**Arduino Game of Life**

# Descriere temă aleasă

Jocul Game of Life reprezintă un exemplu de un automat celular, introdus în anul 1970 de către matematicianul John Horton Conway (1937-2020) ce folosește un set de reguli pentru a evolua. Nu este considerat un joc în adevăratul sens al cuvântului, întrucât nu are jucători și deci, nu există câștigători și pierzători. De asemenea are proprietatea de indecidibilitate, adică nu se poate găsi un tipar, prin care să se afle anumite stări viitoare.

Un automat celular reprezintă un model studiat atât în matematică, cât și în fizică și alte științe, format, de obicei, din două stări: 1 și 0 (”Pornit”, ”Oprit”). În cazul Game of Life, cele două stări pe care le poate avea o celulă vor fi descrise ca ”Viu” și ”Mort”. De asemenea, un automat celular are forma unei grile, în care fiecare celulă poate avea un număr de celule vecine. În cazul Game of Life, grila poate avea orice dimensiune, iar numărul de celule vecine pentru fiecare celulă în parte este de 8. Fiecare etapă a jocului se numește generație. Celulele își pot schimba starea de la o generație la alta. În cadrul Game of Life, acest lucru se face, doar în funcție de celulele vecine.

Regulile „Game of Life” sunt următoarele:

1. O celulă vie cu 2 sau 3 vecini vii, va rămâne vie și in generația următoare;

2. O celulă vie cu mai puțin de 2 vecini vii sau mai mult de 3 vecini vii, va deveni moartă;

3. O celulă moartă cu 3 vecini vii va deveni vie.

Putem observa de exemplu următorul tabel de modificare a stărilor unei configurații de celule:

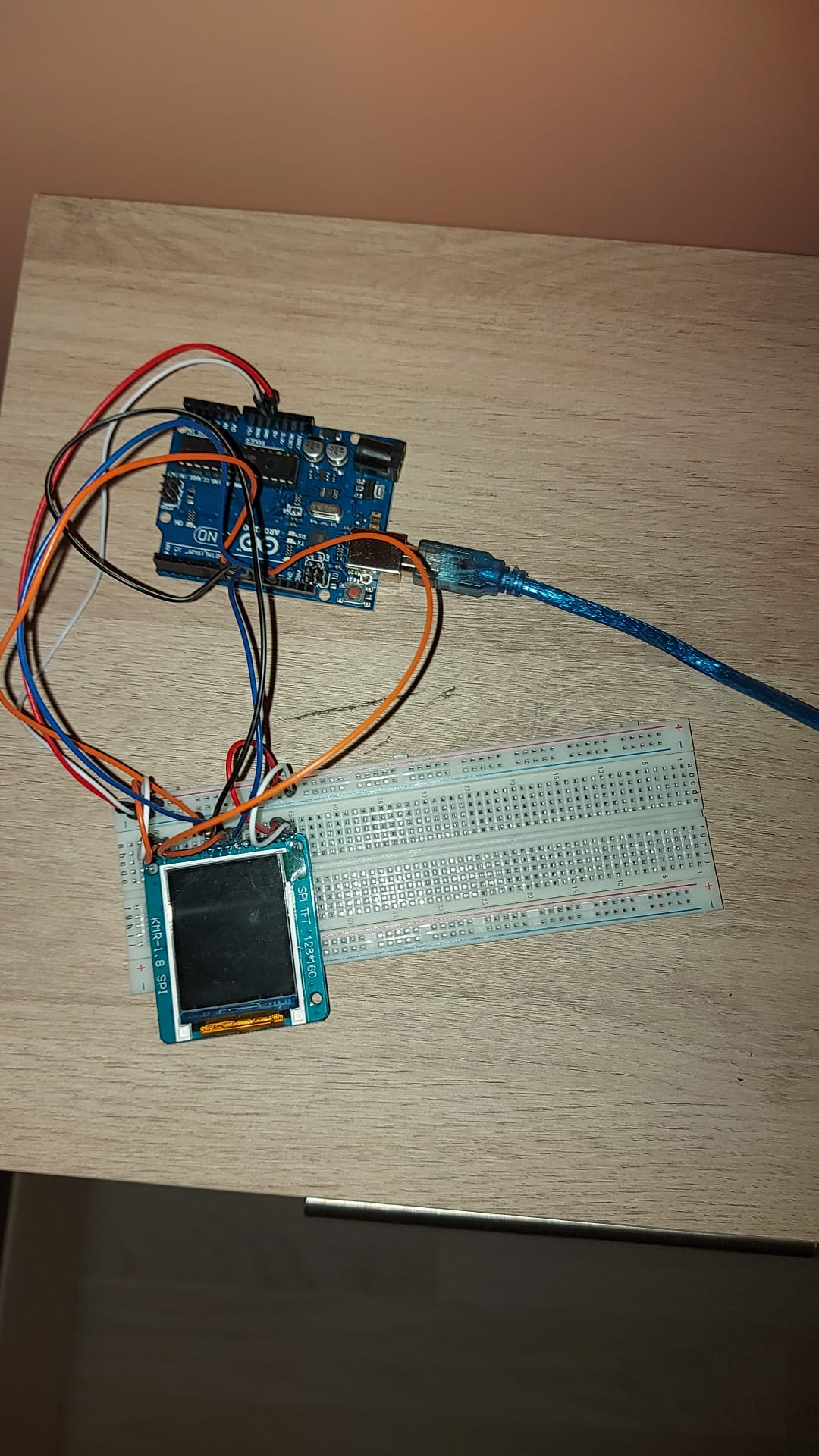
|  |  |  |  |
| --- | --- | --- | --- |
| Generația 1 | Generația 2 | Generația 3 | Concluzie |
| A picture containing shape  Description automatically generated | A picture containing text, clipart  Description automatically generated | A picture containing purple, electronics, keyboard, colorful  Description automatically generated | ”moare” |
| A picture containing electronics  Description automatically generated | A picture containing text, clipart, keyboard  Description automatically generated | A picture containing purple, electronics, keyboard, colorful  Description automatically generated | ”moare” |
| A picture containing shape  Description automatically generated | A picture containing text, clipart  Description automatically generated | A picture containing purple, electronics, keyboard, colorful  Description automatically generated | ”moare” |
| A picture containing shape  Description automatically generated | A picture containing shape  Description automatically generated | A picture containing shape  Description automatically generated | ”stabil” (”block”) |
| A picture containing shape  Description automatically generated | Shape  Description automatically generated | A picture containing shape  Description automatically generated | ”periodic” (perioadă 2) |

