Proiect LCD X&O cu doi jucători

**Microcontrolere**

**Microcontrolere – arhitecturi și programare**

**Studenți:**

**Vasiluț Alexandra**

**Bîrgovan Alexandru-Dumitrel**

Suceava, 2022

Cuprins

[1. Noțiuni teoretice 3](#_Toc156690815)

[2. Descrierea temei alese 4](#_Toc156690816)

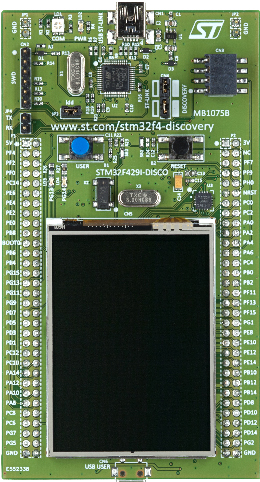
[3. Implementarea si testarea aplicației 5](#_Toc156690817)

[4. Interpretarea rezultatelor și concluzii 18](#_Toc156690818)

[Bibliografie 19](#_Toc156690819)

[Anexe 19](#_Toc156690820)

# Noțiuni teoretice

 Placa de dezvoltare STM32F42I-DISCO (Discovery) este o platformă de evaluare creată de STMicroelectronics pentru a facilita dezvoltarea și testarea aplicațiilor bazate pe microcontrolerele STM32F42x. Placa este echipată cu un microcontroller STM32F429ZIT6, care face parte din familia de microcontrolere STM32F4 de la STMicroelectronics. Acest microcontroller are un nucleu ARM Cortex-M4F care rulează la o frecvență de până la 180 MHz și oferă o gamă largă de funcționalități și periferice. STM32F42I-DISCO dispune de 1 MB de memorie Flash și 256 KB de RAM, oferind spațiu suficient pentru stocarea programelor și a datelor. Placa vine cu diverse opțiuni de conectivitate, inclusiv USB, UART, SPI, I2C și altele, permițând integrarea cu diverse dispozitive și sisteme. Include un ecran LCD TFT color de 2,4 inch pentru afișarea informațiilor și a interfețelor grafice în proiectele tale. Dispune de butoane și LED-uri care pot fi utilizate pentru interacțiunea și semnalizarea în aplicațiile dezvoltate. Placa include un debugger și programator integrat ST-LINK/V2 pentru programarea și depanarea facilă a codului. Poate fi alimentată fie de la un conector USB, fie de la o sursă externă de alimentare. Se integrează bine cu mediul de dezvoltare STM32CubeIDE, iar utilizatorii pot beneficia de biblioteci și resurse software oferite de STMicroelectronics pentru programarea facilă a microcontrolerului. Placa este compatibilă cu mediul de dezvoltare Arduino, ceea ce deschide posibilitatea de a utiliza o varietate de biblioteci și shield-uri Arduino. STMicroelectronics furnizează documentație detaliată, inclusiv schemă electrică, manuale de utilizare și exemple de cod pentru a facilita procesul de dezvoltare.

# Descrierea temei alese

Pentru a crea proiectul de joc X și O cu doi jucători pe placa STM32F42I-DISCO, a trebuit să urmăm mai mulți pași. Acești pași includ inițializarea hardware-ului, programarea butoanelor și afișajului, implementarea logicii jocului și gestionarea interacțiunilor cu jucătorii.

Configurarea GPIO pentru butoane și LED-uri, am identificat GPIO-urile care vor fi utilizate pentru butoanele de joc și LED-urile care indică starea jocului.

Implementarea afișajului LCD, am utilizat bibliotecile și funcțiile de afișare disponibile pentru a inițializa și controla ecranul LCD de pe placă. Am reprezentat grila de joc și am utilizat butonul “reset” pentru reînceperea jocului.

Implementarea logicii jocului, am creat o matrice pentru a reprezenta starea jocului (cu "X", "O" sau gol în fiecare celulă).

Interacțiune cu jucătorii, se face prin atingerea ecranului. Am implementat o rutină de detectare a atingerilor pentru a determina coordonatele punctului atins pe ecran.

Reprezentarea grilei pe ecran, am desenat grila de joc pe ecran astfel încât să fie ușor de înțeles și interactivă. Ne-am asigurat că fiecare celulă a grilei poate fi atinsă individual. Atunci când un jucător atinge o celulă, am verificat dacă mutarea este validă. Alternanța între jucători este gestionată în funcție de atingerile succesive.

Actualizarea ecranului, am actualizat ecranul pentru a reflecta starea curentă a jocului după fiecare mutare.

Feedback vizual, am furnizați feedback vizual pentru a confirma că o mutare a fost înregistrată cu succes.

Optimizări pentru ecran tactil, am asigurat că interfața utilizatorului este optimizată pentru ecranul tactil, cu elemente de dimensiuni adecvate pentru a fi atinse confortabil.

Testare, am testat jocul pe placă și ne-am asigurat că funcționează conform așteptărilor.

