**Aplicație cu micro-controller**

**pentru simularea unui pian**

Proiectare cu micro-procesoare

Prof. Îndrumător: Student:

**Ing. Prof. Paul-Mircea Mureșan** **Mihaela-Cristina Palamaru**

Grupa: **30238**

Table of Contents

[1. Motivația pentru proiectul ales 3](#_Toc60752325)

[2. Descrierea aplicației 3](#_Toc60752326)

[4. Noțiuni teoretice 5](#_Toc60752327)

[4.1.Arduino MEGA 2560 5](#_Toc60752328)

[4.2.JavaFX 6](#_Toc60752329)

[4.3.Comunicare prin UART 7](#_Toc60752330)

[5. Detalii de implementare 8](#_Toc60752331)

[5.1. Interacțiunea cu pianul prin C++ 9](#_Toc60752332)

[5.2. Controlul aplicației și vizualizarea progresului prin Java 12](#_Toc60752333)

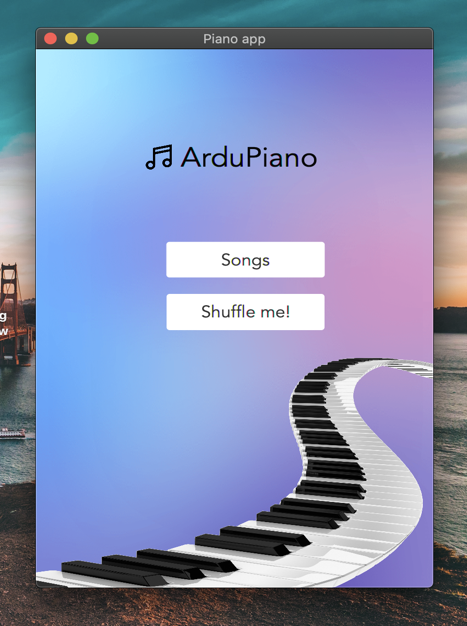
[6. Dezvoltări ulterioare 14](#_Toc60752334)

[7. Bibliografie 14](#_Toc60752335)

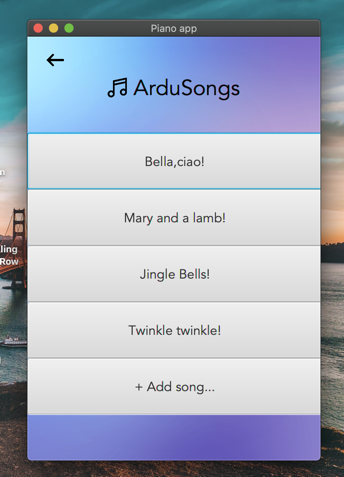
# Motivația pentru proiectul ales

Proiectul ales se dorește a fi un prototip hardware minimalist care să simuleze o pianină electrică. Acesta este conectat la o aplicație Desktop din care utilizatorul poate alege ce cântec dorește să exerseze, iar la final poate asculta înregistrarea cântecului și să observe progresul.  
Studiile au arătat că însușirea unui instrument muzical ajută mult la dezvoltarea creierului și atenției copilului, însă este costisitor să deții o pianină. Mult mai costisitor și mai complicat e să primești lecții specializate (mai ales în cazul copiilor cu deficiențe).  
   
Soluția propusă este de a veni cu cântece predefinite pe care utilizatorul le poate exersa, asociind anumite note de pe butoane și sunetul cu acele note ce vin secvențial pe ecranul prototipului.  
Pentru a crește interactivitatea și spiritul competitiv al copilului, am adăugat și o tabelă de scor în ecranul de progres al aplicației. În cazul în care mai mulți utilizatori ar încerca să se întreacă reciproc, ar putea să își compare scorul și să exerseze pentru a fi tot mai buni.

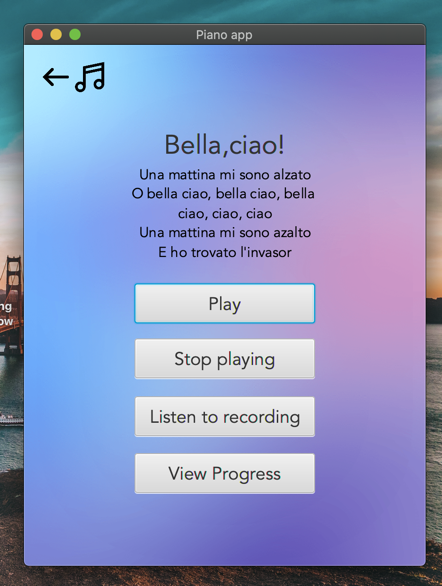
# Descrierea aplicației

 Aplicația dezvoltată pentru acest proiect reprezintă o îmbinare a cunoștințelor de hardware cu cele de software, conectând o aplicație cu interfața grafică la un montaj hardware pe Arduino ce dorește să simuleze un pian.

În continuare, se vor descrie în paralel funcționalitățile de care utilizatorul se poate bucura în folosirea aplicației.

În imagine se poate observa meniul principal al aplicației software, scrisă în JavaFX, pentru a-i oferi un design modern și flexibil. Aplicația vine cu 2 opțiuni pentru alegerea unui cântec pe care utilizatorul îl poate exersa. Fie alege dintr-o listă de cântece disponibile, fie apăsând pe butonul ” Shuffle me”, i se va genera un cântec la întâmplare.

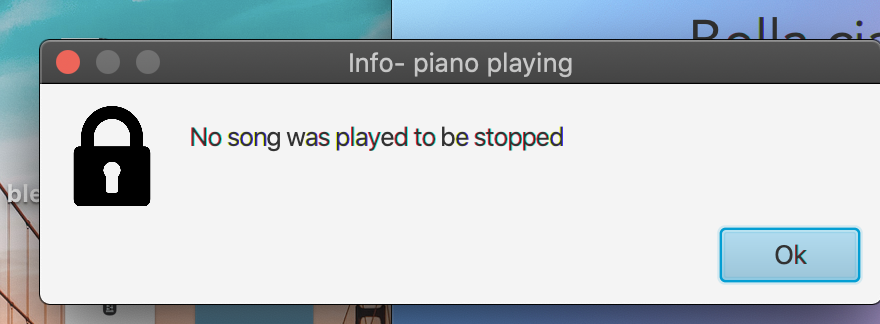
În momentul în care un cântec a fost ales, se deschide o fereastră specifică pentru acesta, din care va putea să trimită comenzi înspre pianul fizic de pe Arduino:



1. Apăsând pe butonul PLAY, pe ecranul lcd conectat la Arduino vor aparea titlul și notele ce trebuie cântate pentru a se potrivi cu linia melodică a cantecului ales, acestea deplasându-se dinspre dreapta spre stânga. Tot în acest moment, Arduino va înregistra cântecul pentru a-l face mai apoi disponibil în redarea progresului.



