**Proiectare cu Microprocesoare**

**Proiect Arduino**

**Flappy Bird**

Student:Balcanu Vlad-Andrei

Grupa:30231

An:3

Profesor indrumator:Itu Razvan

**Cuprins:**

1.Declaratia problemei: proiectul propus, specificarea problemei;

2.Solutia: Descrierea solutiei pentru problema, exemplificare cu cod, implementare, descrierea si explicarea logicii alese pentru implementare;

3.Diagrama circuitelor/Schema;

4.Imaginea proiectului;

5.Bibliografie;

**Declaratia problemei**

Proiectul propus consta in implementarea unei replici a jocului Flappy Bird de pe telefoanele Smart pe o placuta Arduino.

Jocul este relativ simplu, prezentand o pasare ce se afla in mijlocul ecranului pe orizontala. Aceasta se deplaseaza in continuu spre dreapta si se poate deplasa in sus prin apasarea unui buton de comanda, imitand actiunea de dat din aripi a unei pasari normale astfel luand avant, sau poate sa se deplaseze in jos daca nu mai dispune de forta creata de fluturarea aripilor pentru a se deplasa in sus.

Pe parcurs in joc apar obstacole, mai exact perechi de turnuri, unul din turnuri aflandu-se in partea superioara a ecranului iar celalalt aflandu-se in partea inferioara, intre cele doua turnuri existand o gaura/un spatiu liber.

Scopul jocului este de a tine pasarea in aer fara a o lasa sa atinga podeaua, turnurile sau sa se loveasca de tavan si obtinerea unui scor cat mai mare.

Proiectul trebuie realizat pe o placuta de dezvoltare Arduino impreuna cu componentele necesare. Jocul va fi afisat pe un ecran LCD conectat la placuta.

**Solutia problemei**

Solutia problemei consta in proiectarea unui program software cu ajutorul IDE-ului oferit de Arduino, care sa contina logica jocului si sa proiecteze pe un ecran LCD interfata grafica in timp real pentru ca utilizatorul sa se poata juca.

Componente hardware:

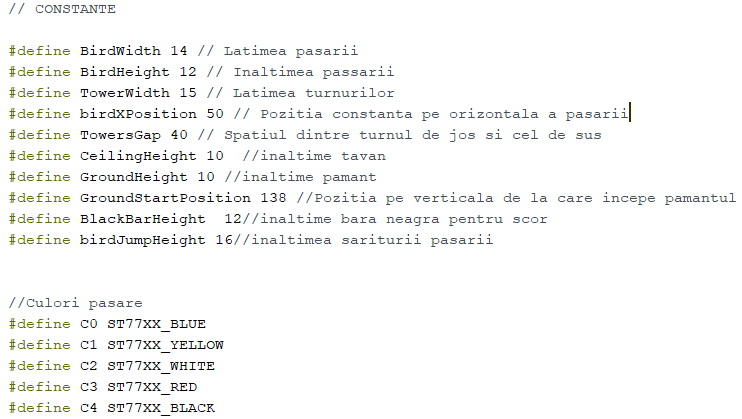
Pentru rezolvarea proiectului am ales sa folosesc o placuta Arduino Uno in locul unei placute Arduino Mega 2560 deoarece nu folosesc foarte multe componente astfel nu am avut nevoie de foarte multi pini iar aceasta placuta a fost de ajuns. Pentru proiect se mai folosesc un breadboard pentru a conecta componentele aditionale, un buton simplu prin care se realizeaza toate functionalitatile jocului adica saritura pasarii si reinceperea jocului in cazul in care am pierdut si un ecran LCD TFT KMR-1.8” SPI , TFT inseamnand ca foloseste thin-film-transistor ceea ce e o tehnologie ce imbunatateste calitatea imaginii iar SPI insemnand Serial Peripheral interface, un protocol de transmitere rapida a datelor serial si sincron pe o distanta mica. Ecranul are o diagonala de 1.8 inch, rezolutia acestuia fiind de 128x160 pixeli. Aceste 2 componente vor fi atasate pe breadboard si conectate la placuta Arduino Uno. Schema de conectare a circuitelor va fi prezentata in capitolele ce urmeaza.

