Simon-Whack

Grupa: 335CC

Nume: Grunberg Edar-Albert

E-mail: Login to see contact info.

Link proiect: https://ocw.cs.pub.ro/courses/pm/prj2021/dbrigalda/simon-whack

Introducere

- Proiectul constă în implementarea a 2 jocuri:
  1. Simon Says (Joc de Memorie) - 1 vs 1
  2. Whack-a-Led (Joc de Concentrare și Memorie) - Single Player

La început, se alege jocul dorit cu ajutorul Joystick-ului. Apoi, buzzer-ul va cânta o melodie de start joc.

- **Simon Says:**

Acest joc se va juca cu ajutorul unui joystick. Pe breadboard vor fi plasate 5 LED-uri (rosu-galben-albastru). Înainte ca jucătorul să realizeze o acțiune, el va aștepta si va memora ordinea în care se vor aprinde x din cele 5 led-uri. Mai departe, înclinând Joystick-ul în stânga sau dreapta, jucătorul va putea selecta led-ul dorit. Dacă nu este sigur ce led urmează într-o secvență, atunci, pierzând din scor un punct, jucătorul poate cere un hint. Acesta va fi realizat printr-un servomotor ce va indica spre led-ul ce trebuie selectat.

- **Whack-a-Led**:

Jocul începe la fel ca primul, în sensul că va cânta melodia de început, apoi led-urile se vor lumina

într-o ordine. Apoi un singur led se va aprinde, la întâmplare. După, începe jocul propriu-zis: cu un delay se vor aprinde led-urile de la stânga la dreapta, iar jucătorul va trebui să apese pe joystick în momentul în care este aprins led-ul pe care l-a văzut la început, pentru a-i scădea numărul de vieți. Led-ul va avea, la întâmplare, între 2 și 5 vieți, iar ca să termine o rundă, va trebui să i se ducă toate viețile led-ului.

Pe parcursul jocurilor, pe un ecran LCD se va afișa textul “Your Score is: …” și un număr ce reprezintă viețile rămase.

Pentru ambele jocuri se va ține o evidență a scorului jucătorilor (pentru primul la final se compară rezultatele și pe LCD se va afișa câștigătorul, iar pentru al doilea se memorează rezultatul pentru a-l afișa la finalul jocului).

Pentru mine, proiectul acesta este fascinant, întrucât pot învăța cum funcționează un joystick (precum cele utilizate în controller-ele de PlayStation sau XBox). De asemenea, îl consider o provocare și un mod de a-mi satisface plăcerea de a inventa ceva.

Descriere generală

Se incepe prin alegerea jocului de catre unul din cei 2 jucatori. Pe ecran se pot vedea la început numele jocurilor (SimSays sau WaL) și 2 triunghiuri ce vor pointa către cele 2 jocuri. Selecția se realizează cu ajutorul Joystick-ului, înclinându-l la dreapta (pentru Whack-a-Led) sau la stânga (pentru Simon Says). În tot acest timp, led-urile de pe placuță luminează la întamplare (adică, cât timp se decide ce joc doresc jucătorii să joace, LED-urile se sting și se aprind la întâmplare, pentru a nu fi totul monoton). Când se alege jocul, acestea se sting și se afișează pe ecran “Game Starts! Player 1”.

- **Pentru Simon Says**:

Simon Says se desfășoară pe runde si nivele (fiecare nivel are 3 runde). Nivelul reprezintă numărul de led-uri ce se vor aprinde în secvență. Dacă jucătorul greșește un led din secvență, atunci led-urile vor face blink de 3 ori, toate, deodată, iar dacă secvența a fost corectă de la cap la coadă, atunci led-urile fac blink o singură dată (adică atunci când se câștigă o runda). Totuși, ca să se înțeleagă când se trece la un nou nivel, led-urile se vor aprinde de la stânga la dreapta cu un delay.

După ce led-urile se aprind singure în secvența întâmplătoare, ele vor face toate blink o singură dată, pentru a anunța jucătorul că este rândul său să reproducă secvența.