# Implementarea temei alese

Pentru implementarea temei alese am folosit următoarele componente:

1. Placuta Arduino Uno
2. Breadboard
3. Firele Jumper
4. Ecran LCD 1.8’’ SPI TFT

Am folosit următorul circuit:



Aplicația trece prin două faze: prima este cea de formare a matricei de joc, printr-o randomizare, iar cea de-a doua fază este cea de simulare efectivă. Mai jos se poate observa în prima imagine faza de formare a matricei, iar în cea de-a doua, faza de simulare.

A picture containing text

Description automatically generated

A picture containing text

Description automatically generated

Matricea are dimensiunea de 20x20 și am considerat că la început jumătate dintre celule sunt “vii” iar celelalte sunt “moarte”. Randomizarea matricei se face astfel: cât timp celulele “vii” sunt mai puțin de jumătate din numărul total de celule se găsesc două numere la întâmplare între 0 și mărimea matricei (în cazul nostru 20), iar dacă elementul de la coordonatele considerate de cele două numere este “mort”, atunci acesta devine viu. După faza de generare a matricei se realizează simularea comportamentului. La fiecare pas, pentru fiecare element în parte se calculează numărul de vecini și se respectă regulile jocului prezentate mai sus. În final, pentru o mai bună vizualizare a elementelor, am considerat fiecare element al matricei ca 4 puncte pe display.

screen.point(x\*3 - 1 + screen\_offset\_x, y\*3 - 1 + screen\_offset\_y);

screen.point(x\*3 - 1 + screen\_offset\_x, y\*3 + screen\_offset\_y);

screen.point(x\*3 - 1 + screen\_offset\_x, y\*3 + 1 + screen\_offset\_y);

screen.point(x\*3 + screen\_offset\_x, y\*3 - 1 + screen\_offset\_y);

screen.point(x\*3 + screen\_offset\_x, y\*3 + screen\_offset\_y);

screen.point(x\*3 + screen\_offset\_x, y\*3 + 1 + screen\_offset\_y);

screen.point(x\*3 + 1 + screen\_offset\_x, y\*3 - 1 + screen\_offset\_y);

screen.point(x\*3 + 1 + screen\_offset\_x, y\*3 + screen\_offset\_y);

screen.point(x\*3 + 1 + screen\_offset\_x, y\*3 + 1 + screen\_offset\_y);

# Bibliografie

1. <http://pi.math.cornell.edu/~lipa/mec/lesson6.html>
2. <https://www.arduino.cc/reference/en/libraries/tft/>
3. <https://docs.arduino.cc/retired/getting-started-guides/TFT>