# Implementarea si testarea aplicației

Bibliotecile folosite pentru implementarea aplicației sunt:

* stm32f429i\_discovery\_lcd.h (această bibliotecă este destinată gestionării ecranului LCD pe placa de dezvoltare STM32F429I Discovery; oferă funcții și definiții pentru a controla și a comunica cu LCD-ul, inclusiv pentru setarea layerelor, desenarea pe ecran);
* stm32f429i\_discovery\_sdram.h (se ocupă de gestionarea SDRAM “Synchronous Dynamic Random Access Memory” de pe placa de dezvoltare; furnizează funcții și definiții pentru inițializarea și gestionarea SDRAM-ului, care este utilizat pentru stocarea temporară a datelor în timpul execuției programului);
* stm32f429i\_discovery\_ts.h (această bibliotecă gestionează ecranul tactil “Touch Screen” de pe placa STM32F429I Discovery; oferă funcții pentru inițializarea și citirea datelor de la ecranul tactil, permițând o interacțiune cu dispozitivul prin atingeri);
* stlogo.h (acest fișier probabil conține o imagine a logo-ului STMicroelectronics; utilizat pentru a încărca logo-ul în program și pentru a-l afișa pe ecran, oferind o interfață vizuală inițială sau de pornire);
* stdio.h (această bibliotecă standard C oferă funcționalități pentru citirea și scrierea datelor pe consolă, cum ar fi funcțiile printf și scanf)
* stdbool.h (această bibliotecă standard C furnizează definiții pentru tipul de date bool și valorile constante true și false; utilizată pentru a adăuga suport pentru variabile booleane în limbajul C)

Sunt declarate variabilele care vor fi utilizate în program și perifericele hardware precum DMA2D, I2C, LTDC, SPI, TIM, UART și SDRAM.

Se face inițializarea și configurarea afișajului LCD pe un dispozitiv bazat pe microcontroler STM32. Ele pot face parte dintr-un program care afișează informații sau efectuează anumite acțiuni vizuale pe ecran.

* BSP\_LCD\_Init(); (inițializează driverul LCD “Liquid Crystal Display”; acesta este un pas important pentru a pregăti interfața hardware pentru a controla afișajul)
* BSP\_LCD\_LayerDefaultInit(LCD\_BACKGROUND\_LAYER, LCD\_FRAME\_BUFFER); (inițializează stratul de fundal “layerele” de pe afișajul LCD cu configurații implicite; funcția primește doi parametri: tipul de strat și buffer-ul de cadru pentru stratul respective)
* BSP\_LCD\_LayerDefaultInit(LCD\_FOREGROUND\_LAYER, LCD\_FRAME\_BUFFER); (similar cu linia anterioară, această linie inițializează stratul de prim-plan al afișajului LCD)
* BSP\_LCD\_SelectLayer(LCD\_FOREGROUND\_LAYER); (selectează stratul de prim-plan ca strat activ pe care se vor desena elementele; aceasta stabilește pe care dintre cele două straturi “fundal sau prim-plan” se va desena în mod curent)
* BSP\_LCD\_DisplayOn(); (activează afișajul LCD pentru a începe afișarea informațiilor pe ecran)
* BSP\_LCD\_Clear(LCD\_COLOR\_BLUE); (curăță stratul activ cu o culoarea albastru)
* HAL\_Delay(400); (întârziere de 400 de milisecunde “0.4 secunde”; acest timp de întârziere poate servi ca o pauză între diverse etape ale programului)
* BSP\_LCD\_Clear(LCD\_COLOR\_WHITE); (curăță stratul activ cu o altă culoare, aici alb)
* BSP\_LCD\_SetTextColor(LCD\_COLOR\_RED); (stabilește culoarea textului pentru afișare la roșu)
* BSP\_LCD\_SetFont(&Font16); (stabilește fontul pentru afișare; se folosește un font cu dimensiunea de 16 pixeli)

Desenarea aspectului vizual al tablei de joc Tic Tac Toe pe ecranul LCD.

* BSP\_LCD\_DisplayStringAtLine(0, (uint8\_t\*)" TIC TAC TOE"); (afisează textul "TIC TAC TOE" la linia 0 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(1, (uint8\_t\*)" Player:"); (afisează textul "Player:" la linia 1 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(2, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 2 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(3, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 3 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(4, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 4 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(5, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 5 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(6, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 6 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(7, (uint8\_t\*)"------+-------+------"); (afisează o linie orizontală care separă rândurile la linia 7 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(8, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 8 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(9, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 9 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(10, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 10 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(11, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 11 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(12, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 12 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(13, (uint8\_t\*)"------+-------+------"); (afisează o linie orizontală care separă rândurile la linia 13 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(14, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 14 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(15, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 15 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(16, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 16 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(17, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 17 a ecranului LCD)
* BSP\_LCD\_DisplayStringAtLine(18, (uint8\_t\*)" | |"); (afisează o linie a tablei de joc la linia 18 a ecranului LCD)

Se vor seta culorile, dimensiunile și fonturile pe ecranul LCD, pentru a face afișarea textului mai atractivă și pentru a gestiona inițializarea funcționalității touch.

* BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLUE); (setează culoarea textului pe albastru)
* BSP\_LCD\_SetFont(&Font24); (setează fontul pentru text la dimensiunea 24)
* BSP\_LCD\_SetFont(&Font20); (setează fontul pentru text la dimensiunea 20)
* HAL\_Delay(1000); (realizează o întârziere de 1000 de milisecunde “1 secundă”)
* BSP\_LCD\_SetFont(&Font16); (setează fontul pentru text la dimensiunea 16)
* BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK); (setează culoarea textului pe negru)
* BSP\_LCD\_SetTextColor(LCD\_COLOR\_RED); (setează culoarea textului pe roșu)
* BSP\_LCD\_SetTextColor(LCD\_COLOR\_BLACK); (setează culoarea textului pe negru)
* if (BSP\_TS\_Init(BSP\_LCD\_GetXSize(), BSP\_LCD\_GetYSize()) == TS\_ERROR) { ... }.. (inițializează funcționalitatea touch și verifică dacă aceasta s-a realizat cu success; afișează un mesaj de eroare pe ecran dacă apare una)
* printf("\n\rCoordonate ecran touch: x=%d, y=%d\n\r", BSP\_LCD\_GetXSize(), BSP\_LCD\_GetYSize()); (afișează dimensiunile ecranului touch în terminal)
* bool player1 = true; (inițializează o variabilă booleană player1 cu valoarea true, pentru a reprezenta faptul că jucătorul 1 începe jocul)
* bool flag1 = false; (inițializează o serie de variabile booleane flag1 până la flag9 cu valoarea false pentru a urmări starea și condițiile în joc)

Urmează o buclă în care se verifică starea ecranului tactil și se iau decizii în funcție de aceasta. Informațiile sunt stocate în variabila TS\_State. Se verifică dacă un touch este detectat pe ecran. Dacă TouchDetected este diferit de zero, înseamnă că un touch a fost detectat și se intră în blocul de cod corespunzător. Se declară două buffer-e de caractere (desc și firstline) pentru a stoca informații despre coordonate și primul rând. Se verifică coordonatele atinse și afișează "X" sau "O" în funcție de jucător și locația pe tablă. De asemenea, se asigură că o celulă poate fi setată doar o dată prin intermediul variabilelor de tip flag. Dacă o celulă a fost deja selectată, nu se mai face nimic pentru acea celulă.

Se configurează și inițializează sistemul de ceasuri al microcontrolerului.

* RCC\_OscInitTypeDef RCC\_OscInitStruct = {0}; (se definește o structură pentru configurarea oscilatoarelor)
* RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0}; (se definește o structură pentru configurarea ceasurilor CPU, AHB și APB)
* RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct = {0}; (se definește o structură pentru configurarea ceasurilor pentru periferice)
* \_\_HAL\_RCC\_PWR\_CLK\_ENABLE(); (se activează ceasul pentru blocul de control al consumului de energie “Power Control”)
* \_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1); (se configurează nivelul de tensiune al regulatorului pentru a obține performanțe optime)
* RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE; (se specifică tipul oscilatorului „High-Speed External – HSE”)
* RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON; (se activează oscilatorul HSE)
* RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON; (se activează modul PLL „Phase-Locked Loop” pentru a ajusta frecvența oscilatorului principal)
* RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE; (se specifică sursa pentru PLL, care este HSE)
* RCC\_OscInitStruct.PLL.PLLM = 8; (se specifică divizorul M pentru PLL)
* RCC\_OscInitStruct.PLL.PLLN = 336; (se specifică factorul de multiplicare N pentru PLL)
* RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2; (se specifică divizorul P pentru PLL)
* RCC\_OscInitStruct.PLL.PLLQ = 5; (se specifică divizorul Q pentru PLL)
* if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK) { Error\_Handler(); } (se aplică configurația oscilatorului și se verifică dacă configurarea a fost realizată cu succes)
* RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK |RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2; (se specifică tipurile de ceasuri care vor fi configurate)
* RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK; (se specifică sursa pentru ceasul de sistem „SYSCLK” ca fiind PLL)
* RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1; (se specifică divizorul pentru ceasul AHB „Advanced High-Performance Bus”)
* RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4; (se specifică divizorul pentru ceasul APB1)
* RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2; (se specifică divizorul pentru ceasul APB2)
* if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_5) != HAL\_OK) { Error\_Handler(); } (se aplică configurația ceasurilor și se verifică dacă configurarea a fost realizată cu succes)
* PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_LTDC; (se specifică perifericul LTDC pentru care se configurează ceasul „Layer Time Domain Controller”)
* PeriphClkInitStruct.PLLSAI.PLLSAIN = 192; (se specifică factorul de multiplicare SAIN pentru PLLSAI „PLL Sub-Block A”)
* PeriphClkInitStruct.PLLSAI.PLLSAIR = 2; (se specifică factorul de reducere SAIR pentru PLLSAI)
* PeriphClkInitStruct.PLLSAIDivR = RCC\_PLLSAIDIVR\_2; (se specifică divizorul pentru PLLSAI)
* if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct) != HAL\_OK) { Error\_Handler(); } (se aplică configurația pentru perifericul LTDC și se verifică dacă configurarea a fost realizată cu succes)