1. Nici o altă comandă nu este disponibilă până în momentul în care cântecul nu a fost oprit.

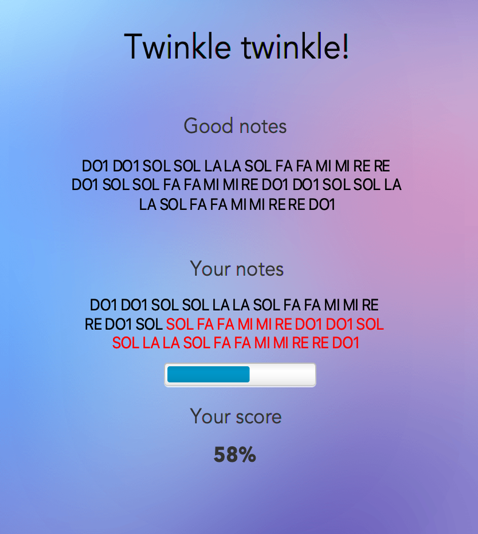
Întreruperea cântecului este marcată și pe ecranul LCD precum în imaginea de mai jos:



Pentru a evita fluxuri diferite de execuție s-au introdus mesaje de eroare precum cel de mai jos, care anunță utilizatorul că încearcă să execute o acțiune ilegală.



1. După oprirea transmiterii cântecului, utilizatorul poate să redea ceea ce a cântat prin apăsarea butonului “Listen to recording” sau să observe progresul într-o altă fereastră.



4. Observarea progresului reiese din numărul de note cântate identic, secvențial cu cele care au fost transmise inițial microcontrollerului. Notele care nu au fost cântate corect vor fi evidențiate prin colorarea lor în roșu.

# Noțiuni teoretice

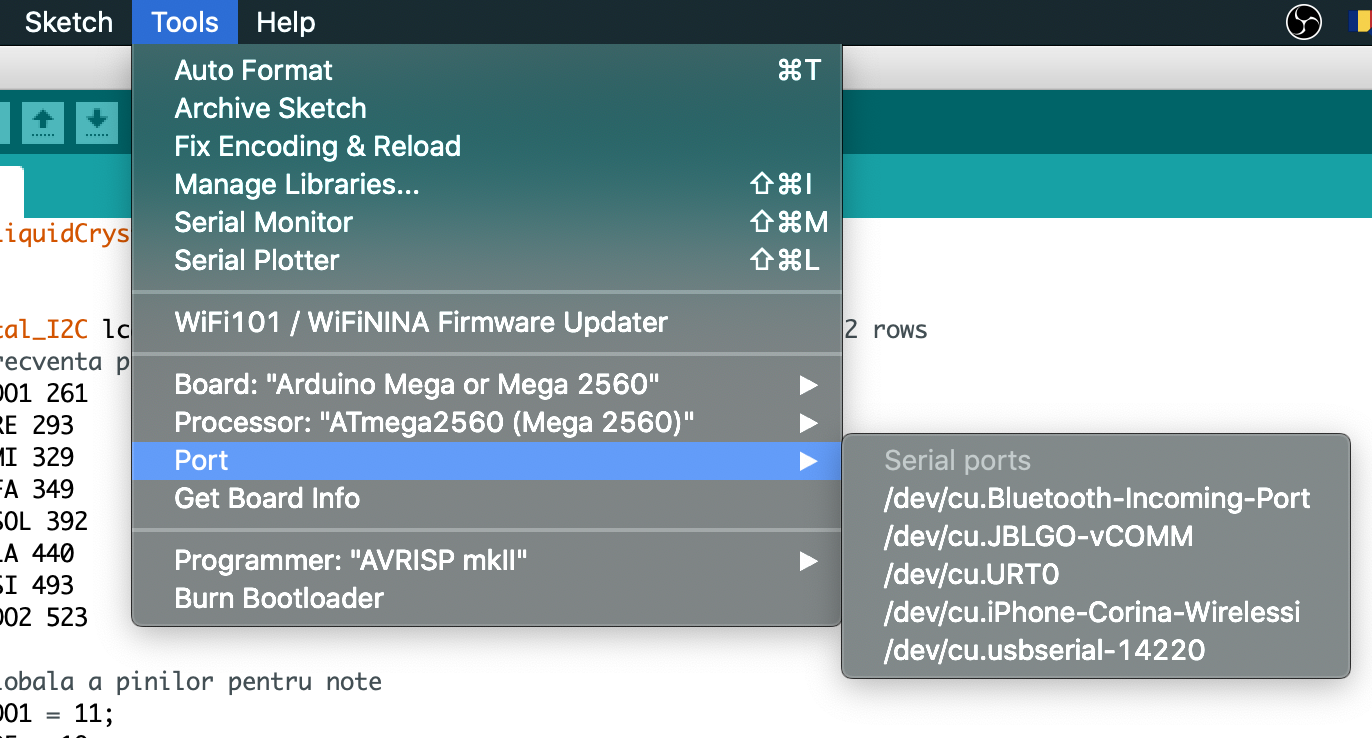
## 4.1.Arduino MEGA 2560

Pentru a crea montajul hardware necesar dezvoltării proiectului, am folosit un Arduino MEGA 2560. Arduino Mega 2560 este o platformă destinată proiectelor complexe, dispunând de 54 porturi digitale I/O (dintre care 15 se pot folosi ca porturi PWM și 4 porturi seriale hardware UART) și 16 analogice.

Platforma are la bază un microcontroler ATMega 2560 AVR pe 8 biți care lucrează la o frecvență de 16Mhz.

Plăcuțele Arduino Mega sunt dotate cu un circuit USB- UART serial ( asigurat de procesorul Atmega 16U2) care comunică prin același cablu cu care se face programarea și alimentarea plăcii**. Va exista un port comun ( COM- pentru calculatoarele ce folosesc Windows, cu.usbserial - pentru calculatoarele ce folosesc MAC-OS).**

Pentru a programa un astfel de dispozitiv este necesară instalarea programului ArduinoIDE și setarea microcontroller-ului astfel încât să fie setată familia MEGA și portul fizic corespunzător celui în care s-a introdus cablul de alimentare.