Implementare:

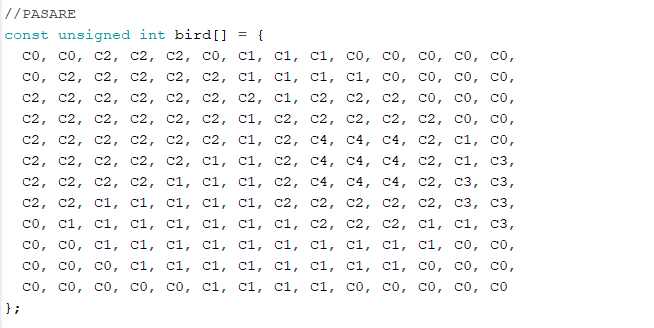
Pentru afisarea pe ecranul LCD se pot folosi o multitudine de biblioteci predefinite. Eu am ales bibliotecile <Adafruit\_GFX.h> si <Adafruit\_ST7735.h> deoarece sunt compatibile cu ecranul prezentat si dupa studierea mai multor librarii, Adafruit pare ca prezinta o imagine mai clara pe display iar schimbarea culorilor se face mai rapid, detalii importante pentru a da iluzia de rulare fluenta a jocului.

Logica jocului este impartita in mai multe functii pentru a face codul mai lizibil si pentru a usura munca pe anumite parti in cazul in care se lucreaza pe o anumita parte.

La inceput avem valori constante definite pentru ca ulterior codul scris sa poata fi inteles mai usor si aceste constante definesc caracteristici ale jocului cum ar fi grosimea si inaltimea pasarii, grosimea turnurilor, pozitia constanta a pasarii pe axa orizontala, culorile pasarii si altele.

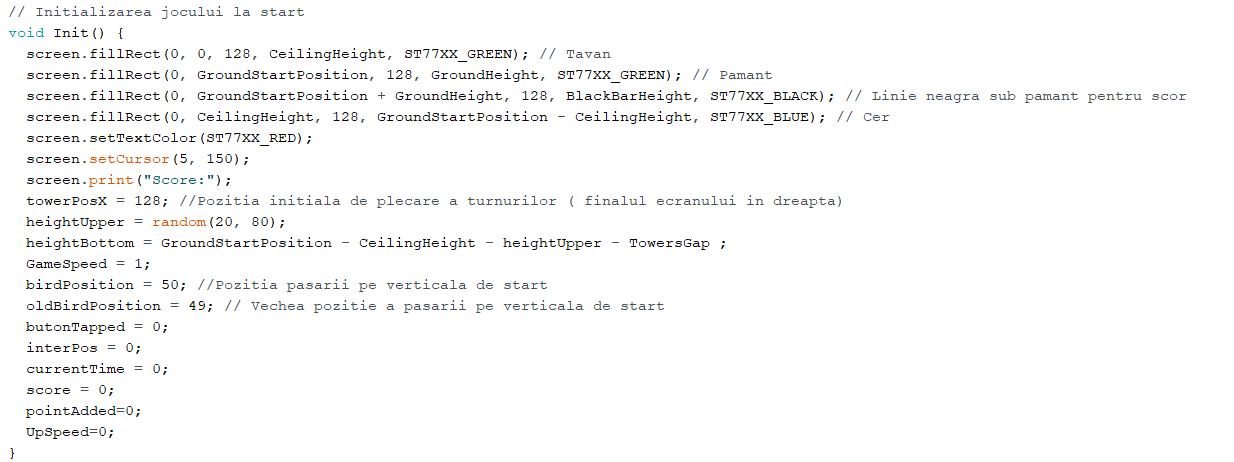


Urmeaza un vector de culori scris sub o forma de matrice prin care reprezentam pixelii pasarii care vor fi colorati cu culoarea corespunzatoare cifrei de pe pozitia respectiva. Aceasta forma de implementare a fost aleasa deoarece construirea cu o matrice propriu-zise in arduino este ineficienta. Dupa aceasta matrice sunt initializate valorile globale ce se schimba pe parcursul jocului.

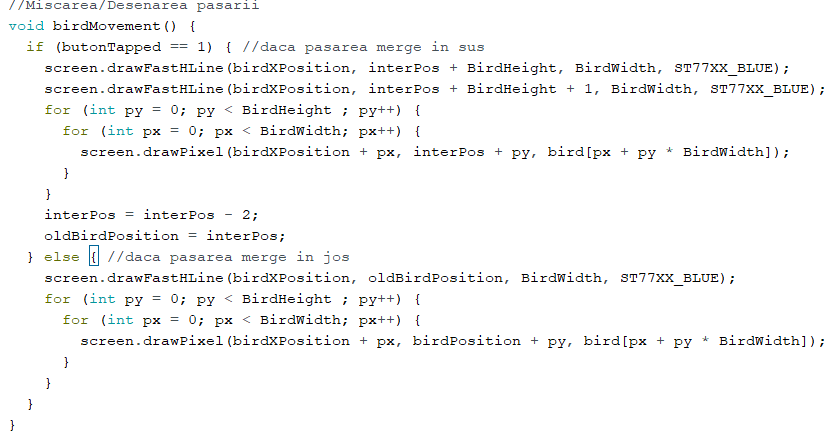


In functia setup() se face doar initializarea ecranului, colorarea lui cu culoare neagra si definirea pinului 7 ca rezistenta de pullup, pin ce va fi conectat la butonul jocului.

