

27.11.2024

Pyry Laine

50282836

Pistelaskuri -projekti

EE.ELE.500-2024-2025-1 Mikrokontrollerijärjestelmät

Idea

Olen jo pidempään yrittänyt löytää aikaa omalle projektilleni ATmega328P -arkkitehtuurilla, jotta oppisin paremmin kuinka se toimii käytännössä. ATmega328P on kaltaiselleni aloittelijalle loistava valinta, sillä sen käskykanta on vielä ymmärrykseni tavoitettavissa, mutta taipuu yksinkertaisuudestaan huolimatta moneen.

Käydessäni läpi ajan saatossa kerryttämäni kasaa komponentteja löysin Arduino Uno rev 3 -kehitysalustan, PN532 NFC RFID -lukijan sekä 16x2 LCD -näytön. Mieleeni nousi kuva lukuisten Yatzy -pelien runtelemapa vihkosesta mökin laatikossa ja päätin tehdä sille korvaajan.

Laitteen toiminta on siis seuraavanlainen: Pelaajat etsivät itselleen RFID kortin, jolla rekisteröityvät laitteeseen. Saadessaan pisteitä pelaaja pitää RFID tunnistettaan sydän merkin päällä kunnes näytöllä on vilissyt oikea määrä pisteitä. Pelaajat voivat pelin edetessä tarkistaa pistesaldonsa käyttämällä korttia lukijalla pikaisesti. Laitteen tiedot voidaan lopuksi resetoitua katkaisemalla virta.

Laitteisto

Kyseiseen mikrokontrollerijärjestelmään kuuluu siis Arduino Uno rev 3 -kehitysalusta, PN532 NFC RFID -lukija sekä 16x2 LCD -näyttö. Lisäksi tilasin paristopitimen, jotta laitteesta saataisiin kätevämpi langaton malli. Varalle löytyi jännityssäädettävä virtajohto.

Jotta sain näytön ja lukijan sopimaan kehitysalustan sisääntuloihin, kytkin RFID lukijan UART väylän mukaan ja näytön I2C mukaan. Paristopitimen kanssa varoin tilaamasta sellaista, joka jännitteellään voisi rikkoa mikrokontrollerin. Myöhemmin opin, että kehitysalusta kestää jännitettä yllättävän hyvin virransäätelynsä ansiosta ja nopeasti tuli myös selväksi, ettei tilaamani paristopitimen jännite edes riitä komponenttien yhteiseen toimintaan.



Laitteiston komponentit

Ohjelmisto

ATmega328 -arkkitehtuurin parhaita puolia on sille soveltuvien oheislaitekirjastojen määrä. Aikaisemmista projekteista viisastuneena varmistin jo ennen osien yhteen liittämistä, että tälle arkkitehtuurille löytyy juuri näitä oheislaitteita vastaavat tuet.

Ohjelman pääsilmutta kutsuu jatkuvasti funktiota *readNFC()*, jolla RFID -lukija haistelee tarjotaanko sille korttia vaiko ei. Kun kortti löytyy, LCD-näytölle päästetään virta ja kortin yksilöivä tunnus otetaan ylös.

Kortin tunnisteen perusteella funktio *PlayerFinder* käy läpi jonon *PlayerRegister*. Mikäli RFID kortin tunniste ei löydy jonosta, luodaan uusi pelaaja tietorakenteella *player*, joka sisältää "Boolean" -arvon *newPlayer*, pistemäärän *score* sekä merkkijonon *id* kortin tunnisteele. Pelaajat asetetaan jonoon *PlayerRegister* rekisteröitymisjärjestyksessä ja pelaajan nimi saadaan näytölle jonon järjestyksnumerosta.

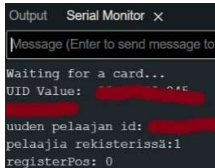
Tämän jälkeen kun kyseisen pelaajan sijainti *PlayerRegister* -jonossa on palautettu funktiosta, alkaa näytössä esitys, jossa pelaajaa tervehditään sen mukaan onko hän uusi vai vanha, kerrotaan pisteet ja pyydetään pitämään tunnistetta vielä lukijassa, mikäli halutaan kerryttää lisää pisteitä.

Mikäli kortti havaitaan olevan vielä tarjolla, päästään silmukkaan, jossa pisteitä lisätään kunnes kortti irtaoo lukijasta. Kortin irrottua kertyneet pisteet kirjataan *PlayerRegister:n* pelaajalle ja näyttö sammutetaan virran säästämiseksi.