În capitolul următor va fi prezentată o schemă de detaliu privind conectarea componentelor.

## 4.2.JavaFX

Am ales să implementez aplicația software utilizând librariile din JavaFx datorită flexibilității în implementare și a aspectului plăcut pe care acesta îl ofera interfețelor grafice, indiferent de sistemul de operare de pe care este rulată. [2]

Pentru a putea avea acces la funcționalitățile acestei librării, este necesară instalarea în prealabil a unui Software Development Kit( SDK), cu o versiune de cel puțin 1.8 [5]. De asemenea, se recomandă utilizarea auxiliarului “Scene Builder” în construcția scenelor grafice, acesta oferind obiecte moderne și metode dinamice de lucru.



Pentru dezvoltarea proiectului Java, am ales să lucrez în IDE-ul Intellij, datorită procesului simplificat de compilare pe care acesta îl pune la dispoziție.

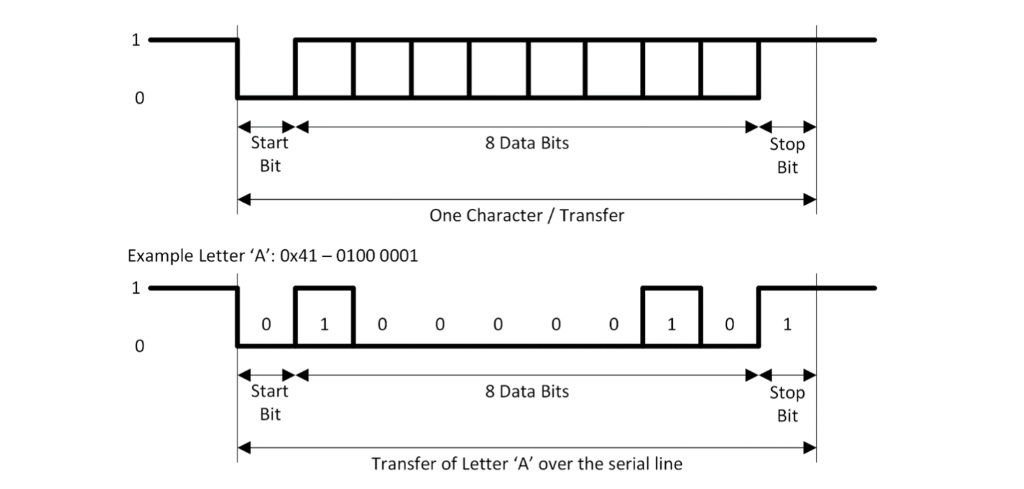
O scenă în JavaFx se compune dintr-o clasă Java ce are rol de Controller și un fișier FXML, în care sunt introduse obiectele grafice de mapat, în mod asemănător unui cod HTML.

## 4.3.Comunicare prin UART

Pentru a realiza comunicarea dintre plăcuță și calculatorul personal, a fost necesară crearea unei legături universale asincrone de transmitere serială a datelor. UART nu este un protocol, ci mai degrabă un mod de conectare fizic, ce permite legarea 1 la 1 a dispozitivelor, făra a avea de a face cu conceptul de master-slave, precum protocolul I2C.

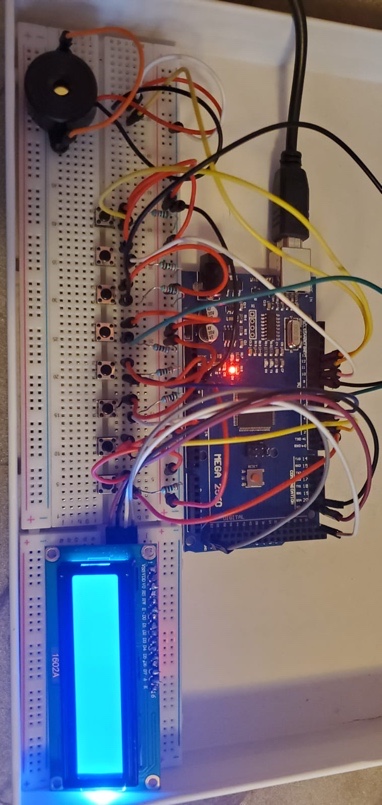
O magistrală serială asincronă conține doar 2 fire – unul pentru trimiterea datelor, iar celălalt pentru recepția lor. Așadar, componentele care doresc să comunice serial vor trebui să aibă 2 pini: pinul de recepție (**RX)** și pinul de transmisie (**TX**).

În cazul comunicației seriale asincrone, fiecare octet transmis este precedat de un bit de START, pentru care linia de comunicație are nivelul logic 0, și este urmat de cel puțin un bit de STOP, pentru care linia de comunicație are nivelul logic 1. Astfel, un octet transmis se încadrează între cei 2 biți de control, START și STOP.