Se setează și inițializează DMA2D pentru a efectua operații de transfer de la memorie la memorie cu anumite configurații de culoare și alfa

* hdma2d.Instance = DMA2D; (setează instanța DMA2D utilizată)
* hdma2d.Init.Mode = DMA2D\_M2M; (configurează modul DMA2D pentru transfer de la memorie la memorie)
* hdma2d.Init.ColorMode = DMA2D\_OUTPUT\_RGB565; (specifică modul de culoare pentru operațiile de transfer)
* hdma2d.Init.OutputOffset = 0; (specifică offset-ul pentru destinația transferului)
* hdma2d.LayerCfg[1].InputOffset = 0; (specifică offset-ul pentru sursa transferului)
* hdma2d.LayerCfg[1].InputColorMode = DMA2D\_INPUT\_RGB565; (specifică modul de culoare pentru sursa transferului)
* hdma2d.LayerCfg[1].AlphaMode = DMA2D\_NO\_MODIF\_ALPHA; (modificarea alfa nu este utilizată)
* hdma2d.LayerCfg[1].InputAlpha = 0; (valoarea alfa pentru sursa transferului)

În cazul în care inițializarea DMA2D nu reușește, declanșează o funcție de gestionare a erorilor. În cazul în care configurarea stratului DMA2D nu reușește, declanșează o funcție de gestionare a erorilor.

Se configurează și inițializează modulul I2C3 pe microcontroller.

* hi2c3.Instance = I2C3; (setează instanța I2C3 utilizată)
* hi2c3.Init.ClockSpeed = 100000; (setează viteza de comunicație la 100kHz)
* hi2c3.Init.DutyCycle = I2C\_DUTYCYCLE\_2; (se setează ciclul de lucru al semnalului I2C la 2)
* hi2c3.Init.OwnAddress1 = 0; (se setează adresa proprie a dispozitivului I2C, nu este folosită în modul 7 biți)
* hi2c3.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;: Setează modul de adresare la 7 biți.
* hi2c3.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE; (se dezactivează modul de adresare duală)
* hi2c3.Init.OwnAddress2 = 0; (se setează a doua adresă proprie)
* hi2c3.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE; (se dezactivează modul de apel general)
* hi2c3.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE; (se dezactivează modul de întindere a perioadelor de ceas)
* if (HAL\_I2C\_Init(&hi2c3) != HAL\_OK) (inițializează I2C3 cu configurațiile specificate și verifică dacă inițializarea a avut succes)
* if (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c3, I2C\_ANALOGFILTER\_DISABLE) != HAL\_OK) (configurează filtrul analog și verifică dacă configurarea a avut succes)
* if (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c3, 0) != HAL\_OK) (se configurează filtrul digital și verifică dacă configurarea a avut succes)

Această secvență de cod este esențială pentru configurarea și inițializarea modulului LTDC, care gestionează afișarea pe ecran a microcontroller-ului.

* LTDC\_LayerCfgTypeDef pLayerCfg = {0}; (structura de configurare a stratului LTDC)
* hltdc.Instance = LTDC; (se setează instanța LTDC utilizată)

Configurările de bază ale LTDC.

* hltdc.Init.HSPolarity = LTDC\_HSPOLARITY\_AL; (polaritatea orizontală a semnalului de sincronizare)
* hltdc.Init.VSPolarity = LTDC\_VSPOLARITY\_AL; (polaritatea verticală a semnalului de sincronizare)
* hltdc.Init.DEPolarity = LTDC\_DEPOLARITY\_AL; (polaritatea semnalului Data Enable)
* hltdc.Init.PCPolarity = LTDC\_PCPOLARITY\_IPC; (polaritatea semnalului de pixel clock)

Configurează caracteristicile temporale și spațiale ale semnalului video care controlează afișarea pe ecran.

