Tilt the maze - Dumitrescu Alexandra 333CA

Demo

Introducere

Proiectul constă într-un labirint ce poate fi rotit de jucător pe 2 axe OX și OY printr-un joystick. Motivația proiectului meu a constat în dorința automatizării unui joc din copilărie. Labirintul dispune 5 găuri, iar jucătorului îi este oferită o singură bilă magnetică. Jocul propune următoarele 2 moduri:

1. *Save The Ball* în acest mod, *labirintul este inamicul*, iar scopul jocului este să salvezi mingea din a cădea în capcanele labirintului. Labirintul urmărește mișcările date de jucător și își recalculează la fiecare pas **modul cel mai optim în care să încurce jucătorul**. În momentele de stagnare ale jucătorului, **labirintul poate avea comportament nedeterminist**. Timpul este cronometrat, iar scopul este să reziști cât mai mult fără să scapi mingea.
2. *Send The Ball* în acest mod, scopul este să distribui mingea în gaura aleasă drept *gaură de finish*. La începutul partidei, o gaură aleatoare este aleasă drept finish, semnalizat printr-un LED adecvat, iar scopul este să **rotești labirintul fără să scapi mingea în găurile nepotrivite într-un timp cât mai scurt**, având o singură încercare.

Descriere generală

Jucătorul primește un controller format dintr-un Arduino Nano la care este conectat un joystick. Arduino Nano va transmite prin intefața serială USART mesaje către un alt Arduino Uno, comunicându-i datele transmise prin joystick. Arduino Uno în funcție de datele primite acționeaza cele 2 servo motoare care deplasează cadranul și labirintul pe 2 axe. La începutul jocului, utilizatorul are o interfață grafică pe ecranul LCD în care trebuie să își selecteze modul jocului dorit. Fiecare gaură din interiorul labirintului are un senzor magnetic Hall care va detecta dacă bila a căzut în gaura potrivită sau nu și va decide rezultatul partidei pe care îl va afișa pe LCD împreună cu timpul cronometrat.

Hardware Design

Componentele necesare realizării proiectului sunt

1. Arduino Uno ATmega328P

2. Arduino Nano
3. senzori magnetici Hall
4. ecran LCD
5. 2 X servomotoare

Mapa labirintului va fi realizată din piese lego și va avea următoarea formă

În vederea realizării proiectului, voi folosi următoarele laboratoare

1. Laborator 1. comunicare pe interfața serială USART între Arduino Uno și Arduino Nano
2. Laborator 2. întreruperi pentru cei 5 senzori magnetici
3. Laborator 6. I2C pentru ecranul LCD
4. Laborator 3. PWM pentru servomotoare

Schema circuitului

Software Design

Descriere generală

Pentru realizarea proiectului am utilizat Arduino IDE, în care am importat următoarele biblioteci

1. **SoftwareSerial.h** - pentru comunicarea pe interfața serială între cele 2 Arduino-uri [1)]
2. **Servo.h** - pentru a deplasa cele două servomotoare [2)]
3. **LiquidCrystal_I2C.h** - pentru a scrie rezultatele obținute pe ecranul LCD cu modulul I2C incorporat

În cadrul proiectului, am utilizat și operații pe regiștri pentru

1. **Întreruperi hardware** - pentru a detecta când este apăsat butonul joystickului am utilizat o întrerupere externă pe pinul PD2
2. **Timer** - pentru a afișa durata jocului am setat Timerul 1 în modul CTC să genereze o întrerupere cu frecvența 1 Hz

Implementare

Arduino Uno 1. - Controller Servo & Joystick - Primul Arduino are următoarele roluri
- primește date de la joystick
- încearcă să reducă pe cât posibil zgomotul posibil provocat de jucător pentru a asigura o deplasare

lină a labirintului
- să mapeze datele de intrare din intervalul [0, 1024) în intervalul [25°, 45°)
- să asigure viteza între rotații să fie corespunzător de mică, tot pentru a asigura o deplasare lină a labirintului - să trimită la începutul jocului prin interfața serială Arduino-ului 2. informații primite de la joystick privind începerea jocului și alegerea game modului *Save the Ball* sau *Send the Ball*.
- în cazul în care a fost selectat modul de joc *Save the ball*, Arduino-ul generează random la fiecare 10s o poziție pe OX și pe OY în care să se deplaseze labirintul.
Putem să le luăm pe rând, în ordinea cronologică începerii jocului

