

המחלקה להנדסת חשמל

יישומי מחשב להנדסת אלקטרוניקה – 20210

פרוייקט סיום – איקס-עיגול בתקשורת RS-232

תאריך הגשת העבודה:

06.07.2025

מגיש:

יריב שוסברגר-316523406

שם המרצה:

פרופ' אלעזר פלקסר

תוכן עניינים

3	הקדמה
4	הצגת החלונות של התוכנית
4	1. הפאנל הראשי – תפריט הבית (MAIN)
5	2. פאנל הטעינה (LOAD_GAME)
6	3. פאנל ההנחיות (INSTRUCT)
7	4. פאנל סטאטוס החיבוריות (POPUP)
8	5. פאנל המשחק (GAME)
9	קבצי קוד הפרויקט
9	1. TicTacToe.c
12	2. ComThread.c – (LogicLib.dll)
15	3. FileLog.c – (LogicLib.dll)
17	4. Logic.c – (LogicLib.dll)
20	5. Audio.c – (LogicLib.dll)
23	הגדרות dll בקוד
24	תיעוד באגים ופתרונות תיקון לקוד
25	קישורים להורדה

הקדמה

היישום שפיתחתי הוא משחק "איקס-עיגול" דו-משתתפים הפועל על-גבי חיבור RS-232 סטנדרטי.

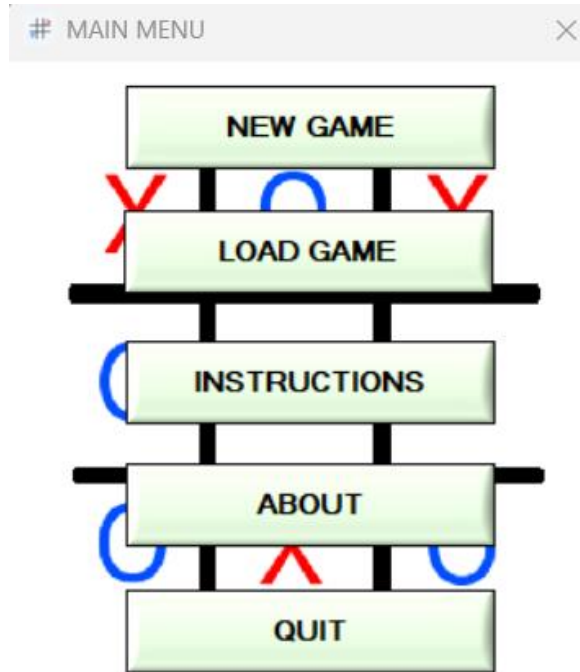
התוכנה מממשת תקשורת א-סינכרונית בין שני מחשבים באמצעות פרוטוקול hand-shake פשוט ('HELLO/OK' ופעימה (Heartbeat) לבדיקת חיבוריות), ומקצה תהליך עובד (Worker Thread) מקבילי לטיפול בחבילות מידע (Packets) לקריאה וכתיבה. ממשק המשתמש נבנה בסביבת LabWindows/CVI ומושתת על קובץ UIR יחיד: **תפריט ראשי, לוח המשחק, חלון טעינת משחקים וחלון סטטוס חיבור.**

מהלך המשחק מנוהל ב-Canvas של CVI: הלוח מצויר כתמונה סטטית, והסמלים (X/O) מצוירים כ-Bitmaps שקופים הממוקמים דינמית בתאי הלוח. כדי להבטיח קצב ריענון אחיד בין שני הצדדים, כל לחיצת עכבר נרשמת תחילה מקומית, נשלחת כ-Packet ('M', row, col) לצד המרוחק ומוזנת ליומן-מהלכים (FileLog) לטובת שמירה או שחזור משחק. מנגנון DLL (Lib Logic) מרכז את חוקי המשחק: בדיקת חוקיות המהלך, זיהוי ניצחון/תיקו, וניהול תור השחקן.

התוכנה מפרידה בבירור בין שכבת הלוגיקה, ממשק המשתמש, ניהול קבצים (CSV בלתי-דחוס) וה-Audio. הצלילים (WAV) מתנגנים בשני ערוצים נפרדים: מוזיקת-רקע (תהליך משנה MenuLoop.exe) וצליל לחיצה מקומי, כך ששניהם יכולים להתנגן בו-זמנית מבלי להפריע זה לזה. להפצת המערכת יצרתי מתקין (NI Installer) הכולל את NI-Serial Runtime 20.0 ואת NI LabWindows/CVI Shared Runtime 2020 f3 כדרייברים להתקנה, ולכן ניתן להריץ את המשחק על-גבי כל מחשב Windows (7 SP1 ומעלה) ללא צורך בהתקנת CVI מלאה.

הצגת החלונות של התוכנית

1. הפאנל הראשי – תפריט הבית (MAIN)



תפריט הבית הוא צומת הניווט הראשי של המשחק, בו כל לחצן פותח את החלון המיועד לו שעליהם אפרט בהמשך.

הוא בעצם מבצע קישור בלבד ולא קיימות בו פעולות מורכבות תוכנית.

NEW GAME – בודק קישוריות ומתחיל משחק מחדש.

LOAD GAME - בודק קישוריות ומאפשר טעינה של משחק שמור.

INSTRUCTIONS - מציג את פאנל ההנחיות.

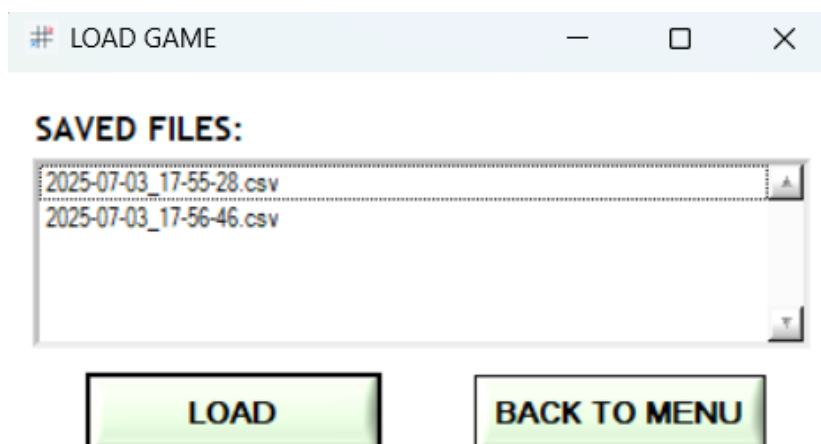
ABOUT – מידע כללי וקרדיטים.