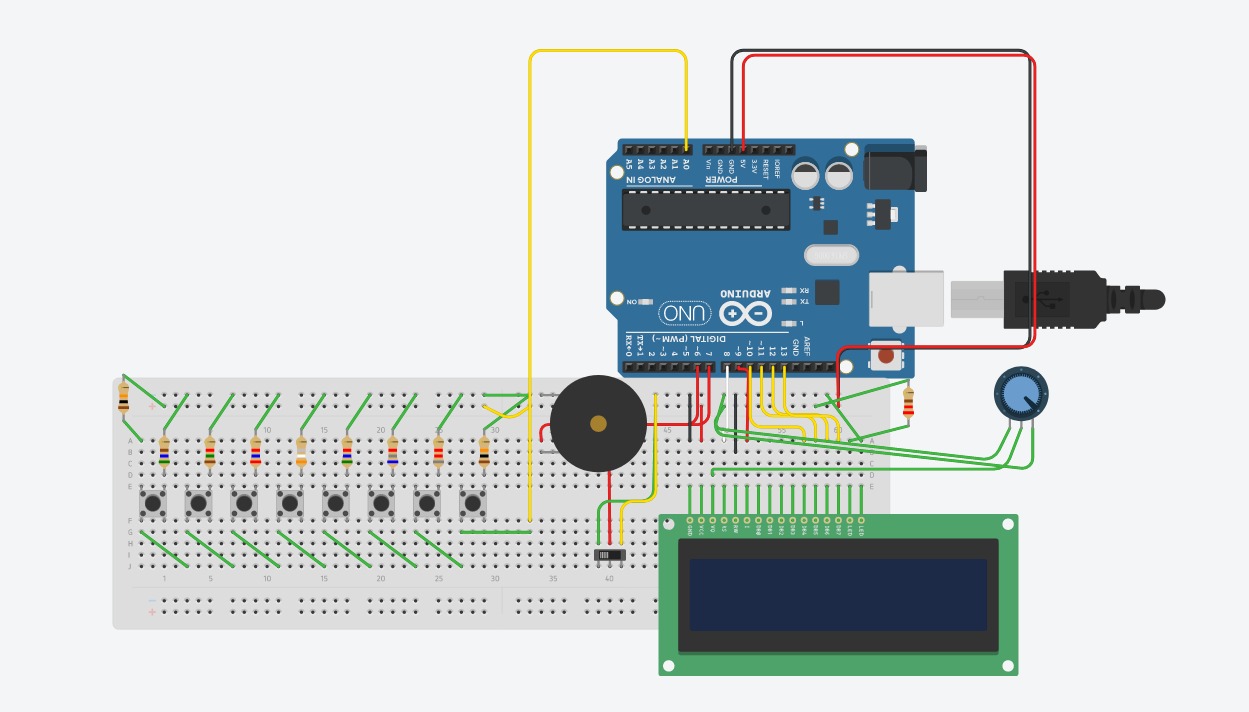


Exemplu de transmitere serială [4].

# Detalii de implementare

Pentru a realiza acest proiect din punct de vedere hardware, pentru montaj au fost necesare 8 butoane, reprezentând clapele pianului și un buzzer pentru redarea sunetului. Fiecare component are atașată o rezistență de 220 Ω pentru protecție.

De asemenea, montajul mai presupune și un LCD 16x2, care este conectat la un modul I2C prin cositorire. Conectarea ansamblului LCD- modul I2C se realizează prin conectarea corespunzătoare a SDL(Serial Data) și SCL(Serial Clock) de pe pinii 20 și 21 de pe Arduino Mega, prin jumpere.

În imaginea din stanga-sus se poate observa montajul realizat fizic pe Arduino Mega. Am realizat un montaj pe platforma de proiectare Tinkercad pentru a observa mai ușor modul de conectare al jumperelor la pini.

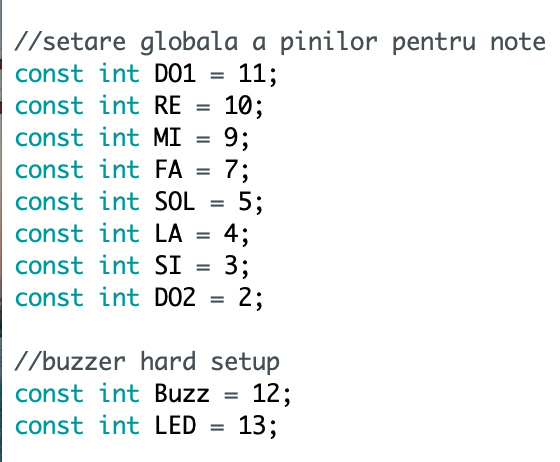
**Observație!**

Este de menționat că Tinkercad lucrează doar cu Arduino UNO, dar principiul de conectare este același ca pentru plăcuța MEGA, diferind numărul pinilor.

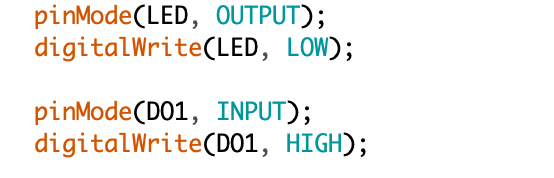
## 5.1. Interacțiunea cu pianul prin C++

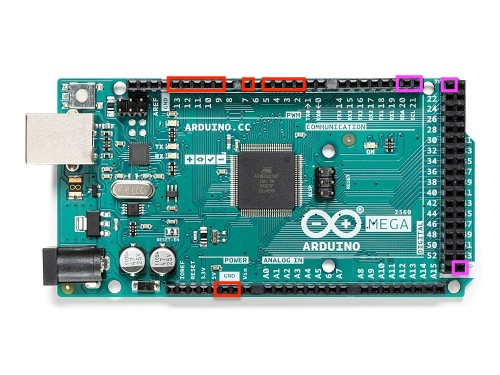
Programul din Java realizează tot controlul funcționalităților aplicației, Arduino executând doar comenzile preluate prin portul serial.

Înainte de a proiecta fluxul de execuție este necesară conectarea componentelor hardware și programarea acestora.



Cu **roșu** au fost marcați pinii corespunzători butoanelor și buzzer-ului.

La apasarea unui buton, un LED intern se va aprinde pentru a marca buna funcționare.



Cu mov se indică pinii de conectare pentru afișorul LCD ( cu I2C). În codul din Arduino a fost necesară includerea librariei **LiquidCrystal\_I2C lcd(0x27, 16, 2)**, unde 0x27 reprezintă adresa slave-ului din protocolul I2C. LCD-ul a fost conectat la pini diferiți de alimentare datorită faptului că tensiunea se diviza și atunci buzzer-ul nu putea primi semnalele potrivite.

Am optat pentru o scriere cât mai modulară a programului, pentru a face ușor distincția între logica de de intrare-ieșire și cea de prelucrare a datelor.

Există 8 note ce pot fii cântate, apelând funcția **tone**(Buzz, T\_NOTA), unde Buzz reprezintă buzzer-ul, iar T\_NOTA, frecvența notei. [1] Frecvențele alese pentru fiecare notă, de la DO1 la DO2 au fost luate dintr-un tabel, reprezentând un standard internațional**\***.

