**Galvos sekimo įrenginys. Duomenų iš daviklio nuskaitymas.**

1. int main(void)
2. {
3. // inializuojam pagrindinius kintamuosius
4. int8\_t ret1,ret2,status;
5. int16\_t acc\_x,acc\_y,acc\_z;
6. int16\_t mag\_x,mag\_y,mag\_z;
7. int16\_t gyro\_x,gyro\_y,gyro\_z;
8. int16\_t ret;
9. int16\_t temperature;
10. char buffer[100];
12. // inicializuojami reikia davikliams išėjimai
13. DDRB=0xFF;
14. PORTB=0xFF;
16. // įjungiame globalias pertrauktis
17. sei();
19. // inicializuojam UART (RS232)
20. uart\_init( UART\_BAUD\_SELECT(UART\_BAUD\_RATE,F\_CPU) );
22. // inicializuojam I2C
23. i2c\_init();
25. // saugumo sumetimais palaukiame 1 s.
26. \_delay\_ms(1000);
28. // inicializuojam daviklius
29. Init\_MAG();
30. Init\_ACC();
31. Init\_GYRO();
33. // saugumo sumetimais palaukiame 2 s.
34. \_delay\_ms(2000);
36. // tikrinimas ar davikliai veikia
37. ret1 = Read\_I2C\_Reg(GYRO\_ADR,0x0f);
38. sprintf(buffer,"GYRO: 0x%2.2x\r\n",ret1);                                                   ");
40. ret=Read\_I2C\_Reg(ACC\_ADR,0x20);
41. sprintf(buffer,"ACC: 0x%2.2x\r\n",ret);
43. ret=Read\_I2C\_Reg(MAG\_ADR,0x21);
44. sprintf(buffer,"MAG: 0x%2.2x\r\n",ret);
45. ");
47. char c;
48. while(1)
49. {
50. {
52. // ciklas sukasi tol kol daviklis nepasiruošęs perduoti duomenų
53. while(!(ret1=Read\_I2C\_Reg(ACC\_ADR,ACC\_STATUS) && (ACC\_READY))){};
54. // nuskaitomi visi daviklio duomenys
55. ret1 = Read\_I2C\_Reg(ACC\_ADR,X\_L\_A);
56. ret2 = Read\_I2C\_Reg(ACC\_ADR,X\_H\_A);
57. acc\_x = Fix\_Value(ret1,ret2);
58. ret1 = Read\_I2C\_Reg(ACC\_ADR,Y\_L\_A);
59. ret2 = Read\_I2C\_Reg(ACC\_ADR,Y\_H\_A);
60. acc\_y = Fix\_Value(ret1,ret2);
61. ret1 = Read\_I2C\_Reg(ACC\_ADR,Z\_L\_A);
62. ret2 = Read\_I2C\_Reg(ACC\_ADR,Z\_H\_A);
63. acc\_z = Fix\_Value(ret1,ret2);
65. // ciklas sukasi tol kol daviklis nepasiruošęs perduoti duomenų
66. while(!(ret1=Read\_I2C\_Reg(MAG\_ADR,MAG\_STATUS) && (MAG\_READY))){};
67. // nuskaitomi visi daviklio duomenys
68. ret1 = Read\_I2C\_Reg(MAG\_ADR,X\_L\_M);
69. ret2 = Read\_I2C\_Reg(MAG\_ADR,X\_H\_M);
70. mag\_x = Fix\_Value(ret1,ret2);
71. ret1 = Read\_I2C\_Reg(MAG\_ADR,Y\_L\_M);
72. ret2 = Read\_I2C\_Reg(MAG\_ADR,Y\_H\_M);
73. mag\_y = Fix\_Value(ret1,ret2);
74. ret1 = Read\_I2C\_Reg(MAG\_ADR,Z\_L\_M);
75. ret2 = Read\_I2C\_Reg(MAG\_ADR,Z\_H\_M);
76. mag\_z = Fix\_Value(ret1,ret2);
78. // ciklas sukasi tol kol daviklis nepasiruošęs perduoti duomenų
79. while(!(ret1=Read\_I2C\_Reg(GYRO\_ADR,GYRO\_STATUS) && (GYRO\_READY))){};
80. // nuskaitomi visi daviklio duomenys
81. ret1 = Read\_I2C\_Reg(GYRO\_ADR,X\_L\_G);
82. ret2 = Read\_I2C\_Reg(GYRO\_ADR,X\_H\_G);
83. gyro\_x = Fix\_Value(ret1,ret2);
84. ret1 = Read\_I2C\_Reg(GYRO\_ADR,Y\_L\_G);
85. ret2 = Read\_I2C\_Reg(GYRO\_ADR,Y\_H\_G);
86. gyro\_y = Fix\_Value(ret1,ret2);
87. ret1 = Read\_I2C\_Reg(GYRO\_ADR,Z\_L\_G);
88. ret2 = Read\_I2C\_Reg(GYRO\_ADR,Z\_H\_G);
89. gyro\_z = Fix\_Value(ret1,ret2);
91. temperature = Read\_I2C\_Reg(GYRO\_ADR,TEMP);
93. // formuojamas duomenų paketas
94. sprintf(buffer,"%d %d %d %d %d %d %d %d %d %d\r\n\0",temperature, gyro\_x, gyro\_y, gyro\_z, mag\_x, mag\_y, mag\_z, acc\_x, acc\_y, acc\_z);
96. // duomenis isunčiamu UART sąsaja atgal į kitą irenginį
97. uart\_puts(buffer);
98. }
99. \_delay\_ms(SEND\_TIMEOUT-2);
101. }
102. }