- Prima dată, am setat un **JOYSTICK_DEADZONE** pentru a asigura că nu există niciun fel de zgomot în momentul în care labirintul stagnează în **SERVO_DEFAULT_POSITION** (35°), practic atunci când jucătorul nu mișcă joystick-ul sau îl mișcă suficient de puțin, labirintul trebuie să sea pe loc pentru a reduce zgomotele. Dacă datele depășesc **JOYSTICK_DEADZONE**, atunci vom mapa datele primite pe OX și OY în intervalul [25°, 45°).

```
int joystick_deadzone = 200;
(...)
void loop() {
  int ox = analogRead(A3);
  int oy = analogRead(A2);

   /* check and adjust deadzone */
  if (abs(ox - 512) < joystick_deadzone) {
    ox = 512;
  }

  if (abs(oy - 512) < joystick_deadzone) {
    oy = 512;
  }

   int servo_ox_pos = map(ox, 0, 1023, 25, 45);
   int servo_oy_pos = map(oy, 0, 1023, 25, 45);
}
```

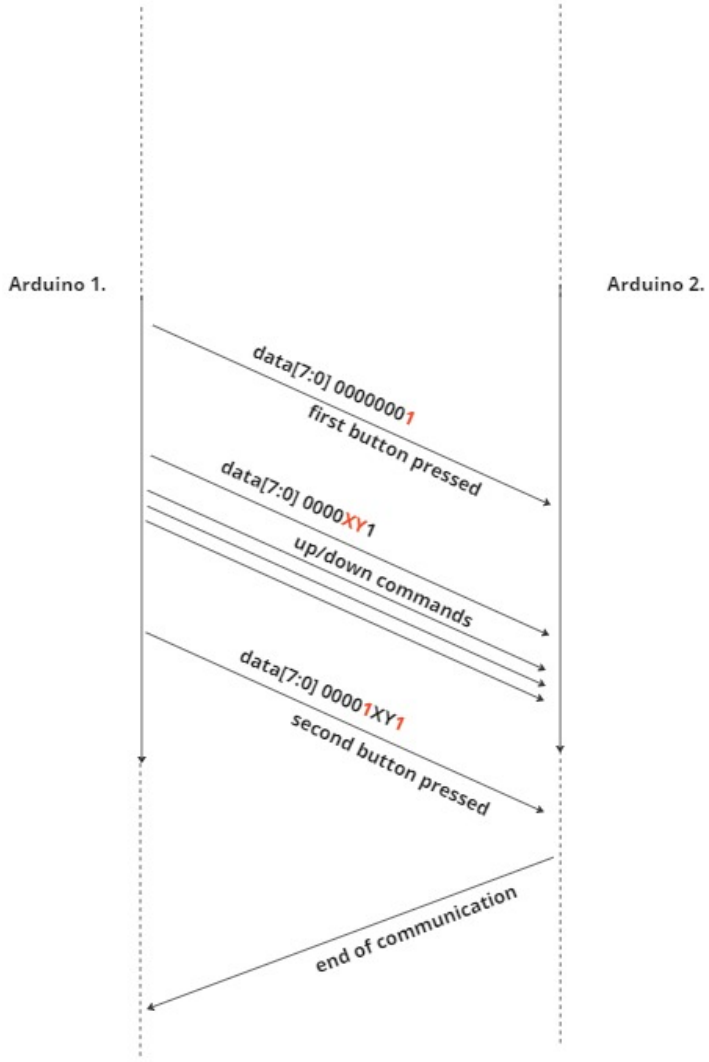
- Variabila **IS_START_GAME** detectează dacă jocul a început sau nu. Pentru a monitoriza acest aspect, contorizăm folosind întreruperea hardware externă generată de apăsarea butonului joystickului, de câte ori a fost apăsat butonul. Prima apăsare anunță începerea jocului, iar a doua selectarea modului de joc. Deoarece pot apărea probleme la apăsare continuă a butonului, în sensul că întreruperea externă se poate activa de mai multe ori pentru o singură apăsare, am implementat o soluție în care se asigură că butonul este apăsat, apoi eliberat și abia apoi o altă apăsare este validată. *Jocul începe o dată ce s-au trimit cele 2 apăsări de buton, iar Arduino 2. trimite înapoi modul de joc selectat*

```
void loop() {
  if(mySerial.available() >= 1) {
      byte finished = mySerial.read();
      is_start_game = true;
      chosen_game_mode = finished;
  }
```

```
}
(...)
ISR(INT0_vect) {
  int value = PIND & (1 << JOYSTICK_BUTTON);
  if(is_button_pressed == true && old_value != 0) {
    is_button_pressed_second = true;
  }
  if(value == 0) {
    is_button_pressed = true;
  }
  old_value = value;
}
```