* hltdc.Init.HorizontalSync = 9; (se specifică câti pixeli sunt utilizați pentru semnalul de sincronizare orizontală)
* hltdc.Init.VerticalSync = 1; (se specifică câte linii sunt utilizați pentru semnalul de sincronizare verticală)
* hltdc.Init.AccumulatedHBP = 29; (se setează numărul total de pixeli pentru sincronizarea orizontală și back porch, aceasta reprezintă suma dintre durata sincronizării orizontale și back porch)
* hltdc.Init.AccumulatedVBP = 3; (se setează numărul total de linii pentru sincronizarea verticală și back porch)
* hltdc.Init.AccumulatedActiveW = 269; (se setează numărul total de pixeli pe linie, inclusiv zona activă a ecranului și zonele de sincronizare orizontală și back porch)
* hltdc.Init.AccumulatedActiveH = 323; (se setează numărul total de linii, inclusiv zona activă a ecranului și zonele de sincronizare verticală și back porch)
* hltdc.Init.TotalWidth = 279; (se setează lățimea totală a liniei video)
* hltdc.Init.TotalHeigh = 327; (se setează înălțimea totală a cadrului video)
* hltdc.Init.Backcolor.Blue = 0; (se setează componenta albastră a culorii de fundal (background) a ecranului)
* hltdc.Init.Backcolor.Green = 0; (se setează componenta verde a culorii de fundal a ecranului)
* hltdc.Init.Backcolor.Red = 0; (se setează componenta roșie a culorii de fundal a ecranului)

În cazul în care initializarea LTDC nu reuseste, declanseaza o functie de gestionare a erorilor.

Configurările stratului (layer) LTDC.

* pLayerCfg.WindowX0 = 0; (coordonata X a colțului stânga-sus al ferestrei straturii (layer). Aceasta indică începutul zonei active pe axa orizontală)
* pLayerCfg.WindowX1 = 240; (coordonata X a colțului dreapta-jos al ferestrei straturii (layer). Aceasta indică sfârșitul zonei active pe axa orizontală)
* pLayerCfg.WindowY0 = 0; (coordonata Y a colțului stânga-sus al ferestrei straturii (layer). Aceasta indică începutul zonei active pe axa verticală)
* pLayerCfg.WindowY1 = 320; (coordonata Y a colțului dreapta-jos al ferestrei straturii (layer). Aceasta indică sfârșitul zonei active pe axa verticală)
* pLayerCfg.PixelFormat = LTDC\_PIXEL\_FORMAT\_RGB565; (formatul pixelilor, unde sunt alocate 5 biți pentru roșu, 6 biți pentru verde și 5 biți pentru albastru)
* pLayerCfg.Alpha = 255; (se setează nivelul de opacitate al întregii straturi. Un nivel de 255 înseamnă că stratura este complet opacă, fără transparență)
* pLayerCfg.Alpha0 = 0; (se setează nivelul de opacitate al pixelilor care au o valoare de transparență (alpha) de 0)

Următoarele setări adaugă informații despre modul în care stratul curent va interacționa cu alte straturi, precum și despre cum se va comporta la nivelul de amestecare și opacitate.

* pLayerCfg.BlendingFactor1 = LTDC\_BLENDING\_FACTOR1\_CA; (se specifică factorul de amestecare pentru primul strat. Se utilizează "Color (C) \* Alpha (A)" pentru factorul de amestecare 1. Aceasta înseamnă că culoarea stratului este multiplicată cu nivelul de transparență)
* pLayerCfg.BlendingFactor2 = LTDC\_BLENDING\_FACTOR2\_CA; (se specifică factorul de amestecare pentru al doilea strat. Se utilizează "Color (C) \* Alpha (A)" pentru factorul de amestecare 2)
* pLayerCfg.FBStartAdress = 0; (adresa de început este setată la 0, ceea ce ar putea însemna că datele sunt stocate în prima zonă a memoriei RAM)
* pLayerCfg.ImageWidth = 240; (lățimea imaginii pe acest strat. Acest lucru indică câți pixeli sunt prezentați pe axa orizontală)
* pLayerCfg.ImageHeight = 320; (înălțimea imaginii pe acest strat. Acest lucru indică câți pixeli sunt prezentați pe axa verticală)
* pLayerCfg.Backcolor.Blue = 0; (componenta de culoare albastră a culorii de fundal a stratului)
* pLayerCfg.Backcolor.Green = 0; (componenta de culoare verde a culorii de fundal a stratului)
* pLayerCfg.Backcolor.Red = 0; (componenta de culoare roșie a culorii de fundal a stratului)

O inițializare a perifericului SPI5 (Serial Peripheral Interface) în cadrul unui proiect bazat pe platforma STM32.