Functia Init(), care va fi apelata la inceputul fiecarui joc, realizeaza initializarea fundamentala a jocului, desenand pasarea in punctul de start, tavanul, podeaua si bara de scor. Pe langa acestea initializeaza sau reseteaza in cazul restartarii jocului valori precum scorul, pozitia de start a pasarii, pozitia de start a turnurilor, dimensiunea turnurilor, dificultatea jocului si altele.



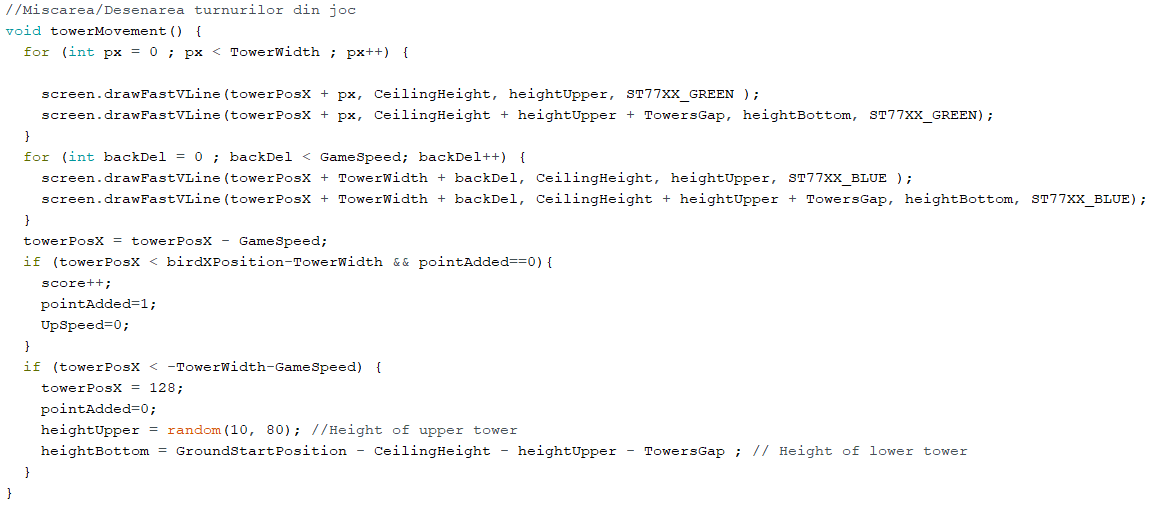
Functia birdMovement() se ocupa cu realizarea animatiei pasarii.



Dupa cum se poate observa avem doua cazuri, unul fiind cand pasarea se deplaseaza in jos, caz in care pozitia pe verticala a pasarii se schimba cu 1 pixel. Cu ajutorul functiei drawFastHLine() vom desena pe vechea pozitie a pasarii doar peste prima linie orizonatala de pixeli o linie albastra, iar cu urmatoarele doua for-uri si cu ajutorul functiei drawPixel, vom desena pasarea la pozitia noua peste pixelii folositi de pasarea veche. Astfel realizam iluzia unei animatii fluente deoarece nu resetam ecranul, lucru ce ar rezulta in pixeli negri pentru o perioada scurta de timp dar observabila de catre ochiul uman ceea ce ar duce la un joc sacadat ce nu da iluzia de animatie continua.

Al doilea caz este cand pasarea a primit comanda de a se deplasa in sus. Deoarece cand apasam butonul, pozitia setata la care trebuie sa ajunga pasarea este cu o multitudine de pixeli mai sus decat pozitia curenta, trebuie sa parcurgem distanta dintre cele doua puncte si sa redesenam pasarea pe parcurs pentru a crea animatia corespunzatoare. Astfel avem o pozitie intermediara care merge din 2 in 2 pixeli, deoarece am dori ca pasarea sa se miste mai repede in sus, la care vom redesena pasarea pana ajunge la pozitia corespunzatoare. Mecanismul folosit este exact acelasi ca si la cazul in care pasarea se misca in jos, aici doar schimband pozitia rapid cu cate 2 pixeli inloc de 1 pixel. Variabila buttonTapped ne specifica ca a fost data comanda de a zbura.