QUIT – יציאה וסגירת הממשק.

כשנמצאים בחלון זה, מתנגנת מוזיקת רקע "menu.wav" ועבור כל לחיצה על כפתור מלווה הצליל "Click.wav".

קבצי האודיו מופעלים מתוך המקור Audio.c בעזרת פונקציית PlaySound אך עבור מוזיקת הרקע תהליך זה קורה חיצונית לקובץ ההפעלה מכיון שPlaySound לא מאפשרת ניגון של שני קבצים במקביל באותו התהליך, לכן Audio.c קוראת ל-API חיצוני בשביל לאפשר ניגון מקבילי ללא הפרעה – על כך ארחיב בהמשך.

2. פאנל הטעינה (LOAD_GAME)



מציג את רשימת קבצי ה CSV השמורים בתיקיית "Saved Games" בעזרת בקר List Box שמאפשרת בחירה בלעדית של קובץ אחד בלבד.

כתיבה וטעינה של קבצים אלו מנוהל ע"י קובץ המקור FileLog.c.

כדי להגיע בהצלחה לתפריט זה יש לקבל אישור על קישוריות בין שני המחשבים – שינתן בחלון הסטאטוס (POPUP), ניהול הקישוריות/כתיבה/קריאה/ACK ינוהל בעזרת קובץ המקור ComThread.c.

לחיצה על LOAD תוביל לחלון המשחק הטעון בעזרת פיענוח קובץ המשחק השמור ותציג אותו עבור 2 המשתמשים – תיזמון התורים וסמל (איקס או עיגול) המשתמשים יישמר ומיובא בעזרת המידע הקיים בקובץ.

אין צורך ששני המשתמשים ילחצו יחדיו על כפתור הLOAD בשביל להתחיל משחק – דבר זה קורה ברגע שאחד המשתמשים עשה זאת.

3. פאנל ההנחיות (INSTRUCT)

INSTRUCTIONS

Tic-Tac-Toe – Quick Rules

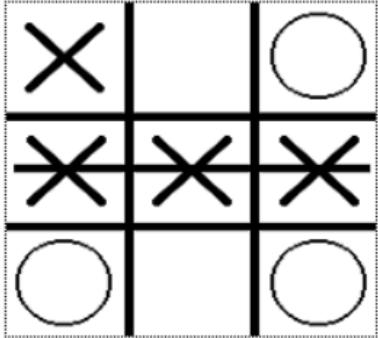
The board is a 3 × 3 grid.

Player X starts; players alternate turns.

On your turn, mark one empty square with your symbol (X or O).

First player to line up three of their symbols horizontally, vertically, or diagonally wins.

If all nine squares are filled with no winner, the game ends in a draw (tie).



Starting a New Game

In the MAIN MENU, click NEW GAME.

A pop-up "Establishing Connection..." will open.

Upon 'Connection Success' , the Tic-Tac-Toe game panel will appear.

Loading a Saved Game

Choose LOAD GAME in the MAIN MENU.

Choose a saved log file to restore the board

Hardware Needed

Two connected computers with an RS-232 serial cable (or any cable/adaptor that supports the RS-232 protocol).

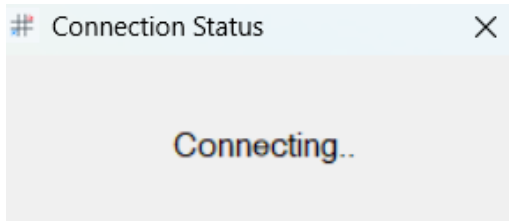
BACK TO MENU

פאנל ההנחיות ממלא את הצורך בהסבר על חוקי המשחק והחומרה הזקוקה להפעלת התוכנה בצורה תקינה.

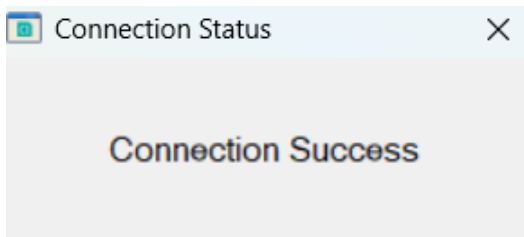
פאנל זה הינו פאנל פשוט למדי.

בעל בקר אנימציה שמפעיל רצף של תמונות טעונות מראש לפי סדר הנקבע מראש וכפתור חזרה לתפריט הראשי BACK_TO_MENU.

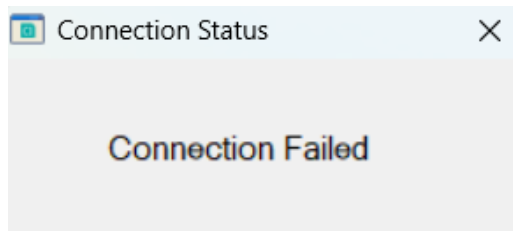
4. פאנל סטאטוס החיבוריות (POPUP)



- בעת ניסיון התחברות, הפאנל יציג Connecting... Connecting.. Connecting.
עד שינוי בסטאטוס חיבוריות
(CTS_FAILED או CTS_CONNECTED)



- חיבור תקין (CTS_CONNECTED) יתקבל עם
הודעת Connection Success בעזרת ACK
שעבור בהצלחה (BYTE_OK ו-BYTE_HELLO).