* hspi5.Instance = SPI5; (se specifică instanța SPI pe care se va face inițializarea SPI5)
* hspi5.Init.Mode = SPI\_MODE\_MASTER; (se configurează modul de operare al SPI ca master, masterul controlează comunicația)
* hspi5.Init.Direction = SPI\_DIRECTION\_2LINES; (configurare pentru două linii de date, adică o linie de date MOSI (Master Out, Slave In) și o linie de date MISO (Master In, Slave Out))
* hspi5.Init.DataSize = SPI\_DATASIZE\_8BIT; (dimensiunea datelor este setată la 8 biți pe transfer. Aceasta indică cât de multe date sunt trimise sau primite la fiecare tranziție)
* hspi5.Init.CLKPolarity = SPI\_POLARITY\_LOW; (se specifică polaritatea semnalului de ceas (CLK) ca fiind LOW în starea de repaus)
* hspi5.Init.CLKPhase = SPI\_PHASE\_1EDGE; (se specifică faza în care datele sunt stabilite sau citite, pe prima margine de urcare (1EDGE))
* hspi5.Init.NSS = SPI\_NSS\_SOFT; (se utilizează NSS software, ceea ce înseamnă că gestionarea semnalului NSS (Slave Select) este realizată software)
* hspi5.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_16; (se setează prescaler-ul pentru stabilirea ratei de transfer de date, este setat la 16)
* hspi5.Init.FirstBit = SPI\_FIRSTBIT\_MSB; (se specifică ordinea de transmisie a bitilor, cel mai semnificativ bit este transmis primul)
* hspi5.Init.TIMode = SPI\_TIMODE\_DISABLE; (dezactivează modul de comunicare prin transfer de mesaje (TI mode))
* hspi5.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE; (dezactivează calculul CRC (Cyclic Redundancy Check) pentru integritatea datelor)
* hspi5.Init.CRCPolynomial = 10; (dacă CRC este activat, aici se specifică polinomul utilizat pentru calcul)

Fragmentul de cod verifică dacă inițializarea perifericului SPI5 (HAL\_SPI\_Init(&hspi5)) a fost realizată cu succes sau nu.

Configurează și inițializează un timer hardware.

* TIM\_ClockConfigTypeDef sClockSourceConfig = {0}; (se declară o structură pentru configurarea sursei de ceas a timerului)
* TIM\_MasterConfigTypeDef sMasterConfig = {0}; (se declară o structură de tip pentru configurarea modului maestru al timerului)
* htim1.Instance = TIM1; (se specifică instanța hardware a timerului)
* htim1.Init.Prescaler = 0; (se setează prescaler-ul timerului la 0. Prescaler-ul este folosit pentru a împărți frecvența de intrare a timerului și pentru a ajusta rezoluția acestuia)
* htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP; (se configurează modulul de numărare al timerului ca fiind "UP", ceea ce înseamnă că timerul va număra de la 0 la valoarea specificată în registrul Period)
* htim1.Init.Period = 0; (se setează valoarea la care timerul va ajunge înainte de a se reseta. Este 0, dar de obicei, această valoare va fi setată în funcție de cerințele aplicației și frecvența dorită)
* htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1; (se configurează divizarea ceasului timerului. Nu se efectuează nicio divizare)
* htim1.Init.RepetitionCounter = 0; (se setează un contor de repetare. Acesta este 0 și nu se efectuează repetare)
* htim1.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_DISABLE; (se dezactivează încărcarea automată a valorii de auto-reîncărcare)

Se inițializează timerul de bază folosind configurația specificată. Dacă inițializarea nu reușește, se apelează Error\_Handler().

* sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL; (se configurează sursa internă de ceas pentru timer)

Se aplică configurarea sursei de ceas timerului. În cazul unei erori, se apelează Error\_Handler().

* sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET; (se configurează semnalul de ieșire al modului maestru pentru a reseta timerul)
* sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE; (se dezactivează modulul maestru-sclav al timerului)

Se aplică configurarea modului maestru al timerului. În cazul unei erori, se apelează Error\_Handler().

* huart1.Instance = USART1; (se specifică instanța hardware a modulului)
* huart1.Init.BaudRate = 115200; (se configurează rata de baud a modulului UART la 115200 biți pe secundă)
* huart1.Init.WordLength = UART\_WORDLENGTH\_8B; (se setează lungimea cuvântului la 8 biți)
* huart1.Init.StopBits = UART\_STOPBITS\_1; (se configurează numărul de biți de stop la 1)
* huart1.Init.Parity = UART\_PARITY\_NONE; (se dezactivează paritatea)
* huart1.Init.Mode = UART\_MODE\_TX\_RX; (se configurează modulul UART pentru transmitere și recepție simultană)
* huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE; (se dezactivează controlul hardware al fluxului)
* huart1.Init.OverSampling = UART\_OVERSAMPLING\_16; (se setează oversampling-ul la 16x)

Se inițializează modulul UART folosind configurația specificată. Dacă inițializarea nu reușește, se apelează Error\_Handler().

* SET\_BIT(huart1.Instance->CR1, USART\_CR1\_RXNEIE); (se setează bitul RXNEIE din registrul de control CR1. Acesta activează întreruperea pentru evenimentul de recepție)
* HAL\_NVIC\_SetPriority(USART1\_IRQn, 0, 0); (se configurează prioritatea pentru întreruperea legată de USART1)
* HAL\_NVIC\_EnableIRQ(USART1\_IRQn); (se activează întreruperea pentru USART1)

Această secvență de cod configurează și inițializează modulul de control a memoriei flash externe (FMC) pentru utilizarea unei memorii SDRAM.