Functia towerMovement() se ocupa cu realizarea animatiei turnurilor.

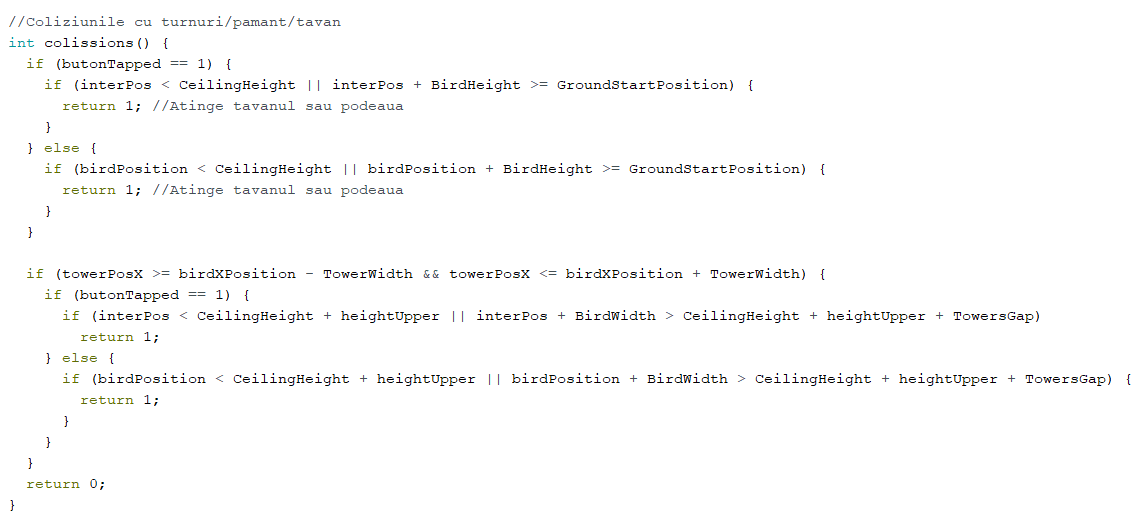


Mecanismul folosit pentru desenare este foarte asemanator cu cel folosit pentru pasare, redesenand turnurile noi la pozitia noua, peste pixelii ce formau turnurile vechi, si colorand cu albastru coloanele de pixeli de pe ultimele pozitii ale turnurilor vechi in functie de dificultatea jocului.

Daca marginea din dreapta a turnurilor se afla mai la stanga decat pozitia pe orizontala a pasarii iar variabila pointAdded este 0, scorul va fi incrementat. Am scris conditiile in acest mod deoarece puteam sa comparam direct pozitia turnurilor cu pixelul 0 pe orizontala dar cum turnurile se muta la stanga cu un anumit numar de pixeli ce depinde de dificultatea jocului uneori am avea exceptia de a nu ajunge niciodata pe pixelul 0 si astfel nu s-ar adauga in acel caz punctul.

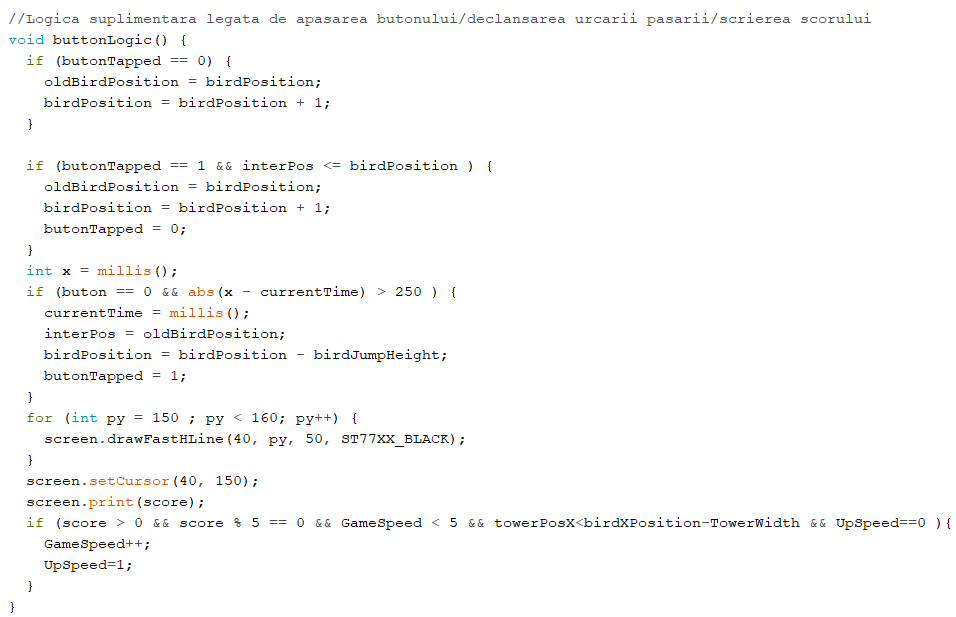
Daca pozitia turnurilor a trecut de pixelul 0 pe orizontala cu o distanta egala cu grosimea turnurilor adunata cu dificultatea jocului, pozitia turnurilor se reseteaza la pixelul 128, adica in dreapta ecranului, se recalculeaza dimensiunea fiecarui turn in parte si variabila pointAdded ia valoarea 0 pentru ca un nou punct sa poata fi adaugat la scor.

