***Grunberg Edar-Albert***

***335CC***

**Proiect PM – Simon-Whack – README**

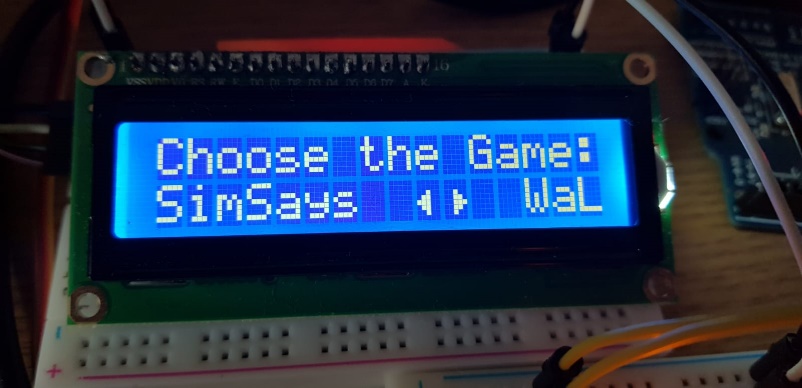
**Descrierea implementării Software:**

* + **Variabile importante**:
  + uint8\_t LEDS[5] = {LED7, LED8, LED9, LED10, LED11}; (vectorul în care salvez pinii la care sunt conectate LED-urile).
  + int ledSounds[5]; (sunetele asociate fiecărui led în parte)
  + int scoruriJucatori[2]; (scorurile ambilor jucători)
  + int lives[2] = {3, 3}; (viețile celor 2 jucători)
  + int randomArr[Sequence\_Len]; (se salvează aici ordinea în care s-au aprins LED-urile, mai exact indecșii LED-urilor, pentru a le putea accesa din vectorul LEDS)
  + int servoPositions[5] = {180, 135, 80, 30, 1}; (valorile, în grade, pentru ca servomotorul să știe să arate spre led-ul corect)
  + uint8\_t duck[8], leftTrangle[8], rightTrangle[8], sadFace[8], happyFace[8] (acești vectori sunt folosiți pentru a desena caracterele proprii pe ecran)
* **Funcții importante**:

1. **ISR(PCINT2\_vect)** - aici se intră atunci când se apasă butonul joystick-ului, conectat la prinul digital 4 și se tratează input-ul dat de jucător pentru ambele jocuri
2. **ISR(TIMER1\_COMPA\_vect)** - aceasta este întreruperea pe timer 1, cu TOP OCR1A, setat in setup() astfel încât să se intre la fiecare 20ms. Această întrerupere controlează servomotorul (**digitalWrite(SERVOPIN, HIGH)**), aprinde și stinge led-urile în cadrul meniului de selecție a jocurilor la fiecare 1 secundă trecută (adică atunci când s-a intrat de 50 de ori pe întrerupere) și ajută la alegerea led-ului în cardul jocului Simon Says.

Funcția **setup()** se ocupă de afișarea inițială pe ecran a jocurilor ce pot fi alese și de setarea corectă a regiștrilor. (pentru timer 1 COMPA și COMPB, LED-uri, etc.).

Totul începe în funcția **loop()**. La început aceasta apelează funcția **wait\_for\_choice()**, unde se stă într-o buclă până se selectează jocul dorit, cu ajutorul Joystick-ului (stânga-dreapta). Pentru ca să se înțeleagă ce joc este selectat, am creat câteva caractere pentru LCD-ul meu. Pentru început, pe ecran sunt 2 triunghiuri ce pointează către un joc fiecare. De exemplu: Dacă se alege jocul din dreapta (WaL), atunci triunghiul din dreapta se transformă într-o față veselă, iar cel din stânga într-una tristă.

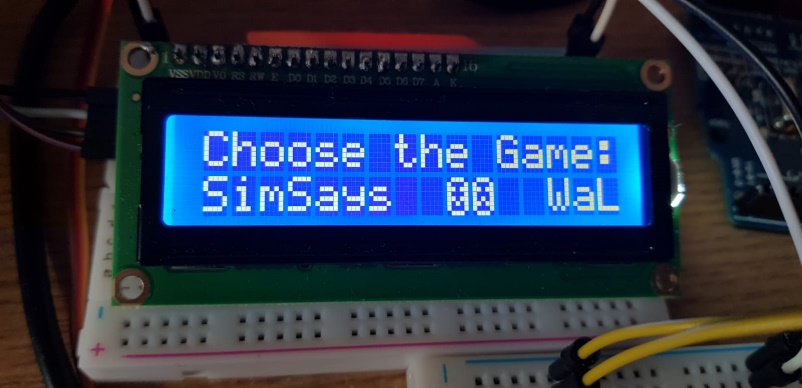
[](https://ocw.cs.pub.ro/courses/_detail/pm/prj2021/dbrigalda/selectie-joc-sim-whack.jpg?id=pm%3Aprj2021%3Adbrigalda%3Asimon-whack)

Mențiune: butonul Joystick-ului funcționează pe intrerupere (pe pin-ul digital 4), iar înclinarea Joystick-ului la stânga sau la dreapta o citesc din registrul ADCH (ce ia valori între 0 și 3), pentru că ADCL e mereu 255, normal, adică, folosesc noțiuni de AVR pentru citirea valorilor date de Joystick.