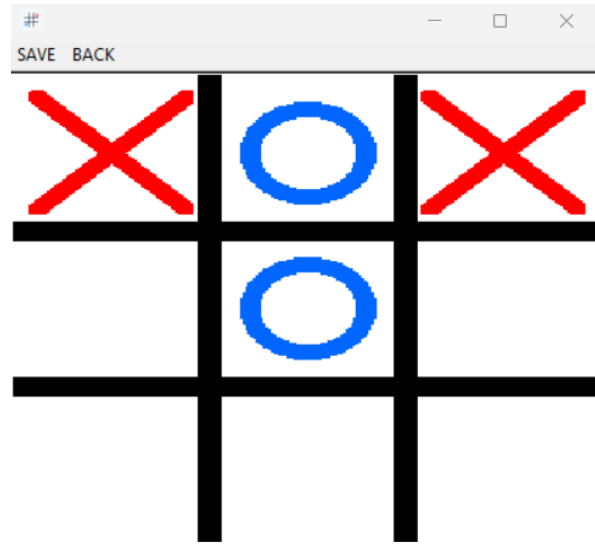
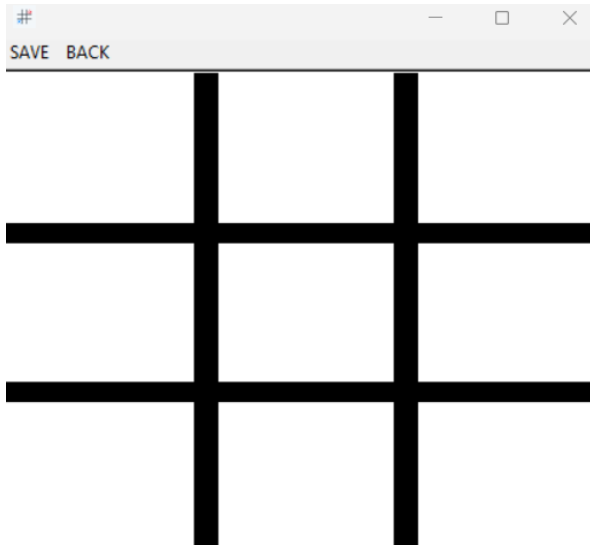


- ניתוק (CTS_FAILED) יתקבל עם הודעות
Connection Failed כתוצאה מ-TIMEOUT או
שגיאה שהתקבלה מ-ComDisconnectCallback
שהוא interrupt הנוצר מדגלי שינוי בקו הRS232
(LWRS_DSR | LWRS_RLSD | LWRS_ERR).

תהליך הכתיבה והקריאה מנוהל בצורה טורית ומקבילית:

- קריאה וACK מקבילית: ComWorker ה-Thread הראשי של התוכנית שמוודא חיבוריות ופיענוח של הביטים שנשלחים בקו.
- כתיבה טורית: ComThread_SendMove מעביר את מהלך המשתמש הנוכחי.

5. פאנל המשחק (GAME)



פאנל המשחק הינו פאנל אינטראקטיבי שמגיב ללחיצת העכבר לפי תור המשתמש.

הפאנל מורכב כולו מבקר מסוג Canvas, הטעון בתמונת הלוח ("Board.png") בעת אתחול המשחק ו-menubar שמאפשר שמירה של מצב הלוח הנוכחי וחזרה לתפריט הראשי.

לוח המשחק מחולק לריבועים שהקורדינאטות שלהם מחולקות לפי תאי הלוח. כשמגיע תור המשתמש – כל לחיצה על תחום אחד הריבועים תטעין את התמונה התואמת לסמל שנבחר עבורו (איקס או עיגול – "x.png" "o.png") אל תוך התא הרלוונטי. למשתמש לא ניתן לבצע אינטראקציה עם הלוח כשזה לא תורו.

בסוף המשחק תוצג הודעה התואמת את סטאטוס התרחשותו (win/loss/draw) לכל משתמש.

קבצי קוד הפרויקט

1. TicTacToe.c

קובץ המקור המרכזי שמחבר בין ממשק המשתמש (UI), מנוע הלוגיקה (Logic.c) ומודולי התקשורת/קבצים. הוא טוען את הפאנלים, מפעיל את מוזיקת-הרקע ויוצר את סבב האירועים הראשי בקוד (RunUserInterface). כל לחיצה על כפתור בתפריט הראשי מובילה אל Callback מתאים בקובץ זה. מכאן נקרא לחיבור התקשורת בין המחשבים, שמירה ואכלוס קבצי משחק ו-cleanup בעת יציאה/סגירה. הקובץ כמעט ואינו "מחשב" בעצמו – הוא בעיקר מתווך בין שכבות אחרות.

פונקציות מרכזיות:

```
int CVICALLBACK ButtonCallback (int panel, int control, int event,
                                void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            PlayClick(); // Click sound

            switch (control)
            {
                case MAIN_NEW_GAME:
                    currentHandle = game;
                    EstablishConnection(); /* start worker */
                    SetCtrlAttribute (mainHandle, MAIN_TIMER, ATTR_ENABLED, 1); // polling
                    break;

                case MAIN_LOAD_GAME:
                    currentHandle = loadGame;
                    EstablishConnection(); /* start worker */
                    SetCtrlAttribute (mainHandle, MAIN_TIMER, ATTR_ENABLED, 1); // polling
                    break;

                case MAIN_INSTRUCTIONS:
                    HidePanel(mainHandle);
                    DisplayPanel(instruct);
                    break;

                case MAIN_ABOUT:
                    DisplayAbout();
                    break;

                case MAIN_QUIT:
                    ComThread_ForceExit();
                    StopMenuMusic();
                    QuitUserInterface(0);
                    break;
            }
        }
}
```

מימוש כל אחד מכפתורי התפריט.

עבור לחיצה על NEW/LOAD GAME – מופעל ה Thread המרכזי והטיימר שמבצע polling להפעלת המשחק.

```

int CVICALLBACK CheckConnStatusCallback (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
    if (event != EVENT_TIMER_TICK)
        return 0;

    /* poll until connected or failed */
    if (GetConnectionState() == CTS_CONNECTED)
    {
        SetCtrlAttribute (mainHandle, MAIN_TIMER, ATTR_ENABLED, 0); // Stop polling

        if (currentHandle == game)
        {
            Logic_LaunchGame(GetSymbol());
            StopMenuMusic();
        }

        if (currentHandle == loadGame)
        {
            HidePanel(mainHandle);
            FileLog_PopulateSavedList(); // panel init - populate listbox
            DisplayPanel(loadGame);
        }
    }

    if (GetConnectionState() == CTS_FAILED)
    {
        SetCtrlAttribute (mainHandle, MAIN_TIMER, ATTR_ENABLED, 0); // Stop polling
        MessagePopup ("Serial", "Could not establish connection.");
    }

    return 0;
}

```

מימוש טיימר התוכנית המבצע את polling בעזרת בדיקה של מצב החיבור.