Functia colissions() se ocupa cu coliziunile dintre pasare si tavan, podea sau turnuri si returneaza valoarea 1 in cazul in care s-a produs o coliziune sau 0 in cazul in care nu. Aceasta functie va determina daca jocul continua sau utilizatorul a pierdut.



Se testeaza daca prima linie de pixeli a pasarii coincide cu ultima linie de pixeli a tavanului sau a turnului de sus. Se mai testeaza daca pixelii de pe ultima coloana a pasarii coincid cu pixelii de pe marginea din stanga a turnului de sus sau de jos si daca ultima linie de pixeli a pasarii coincide cu prima linie de pixeli a podelei sau a turnului de jos. Daca vreunul din cazuri este adevarat functia returneaza valoarea 1.

Functia buttonLogic() contine logica suplimentara referitoare la detaliile jocului.



Variabila butonTapped ne specifica daca s-a primit comanda de la utilizator pentru pasare pentru a se deplasa in sus. Daca aceasta variabila este 0, pasarea se deplaseaza in jos cu cate un pixel constant. In cazul in care aceasta este 1, pasarea se deplaseaza in sus pana cand se ajunge la pozitia dorita. Daca pozititia intermediara a pasarii a ajuns la pozitia setata, variabila devine 0 si pasarea incepe sa se indrepte spre podea.

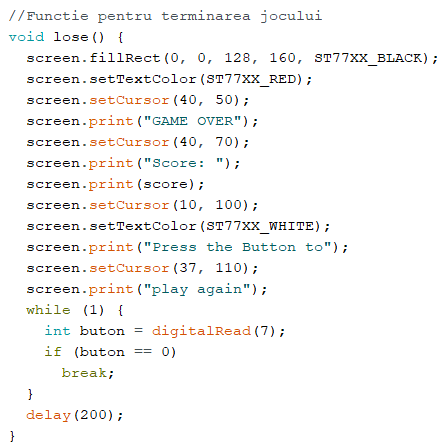
Pentru a evita citirea mai multor semnale de 0 de la utilizator cand apasa butonul cum se poate intampla in unele cazuri cum ar fi daca tine putin mai mult apasat butonul, folosim functia millis() pentru a testa daca a trecut o anumita perioada de timp de la apasarea butonului pentru a putea primi un alt semnal de la utilizator. Astfel citim constant timpul masurat in timpul jocului. La fiecare apasare de buton variabila currentTime retine timpul la care acesta a fost apasat ultima data pentru a putea face o diferenta intre ele la urmatoarea apasare de buton. Folosim o valoare absoluta a acestei diferente deoarece, dupa un anumit timp, functia millis() returneaza o valoare negativa si face ca functia sa fie apelata in continuu. Acest fapt are ca rezultat pasarea primind in continuu comanda de a se indrepta spre tavan incontrolabil. Prin folosirea valorii absolute rezolvam aceasta problema.

Daca conditia este indeplinita, variabila buttonTapped devine 1 pentru a semnala faptul ca pasarea trebuie sa se indrepte in sus, pozitia pasarii este setata la pozitia actuala din care scadem valoarea sariturii pasarii iar pozitia intermediara este setata la pozitia actuala pentru a incepe algoritmul descris de functia birdMovement().

Cu ajutorul for-ului desenam bara neagra de jos peste care va fi afisat scorul constant.