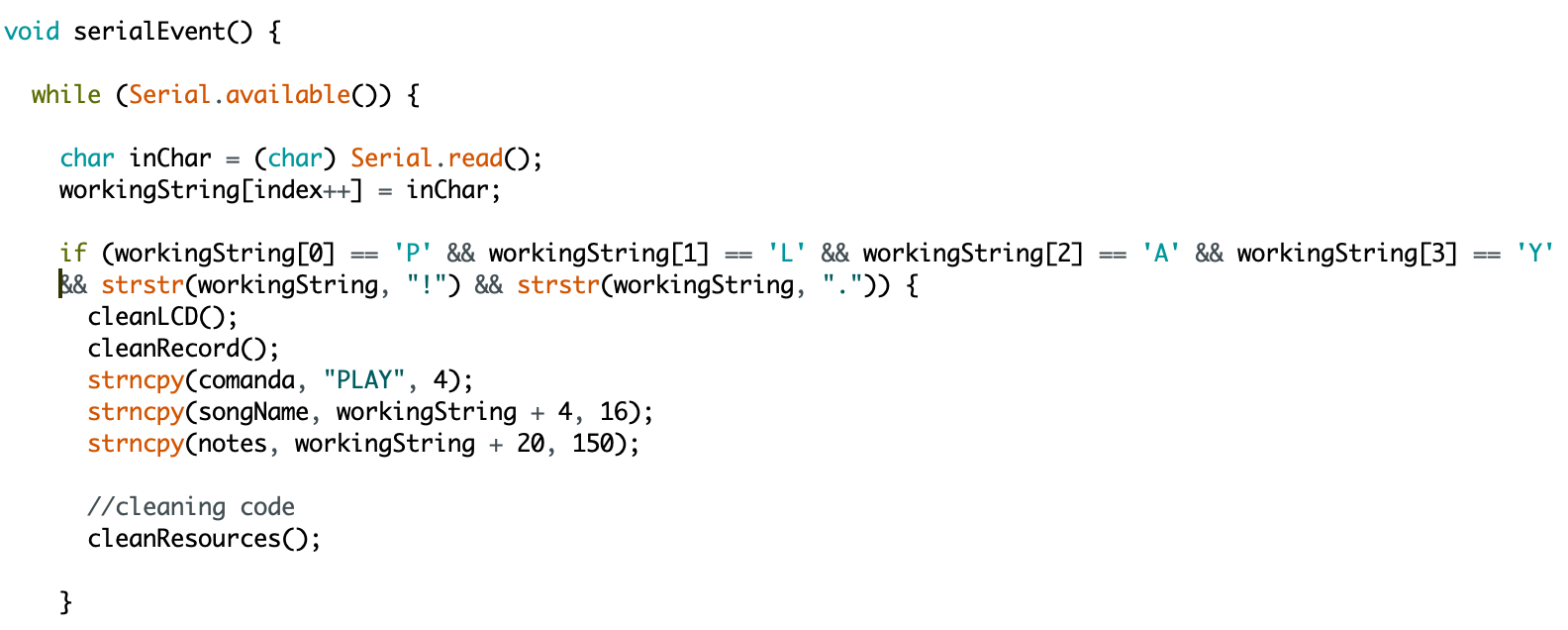
Pentru a realiza înregistrarea, se va lua un vector de frecvențe și durata în care un buton a fost ținut apăsat, după cum urmează:



Durata se calculează ca momentul curent de execuție al programului – timpul la care butonul a fost apăsat.

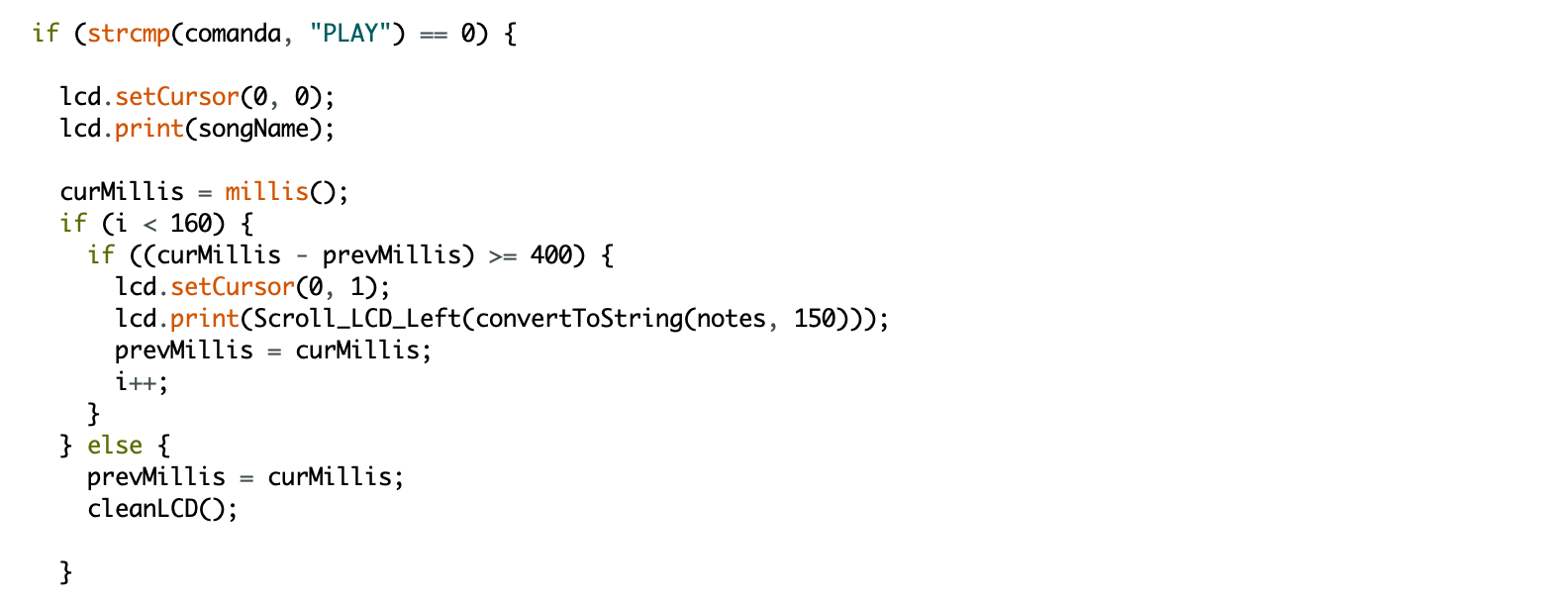
Am introdus un flag în care marchez cântarea unei note, pentru a o putea adăuga la înregistrare în afara buclei while,

După setarea modului în care butoanele vor reacționa la interpretarea unei melodii, a fost necesară descrierea ciclului de execuție al aplicației. Vor exista 3 comenzi principale din punct de vedere al aplicației pe partea de hardware: PLAY, STOP, RECORD.