Mențiune: Scrierea pe LCD o fac în marea majoritate cu funcția LCD\_print\_prompt care scrie 2 string-uri pe cele 2 linii. Unde nu am folosit funcția **LCD\_print\_prompt()** pentru a scrie pe ecran, înseamnă ca am scris si caracterele mele speciale, cu funcția **lcd.write()**.

După ce s-a ales jocul, buzzer-ul începe să cânte o melodie realizată de mine în funcția **play\_start\_song()** din **songs.h** și led-urile încep să facă un “joc de lumini”, se aprind toate pe rând si apoi se sting. (cu funcția **turn\_on\_all\_leds()** din **ledPlay.h**). Acest lucru se întâmplă pentru ambele jocuri la fel.

* Dacă jocul ales este Simon Says:

[](https://ocw.cs.pub.ro/courses/_detail/pm/prj2021/dbrigalda/selectie-simon.jpg?id=pm%3Aprj2021%3Adbrigalda%3Asimon-whack)

Apoi, după această introducere în joc se aprind la întâmplare led-uri (cu funcția **random()** ce are seed pin-ul analogic A3), iar ordinea în care s-au aprins este salvată în vectorul randomArr. Ca să se delimiteze secvența generată random de momentul în care jucătorul trebuie să o reproducă, led-urile se aprind si se sting toate după ce s-a terminat secvența (**turn\_off\_all\_leds()** + **turn\_on\_all\_leds()**).

Acum se intră în funcția **wait\_for\_sequence()**, unde au loc următoarele lucruri: jucătorul poate folosi Joystick-ul (stânga-dreapta) pentru a selecta led-ul dorit. Acțiunea aceasta se realizează în 2 locuri: in loop() se incrementează sau se decrementează o variabilă ce reprezintă index-ul în cadrul vectorului LEDS, iar pe timer 1 (cu TOP la OCR1A, ce se activează la fiecare 20ms) se stinge și se aprinde la fiecare o secundă led-ul selectat, adică cel aflat la index-ul incrementat sau decrementat de Joystick.

În tot acest timp, cu diverse funcții, afișez pe ecran Scorul si Viețile rămase.

Tot în acest moment, când se așteaptă ca jucătorul să termine sau să greșească secvența, se mai poate face un lucru. Apăsând butonul conectat la pinul digital 2 (cu intrerupere atașată cu funcția attachInterrupt()), se activează servomotorul, adică știind index-ul LED-ului ce trebuie selectat pentru a continua secvența, servomotorul va arăta spre valoarea de la același index, din vectorul servoPositions[5] (care conține valori în grade, de la 1 la 180). Servomotorul este acționat astfel: la fiecare 20 ms, **ISR(TIMER1\_COMPA\_vect)** activează pin-ul digital 3 (cu PWM) și se intră pe **ISR(TIMER1\_COMPB\_vect)**, unde se pune același pin pe valoarea LOW. Pentru a schimba poziția servomotorului trebuie modificată valoarea lui OCR1B. Făcând calculele (regula de 3 simplă), am aflat intervalul de valori pe care OCR1B le poate lua pentru a muta servomotorul între 0 si 180 grade. Aici, deoarece am setat prescaler-ul pentru timer la valoarea 64, rezoluția servomotorului este una bună, valorile fiind între 250 si 500 pentru OCR1B. Totuși, deoarece Servomotorul începea să se miște brusc la 1s și să se întoarcă la loc, am decis ca de fiecare dată când se cere hint, să arate spre led-ul bun, iar apoi să se întoarcă la o valoare de referință (respectiv 125 - adică SERVONOTICK).

**int SERVO\_convert\_from\_degrees(int grade);**

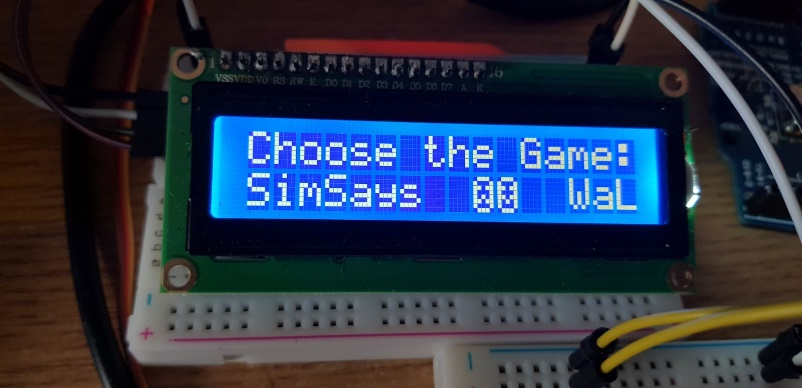
Extra, aici am făcut o funcție de conversie din valori 200-500 în 0-180 ca să lucrez mai ușor, în grade.