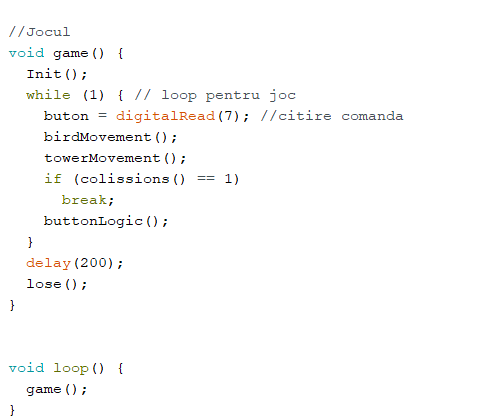
Prin ultima conditie testam daca dificultatea jocului trebuie crescuta. Daca scorul obtinut pana la punctul curent de catre jucator se imparte exact la o anumita valoare, dificultatea jocului, ce consta in viteza de deplasare a turnurile, va creste. Variabila upSpeed ne semnaleaza daca dificultatea jocului a fost crescuta. Astfel daca este 0, dificultatea nu a fost crescuta, daca este 1 aceasta a fost crescuta. Folosim aceasta variabila deoarece scorul va ramane constant pentru o perioada de cateva secunde, fapt ce ar duce la cresterea scorului incontrolabil pe acea perioada. Cu aceasta variabila rezolvam aceasta problema.

Functia lose() se apeleaza in cazul in care utilizatorul pierde un joc.



Daca utilizatorul a pierdut jocul, ecranul este colorat cu culoarea neagra si pe acesta vor aparea mesajul de ”GAME OVER”, urmat de scorul obtinut in aceasta runda si de mesajul ”Press the button to play again”. Urmeaza un while infinit in care citim constant valoarea de pe pinul 7 (butonul) pentru a astepta semnalul de reincepere a jocului. Daca utlizatorul apasa butonul, programul iese din while si porneste alt joc.

Toata logica scrisa mai sus este combinata ulterior in functia game().



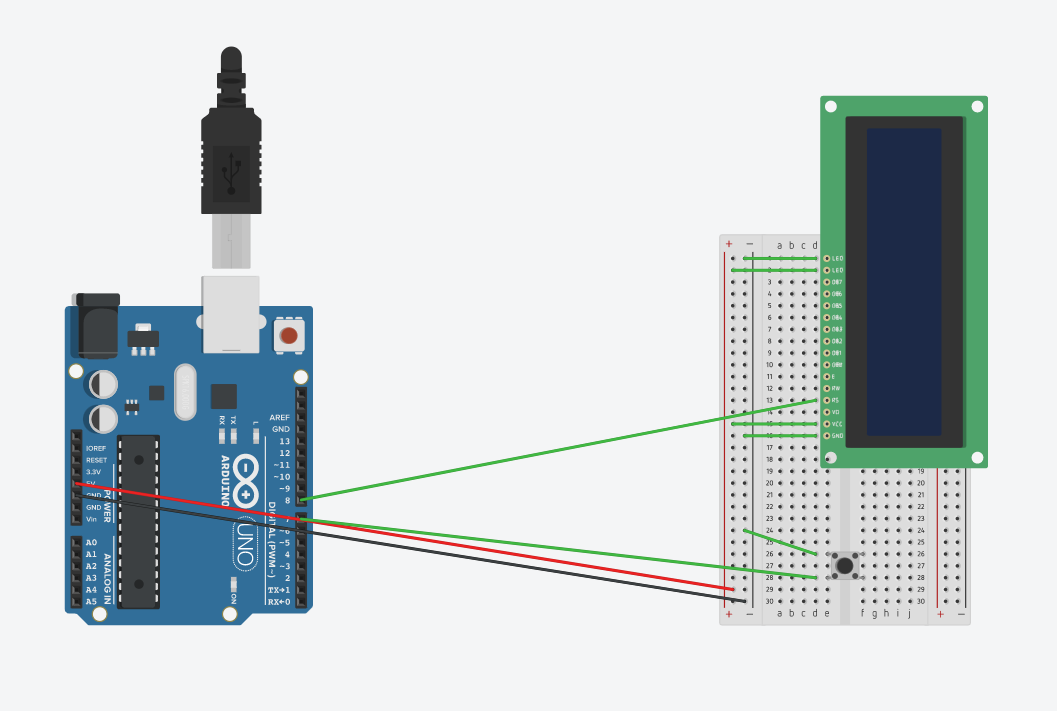
Prima oara se realizeaza initializarea jocului prin functia Init(). Dupa aceea avem o bucla infinita in care citim constant valoarea de pe pinul 7 (butonul) si apelam functiile de birdMovement(), towerMovement(), colissions() cu testarea rezultatului returnat de aceasta functie si buttonLogic(). Astfel la fiecare bucla desenam pasarea, desenam turnurile, verifica daca s-a produs o coliziune intre pasare si restul obiectelor din ecran, caz in care se iese din bucla infinita, altfel se verifica logica suplimentara a jocului. In momentul in care se iese din bucla infinita este apelata functia lose() ,dupa un mic delay, pentru a semnala ca jocul a fost pierdut. Functia game() este ulterior apelata in functia loop() pentru a se intampla in continuu atat timp cat placuta este conectata la curent.