Aceste comenzi for fi citite in funcția **serialEvent(),** atât timp cât portul serial este disponibil. Datele vor fi citite și scrise la un BaudRate de 9600, setat în funcția **setup**() pentru Arduino.[3]

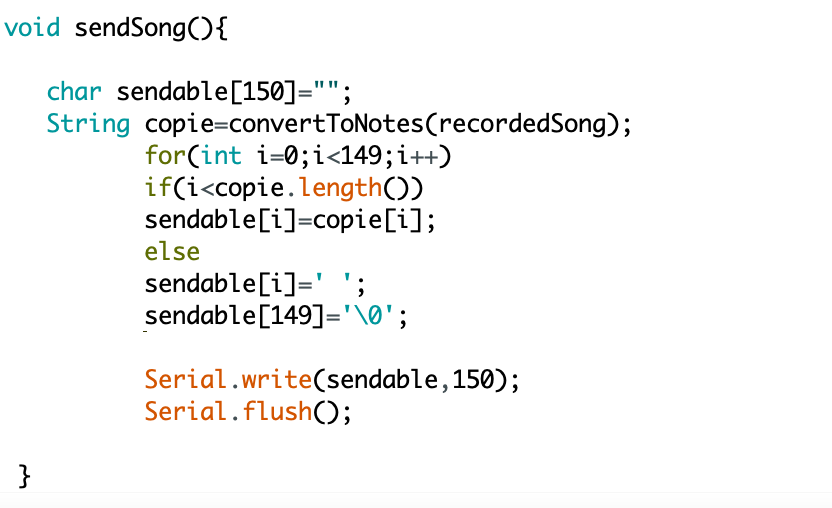
Citirea unei comenzi se realizează bit cu bit, comenzile fiind apoi construite ca șir de caractere și copiate cu ajutorul funcției **strncpy.** De fiecare dată când se percepe o nouă comandă, LCD-ul și resursele folosite în comenzile anterioare vor fi curățate, pentru a nu exista conflicte și **buffer overflow**.

În momentul în care o comandă este disponibilă ca șir de caractere, execuția acesteia se va realiza în funcția loop(). Un bun exemplu de acest fel îl reprezintă comanda “PLAY”:



În această secvență se va seta pe prima linie a afișorului numele melodiei, iar pe a doua linie, notele vor veni secvențial de la dreapta la stânga. Animația notelor se realizează în funcția **Scroll\_LCD\_Left**, o dată la 400ms.

Aveam posibilitatea de a aștepta ca o notă să fie afișată cu ajutorul functiei **delay**, dar acest lucru blochează accesul la pianul din butoane și întregul flux de execuție, ceea m-a determinat să verific timpul cu ajutorul funcției **millis().** [1]

Am ales să lucrez doar cu funcții care sunt non-blocking, pentru a evita posibilele blocaje ale fluxului de execuție.

În cazul transmiterii notelor cântate dinspre Arduino spre aplicația software cu scopul de a evidenția progresul utilizatorului, s-a folosit funcția **Serial.write().**

Aceasta este apelată în cadrul funcției **sendSong()**, în momentul în care Arduino primește comanda de încetare a înregistrării.

Datele vor rămâne în coadă, așteptând să fie preluate prin citire de către aplicația din Java.

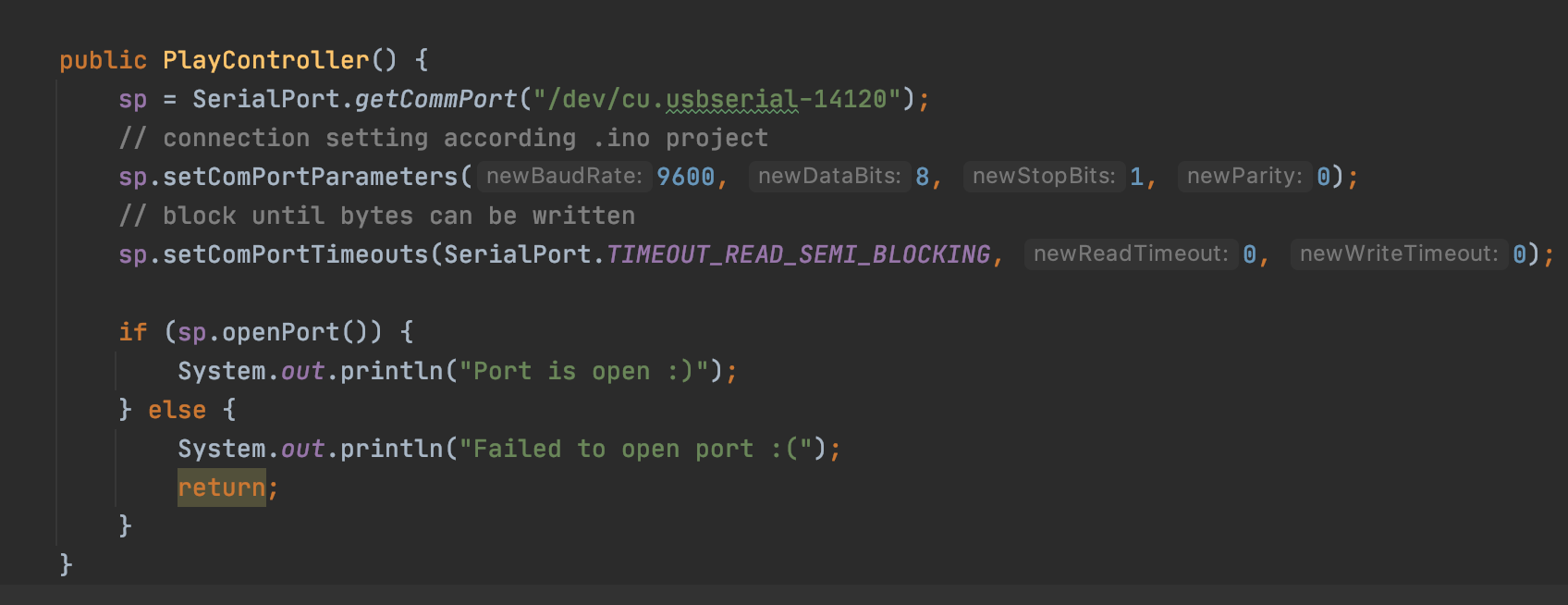
## 5.2. Controlul aplicației și vizualizarea progresului prin Java

După cum s-a menționat anterior, funcționalitățile aplicației sunt controlate prin intermediul aplicației software, având interfața grafică scrisă în JavaFx.