סטאטוס החיבור מוגדר בעזרת שני מצבים סופיים מצב אחד התחלתי ומצב ביניים:

```

/* public connection states */
typedef enum {CTS_IDLE = 0, CTS_CONNECTING, CTS_CONNECTED, CTS_FAILED} ConnState;

```

(מתוך ComThread.h)

כלומר כל עוד אנחנו בלא במצב סופי (CONNECTED || FAILED) אנחנו נמשיך את תהליך polling.

המצב ההתחלתי (IDLE) ישתנה בעת קריאה ל-EstablishConnection ולכן אין צורך לטפל בו במנגנון זה.

בכל מצב סופי – נסגור את הטיימר.

בחיבור – נאתחל את המשחק וסימבול השחקן.

בניתוק – נשלח הודעה לחיווי של נתק.

```

case LOAD_GAME_LOAD_FILE:

    PlayClick(); // Click sound
    int index;
    GetCtrlVal (panel, LOAD_GAME_SAVED_GAMES, &index); /* With a HOT list box, GetCtrlVal gives the *item value* (here: i) */
    if (index < 0)
    {
        MessagePopup ("Load", "Please choose a file first.");
        break;
    }

    char fname[MAX_PATHNAME_LEN];
    GetLabelFromIndex (panel,
        LOAD_GAME_SAVED_GAMES,
        index,
        /* same value we stored */
        fname);
        /* returns the label string */

    /* Build full path and load */
    char path[MAX_PATHNAME_LEN];
    sprintf (path, "%s\\%s", SAVE_DIR, fname);

    Move seq[MAX_MOVES];
    int n, myS, oppS;

    if (FileLog_LoadGame (path, seq, &n, &myS, &oppS) == 0)
    {
        HidePanel (loadGame);
        Logic_LaunchGame (myS); /* show board, then apply moves one by one */

        for (int i=0;i<n;++i)
            Logic_ApplyMove (seq[i].row, seq[i].col, seq[i].sym);

        ComThread_SendByte ('B'); /* header */
        ComThread_SendByte ((char)oppS); /* peer symbol */
        ComThread_SendByte ((char)n); /* how many moves */
        for (int i = 0; i < n; ++i) {
            ComThread_SendByte ((char)seq[i].row);
            ComThread_SendByte ((char)seq[i].col);
            ComThread_SendByte ((char)seq[i].sym);
        }
        ComThread_SendByte ('E'); /* tail */
    }

    break;

```

כפתור טעינת קובץ שמירת משחק בפאנל הטעינה.

- פותח את קובץ ה-CSV שנבחר ומחזיר מספר-מהלכים + סמלי השחקנים.
- שולח חבילת-סנכרון אל הצד השני ('Byte 'B' + כמות המהלכים + oppSym).
- משגר את כל המהלכים דרך ComThread_SendMove ומריץ Logic_ApplyMove מקומית.
- בסיום מציג את הלוח ומשנה את myTurn בהתאם לשורה האחרונה שנשמרה.

2. ComThread.c – (LogicLib.dll)

קובץ-המקור שאחראי על כל שכבת ה-RS-232: פתיחת הפורט, ביצוע Handshake, שרות-רקע לקבלת נתונים וסנכרון מצב-החיבור מול ה-UI.

בתוכו המודול ComWorker רץ כ-Thread נפרד כדי שלא לחסום את הלולאה הגרפית של CVI.

הוא מגדיר פרוטוקול מינימלי (HELLO/OK, חבילות-M, row, col וכו') וממפה כל בת-קלט לפעולה במשחק באמצעות PostDeferredCall.

בנוסף מתבצעת כאן השגחה תקופתית על בריאות הקשר (Heartbeat / Line-Status) וסגירה נקייה במקרה של שגיאה.

פונקציות מרכזיות:

```
/* Called (later) from NEW_GAME: port is already open by AutoInitComPort() */
int EstablishConnection (void)
{
    if (AutoInitComPort(&gPort) < 0)
    {
        MessagePopup ("Serial Error",
                      "Could not find an available COM port.\n"
                      "Check the cable and try again.");
        return 0;          /* stay in MAIN menu */
    }

    InstallComCallback (gPort, LWRS_DSR | LWRS_RLSD | LWRS_ERR, 0, 0, ComDisconnectCallback, NULL);

    SetComTime(gPort, COM_TIMEOUT); // setting com timeout

    if ((popup = LoadPanel (0, "TicTacToe.uir", POPUP)) < 0)
        return -1;

    DisplayPanel(popup);
    SetCtrlVal (popup, POPUP_TXT_STATUS, "Connecting");

    int err = CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,
                                             ComWorker,
                                             NULL,          /* data ptr */
                                             &gThread);     /* ID */

    if (err < 0) {
        gState = CTS_FAILED;
        return err;
    }

    return 0;          /* thread launched OK */
}
```

פותח ומאתחל את הפורט (AutoInitComPort) לפי ערכי baud/parity ומגדיר Timeout.

מתקין את ה-interrupt שיגרום לניתוק כאשר נפגעה התקשורת עם המחשב השני.