Koska mahdollisuutta debugaamiseen ei ollut, ohjelmakoodin testaaminen suoritettiin perinteisemmällä menetelmällä eli viljelemällä Serial.print -komentoja koodin suorituksen varrelle, jotta päästiin näkemään mihin asti päästiin ja missä muodossa keskeiset tietorakenteet milloinkin olivat.

```
Serial.print("uuden pelaajan id: ");
for (uint8_t i = 0; i < uidlength; i++){Serial.print(newp.id[i]);}
Serial.println();

PlayerRegister[players_added] = newp;
++players_added;
Serial.print("pelaajia rekisterissä:"); Serial.println(players_added);
```



Ohjelman testaus käytännössä

Yritin aikarajoitusteni puitteissa muodostaa ohjelmakoodistani sellaista, että se suoriutuisi tehtävistään mahdollisimman tehokkaasti niin laskennan kuin muistinkin näkökulmasta, mutta jo näinkin pienessä koodissa huomasi kuinka parannettavaa on aina.

Haasteet ja opit

Projektin alussa yritin olla kaukaa viisas ja valita ohjelmointiympäristökseni tutun Microchip Studion. Emme kuitenkaan tule kyseisen ympäristön kanssa juuri toimeen jostain syystä ja

kaksi päivää suostuteltuani ympäristöä hyväksymään projektin oheislaittekirjastoja jouduin luovuttamaan. Yritin vielä sovittaa Arduinon komentorivipohjasta versiota VSCode ympäristööni, mutta sen tuki oli ilmeisesti lakkautettu jo aikapäiviä sitten. Näistä nöyrytyneenä jouduin siirtämään projektin perinteiselle Arduino IDE:lle, vaikka se ei sisältänyt edes debuggaus mahdollisuutta.

Kuten demotessani valmiihkoa projektiani huomattiin, RFID lukija ei oikein jaksanut ylläpitää kortin lukua pahvilaatikon lävitse. Tämän sain kuitenkin korjattua ohjelmallisesti lisäämällä tarkastuskertoja. Tarkistin vielä, että ohjelmakoodiin määrittämäni standardi vastasi RFID korttien standardia, sillä lukija tuki aika joukkoa erilaisia standardeja.



RFID kortin standardin määrittäminen

Kuten aikaisemmin jo mainittiin, tilaamani paristokotelon jännite ei riittänyt laitteen LCD näytölle. Nopealla aikataululla en onnistunut löytämään laitteen sisälle mahtuvaa vastinetta, joka olisi saman aikaa paloturvallinen ja riittävän halpa.

Vaikka tämä projekti olikin suhteellisen yksinkertainen ja nopea tunkkaus, sisälsi se lukuisasti oppeja, joita aineistoa lukemalla ei oppisi. Edellä mainittujen kommellusten lisäksi ymmärrän nyt miksi sulautettujen järjestelmien parissa työskentelevä ystäväni ei pidä Arduinon kehitysalustoja juuri minään, mutta sen sijaan hehkuttaa STM:n vastaavia laitteita. Syy tähän on hinnan lisäksi STM:n ohjelmointiympäristö *CubeIDE*, joka mahdollistaa hieman monimutkaisemmatkin projektit.