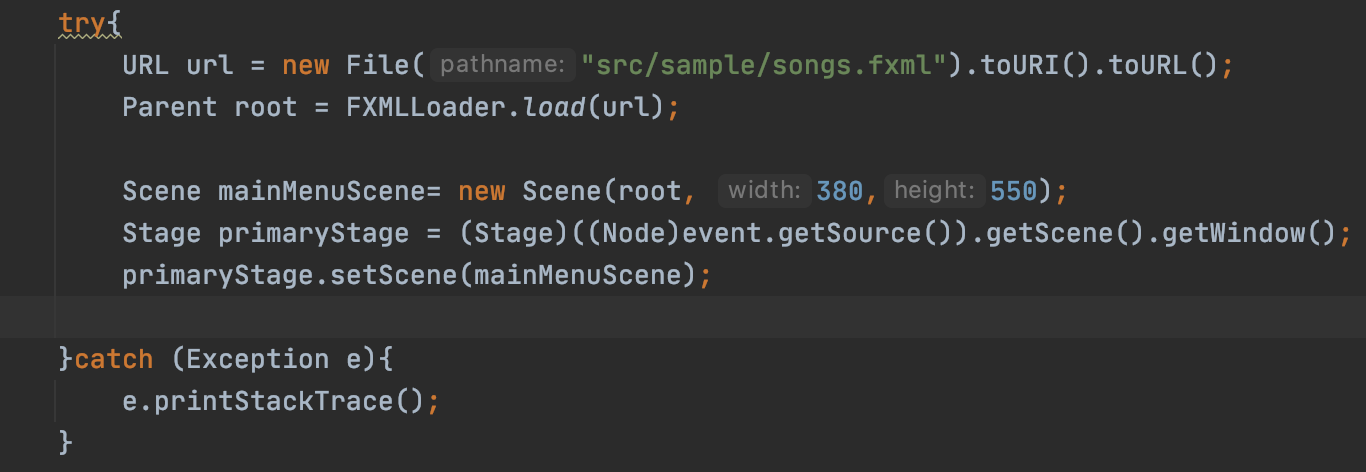
Pentru realizarea conexiunii cu UART pe partea de aplicație software, s-a folosit librăria externă (**jSerialComm**), librărie care oferă o bună încapsulare a metodelor de interacțiune cu hardware [6].

Conexiunea a fost setată conform specificațiilor din Arduino, la un baud rate de 9600, linia de date având 8biți și un singur bit de stop. Se va verifica dacă portul este deschis.

Portul este deschis în momentul în care utilizatorul se află în fereastra unui cantec.

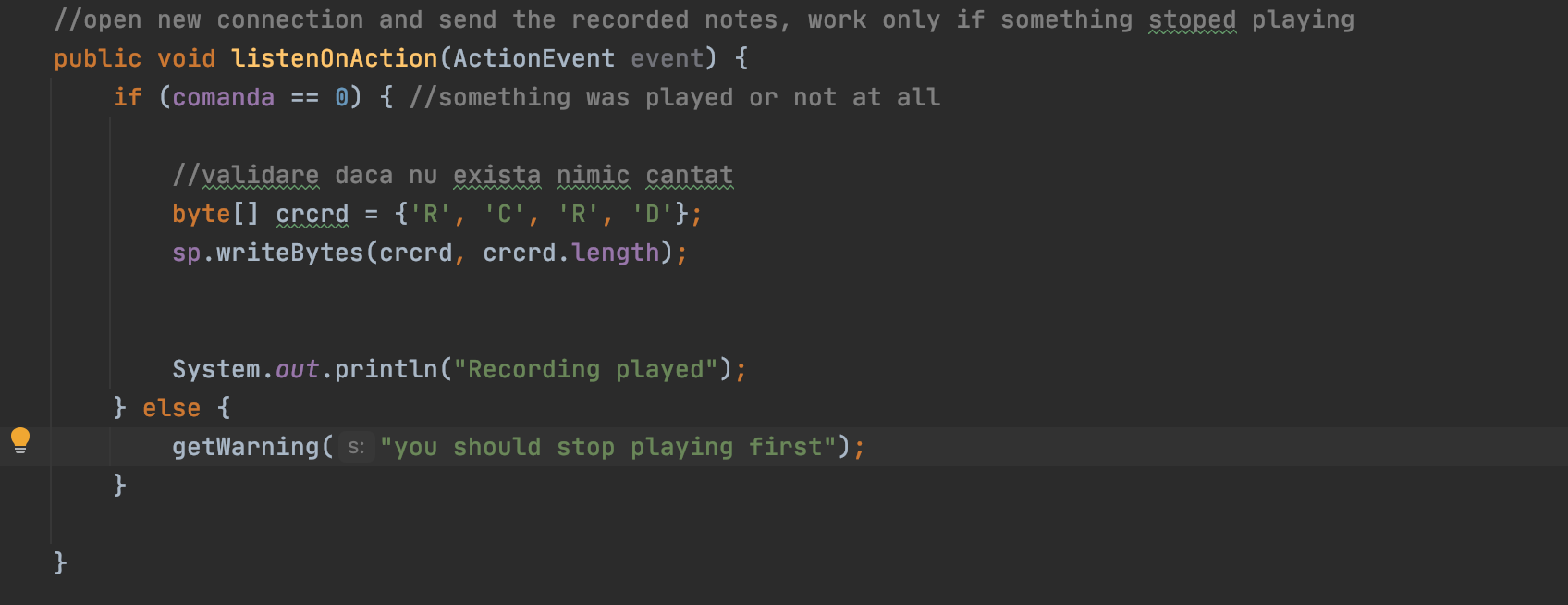


Acesta se va închide de fiecare dată când utilizatorul va trece dintr-un meniu în altul al aplicației.

Trecerea dintr-un meniul în altul aplicației se realizează după cum urmează, prin încărcarea fișierului FXML și a controller-ului corespunzător.

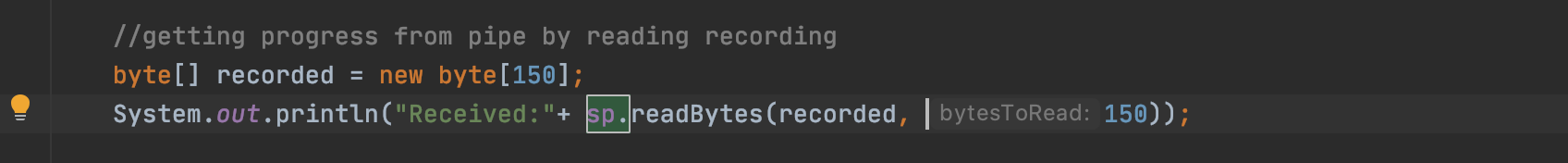
Comenzile disponibile pentru a activa fluxul de execuție din Arduino sunt executate cu ajutorul butoanelor specifice comenzii din interfața grafică.

Acțiunile de pe butoane vor fi ascultate cu ajutorul ActionListeners din JavaFX.