- Pentru a nu aglomera comunicarea pe interfața serială (deoarece ar putea interveni desincronizări în primirea/trimiterea datelor în Receiver / Transmitter sau datele ar putea fi primite cu o întârziere care nu este necesară), am propus o soluție în care pe interfață se comunică doar la începutul jocului câte un **singur byte la un moment dat, în care biții codifică starea actuală a joystickului**. Nu ne interesează decât up/low/pressed pentru că interfața de pe LCD permite alegerea game modului prin swipe up/low. **Între cele Arduinouri se stabilește un protocol de comunicație. Ambele părți trebuie să cunoască decizia de game mode luat și, implicit încheierea conexiunii. Din punctul alegerii jocului, Arduino-urile au taskuri separate specializate pe modul de joc.**

button[0:7]	Semnification
button[0]	Button Pressed
button[1]	Up
button[2]	Low
button[3]	2nd Button Pressed



- O dată ce game modeul a fost setat se așteaptă un byte înapoi de sincronizare între cele 2 Arduino-uri, care semnifică game modeul selectat, după care începe jocul, pornind servomotoarele și dându-le ca direcție datele mapate [25°, 45°) de la joystick. Pentru a asigura o viteză acceptabilă astfel încât să nu se miște brusc/dezordonat labirintul am lasat un delay pentru fiecare unghi în care se deplasează servomotorul.

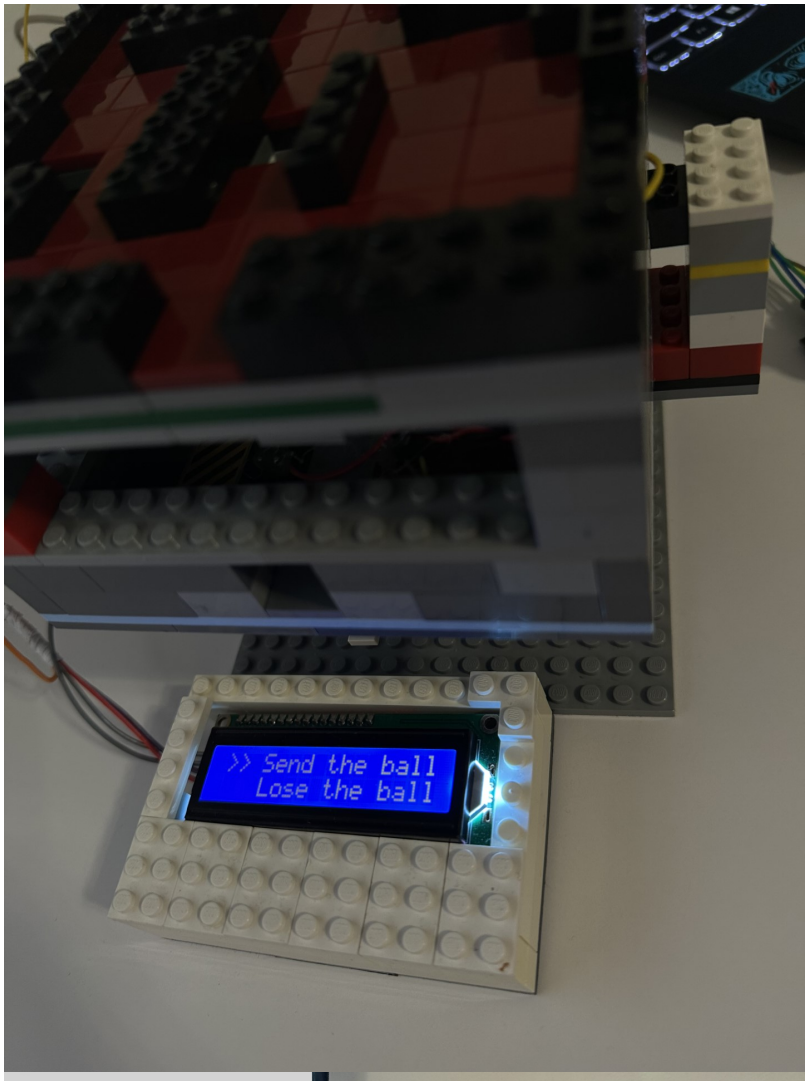
Arduino Uno 2. - Controller Senzori & LCD & Led - Al doilea Arduino are următoarele roluri
- Primind datele de pe interfața serială de la Arduino 1. setează logica celor 2 moduri de joc.
- Primul mod, *Save the Ball* așteaptă ca pentru fiecare gaură să detecteze senzorul magnetic bila magnetică. Dacă bila a fost detectată, se oprește și afișează timpul jocului.
- Al doilea mod, *Send The Ball*, generează random o gaură, implicit pornește LED-ul găurii respective, pentru a anunța jucătorul unde trebuie să trimită mingea, și așteaptă ca senzorii să detecteze fie dacă mingea a picat în gura corectă, fie nu. Va afișa rezultatul **FAIL/PASS** și durata jocului.