מפעיל את ה-Worker Thread של התוכנית.

```

static int CVICALLBACK ComWorker (void *data)
{
    int dotPhase = 0; /* 0-3 */
    gState = CTS_CONNECTING;

    ComWrtByte (gPort, BYTE_HELLO); /* kick off our own HELLO ? both sides do this once */

    int timeoutTicks = TIMEOUT_TICKS; /* TIMEOUT_TICKS * (5 ms + COM_TIMEOUT) = 15.075 s timeout */
    while (timeoutTicks-- > 0 && gState == CTS_CONNECTING)
    {
        int ch = ComRdByte (gPort);
        if (ch < 0) {
            PostDeferredCall (UpdatePopupDots, (void*)(intptr_t)dotPhase);
            dotPhase = (dotPhase + 1) & 3; /* 0->1->2->3->0 */
            Delay (0.005);
            continue;
        }

        if (ch == BYTE_HELLO)
            ComWrtByte (gPort, BYTE_OK); /* peer's hello ? answer */
        else if (ch == BYTE_OK)
            gState = CTS_CONNECTED; /* handshake done */
    }

    if (gState != CTS_CONNECTED)
        gState = CTS_FAILED;

    PostDeferredCall (FinalPopup, (void*)(intptr_t)gState);

    DecideSymbol(); // calling this function from here for reading sync of symbol

    while (gState == CTS_CONNECTED)
    {
        int ch = ComRdByte (gPort);
        if (ch >= 0)
            ProcessIncomingByte ((unsigned char) ch);
        Delay (0.005);
    }
    return 0;
}

```

ה-Worker Thread של התוכנית שתפקידו:

- לבצע ACK ראשוני לתקשורת בין המחשבים אשר הסטאטוס שלו מוצג למשתמש בפאנל סטאטוס החיבוריות.
- קובע את סמל השחקן ומעדכן אותו במשתנה gMySymbol.
- נכנס ללולאת-רקע: while (gState == CTS_CONNECTED) שקוראת כל 5ms ומעבירה כל בייט ל-ProcessIncomingByte לעיבוד הבייט.

```

static void CVICALLBACK ComDisconnectCallback (int port, int eventMask,
                                              void *cbData)
{
    /* confirm peer still asserts carrier / DSR ----- */
    int lineBits = GetComLineStatus (port);          /* always safe - no GP fault */
    int peerAlive = (lineBits & kRS_DSR_ON) &&        /* DSR asserted */
                   (lineBits & kRS_RLSD_ON);          /* Carrier-Detect on */

    if (!peerAlive)
    {
        gState = CTS_FAILED;
        PostDeferredCall ((void*)Logic_EndGame, NULL);
    }
    /* else: benign transition, ignore and continue */
}

```

Callback שנרשם מול InstallComCallback עבור האירועים:

LWRS_RLSD – דרגת-המתח על קו CD/RLSD (Carrier Detect/Receive Line Signal Detect) משתנה.

LWRS_DSR – החומרה (מודם/USB-Serial) מודיעה שהיא מוכנה, כאשר המכשיר בצד השני מכבה את DSR (Data-Set-Ready) ייתכן שנסגר חיבור RS-232.

LWRS_ERR – דרייבר ה-UART זיהה בתעבורה שגיאת מסגרת, עודף בתור או שגיאת parity.

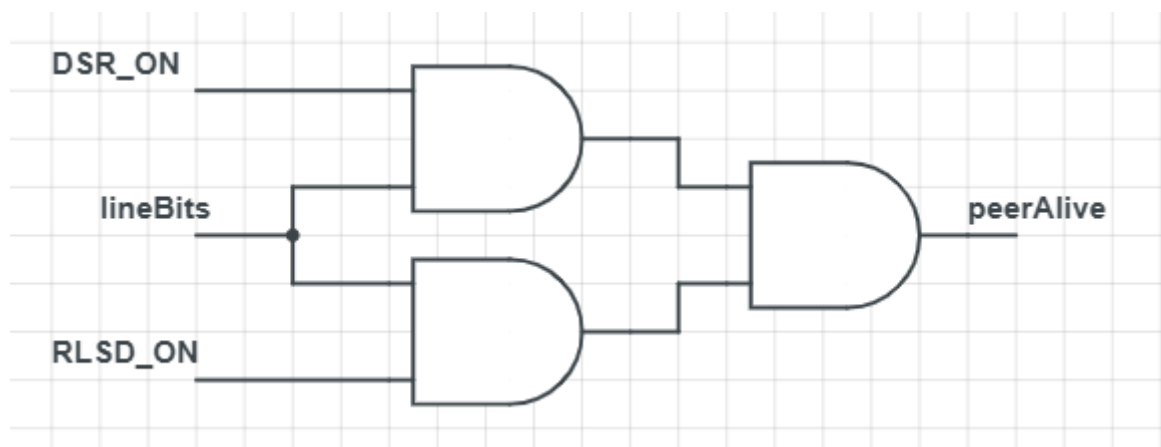
כאשר קו ה-RS-232 מתנתק-פיזית או נוצרה שגיאת-קו, הפונקציה מופעלת.

הקריאה GetComLineStatus(port) בודקת שה-DSR/CD עדיין ON. אם לא – קובעת $alive = 0$.

עדכון גלובלי – אם נמצא ניתוק מעבירה לאשכול ה-UI הודעת "Connection Lost" ומונעת קריאות RS-232 נוספות.

האלגוריתם מנצל Interrupt-Driven Error Handling במקום Polling (זמן-תגובה מיידי ללא לולאה).

הסבר על peerAlive:



סטטוס הקווים שהפעיל את הInterrupt (lineBits) ואם הקו הספציפי (RLSD/DSR) במצב ON נבדקים ביניהם ותוצאתם אחת בשנייה כדי לתת אינדיקציה מוחלטת האם הקו 'בחיים'.

3. FileLog.c – (LogicLib.dll)

קובץ FileLog.c אחראי לשכבת התיעוד (Logging) ולניהול קובצי שמירה בפורמט CSV. הוא שומר כל צעד במשחק בזמן אמת, רושם מי-הסמל (X/O), ותאריך-שעה ייחודי עבור קובץ השמירה.

בנוסף הוא בונה את רשימת הקבצים הקיימים ומציג אותה בלוח LOAD GAME ב-UI.

הקובץ גם מטפל בשחזור משחק: קריאת הקובץ, שליחת המהלכים ל-Logic.dll וסנכרון עם הצד המרוחק.

כל הלוגיקה נמנעת מאלגוריתמים מורכבים – הדגש הוא על I/O בטוח, פורמט עקבי, והפרדת אחריות מה-UI.