Comenzile vor fi trasnmise serial ca șir de bytes. În cazul în care un cântec nu a fost cântat sau se încearcă execuțuia unui flux ne-conform, se vor genera **pop-ups** care vor anunța utilizatorul că acțiunea lor este ilegală.

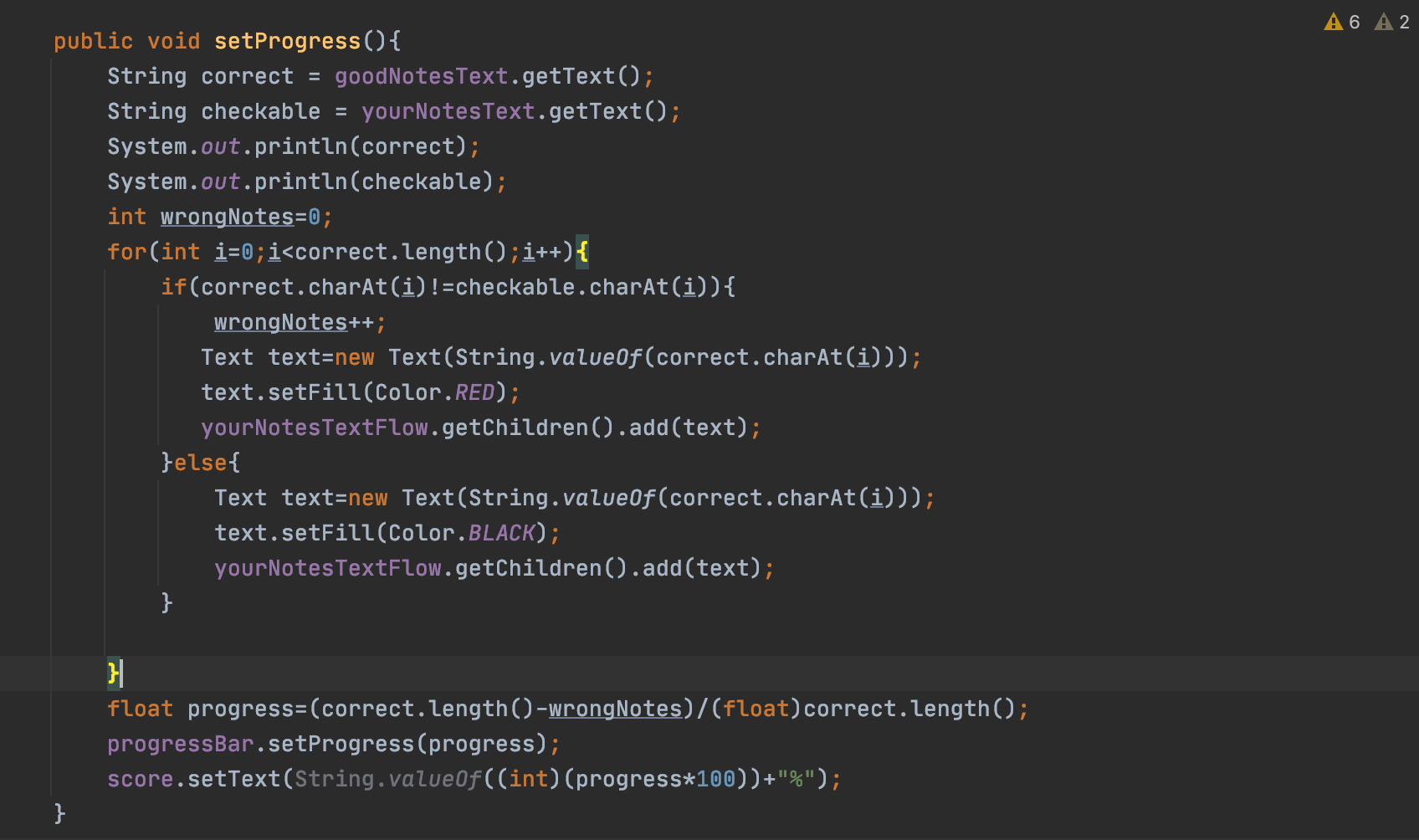
Progresul este citit în momentul apăsării pe butonul de progres, acesta constând în preluarea notelor înregistrate pe Arduino, într-un șir maxim de 150 de bytes.



În **ProgressController** este definită metoda **setProgress()** care va afișa notele corecte ce trebuiau cântate și le va compara cu cele primite de la Arduino. Pentru a evidenția locul în care utilizatorul a trimis o notă eronat, aceasta va fi marcată cu o culoare roșie.

Colorarea este posibilă prin concatenarea mai multor obiecte de tip **Text**, într-un obiect de tip **TextFlow**, pus la dispoziție de JavaFX.

Nivelul de evoluție este marcat prin încărcarea unei valori în progressBar, calculată ca raport între totalitatea de note cântate corect și secvența de note inițiale ale cântecului.



# Dezvoltări ulterioare

Din proiectul curent există o dezvoltare imediată ce se poate realiza strict la nivel software, anume adăugare preferențială de cântece. Acest lucru se poate realiza prin adăugarea unor textFields (javafx.control) în care textul necesar pentru denumirea cântecului, versuri și lungimea secvenței de note să fie restricționată pentru a putea fi afișate și redate cu ajutorul Arduino.

Pe viitor, pentru a putea oferi aplicația în scop didactic utilizatorilor care vor să își îmbunătățească motricitatea prin studiul la pian, doresc să adaug o carcasă fizică care să încapsuleze circuitele de alimentare și de interconectare.

# Bibliografie

[1]*Proprietăți funcții intrare- ieșire,* de pe pagina: <https://www.arduino.cc/reference/en/>, accesată la data de 20.12.2020

[2] *Setare proiect JavaFx,* de pe pagina: <https://www.geeksforgeeks.org/javafx-progressbar/>, accesată la data de 3.01.2021

[3] *Restrictii citire seriala*, de pe pagina: <https://github.com/Fazecast/jSerialComm/wiki/Modes-of-Operation>, accesată la data de 2.01.2021

[4] *Automate cu stări finite / Comunicație serială* - *Laboratorul 11*, din “Arhitectura Calculatoarelor”, autori - Florin Oniga și Mihai Negru, Editura UTPRESS, Cluj-Napoca, 2019

[5] *SDK 1.8*, de pe pagina: [https://gluonhq.com/products/javafx/](https://gluonhq.com/products/javafx/a) accesată la data de 23.12.2020

[6] *Librăria Fazecast*, website: <https://fazecast.github.io/jSerialComm>

**\* Tabelul cu frecvențele standard se regăsește în fișierul Assets al proiectului.**