* FMC\_SDRAM\_TimingTypeDef SdramTiming = {0}; (se definește o structură numită SdramTiming pentru a configura timpii specifici SDRAM)
* hsdram1.Instance = FMC\_SDRAM\_DEVICE; (se specifică instanța hardware a modulului FMC pentru memorie SDRAM)
* hsdram1.Init.SDBank = FMC\_SDRAM\_BANK2; (se specifică bancul de memorie SDRAM utilizat)
* hsdram1.Init.ColumnBitsNumber = FMC\_SDRAM\_COLUMN\_BITS\_NUM\_8; (se configurează numărul de biți pentru adresa coloanei SDRAM la 8 biți)
* hsdram1.Init.RowBitsNumber = FMC\_SDRAM\_ROW\_BITS\_NUM\_12; (se configurează numărul de biți pentru adresa rândului SDRAM la 12 biți)
* hsdram1.Init.MemoryDataWidth = FMC\_SDRAM\_MEM\_BUS\_WIDTH\_16; (se setează lățimea datelor la 16 biți)
* hsdram1.Init.InternalBankNumber = FMC\_SDRAM\_INTERN\_BANKS\_NUM\_4; (se specifică numărul intern de bănci SDRAM)
* hsdram1.Init.CASLatency = FMC\_SDRAM\_CAS\_LATENCY\_3; (se configurează latenta CAS a SDRAM la 3 cicluri)
* hsdram1.Init.WriteProtection = FMC\_SDRAM\_WRITE\_PROTECTION\_DISABLE; (se dezactivează protecția la scriere pentru SDRAM)
* hsdram1.Init.SDClockPeriod = FMC\_SDRAM\_CLOCK\_PERIOD\_2; (se configurează perioada de ceas pentru SDRAM la 2 cicluri)
* hsdram1.Init.ReadBurst = FMC\_SDRAM\_RBURST\_DISABLE; (se dezactivează modul de citire burst)
* hsdram1.Init.ReadPipeDelay = FMC\_SDRAM\_RPIPE\_DELAY\_1; (se configurează delay-ul pentru read pipe la 1 ciclu)
* SdramTiming.LoadToActiveDelay = 2; (se configurează delay-ul de la Load Mode la Active Mode)
* SdramTiming.ExitSelfRefreshDelay = 7; (se configurează delay-ul pentru a ieși din Self Refresh Mode)
* SdramTiming.SelfRefreshTime = 4; (se configurează timpul de auto-refresh pentru SDRAM)
* SdramTiming.RowCycleDelay = 7; (se configurează delay-ul pentru ciclul de rând al SDRAM)
* SdramTiming.WriteRecoveryTime = 3; (se configurează timpul de recuperare pentru scriere)
* SdramTiming.RPDelay = 2; (se configurează delay-ul pentru Precharge)
* SdramTiming.RCDDelay = 2; (se configurează delay-ul pentru Read to Write Delay)

Se inițializează modulul SDRAM folosind configurația specificată în structura SdramTiming. Dacă inițializarea nu reușește, se apelează Error\_Handler().

Această secvență de cod configurează și inițializează porturile GPIO.

* GPIO\_InitTypeDef GPIO\_InitStruct = {0}; (se declară o structură pentru a configura parametrii GPIO)
* \_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOC)
* \_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOF)
* \_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOH)
* \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOA)
* \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOB)
* \_\_HAL\_RCC\_GPIOG\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOG)
* \_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOE)
* \_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE(); (se activează ceasul pentru portul GPIOD)
* HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_4, GPIO\_PIN\_RESET); (se scrie nivelul logic "RESET" pe pinii 1, 2 și 4)
* HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET); (se scrie nivelul logic "RESET" pe pinul 7)
* HAL\_GPIO\_WritePin(GPIOD, GPIO\_PIN\_12|GPIO\_PIN\_13, GPIO\_PIN\_RESET); (se scrie nivelul logic "RESET" pe pinii 12 si 13)
* HAL\_GPIO\_WritePin(GPIOG, GPIO\_PIN\_13|GPIO\_PIN\_14, GPIO\_PIN\_RESET); (se scrie nivelul logic "RESET" pe pinii 13 si 14)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_4; (se specifică pinii 1, 2 si 4 de pe portul GPIOC care vor fi configurați)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP; (se configurează modul de lucru pentru pinii specificați ca ieșiri cu push-pull)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (se configurează modul de pull-up/pull-down pentru pinii specificați ca "None")
* GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW; (se configurează viteza de comutare pentru pinii specificați la o frecvență redusă)
* HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct); (se aplică configurarea pentru portul GPIOC)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_15; (se specifică pinii 0, 1, 2 si 15 de pe portul GPIOA care vor fi configurați)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING; (se configurează modul de lucru pentru pinii specificați ca intrări cu declanșare la nivel crescător)
* HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct); (se aplică configurarea pentru portul GPIOA)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_7; (specifică pinul 7 de pe portul GPIOA pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP; (configurează pinul 7 de pe portul GPIOA ca ieșire cu push-pull)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinul 7 de pe portul GPIOA)
* GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