Am folosit aceasta implementare deoarece daca scriam logica in cadrul functiei loop(),fara a folosi o bucla infinita, dupa terminarea fiecarui set de instructiuni, ecranul era resetat si redesenat. Acest lucru duce la o perioada scurta de timp dar observabila in care pixelii ecranului sunt negri. Rezultatul ar fi fost un joc sacadat ce nu da impresia de fluiditate si nu pacaleste ochiul uman cu o animatie continua.

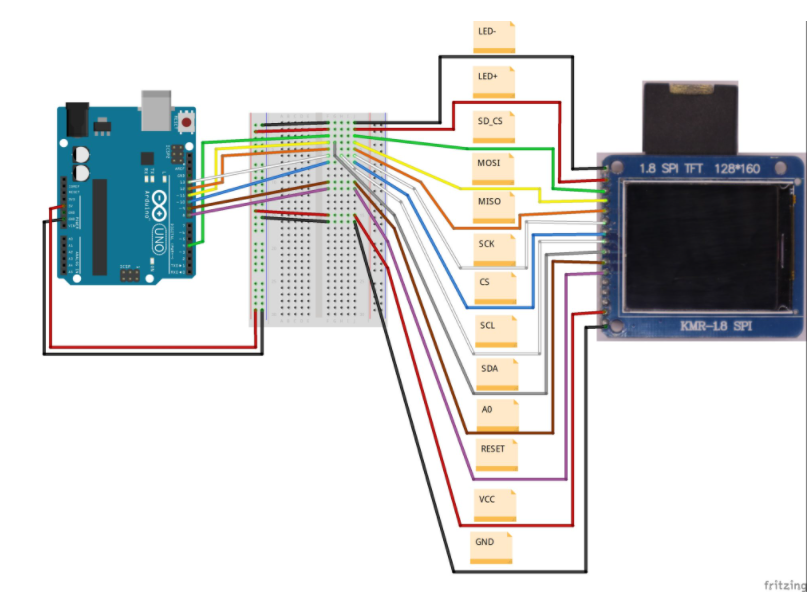
Scriind logica de desenare in cadrul unei bucle infinite, tinem ecranul activ fara a-l reseta si realizam animatia prin recolorarea pixelilor in functie de nevoie la fiecare moment. Prin aceasta implementare, colorarea se face aproape instant si se creaza iluzia unei animatii fluente ce ruleaza la mai mult de 24 de cadre, destul pentru a pacali creierul uman.