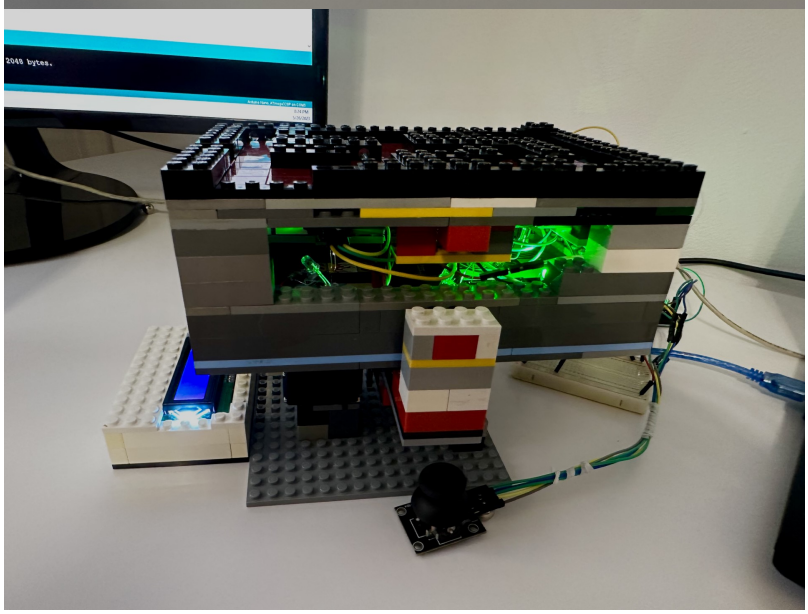
```
void loop() {
if(mySerial.available() >= 2) {
    /* extract info from protocol */
    byte received = mySerial.read();
    byte button_pressed = (received & 0x01);
```

