პანელის კონტროლერის შესასვლელებს ენიჭებათ მნიშვნელობები **DRIVE_A = 0; DRIVE_B = 0** (30-31 სტრიქონი), რომლის საფუძველზე კონტროლერი პანელის გამომყვანებს აყენებს შემდეგ მდგომარეობაში: ქვედა ფენის სალტეები (**LEFT და RIGHT**) გათიშულია კვების წყაროდან, ზედა ფენის **TOP** სალტეს ეწოდება მაღალი პოტენციალი, **BOTTOM** სალტე აგრეთვე გათიშულია. პანელზე დაჭერით მის **LEFT** გამოსასვლელზე დაფიქსირდება Y კოორდინატას შესაბამისი ძაბვის მნიშვნელობა, ეს გამოსასვლელი დაკავშირებულია აცგ-ს 1 ანალოგურ შესასვლელთან. გარდასახვის შედეგის წაკითხვა სრულდება კომპილატორის ბრძანების **ADC_Read (READ_Y_CHANNEL)** საშუალებით (სტრიქონი 33), რომლის პარამეტრს წარმოადგენს არხი 1-ის ნომერი. წაკითხული მნიშვნელობა იწერება ცვლადში **adc_rd.** როგორც ზევით იყო ნათქვამი, დაჭერის ფაქტი ფიქსირდება, როდესაც გარდასახული კოდის მნიშვნელობა მეტია აცგ-ს გარდასახვის ზღვარზე (პროგრამაში 900) **result** ცვლადში ლოგიკური ერთის ჩაწერით (სრიქონი 34). იმის გასარკვევად, ნამდვილად განხორციელდა დაჭერა თუ არა, 2მწ შემდეგ კვლავ მოწმდება პანელზე დაჭერა. თუ დაჭერა გრძელდება, ეს ნიშნავს, რომ დაჭერა არ ყოფილა შემთხვევითი (35-38 სტრიქონი) და ფიქსირდება დაჭერის ფაქტი **result** ცვლადში.

39-45 სტრიქონებში განსაზღვრულია X კოორდინატას წაკითხვის ფუნქცია **GetX().** ფუნქციის დასაწყისში სენსორული პანელის კონტროლერის შესასვლელებს მიკროკონტროლერიდან მიეწოდება კოდი **DRIVE_A = 1; DRIVE_B = 0** (41,42 სტრიქონი), რის საფუძველზე პანელის შესასვლელებზე იქმნება შემდეგი მდგომარეობა: ქვედა ფენის **LEFT** სალტეს მიეწოდება მაღალი პოტენციალი, **RIGHT** სალტეს - მიწა**,** ზედა ფენის **TOP და BOTTOM** სალტეები გათიშულია კვებისაგან. შედეგად პანელზე დაჭერის შემთხვევაში იზომება ძაბვა ქვედა ფენის დაჭერის წერტილში (X კოორდინატა) და მისი მნიშვნელობა ზედა ფენის **BOTTOM** სალტით მიეწოდება აცგ-ს 0 შემავალ ანალოგურ არხს. 44 სტრიქონზე სრულდება გარდასახული კოდის აცგ-დან ამოკითხვა და **result** ცვლადში ჩაწერა.

46-52 სტრიქონებში განსაზღვრულია Y კოორდინატას წაკითხვის ფუნქცია **GetY().** ფუნქციის დასაწყისში სენსორული პანელის კონტროლერის შესასვლელებს მიეწოდება კოდი **DRIVE_A = 0; DRIVE_B = 1** (48-49 სტრიქონი), რის საფუძველზე

პანელის სალტეებზე იქმნება შემდეგი მდგომარეობა: ქვედა ფენის **LEFT** და **RIGHT** სალტეები გამორთულია, ზედა ფენის **TOP** სალტეს მიეწოდება მაღალი ძაბვა, **BOTTOM** სალტეს -მიწა. ამის შედეგად პანელზე დაჭერის შემთხვევაში იზომება ძაბვა ზედა პანელის დაჭერის წერტილში (Y კოორდინატა) და მისი მნიშვნელობა ქვედა ფენის **TOP** სალტით მიეწოდება აცგ-ს 1 შესასვლელ ანალოგურ არხს. 51 სტრიქონში სრულდება გარდასახული კოდის აცგ-დან ამოკითხვა და **result** ცვლადში ჩაწერა.

53-66 სტრიქონში განსაზღვრულია დაკალიბრების ფუნქცია **Calibrate().** ამ ფუნქციის დანიშნულებაა სენსორული პანელიდან წაკითხული კორდინატი შეუსაბამოს გრაფიკული დისპლეის კოორდინატთა არეს. ფუნქციის დასაწყისში (სტრიქონი 54) დისპლეის ეკრანზე გამოდის წერტილი, რომელიც გვიჩვენებს კოორდინატთა არის მარცხენა ქვედა ზღვარს და აგრეთვე ტექსტი **TOUCH BOTTOM LEFT** (სტრიქონი 55**).** იმ შემთხვევაში, როდესაც სენსორული პანელის კოორდინატთა ბადე ემთხვევა გრაფიკული დისპლეის კოორდინატთა ბადეს, აღნიშნულ წერტილზე დაჭერით უნდა დავაფიქსიროთ იგი როგორც პანელის ქვედა ზღვრული კოორდინატა (56 სტრიქონი). ვინაიდან ბევრ შემთხვევაში პანელიდან მოწოდებული კორდინატები არ ემთხვევიან დისპლეის ზღვრული წერტილის კოორდინატას, სრულდება მაკორექტირებელი კონსტანტების **cal_x_min, cal_y_min** გამოთვლა (სტრიქონი 57,58).