אלמנטים גלובליים חשובים:

```
typedef struct { int row, col, sym; } Move;    MOVE = DATA PACKET
static Move moveBuf[MAX_MOVES];               מערך של חבילות
static int moveCnt = 0;
static int dirty = 0;
```

פונקציות מרכזיות:

```
void FileLog_RegisterMove (int r,int c,int s)
{
    if (moveCnt < MAX_MOVES) {
        moveBuf[moveCnt++] = (Move){r,c,s};
        dirty = 1;
    }
}
```

נקראת מכל צד אחרי כל מהלך תקף. מוסיפה רשומה למערך המהלכים (struct Move), מעלה את מונה המהלכים ומסמן `dirty = TRUE`.

מנגנון זה מבטיח שה- SAVE GAME יוצג כפעיל רק כאשר בוצע שינוי מאז השמירה הקודמת.


```

int FileLog_SaveCurrentGame (void)
{
    if (!dirty) return 0;                /* nothing to save */

    EnsureSaveDir();

    char file[MAX_PATHNAME_LEN];
    BuildFilename (file);

    FILE *fp = fopen (file, "w");
    if (!fp) { MessagePopup("Save","Cannot create file"); return -1; }

    fprintf (fp, "mySym,%d,oppSym,%d\n", mySym, oppSym);
    for (int i=0;i<moveCnt;++i)
        fprintf (fp, "%d,%d,%d\n",
                    moveBuf[i].row, moveBuf[i].col, moveBuf[i].sym);
    fclose (fp);

    dirty = 0;
    return 0;
}

```

בונה שם קובץ ייחודי על-פי חותמת תאריך-שעה (YYYY-MM-DD_hh-mm-ss.csv), יוצר את תיקיית "Saved Games" במידת הצורך (פונקציות עזר EnsureSaveDir ו-BuildFileName).

כותב שורת-(mySym/oppSym) header ואחריה את כל המהלכים בפורמט row,col,sym. האלגוריתם פשוט אך משתמש ב-fprintf במצב "uncompressed" למניעת תלויות-קוד.

```

int FileLog_LoadGame (const char *path, Move moves[], int *num, int *oMy, int *oOpp)
{
    FILE *fp = fopen (path, "r");
    if (!fp) return -1;

    char line[64];
    if (!fgets(line,sizeof(line),fp)) { fclose(fp); return -1; }
    sscanf (line, "mySym,%d,oppSym,%d", oMy, oOpp);

    int n=0;
    while (fgets(line,sizeof(line),fp) && n<MAX_MOVES)
        sscanf (line, "%d,%d,%d",
                    &moves[n].row,&moves[n].col,&moves[n].sym), ++n;

    fclose(fp);
    *num = n;
    return 0;
}

```

קורא קובץ CSV קיים, שולף את סמלי המשתמשים, סורק כל שורה למערך של Move, ומעדכן את הערכים חזרה ל-Logic. בעת טעינה הוא גם מחזיר num (מס' המהלכים) כדי שה-Logic ישחזר בזמן נכון.

4. Logic.c – (LogicLib.dll)

Logic.c הוא "מוח" המשחק - הוא מחזיק את מטריצת הלוח board[3][3], מונה מהלכים, סמלי השחקנים ואת דגל myTurn. הקובץ מממש את כללי Tic-Tac-Toe – זיהוי פגיעה בתא, ציור הסמל ל-UI, בדיקת ניצחון/תיקו והחלפת תור. הוא גם מגשר בין שכבת התקשורת (ComThread) ובין ה-UI: מקבל קליק מקומי, מקבל מהלך מרוחק.

אלמנטים גלובליים חשובים:

```
/* 9 board rectangles ----- */
typedef struct { int l,t,r,b; } CellRect; /* left, top, right, bottom */
static CellRect cell[3][3];
```

פונקציות מרכזיות:

```
void Logic_LaunchGame (int mySymbol)
{
    moveCount = 0;

    if(!LoadBitmaps())
        { MessagePopup("Error","Cannot load symbol PNGs"); return; }

    HidePanel(mainHandle);
    DisplayPanel(game);
    StopMenuMusic();

    memset(board,0,sizeof(board)); // resets board variable
    BuildCellRects();

    mySym = mySymbol;
    oppSym = (mySym==SYMBOL_X)? SYMBOL_O : SYMBOL_X;
    myTurn = (mySym==SYMBOL_X);          /* X starts */

    FileLog_Reset (mySym, oppSym); // call after computing the symbols
    CanvasClear (game,GAME_CANVAS,VAL_ENTIRE_OBJECT);
    CanvasDrawBitmap (game,GAME_CANVAS,bmpBoard, VAL_ENTIRE_OBJECT,VAL_ENTIRE_OBJECT); /* draw fresh board background */
}
```

מאותחלת לאחר שה-RS-232 הגיע למצב CTS_CONNECTED. מאפסת את הלוח ואת מונה המהלכים, קובעת מי מתחיל (X תמיד ראשון), טוענת את Board.png ל-Canvas, ומעדכנת את myTurn. אם המשחק נטען מקובץ היא קוראת ל-FileLog_Reset (...) כדי לסנכרן סמלים לפני שחזור.

הפונקציה נקראת מתוך טיימר ה-polling מ-TicTacToe.c.

```

void Logic_ApplyMove (int r,int c,int sym)
{
    if (board[r][c]) return;           /* already taken - ignore */

    board[r][c] = sym;
    DrawSymbol (r, c, sym);
    moveCount++;

    int wr,wc,dg;
    if (CheckWin (&wr,&wc,&dg) == sym) {
        DrawStrike (wr, wc, dg);
        MessagePopup (sym == mySym ? "Win" : "Game Over",
                      sym == mySym ? "You won!" : "You lost.");
        FileLog_Reset (mySym, oppSym);    /* clear buffer / dirty */
        Logic_EndGame();
        return;
    }
    if (moveCount == 9) {
        MessagePopup ("Draw", "No winner - game tied.");
        FileLog_Reset (mySym, oppSym);
        Logic_EndGame();
        return;
    }
    myTurn = (sym != mySym);             /* switch turn */
}

```

מיישמת את חבילת המידע שהתקבלה אליה כפרמטרים אל הלוח.

ליבה אלגוריתמית: שומרת את sym (symbol) בלוח כדי לקבוע איזה מהסמלים ביצע את התור הנוכחי (על מנת ליצור תיעוד מדויק שיעזור בשחזור בעת טעינת משחק שמור), מציירת את הסמל בקואורדינטות המדויקות, מעלה את moveCount ובודקת ניצחון באמצעות CheckWin (מעבר על 8 קומבינציות).