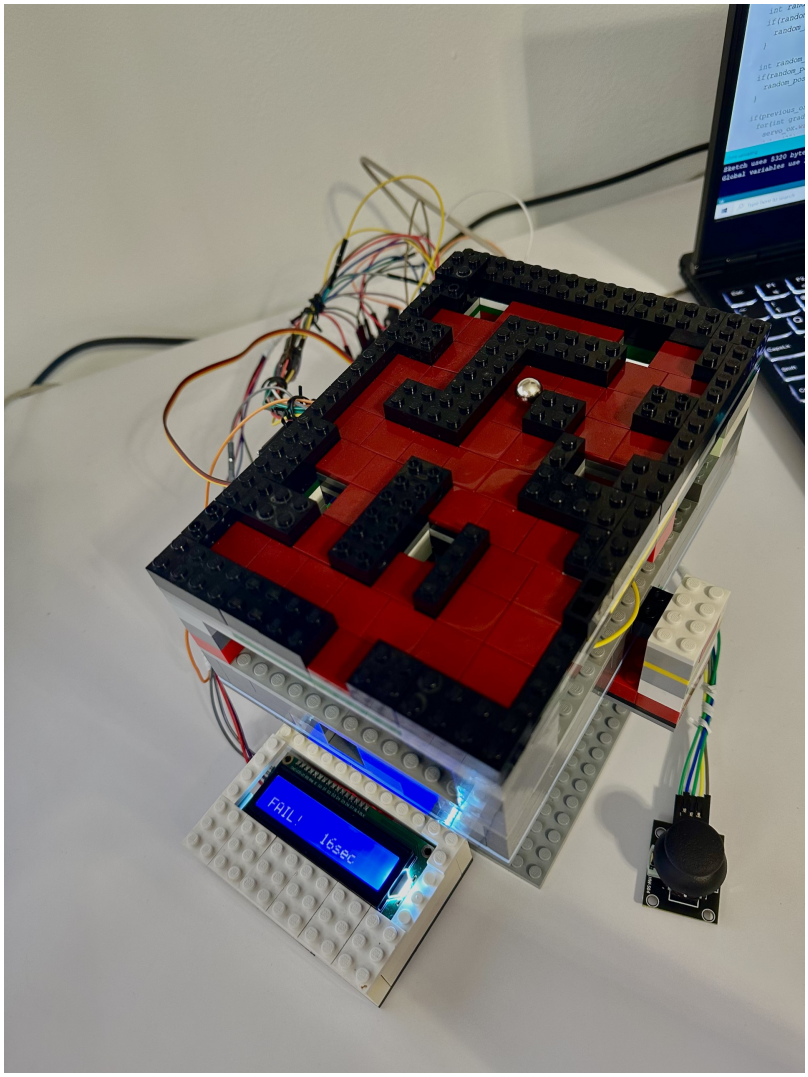
```
    byte upper_cursor = (received >> 1) & 0x01;
    byte lower_cursor = (received >> 2) & 0x01;
    byte second_button_pressed = (received >> 3) & 0x01;

    if(button_pressed && !start_game) {
      start_game = true;
      set_LCD_start_game();
    }

    if(start_game && !set_game_mode && (upper_cursor != 0 || lower_cursor
!= 0)) {
      /* choose game mode */
      last_game_picked = upper_cursor != 0 ? 1 : 2;
      byte cursor_LCD = upper_cursor != 0 ? 1 : 0;
      if(cursor_LCD != previous_cursor_state) {
        previous_cursor_state = cursor_LCD;
        parse_cursor(cursor_LCD);
      }
    }
  }
```