ანალოგიურად სრულდება მარჯვენა ზედა ზღვარის კორექტირება (სტრიქონი 61-66). ეკრანის გასუფთავების შემდეგ (სტრიქონი 60), გამოგვყავს ეკრანის მარჯვენა ზედა წერტილი (სტრიქონი 61) და ტექსტი **TOUCH UPPER RIGHT**. პანელზე აღნიშნულ წერტილზე დაჭერით ფიქსირდება მისი კოორდინატები, საჭიროების შემთხვევაში მათი კორექტირების მიზნით გამოითვლება მაკორექტირებელი კონსტანტები **cal_x_max , cal_y_max** (სტრიქონები 64,65).

67-70 სტრიქონებში განსაზღვრულია ინიციალიზაციის ფუნქცია **Initialize().** ამ ფუნქციის შესრულების დროს ცხადდება **DRIVE_A, DRIVE_B** გამომყვანების გადაცემის მიმართულება (სტრიქონი 68,69) და გრაფიკული დისპლეის ინიციალიზაცია (სტრიქონი 70).

71-ე სტრიქონიდან იწყება მთავარი main() პროგრამა.72 სტრიქონში სრულდება სისტემის ინიციალიზაცია ზევით განხილული **Initialize()** ფუნქციის გამოყენებით. 78 სტრიქონში სრულდება მაკორექტირებელი კონსტანტების გამოთვლა **Calibrate()** ფუნქციის საშუალებით.

90-ე სტრიქონიდან სრულდება დაჭერის აღმოჩენის უსასრულო ციკლი. პანელზე დაჭერის შემთხვევაში (სტრიქონი 91) სრულდება მიღებული კოორდინატების კორექტირება მაკორექტირებელი კონსტანტების საშუალებით (სტრიქონი 92,93).

პროგრამის მომდევნო სტრიქონებში სრულდება დაჭერის ადგილის შესაბამისი ინფორმაციის გამოყვანა ეკრანზე. 96 სტრიქონში მოწმდება სენსორული პანელის არე, რომელშიც იმყოფებიან დაჭერის კოორდინატები. თუ დაჭერის წერტილი იმყოფება წინასწარ განსაზღვრულ არის ფარგლებში (ზევით მოცემულ საილუსტრაციო პროგრამაში x_coord > 88 && y_coord > 60), ეკრანზე გამოვა გარკვეული გამოსახულება (განხილულ პროგრამაში ტექსტი). თუ დაჭერის ადგილი არ იმყოფება მითითებულ არეში, მაშინ ეკრანზე გამოდის სხვა გამოსახულება (განხილულ პროგრამაში ნახატი).

ზოგადად შეგვიძლია განვსაზღვროთ პანელის რამდენიმე არე უფრო რთული პირობით, სხვადასხვა არეზე დაჭერის შემთხვევაში ეკრანზე გამოვა ჩვენ მიერ წინასწარ შექმნილი შესაბამისი გრაფიკული ან ტექსტური გამოსახულება.

ლიტერატურა - References – Литература:

1. ქართველიშვილი ო., ხოშტარია ც., ხოშტარია ს. (2015). მიკროპროცესორული სისტემები. I ნაწ., მიკროკონტროლერის არქიტექტურა. სტუ. „ტექნიკური უნივერსიტეტი“. თბ.

2. ქართველიშვილი ო., ხოშტარია ს. (2016). მიკროპროცესორული სისტემები. მეთოდ. მით. ლაბ.სამუშაოსთვის. „ტექნიკური უნივერსიტეტი“. 2016.

3. ქართველიშვილი ო., ხოშტარია ს. (2017). მიკროპროცესორული სისტემები. II ნაწ., მიკროპროცესორული სისტემების დაგეგმარება. ელ-სახელმძღ., სტუ-ს ცენტრალური ბიბლიოთეკა. Atmega 128 Datasheet. http://www.atmel.com/images/doc2467.pdf

4. Подключение резистивной сенсорной панели к микроконтроллеру. http://avrproject-.ru/publ/kak_podkljuchit/touchscreen_avr_bascom/2-1-0-45

CONTROL OF A TOUCH-SENSITIVE PANEL USING MICROCONTROLLER

Kartvelishvili Otar, Khoshtaria Simon
Georgian Technical University

Summary

The present article discusses possibilities of controlling the touch-sensitive screen by microcontroller. A touch-sensitive panel, graphical display and AVR family microcontroller Atmega 128 are used in this project. The control algorithm and one of the program options in C language are shown, which allows output of various types of information – either text or graphical, depending on the touch location. System operation is confirmed at the lab stand.

УПРАВЛЕНИЕ СЕНСОРНЫМ ЭКРАНОМ С ПОМОЩЬЮ МИКРОКОНТРОЛЛЕРА

Картвелишвили О., Хоштария С.
Грузинский Технический Университет

Резюме

Рассмариваются возможности управления сенсорного экрана с помощью микроконтроллера. В проекте используются сенсорная панель, графический дисплей и микроконтроллер Atmega 128 семейства AVR. Показан алгоритм управления и один из вариантов программы на языке Си, с помощью которой возможно выводить на екран информацию различного характера – текстовую или графическую, в зависимости от места нажатия экрана. Работа системы была проверена на лабораторном стенде.