Hyödynnetyt lähteet

<https://github.com/elechouse/PN532>

<https://www.arduino-libraries.info/libraries/liquid-crystal-i2-c>

<https://how2electronics.com/interfacing-pn532-nfc-rfid-module-with-arduino/>

Liitteet



Demo laitteen toiminnasta

```
1. //proessorin kirjastot
2. #include <avr/io.h>
3. #include <util/delay.h>
4. #include <avr/sleep.h>
5. #include <stdio.h>
6. #include <string.h>
7.
8. //lisälaitteiden kirjastot
9. #include <SoftwareSerial.h>
10. #include <PN532_SWHSU.h>
11. #include <PN532.h>
12. #include <LiquidCrystal_I2C.h>
13.
14. // Oheislaitteikirjastojen määrittelyjä
15. SoftwareSerial SWSerial( 3, 2 ); // RX, TX
16. PN532_SWHSU pn532swhsu( SWSerial );
17. PN532 nfc( pn532swhsu );
18. String tagId = "None", dispTag = "None";
19. byte nuidPICC[4];
20.
21. LiquidCrystal_I2C lcd(0x27,16,2);
22.
23. struct player
24. //Tietorakenne rekisteröidylle pelaajalle
25. {
26.     uint8_t newPlayer = 1;
27.     int score;
28.     uint8_t id[7];
29. };
30.
31. // Global muuttujat
32. uint8_t player_count = 10; // vaikka kymmenen pelaajan maksimi
33. uint8_t players_added = 0;
34. player PlayerRegister[10];
35.
36. void setup(void)
37. //määrittelyt omissa funktioissaan modulaarisuuden parantamiseksi
38. {
39.     Display_setup();
40.     NFCsetup();
41. }
42.
43. void NFCsetup()
```

```

44. //RFID lukijan määrittelykset
45. {
46.   Serial.begin(115200); //Baud rate
47.   nfc.begin();
48.   nfc.SAMConfig();
49. }
50.
51. void Display_setup()
52. //LCD näytön määrittelykset
53. {
54.   lcd.begin(16,2);
55.   lcd.noBacklight(); //näyttö pois päältä
56.   //lcd.backlight();
57. }
58.
59. void loop()
60. //Pääsilukija
61. {
62.   readNFC(); // Näyttöön valot vasta kun kortti havaitaan
63. }
64.
65. uint8_t CardPresent()
66. {
67.   uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0, 0 }; // Bufferi tunnisteen varastointiin
68.   uint8_t uidLength;
69.   return nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);
70. }
71.
72. void readNFC() //kortinlukijan logiikka
73. {
74.   uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0, 0 };
75.   uint8_t uidLength;
76.   delay(500); //UX
77.
78.   if(nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength))
79.   {
80.     uint8_t registerPos = PlayerFinder(uid, uidLength); //käydään Pelaajarekisteri läpi
81.     printDisplay(registerPos); //näyttö esitys alkaa
82.   }
83. }
84.
85. void printDisplay(uint8_t registerPos)
86. {
87.   uint8_t player = registerPos + 1;
88.   uint8_t score = PlayerRegister[registerPos].score;
89.
90.   lcd.backlight();
91.
92.   if(PlayerRegister[registerPos].newPlayer == 1)
93.   // uusi käyttäjä tervehdys
94.   {
95.     lcd.setCursor(0,0); lcd.print("Hello");
96.     delay(1000);
97.     lcd.setCursor(0,1); lcd.print("stranger");
98.     delay(2000);
99.     lcd.clear();
100.    delay(1000);
101.
102.    lcd.setCursor(0,0); lcd.print("You're ");
103.    delay(1000);
104.    lcd.setCursor(7,0); lcd.print("player ");
105.    lcd.setCursor(14,0); lcd.print(player);
106.    delay(1000);
107.
108.    lcd.setCursor(0,1); lcd.print("with score: ");
109.    delay(1000);
110.    lcd.setCursor(12,1); lcd.print(score);
111.    delay(2000);
112.    lcd.clear();
113.    delay(1000);

```

```

114. }
115. else
116. // vanhan käyttäjän tervehtiminen
117. {
118.     lcd.setCursor(0,0);
119.     lcd.print("Hello ");
120.     delay(1000);
121.     lcd.setCursor(6,0); lcd.print("player ");
122.     lcd.setCursor(13,0); lcd.print(player);
123.     delay(1000);
124.
125.     lcd.setCursor(0,1);
126.     lcd.print("Your score: ");
127.     delay(1000);
128.     lcd.setCursor(12,1);
129.     lcd.print(score);
130.     delay(2000);
131.     lcd.clear();
132.     delay(1000);
133. }
134. // ohjeet
135. lcd.setCursor(0,0);
136. lcd.print("Hold card to");
137. lcd.setCursor(0,1);
138. lcd.print("increase score");
139. delay(2000);
140. lcd.clear();
141. delay(1000);
142.
143. while(CardPresent() || CardPresent() || CardPresent())
144. // Pisteiden lisäys. Kolme tarkastusta joka pisteen kohdalla viansiedollisista syistä
145. {
146.     lcd.setCursor(0,0); lcd.print("Player ");
147.     lcd.setCursor(7,0); lcd.print(player);
148.     lcd.setCursor(0,1); lcd.print("Score: ");
149.     lcd.setCursor(8,1); lcd.print(score);
150.     delay(1000);
151.     ++score;
152. }
153.
154. if(PlayerRegister[player-1].score != score){PlayerRegister[player-1].score = score-1;}
155.
156. lcd.clear();
157. lcd.noBacklight();
158. }
159.
160. uint8_t PlayerFinder(uint8_t uid[], uint8_t uidlength)
161. //Luo tarvittaessa pelaajan ja palauttaa tunnistekortin ID:tä vastaavan pelaajan paikan
162. //pelaajarekisterissä
163. {
164.     if(players_added != 0)
165.     //Pelaajarekisteri tyhjä
166.     {
167.         for(int i=0;i<players_added;i++) //etsitään pelaajaa rekisteristä
168.         //per pelaaja
169.         {
170.             uint8_t new_player = 0;
171.             for (uint8_t ii = 0; ii < uidlength; ii++)
172.             {
173.                 if(PlayerRegister[i].id[ii] != uid[ii])
174.                 //per tunnisteen merkki
175.                 {
176.                     new_player = 1;
177.                     break;
178.                 }
179.             }
180.             if(new_player == 0)
181.             //tunniste löytyy pelaajarekisteristä
182.             {
183.                 PlayerRegister[i].newPlayer = 0;

```

```
183.         return i;
184.     }
185. }
186. }
187.
188. struct player newp;
189. for (uint8_t i = 0; i < uidlength; i++){newp.id[i] = uid[i];}
190. newp.score = 0;
191. newp.newPlayer = 1;
192.
193. PlayerRegister[players_added] = newp;
194. ++players_added;
195.
196. return players_added-1;
197. }
198.
```

Ohjelmakoodi