במקרה ניצחון/תיקו קוראת ל-DrawStrike ומפעילה את Logic_EndGame.

האלגוריתם דטרמיניסטי ומחזורי O(1).

```

void Logic_RemoteMove (void *data)
{
    typedef struct { int r,c; } Move;
    Move *mv = (Move *) data;
    FileLog_RegisterMove (mv->r, mv->c, oppSym); /* record for save/load */
    Logic_ApplyMove (mv->r, mv->c, oppSym);
    free (mv);
}

```

עטיפה ל-PostDeferredCall.

מקבלת מצמד row/col/sym מ-ComWorker, ממירה את void* לMove, ומעבירה ל-

Logic_ApplyMove.

כך נשמר Thread-Safety – ציור ועבודה על הלוח תמיד מתבצעים ב-Thread ה-UI.

הפונקציה הזו מופעלת בתוך ProcessIncomingByte השייכת ל-ComWorker בתוך ComThread.c.

```

void Logic_EndGame (void)
{
    /* stop serial + worker thread */
    ComThread_EndGame();

    /* offer to save only if something changed */
    if (FileLog_IsDirty())
    {
        if (ConfirmPopup ("Quit", "Save game before exiting?"))
            FileLog_SaveCurrentGame();
    }

    /* reset local state for the next match */
    moveCount = 0;
    myTurn    = 0;

    /* return to main menu UI */
    HidePanel (game);
    DisplayPanel (mainHandle);
    StartMenuMusic();
}

```

פונקציית ניקוי מרוכזת.

בודקת FileLog_IsDirty ומציגה ConfirmPopup אם צריך, קוראת ComThread_EndGame ו-StartMenuMusic לחזרה למסך הראשי.

5. Audio.c – (LogicLib.dll)

קובץ Audio.c אחראי לשכבת הצליל של המשחק.

הוא מספק תפעול של מוזיקת-רקע בלולאה ושמיעת "קליק" בעת לחיצה על כפתורי ה-UI.

הקוד מטפל במגבלת PlaySound (רק צליל אסינכרוני אחד בכל תהליך) באמצעות שימוש בתהליך-עזר נפרד (MenuLoop.exe) שמנגן את קובץ WAV ברצף.

בנוסף מתבצע חיפוש ודחיסה (8.3) של נתיב ה-WAV כדי למנוע שגיאת MCI בשמות ארוכים.

ה-API בקובץ נחשף החוצה כ-DLL בעזרת סמל LOGIC_API לצורך שימוש מהאפליקציה הראשית.

אלמנטים גלובליים חשובים:

```
static HANDLE bgmProc = NULL; /* helper process handle */
static HANDLE bgmThread = NULL; /* helper primary thread */

#define AUDIO_BGM_PATH "Sounds\\menu.wav"
```

תהליך חיצוני – MenuLoop.exe

```
int APIENTRY WinMain (HINSTANCE h, HINSTANCE p, LPSTR c, int n)
{
    PlaySound ("Sounds\\Menu.wav",
              NULL,
              SND_FILENAME | SND_ASYNC | SND_LOOP | SND_NODEFAULT);
    /* message-only loop: close on WM_CLOSE */
    MSG msg;
    while (GetMessage (&msg, NULL, 0, 0))
        if (msg.message == WM_CLOSE) break;
    PlaySound (NULL, NULL, 0); /* stop */
    return 0;
}
```

התהליך הקטן "MenuLoop" הוא קובץ exe. נפרד שמנגן בלולאה אינסופית את הקובץ Sounds\\Menu.wav. הוא נכתב בכ-20 שורות בלבד ומשתמש ב-API Win32 ב-PlaySound כדי להתחיל להשמיע את הקובץ בצורה אסינכרונית וחוזרת (הדגלים SND_ASYNC | SND_LOOP). לאחר מכן הוא נכנס ללולאת-הודעות מינימלית. כל עוד לא מתקבל WM_CLOSE הוא ממשיך לרוץ ברקע ולשמור על המוזיקה פעילה, מבלי לחסום את תוכנת TicTacToe הראשית. עם קבלת WM_CLOSE הוא מפסיק את הסאונד ע"י קריאה נוספת ל-PlaySound(NULL, ...) ומשחרר משאבים לפני סיום. מבנה זה מאפשר להריץ מוזיקה ורעשי-לחיצה יחד – תהליך אחד לטיפול באודיו רציף, ותהליך אחר להפעלת צלילי-מערכת קצרים.

פונקציות מרכזיות:

```
int StartMenuMusic (void)
{
    /* if already running, do nothing */
    if (bgmProc) return 0;

    STARTUPINFO      si = { sizeof si };
    PROCESS_INFORMATION pi = {0};

    BOOL ok = CreateProcess ("MenuLoop.exe",
                             NULL,          /* command line */
                             NULL, NULL,     /* security     */
                             FALSE,         /* inherit hndl */
                             CREATE_NO_WINDOW,
                             NULL, NULL,    /* env / dir    */
                             &si, &pi);

    if (!ok)
    {
        MessagePopup ("Audio - helper launch failed",
                      "MenuLoop.exe not found or cannot start.");
        return -1;
    }

    bgmProc  = pi.hProcess;
    bgmThread = pi.hThread;
    return 0;    /* success */
}
```

- בודקת אם מוזיקה כבר רצה (bgmProc) וחוזרת במקרה כזה.
- מחשבת נתיב קובץ menu.wav קצר (GetBgmShortPath) ומזניקה תהליך MenuLoop.exe עם CreateProcess ללא חלון.
- שומרת ידיות לתהליך ול-Thread הראשי כדי לאפשר שליטה בהמשך.
- אם ההרצה נכשלת מוצג MessagePopup עם פירוט השגיאה.

```

void StopMenuMusic (void)
{
    if (!bgmProc) return;          /* nothing to do */

    /* ask the helper nicely */
    PostThreadMessage (GetThreadId (bgmThread), WM_CLOSE, 0, 0);

    /* wait up to 1 s - then kill */
    if (WaitForSingleObject (bgmProc, 1000) == WAIT_TIMEOUT)
        TerminateProcess (bgmProc, 0);

    CloseHandle (bgmThread);
    CloseHandle (bgmProc);
    bgmThread = bgmProc = NULL;
}

```

- שולחת לתהליך העזר הודעת WM_CLOSE (באמצעות PostThreadMessage) כדי לבקש סיום מסודר.