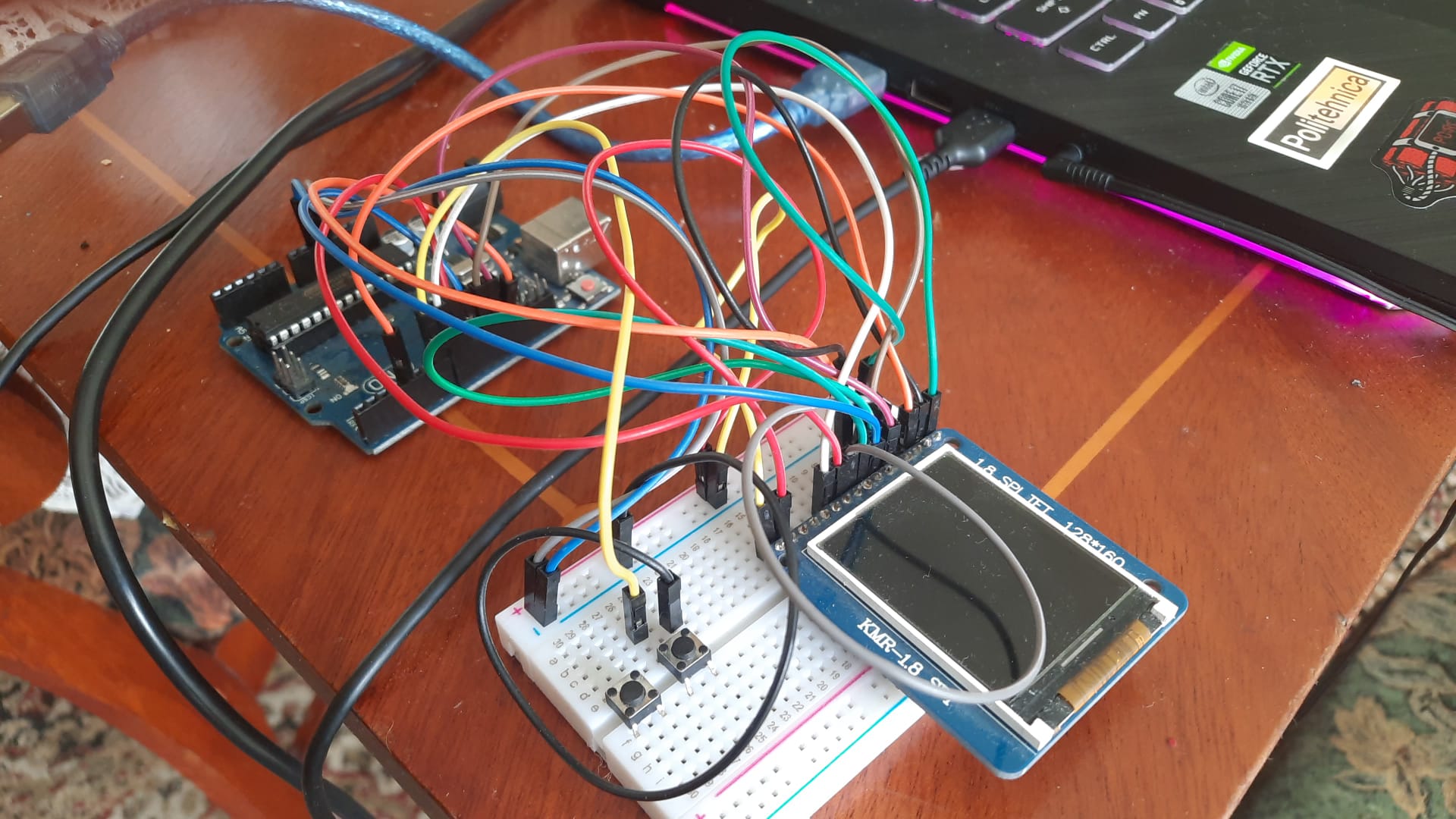
**Schema de montaj a circuitelor**

Datorita faptului ca pe niciuna din aplicatiile de creare de scheme de montaj am optiunea de a adauga un LCD-ul folosit pentru acest proiect am creat o schema in thinker Cad in care am folosit un arduino uno impreuna cu un breadboard pe care am atasat un buton si un LCD liquid crystal pentru a proiecta modul in care au fost conectate butonul si displayul pe breadboard si mai apoi legate la placuta Arduino Uno. Conectarea display-ului liquid crystal nu este facuta in totalitate, aparatul a fost atasat doar pentru a exemplifica cum arata proiectul



In plus am adaugat o schema de montaj importata de pe o pagina web ce continea documentatia despre cum se conecteaza LCD-ul ales de mine la o placuta Arduino. Am adaugat si aceasta fotografie pentru a se vedea clar cum se leaga acest LCD de placuta si pentru a specula de unde am avut cunostintele de a realiza acest lucru. Desi schema contine foarte multe fire ce leaga placuta de LCD, in initializarea acestuia in cod, trebuie definiti doar pinii de RS (reset), DC (sau A0/ data or command selector) si CS (chip select). Acestia sunt toti pinii necesari pentru scopul in care este folosit LCD-ului in acest proiect.



**Imaginea Proiectului**

**Bibliografie**

Intelegerea modului de lucru cu o placuta de dezvoltare Arduino, conectarea componentelor, folosirea IDE-ului oferit de Arduino : <https://users.utcluj.ro/~razvanitu/teaching.html#pmp>

Conectarea ecranului LCD, KMR 1.8” ISP TFT, la placuta Arduino:

<https://mschoeffler.com/2019/06/20/arduino-tutorial-making-the-kmr-1-8-spi-tft-display-work/>

Intelegerea modului de lucru cu biblioteca Adafruit\_GFX:

<http://adafruit.github.io/Adafruit-GFX-Library/html/class_adafruit___g_f_x.html>

<https://github.com/adafruit/Adafruit-GFX-Library/blob/master/Adafruit_GFX.h>

<https://github.com/adafruit/Adafruit-GFX-Library>