Jocul se încheie atunci când jucătorul a făcut 3 greșeli, adică când și-a consumat toate cele 3 vieți.

Pentru următorul jucător se reia procesul, diferit pentru el este jocul led-urilor de la început.

La final, după ce s-a terminat jocul, tot procesul se reia.

* Dacă jocul ales este Whack-a-Led:

[](https://ocw.cs.pub.ro/courses/_detail/pm/prj2021/dbrigalda/selectie-whack.jpg?id=pm%3Aprj2021%3Adbrigalda%3Asimon-whack)

Jocul acesta este mai simplu, se aprinde un singur LED la întâmplare după ce a cântat melodia la început și led-urile și-au terminat jocul de lumini. Acest led este memorat în variabila **whackLed**.

La fel ca la Simon Says, pentru a anunța jucătorul că e rândul său să se joace, se aprind toate led-urile, apoi se sting.

Pe ecran acum se afișează scorul, numărul de vieți rămase și căte vieți are LED-ul ce trebuie înfrânt.

Acum, Din loop() se intră în funcția **wait\_for\_whacking()**. Aici se aprinde de la stânga la dreapta câte un LED (într-un while din care se iese ori când led-ul nu mai are vieți, ori când nu mai are jucătorul vieți). În momentul în care este aprins led-ul pe care trebuia să-l țină minte jucătorul, se apasă butonul Joystick-ului (se intră pe întreruperea asociată lui, unde se verifică dacă este led-ul bun și se activează flag-uri, astfel încât când se reia funcția wait\_for\_whacking să se poată actualiza ecranul) și scorul jucătorului creste cu 1 punct, iar led-ului i se va lua o viață. Dacă a apăsat pe joystick când alt led s-a este aprins, jucătorului i se scade o viață și se reia de la început până când nu mai are vieți rămase.

Asemenea jocului Simon Says, și în acest joc paote fi acționat Servomotorul, pentru a-i aduce aminte jucătorului care este LED-ul ce trebuie înfrânt.

La final, după ce s-a terminat jocul, tot procesul se reia.

* **Bibliotecile extrene** pe care le folosesc:

1. **LiquidCristal.h** (+/- **Wire.h**): pentru ecranul LCD
2. **pitches.h**: pentru notele/frecventele emise de Buzzer (multe define-uri pentru fiecare notă și frecvența aferentă)
3. **songs.h**: aici sunt funcțiile pentru melodia de început de joc și pentru sunetele de pierdere rundă
   1. void play\_start\_song(int theDelay); (melodie de început de joc)
   2. void play\_fail\_song(); (greșire secvență)
   3. void play\_next\_lvl\_song(); (trecere la următorul nivel)
   4. void play\_right\_song(); (secvență reprodusă corect \ trecere la următoarea rundă)
4. **ledPlay.h**: aici sunt jocurile de lumini (de început de rundă, de pierdere rundă, de trecere la următorul nivel)
   1. void turn\_on\_all\_leds(int rounds, bool sequence, bool leftRight, int myDelay, int LEDSNO);
   2. void turn\_off\_all\_leds(int LEDSNO);

**Descrierea implementării Hardware:**

Modul în care am asamblat toate modulele / piesele chiar din prima zi în care m-am apucat îl aveam stabilit în minte.

Ecranul LCD și modului I2C de pe spatele lui a trebuit să le lipesc cu Letcon-ul pentru că au venit separat, întrucât nu era în stoc ecranul lipit deja. Prima oară am greșit destul de mult, nu se aprindea LCD-ul, lipisem prea mult fludor, iar Letconul a decedat în mâna mea. Am comandat alt LCD, am continuat proiectul cu acesta nou, dar după ce am terminat de scris codul, am decis să repar greseala, asa ca am cumparat un kit complet pentru lipit si totul a decurs mult mai bine, pentru că in momentul acesta LCD-ul pe care il folosesc este primul pe care l-am cumparat.

Servomotorul l-am lipit cu picătura de Breadboard ca să nu mai cadă atunci când firul este tras din greseala de mine sau cand este actionat.

În rest, totul a decurs perfect. Nu am avut nevoie de Tinkercad pentru a testa, am facut totul live, pe placuta.