Pentru a ști ce led este selectat, acesta va face blink (în continuu, pe o intrerupere).

Joystick-ul va incrementa sau decrementa (stânga / dreapta) un index, ce va referi câte un LED. Scorul va depinde de corectitudinea cu care a apăsat acel led.

Buzzer-ul este folosit pentru a “reda o melodie de start” când începe jocul, pentru a anunța jucătorul că a selectat LED-ul potrivit sau nu (sunete pentru selecție bună și pentru selecție greșită).

Servomotorul este acționat cu ajutorul unui buton atunci când jucătorul trebuie să restabilească secvența de culori și va indica spre led-ul ce trebuie selectat, dar jucătorul va pierde 1 punct din scor-ul actual.

Butonul este folosit pentru a folosi Servomotorul (pentru a cere hint)

Pe **LCD** se afișează scorul, viețile rămase, jucătorul este anunțat că va începe jocul, iar la final sunt afișate scorurile ambilor jucători și câștigătorul (cel care a avut scorul cel mai mare)

- **Pentru Whack-a-Led**:

La Whack-a-Led nu mai sunt nivele si runde, este doar un joc repetitiv, singurul lucru diferit la fiecare "rundă" este LED-ul pe care trebuie să îl infrângă jucătorul și delay-ul cu care se schimbă ce led se arpinde (pentru că este un număr random).

Pentru a alege un led trebuie apăsat **butonul Joystick-ului**.

Servomotorul este folosit pentru a aduce aminte jucătorului care este led-ul ce trebuie înfrânt

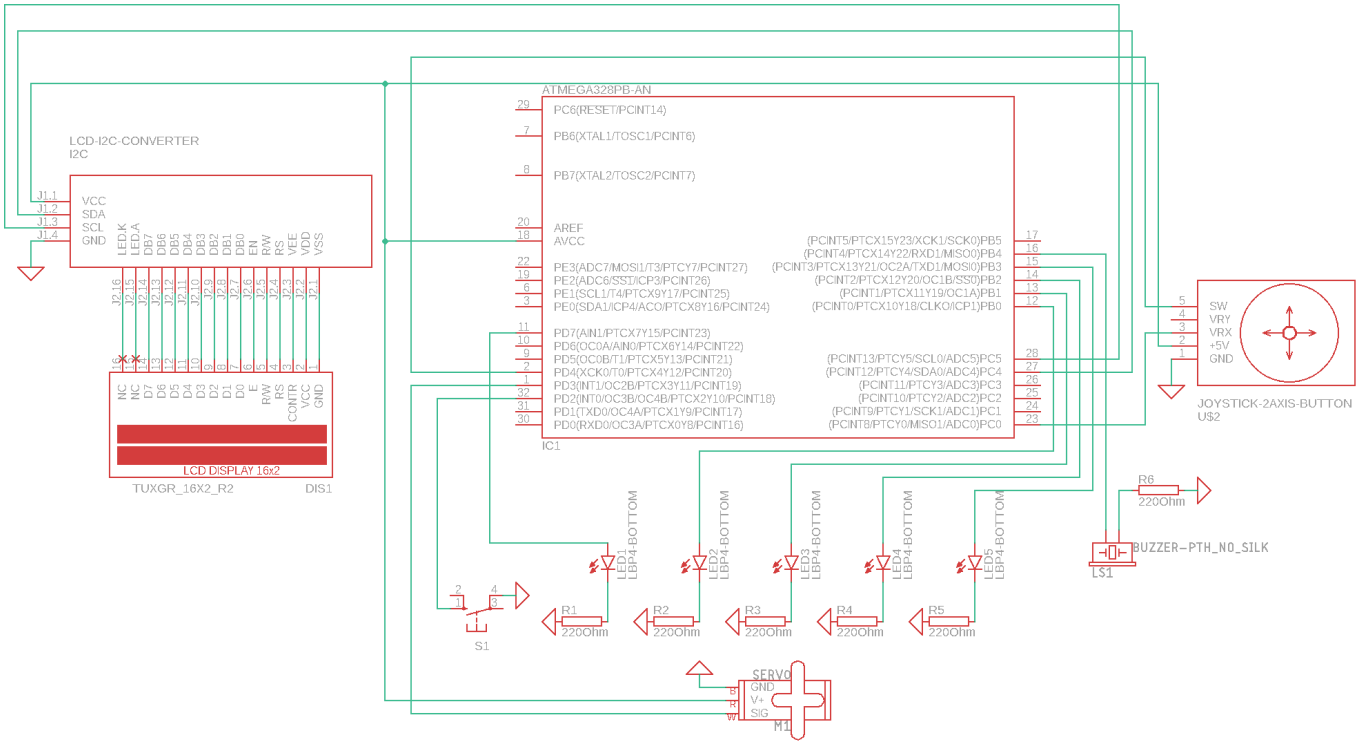
- **Concepte de Laborator Folosite**:
  1. **Întreruperi**: pentru butoane și pentru întreruperile pe timer
  2. **PWM**: Servomotorul
  3. **ADC**: Joystick-ul
  4. **I2C**: Ecranul LCD 1602 (ce are și potențiometru, pentru ajustarea luminozității)
  5. **Laboratoarele introductive**: Led-urile
- **Schema Bloc**:

Hardware Design

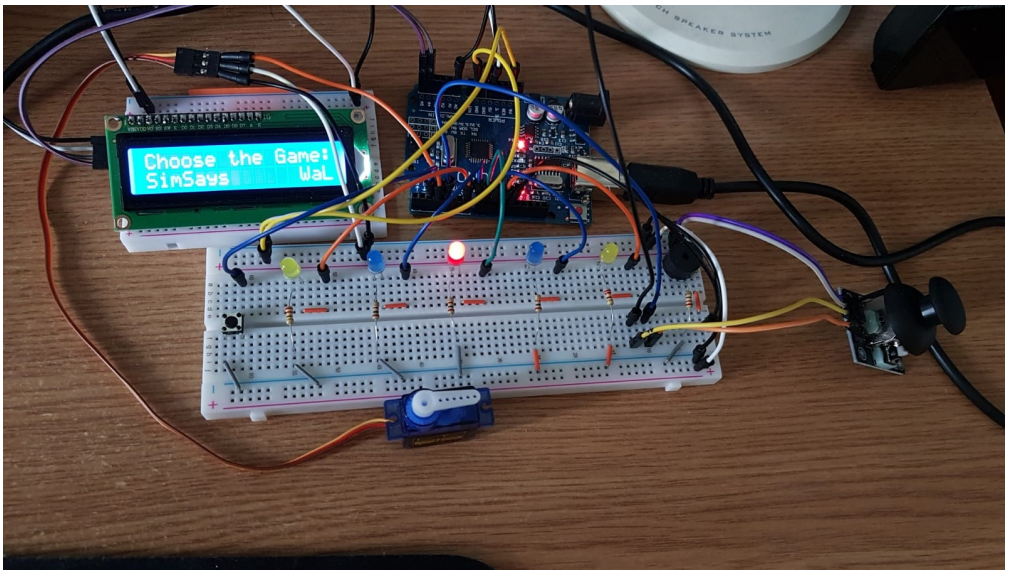
- **Lista de Componente**:

Număr Piese	Denumire Piese	Descriere
1	Arduino UNO	29 lei (Ardushop)
2	Cablu USB alimentare	2 lei (Ardushop)
30	Rezistențe de 2k și de 220	3 lei (Ardushop)
100	Fire Jumper și rigide	12 lei + 12 lei (Ardushop + Optimus Digital)
2	Modul Joystick Biaxial Negru cu 5 pini	2 x 5.49 lei (Optimus Digital)
30	Led-uri: albastre, roșii, galbene	~5 lei (Ardushop)
2	Ecran LCD 1602 cu convertor I2C	2 x (12 lei + 5 lei) (Ardushop)
1	Servomotor	14 lei (Ardushop)
1	Breadboard Micutz + Breadboard Măricel	3 lei + 10 lei (Ardushop + Optimus Digital)