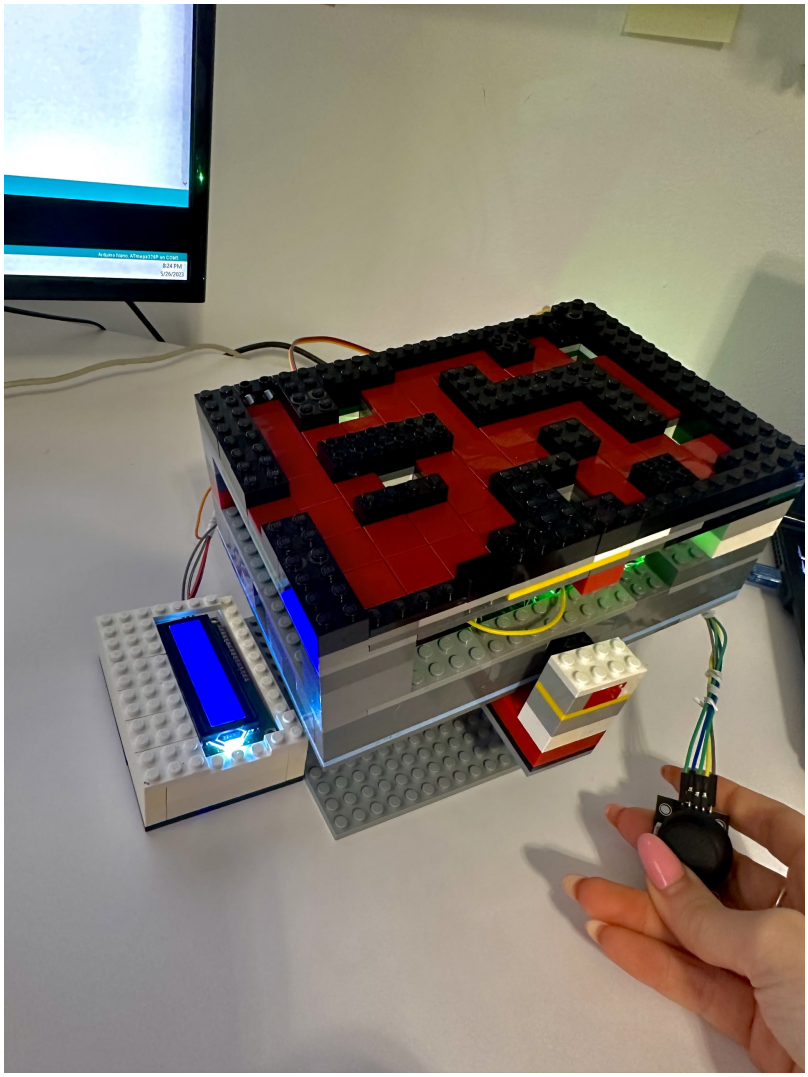
Rezultate obținute

Consider că partea care mi-a provocat cele mai multe probleme a fost fizica și dinamica jocului. Au apărut dificultăți atât în montarea platformei și a senzorilor și a ledurilor, cât și în fizica deplasării labirintului (viteză prea mare, mișcare bruscă etc.). În final, am reușit să implementez ce mi-am propus. Multe probleme au apărut și din cauza bibliotecilor folosite, primordial SoftwareSerial.h, care mi-a limitat posibilitatea lucrului cu regiștrii.


>> Send the ball
Lose the ball




FAIL!
16sec



Concluzii

Mă bucur că am reușit să duc la bun final proiectul pe care mi l-am propus, contrar numeroaselor probleme care s-au ivit. Țin să menționez că deși am reușit să implementez tot ce mi-am propus există câteva lipsuri ale proiectului, pe care cu siguranță le voi adăuga în viitor și care ar face experiența jocului mai bună. O problemă de luat în calcul este că atunci când doresc resetarea jocului, trebuie re-rulat codul, deoarece, la cum a fost gândită soluția inițial, conexiunea pe interfața serială între cele două Arduino-uri se finalizează o dată cu începerea jocului. Overall, mi s-a părut fun și mă consider mulțumită de proiectul obținut, mai ales că este primul proiect la care lucrez.
Am învățat ca folosirea bibliotecilor (mai ales Software Serial) poate provoca numeroase probleme, precum limitarea flexibilității și a lucrului cu regiștrii.

Download

Codul sursă pentru cele două Arduino-uri poate fi găsit la următorul repository de Github https://github.com/adumitrescu2708/TiltTheMaze.

Bibliografie

Pentru realizarea proiectului am folosit următoarele tool-uri, pentru schema bloc Miro, pentru schema circuitului electric Fritzing, iar pentru implementare am urmărit laboratoarele și m-am folosit de articolele
- https://diyodemag.com/projects/arduino_uno_servo-operated_ball_maze_marble_mayh
- https://www.hackster.io/michael-engel/automatic-marble-labyrinth-solver-d54ad3
- https://ocw.cs.pub.ro/courses/_media/pm/atmel-7810-automotive-microcontrollers-atmega328p_datasheet.pdf
- https://forum.arduino.cc/t/joystick-huge-outer-deadzone/939904

[1] Un Arduino se ocupă de cele 2 servomotoare și de joystick, iar al doilea de senzori, LCD și leduri
[2] Un servo acționează planul pe axa OX, iar al doilea pe OZ