- ממתינה עד v1 שנייה ל-Exit. אם עבר הזמן – קוראת TerminateProcess גס.

- משחררת את כל הידיות (CloseHandle) ומאפסת bgmThread / bgmProc.

- כך נמנעת דליפת משאבים גם אם התהליך לא נענה לבקשה.

```

void PlayClick (void)
{
    /* play click */
    PlaySound ("Sounds\\Click.wav",
              NULL,
              SND_FILENAME | SND_ASYNC | SND_NODEFAULT);
    return;
}

```

מנגנת את Click.wav באופן אסינכרוני כמעטפת ל-PlaySound עם דגלים SND_FILENAME ו-SND_NODEFAULT.

הגדרות dll בקוד

```
#ifndef _WIN32
#define LOGIC_EXPORTS /* defined when building LogicLib.dll */
#define LOGIC_API __declspec(dllexport)
#else
#define LOGIC_API __declspec(dllimport)
#endif
#endif
#define LOGIC_API
#endif
```

– `__declspec(dllexport)` ומתווסף `LOGIC_EXPORTS` מוגדר הלחצן `LogicLib.dll` קומפילציית

כך שמנהל הקישור מייצר טבלת יצוא.

כאשר הפרויקט הראשי כולל לדוגמא את `Logic.h` ללא `LOGIC_EXPORTS`, המקרו הופך ל-`dllimport` כדי שהלינקר ידע למשוך את הסמל מה-DLL.

מחוץ ל-Windows (למשל Linux/-macOS) אין צורך במאפיינים מיוחדים, ולכן המקרו ריק.

בקבצי המקור נגדיר:

```
#define LOGIC_EXPORTS /* enables __declspec(dllexport) */
```

דוגמא לחתימות הפונקציות:

```
LOGIC_API int StartMenuMusic (void);
LOGIC_API void StopMenuMusic(void);
LOGIC_API void PlayClick (void);
```

עבור משתנים חיצוניים (המזוהים ביותר מקובץ אחד) נגדיר ספציפית.
לדוגמא:

```
/* main game panel */
__declspec(dllexport) int game; /* GAME panel */
__declspec(dllexport) int mainHandle; /* MAIN_MENU panel */
__declspec(dllexport) int loadGame; /* LOAD_GAME panel */
extern __declspec(dllexport) int gPort;
```

תיעוד באגים ופתרונות תיקון לקוד

#	תיאור הבעיה	הגורם	דרך הפתרון
1	המהלך האחרון לא שודר ליריב – הלוח בצד אחד עודכן, ובצד השני לא.	סגירת ה-COM מיד לאחר Logic_ApplyMove, לפני הקריאה ל-ComThread_SendMove.	הקדמתי את שליחת המהלך והלוג (RegisterMove) לפני בדיקת הניצחון/תיקו, כך שה-COM נשאר פתוח עד לסיום השידור.
2	מוזיקת תפריט וצליל-לחיצה לא יכלו להתנגן בו-זמן (PlaySound מוגבל לצליל אסינכרוני אחד לתהליך).	מגבלות WinMM.	יצרתי תהליך חיצוני זעיר MenuLoop.exe שמריץ PlaySound בלולאה. האפליקציה הראשית שומרת על צלילי-לחיצה באמצעות sndPlaySound, כך שכל תהליך מגן ערוץ נפרד.
3	איתור ניתוק יריב – קושי להבחין אם רק "אין תנועה" או שה-COM של היריב נותק.	בדיקת Heartbeat לא מספיקה אם השחקן ממתין לתורו.	הוספתי InstallComCallback עם LWRD_DSR.
4	קובצי wav. עם שמות ארוכים* גרמו לשגיאת "invalid filename" ב-mciSendString.	WinMM דורש מסלול לא מוגן ברווחים+גרשיים.	עטפתי את הנתביב ב-R"..." והעברתי כ-absolute path. כמו כן הוספתי טיפול שגיאות עם mciGetErrorString.
5	קישור DLL – סמלים כפולים או חסרים (__imp__game, gPort וכו') בין LogicLib לבין TicTacToe.exe.	משתנים גלובליים הוגדרו פעמיים / לא יוצאו כראוי.	העברתי קבועים גלובליים ל-DLL וצירפתי extern באפליקציה. השתמשתי במאקרו LOGIC_API עם __declspec(dllexport/dllimport) למניעת הגדרות כפולות
6	חסרים קבצי הרצה (cvi32.dll) במחשבים ללא CVI.	ההתקנה המקורית לא כללה את CVI Runtime.	בהתקנת ה-Distribution הוספתי NI LabWindows/CVI Shared Runtime + NI-Serial Runtime בלשונית Drivers & Components. בניתי קובץ התקנה וdeploy עם README של כל הקבצים ההכרחיים.
7	PlaySound ביטל את המוזיקה בזמן פקדי UI נוספים (חיבור/ניתוק).	PlaySound עם דגל SND_NOSTOP לא פתר לגמרי.	עברתי לשילוב MCI עבור מוזיקת-רקע, PlaySound א-סינכרוני קצר לצלילי-לחיצה, וסדר עדיפויות מחרוזות-פקודה (play ... repeat). (לפני פתרון סעיף 2 – לא פתר את הבעיה)
8	שגיאת "File not found" ב-LoadPanel במחשב יעד.	קובץ TicTacToe.uir לא הוטמע בקובץ ההפעלה.	סימון הטמעת קובץ ה-uir ב-build >Target Settings

קישורים להורדה

הקוד בpdf



TicTacToe.c

- TicTacToe.proj



LogicLib.proj.zip

- LogicLib.proj



MenuLoop.c.pdf

- MenuLoop.proj

Installer



RS232-TicTacToe_Setup.zip

- TicTacToeSetup.exe