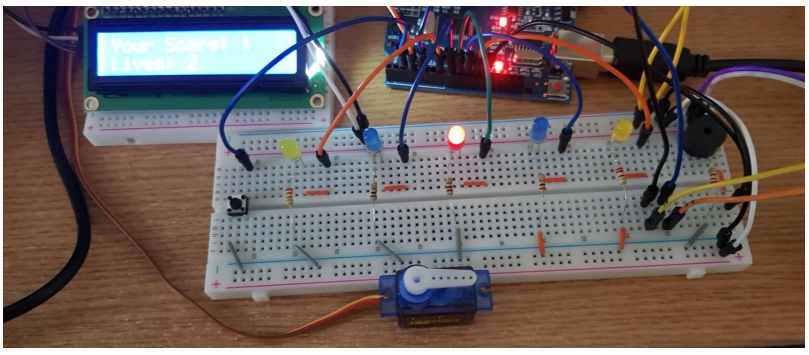
- **Schema Electrică**:



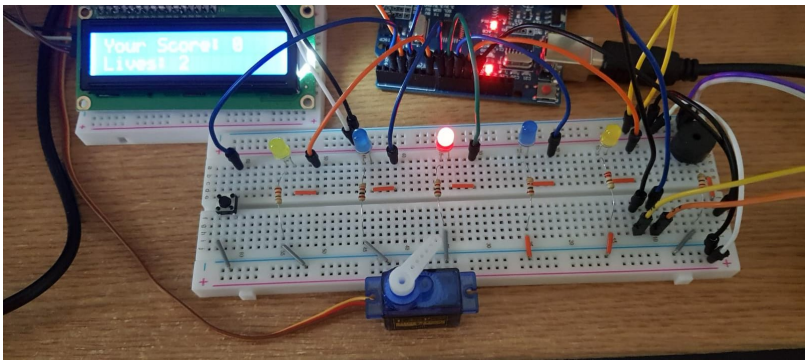
- **Hardware-ul realizat**:
- **Meniu Selecție Joc**:



- **Înainte de a cere hint**:



- **După ce s-a cerut hint**:



Software Design

Am ales sa folosesc drept **mediu de dezvoltare** Arduino IDE.

- **Bibliotecile externe** 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:
  - **Variabile importante**:
  1. uint8_t g_LEDS[5] = {LED7, LED8, LED9, LED10, LED11}; (vectorul în care salvez pinii la care sunt conectate LED-urile).
  2. int g_ledSounds[5]; (sunetele asociate fiecărui led în parte)
  3. int g_scoruriJucatori[2]; (scorurile ambilor jucători)
  4. int g_lives[2] = {3, 3}; (viețile celor 2 jucători)
  5. int g_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)
  6. int g_servoPositions[5] = {180, 135, 80, 30, 1}; (valorile, în grade, pentru ca servomotorul să știe să arate spre led-ul corect)

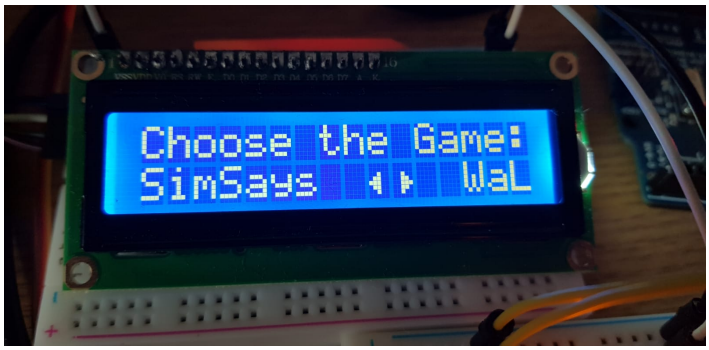
7. uint8_t g_duck[8], g_leftTrangle[8], g_rightTrangle[8], g_sadFace[8], g_happyFace[8] (acești vectori sunt folosiți pentru a desena caracterele proprii pe ecran)

- **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.