(configurează viteza de comutare a pinului 7 de pe portul GPIOA la o frecvență scăzută)

* HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOA pe pinul 7)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_5; (specifică pinul 5 de pe portul GPIOC pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING; (configurează pinul 5 de pe portul GPIOC ca intrare cu declanșare la nivel crescător)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinul 5 de pe portul GPIOC)
* HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOC pe pinul 5)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_2; (specifică pinul 2 de pe portul GPIOB pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT; (configurează pinul 2 de pe portul GPIOB ca intrare)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinul 2 de pe portul GPIOB)
* HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOB pe pinul 2)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_12|GPIO\_PIN\_14|GPIO\_PIN\_15; (specifică pinii 12, 14 și 15 de pe portul GPIOB pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP; (configurează pinii 12, 14 și 15 de pe portul GPIOB ca ieșiri alternative cu push-pull)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinii 12, 14 și 15 de pe portul GPIOB)
* GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_VERY\_HIGH; (configurează viteza de comutare pentru pinii 12, 14 și 15 de pe portul GPIOB la o frecvență foarte înaltă)
* GPIO\_InitStruct.Alternate = GPIO\_AF12\_OTG\_HS\_FS; (configurează funcția alternativă a pinilor 12, 14 și 15 de pe portul GPIOB)
* HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOB pe pinii 12, 14 și 15)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_13; (specifică pinul 13 de pe portul GPIOB pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT; (configurează pinul 13 de pe portul GPIOB ca intrare)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinul 13 de pe portul GPIOB)
* HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOB pe pinul 13)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_11; (specifică pinul 11 de pe portul GPIOD pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT; (configurează pinul 11 de pe portul GPIOD ca intrare)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinul 11 de pe portul GPIOD)
* HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOD pe pinul 11)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_12|GPIO\_PIN\_13; (specifică pinii 12 și 13 de pe portul GPIOD pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP; (configurează pinii 12 și 13 de pe portul GPIOD ca ieșiri cu push-pull)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinii 12 și 13 de pe portul GPIOD)
* GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW; (configurează viteza de comutare pentru pinii 12 și 13 de pe portul GPIOD la o frecvență scăzută)
* HAL\_GPIO\_Init(GPIOD, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOD pe pinii 12 și 13)
* GPIO\_InitStruct.Pin = GPIO\_PIN\_13|GPIO\_PIN\_14; (specifică pinii 13 și 14 de pe portul GPIOG pentru configurație)
* GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP; (configurează pinii 13 și 14 de pe portul GPIOG ca ieșiri cu push-pull)
* GPIO\_InitStruct.Pull = GPIO\_NOPULL; (configurare fără rezistență de pull pentru pinii 13 și 14 de pe portul GPIOG)
* GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW; (configurează viteza de comutare pentru pinii 13 și 14 de pe portul GPIOG la o frecvență scăzută)
* HAL\_GPIO\_Init(GPIOG, &GPIO\_InitStruct); (aplică configurarea pentru portul GPIOG pe pinii 13 și 14)

# Interpretarea rezultatelor și concluzii

Se utilizează un microcontroller STM32F429 cu un set variat de periferice și interfețe, inclusiv ecran LCD, memorie SDRAM, interfeță I2C, SPI, UART, și altele.

Biblioteca STM32F429 Discovery este folosită pentru a interacționa cu ecranul LCD. Se realizează afișarea de text și desene pe ecran.

Există o configurare detaliată a sistemului, inclusiv inițializarea sistemului de ceas, configurarea perifericelor și interfețelor hardware.

Se folosesc funcții de callback pentru a trata evenimente precum expirarea unui timer (TIM6), iar unele funcții sunt apelate în cadrul acestor callback-uri.

Se realizează comunicare serială prin intermediul UART, cu o viteză de baud de 115200 bps.

Se configurează LTDC pentru a controla afișarea pe ecran. Aceasta include setarea parametrilor de sincronizare, dimensiunea ecranului și configurarea layer-elor pentru grafică.

Este configurată o interfață I2C la o viteză de 100 kbps.

Este configurată o interfață SPI în modul master cu specificații precum modul de date de 8 biți și polarițatea semnalului.

Se configurează DMA2D pentru transfer de date între memorie și ecran LCD.

Sunt configurate și utilizate porturi GPIO pentru controlul diverselor componente, cum ar fi LED-uri, butoane și altele.

Se configurează și inițializează memorie SDRAM pentru stocarea datelor.

Este un joc interactiv si distractiv în care te poți juca împreună cu un prieten.

# Bibliografie

<https://www.st.com/>

<https://ocw.cs.pub.ro/courses/pm/prj2019/rbarbascu/xsior336>

<https://ocw.cs.pub.ro/courses/pm/prj2013/rdobre/x-si-0>

<https://ocw.cs.pub.ro/courses/pm/prj2021/amocanu/tic-tac-toe_ar>

<https://github.com/topics/stm32f429i-disco>

<https://en.wikipedia.org/>

# Anexe