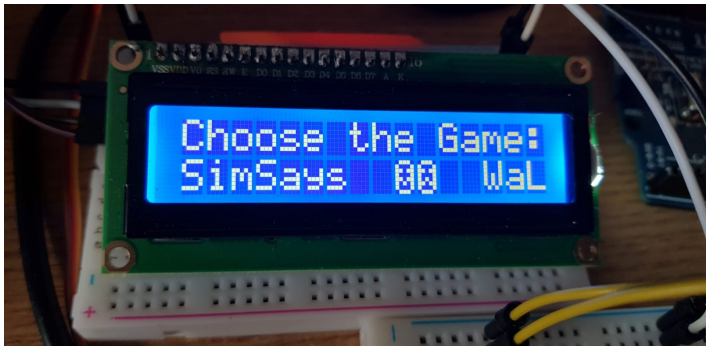


Mențiune: butonul Joystick-ului funcționează pe intrerupere (pe pin-ul digital 4), iar înclinarea Joystick-ului la stânga sau la dreapta o citesc din registrul ADCH (ce ia valori între 0 și 3), pentru că ADCL e mereu 255, normal, adică, folosesc noțiuni de AVR pentru citirea valorilor date de Joystick.

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:



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);

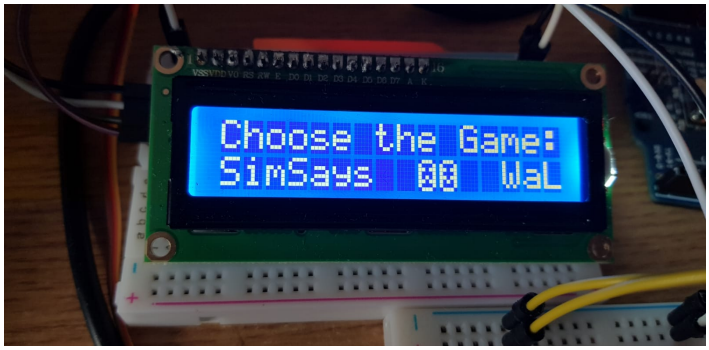
Extra, aici am făcut o funcție de conversie din valori 200-500 în 0-180 ca să lucrez mai ușor, în grade.

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:



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 **g_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 câte 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.

Rezultate Obţinute

Proiectul mi-a ieșit de la cap la coadă, exact așa cum am vrut eu. Totuși, mă gândeam ca al doilea joc, pe care inițial l-am numit Whack-a-Mole, să fie o copie a jocului cu același nume, dar mi-am dat seama că îmi va fi greu să am câte un buton pentru fiecare LED. De aceea am decis să îl redenumesc Whack-a-Led și să inventez propriile mele reguli.

- **Link-ul** către filmul proiectului meu:

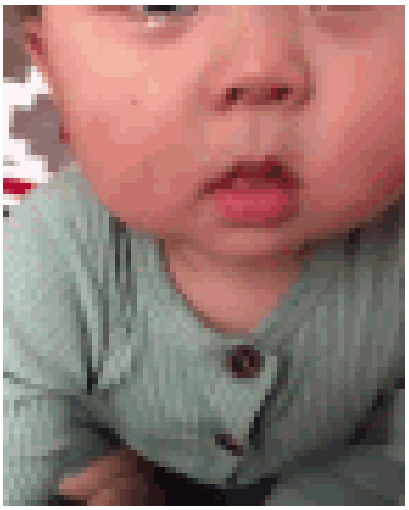
Concluzii

După multe momente în care credeam că nu o să pot să termin proiectul deoarece vedeam că nu voi putea să folosesc toate componentele pe care doream să le folosesc, am reușit într-un final să finalizez, totul fiind funcțional.

Totuși, cele mai importante realizări pentru mine au fost faptul că am înțeles cum se lucrează cu un Servomotor, un buzzer și un LCD în același timp. Inițial, deoarece doream să folosesc funcții deja existente precum Servo.write() sau tone() sau lcd.print(), mi-am dat seama că este aproape imposibil. Multe din ele se foloseau de același timer (1) pe care îl foloseam și eu pentru blink-ul led-urilor. Astfel, am fost nevoit să învăț cum controlez Servomotorul direct pe Timer, cu întreruperi, iar scrierea pe LCD a trebuit să o fac numai în afara întreruperilor, deoarece biblioteca pentru LCD-ul meu cu convertor I2C nu permitea acest lucru.

De asemenea, cu ocazia acestui proiect am învățat cum să folosesc letcon-ul pentru a lipi pinii modulului I2C pe spatele ecranului LCD 1602, lucru care prima oară a eșuat, dar am rectificat această greșeală pe parcurs.

Realizarea acestui proiect m-a motivat să vreau să mai folosesc plăcuța Arduino și pentru alte proiecte personale legate în special de senzori.



Download

- Arhiva cu toate sursele:

Simon-Whack.zip

README_SOFT_HARD-WARE
LiquidCrystal_I2C.h

Jurnal

- **22.04.2021**: Au sosit toate piesele de care am nevoie.
- **23.04.2021**: Am implementat logica de selectare a led-urilor cu ajutorul Joystick-ului și am conectat toate piesele (am făcut un prototip), în afară de servomotor. (Încep cu jocul Simon Says). Am creat această pagină de wiki pentru proiect.
- **25.04.2021**: Am terminat intreaga logica a jocului Simon Says. (Am implementat calculul scorului, pierderea vieților si hotărârea câștigătorului pentru acest joc, am reușit sa-l fac pe ture. Mai intai joaca player 1, face un scor, apoi joaca player 2 si face si el un scor, iar la final, când își pierd viețile, pe ecranul lcd se afiseaza si castigatorul.) Am definitivat pagina de wiki pentru etapa 1.
- **27.04.2021**: Am adăugat si buzzer-ul, alături de sunete pentru trecerea la următoarea runda, la următorul nivel, sunet pentru fiecare led in parte și pentru pierderea unei runde.
- **30.04.2021**: Am adăugat și servomotorul. Am realizat controlul său cu ajutorul Timer-ului 1 (pe OCR1B).
- **07.05.2021**: Am rezolvat diverse bug-uri legate de selecția LED-urilor cu ajutorul Joystick-ului (este mult mai rapidă selecția).
- **17.05.2021**: Am schimbat prescaler-ul timer-ului 1 și precizia este mai bună acum pentru servomotor. De asemenea am modificat din analogRead în lucru pe regiștrii pentru a lua valoarea dată de Joystick din ADCH. Primul joc este complet.
- **19.05.2021**: Am realizat schema electrică în Eagle.
- **20.05.2021**: Am realizat cel de-al doilea joc. Mi-am dat seama că pot crea propriile mele caractere pe LCD și din acest motiv am schimbat radical modul în care selectez jocul la început, nu mai face blink numele jocului, ci se selecteaza înclinând Joystick-ul stânga sau dreapta și prin afișarea de caractere speciale pe ecran. Am înfrumusețat aspectul codului.
- **21.05.2021**: Am rectificat câteva greșeli pe pagina de OCW.
- **23.05.2021**: Am definitivat pagina de OCW.

Bibliografie și Resurse

- **Youtube**:

1. https://www.youtube.com/watch?v=MlDi0vO9Evg&ab_channel=Brainy-Bits
2. https://www.youtube.com/watch?v=q9YC_GVHy5A&t=322s&ab_channel=Robojax (foarte frumos lipitul de pini)
3. https://www.youtube.com/watch?v=XN7a0z1gjsI&t=280s&ab_channel=Barqunics (un fel de inspiratie)

- **Laboratoarele de pe OCW**: in special cel cu intreruperi si PWM pentru servomotor.
- **Google**
- **Datasheet-ul Atmega 328P**:
  - https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf
- **Biblioteca pentru LCD cu modul I2C**:
  - https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
- **Informații pentru funcția tone()**:
  - https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

Documentul în format PDF: 335CC_Grunberg_Edar-Albert

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2021/dbrigalda/simon-whack

Last update: **2021/06/04 19:17**